A Cost-Efficient Method for Big Geospatial Data on Public Cloud Providers

Joao Bachiega Junior, Marco Antonio Sousa Reis *
Aletéia Patrícia Favacho de Araújo, Maristela Holanda **
Department of Computer Science
University of Brasilia
Brasília/DF, Brazil
* e-mail: {joao.bachiega.jr, marco.antonio.sousa.reis}@gmail.com
** e-mail: {aleteia, mholanda}@unb.br

***Abstract*—The rise of big geospatial data creates the need for an environment with powerful computational resources to process this large amount of geographical information. Spatial Cloud Computing is a solution to this problem as it offers facilities to overcome the challenges of a big data environment, providing significant computer power and vast storage. However, the software to process this data requires great performance capacity. These requirements are met by SpatialHadoop, a fully-fledged MapReduce framework with native support for spatial data. This paper presents a cost-efficient method for processing geospatial data on public cloud providers, optimizing the number of data nodes in a Hadoop cluster according to dataset size. Tests have proven that it can optimize the use of computational resources for available SpatialHadoop datasets.**

Keywords - Big geospatial data; Hadoop; SpatialHadoop; Spatial Cloud Computing

I. INTRODUCTION

Big geospatial data is the emerging paradigm for the infinite amount of information that has become available to users with the development and widespread use of Geographical Information System (GIS) softwares, delivering hundreds of TiB up to several PiB per hour [1][7]. Big data is defined by some authors [2][3] according to three essential aspects: i) *Variety* – referring to the different types of data, with more than 80% of them in an unstructured form; ii) *Volume* – the tremendous amounts of data generated each second; iii) *Velocity* – the speed at which new data is being produced. Recently, new aspects were included in the big data definition [7]: *Veracity* – how trustworthy the data is; *Value* – referring to the importance that the data has to the business; *Variability* - referring to data whose meaning is constantly changing and *Visualization* – how the data is presented, readable and accessible to users.

The rise of cloud computing and cloud data stores has been a precursor to, and a facilitator of the emergence of big data [9]. Consequently, to support the computational demand big data has caused, mainly for geospatial data, Yang and Huang [4] have proposed Spatial Cloud Computing, an infrastructure that helps conduct relevant computing and data processing, which is characterized by its sufficient computing capability, low energy cost, fast response to spike computing needs, and a wide accessibility to the public when needed.

An important property of clouds is their capability to increase and decrease computational resources without impact on applications. NIST [4] identified elasticity as an essential characteristic of cloud computing: "capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time".

Moreover, Kramer and Senner [1] assert that the cloud offers virtually unlimited resources in terms of processing power and memory, and that the faster geospatial data is processed, the higher its practical value will be. However, public cloud providers, such as Amazon AWS [24], Microsoft Azure [25], Google Cloud [26] among others, charge computational resources by the minute or by byte transferred, which is extremely costly. Therefore, using the computational resources in the most efficient way is essential to minimize costs.

Big Geospatial data demands a large number of resources to store and process information. The amount of computational resources required by this vast volume of information grows in an asymptotic way and each wasted resource represents a significant financial loss, which could have been avoided. Faced with this dilemma, some methods have been developed to process big data [2]. Among them, Apache Hadoop, a programming framework for distributed computing using the divide and conquer (or Map and Reduce) method to break down complex big data problems into small units of work and process them in parallel. Specifically for big geospatial data, some applications have been developed using Hadoop concepts [9], such as the following: i) "GIS Tools on Hadoop", that work with the ArcGIS product; ii) Parallel-Secondo as a parallel spatial Data Base Management System (DBMS) that uses Hadoop as a distributed task scheduler; iii) MD-HBase extends HBase, a non-relational database for Hadoop, to support multidimensional indexes; iv) Hadoop-GIS extends Hive, a data warehouse infrastructure built on top of Hadoop with a uniform grid index for range queries and self-join. Finally, Eldawy and Mokbel [9] presented SpatialHadoop, a fully-fledged MapReduce framework with native support for

 ISBN: 978-1-61208-539-5

spatial data with better performance when compared to all the other applications listed.

This article presents a cost-efficient method for determine the cluster size for processing big geospatial data using SpatialHadoop on public cloud providers. The goal is to optimize the use of computational resources to reduce costs. Section II covers concepts of SpatialHadoop. Section III presents concepts about Spatial Cloud Computing. Section IV presents the system architecture. The methodology used to develop this method is explained in Section V and the previous work done in this area is discussed in Section VI. The tests carried out and results obtained are shown in Section VII and Section VIII contains the conclusion and some suggestions for future work.

II. SPATIALHADOOP

Hadoop is the most popular technique for working with big data. It uses a MapReduce paradigm that break big problems in small ones and process these jobs in a distributed computation [9]. SpatialHadoop was developed as a fully-fledged MapReduce framework with native support for spatial data. It was built on Hadoop base code, adding spatial constructs and the awareness of spatial data inside the core functionality of traditional Hadoop.

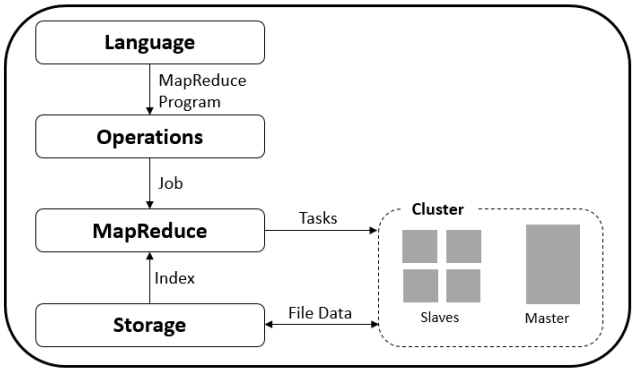

Figure 1. SpatialHadoop high-level architecture. Adapted by [9]

SpatialHadoop is composed of four main layers, namely language, operations, MapReduce and storage. All of them execute in a cluster environment with one master node that breaks a MapReduce job into smaller tasks, carried out by slave nodes [9]. The high-level architecture of SpatialHadoop is shown in Fig. 1.

A. Language Layer

The language used by SpatialHadoop is Pigeon, a simple high-level SQL-like language, derived from Pig Latin. It is compliant with the Open Geospatial Consortium's (OGC) simple feature access standard, which is supported in both open source and commercial spatial DBMS. Pigeon supports OGC standard *data types* including point, linestring and polygon, as well as OGC standard *functions* for spatial data import/export, querying and manipulation. Spatial *operations* are also included.

The spatial functionality is implemented as user-defined functions (UDFs), which are seamless to integrate with existing non-spatial operations in Pig Latin and also makes it compatible with all recent versions of Pig that support UDFs [11].

B. Operations Layer

This layer encapsulates the implementation of various spatial operations that use the spatial indexes and the new components in the MapReduce layer. According to [10], the operations layer is composed of:

- *Basic Operations*: among the available spatial operations, three of them were chosen as basic operations in SpatialHadoop due to their popular use. These basic operations are range query, k-nearest neighbor (knn) and spatial join [9].
- *CG_Hadoop*: a suite of scalable and efficient MapReduce algorithms for various fundamental computational geometry problems, namely, *polygon union, skyline, convex hull, farthest pair, and closest pair* [12]. These operations take advantage of spatial indexes available in SpatialHadoop to achieve better performance than traditional Hadoop environments.
- *Spatial Data Mining*: operations developed using spatial data mining techniques.

C. MapReduce Layer

Similar to Hadoop, the MapReduce layer in SpatialHadoop is the query processing layer that runs MapReduce programs [9]. However, contrary to Hadoop where the input files are non-indexed heap files, SpatialHadoop supports spatially-indexed input files. In Hadoop, the input file goes through a FileSplitter that divides it into n splits, where n is set by the MapReduce program, based on the number of available slave nodes. Then, each split goes through a RecordReader that extracts records as key-value pairs that are passed to the map function.

SpatialHadoop enriches traditional Hadoop systems with two main components: i) *SpatialFileSplitter* - an extended splitter that exploits the global index in input files to perform early pruning of file blocks not contributing to the answer, and ii) *SpatialRecordReader* - which reads a split originating from spatially indexed input files and exploits the local indexes to process it efficiently.

D. Storage Layer

There are two challenges when using traditional spatial indexes in Hadoop. First, traditional indexes are designed for the procedural programming paradigm, while SpatialHadoop uses the MapReduce programming paradigm. Secondly, traditional indexes are designed for local file systems, while SpatialHadoop uses the Hadoop Distributed File System (HDFS), which is inherently limited as files can be written in an append-only manner, and once written, they cannot be modified [10].

To solve this limitation, SpatialHadoop creates two index layers - global and local. The global index is applicable on a cluster's master node, while local indexes organize data in each slave node. It is therefore possible for SpatialHadoop to support the following spatial index structures [9]:

- *Grid file:* a simple flat index that partitions the data according to a grid such that records overlapping each

grid cell are stored in one file block as a single partition. To simplify, we use a uniform grid assuming that data is uniformly distributed;

- *R-tree:* in this indexing technique records are not replicated, which causes partitions to overlap. This makes it more efficient for range queries where partitions that are completely contained in query range can be copied to output and no reduplication step is required;
- *R+-tree:* a variation of the R-tree where nodes at each level are kept disjoint, while records overlapping multiple nodes are replicated to each node to ensure efficient query answering. In this indexing technique, SpatialHadoop adjusts the size of each partition based on data distribution such that the contents of each partition ensure load balancing. Records in each partition are stored together as one HDFS block in one machine.

Eldawy et al. [13] developed four more indexing techniques for SpatialHadoop, namely, Z-curve, Hilbert curve, Quad tree, and K-d tree, but these techniques are not as widely used as the others are.

Before executing queries and operations, the dataset needs to be indexed and this task occurs in the partitioning phase. The indexing algorithm runs in three steps, where the first step is fixed and the last two steps are customized for each partitioning technique. The first step computes the number of desired partitions, *n*, based on file size and *HDFS block capacity*, both of which are fixed for all partitioning techniques. The second step reads a random sample, with a sampling ratio, from the input file and uses this sample to partition the space into n cells such that the number of sample points in each cell is at most ⌊k/n⌋, where k is the sample size. The third step actually partitions the file by assigning each record to one or more cells. Boundary objects are handled using either the distribution or replication methods. The distribution method assigns an object to exactly one overlapping cell and the cell has to be expanded to enclose all contained records. The replication method avoids expanding cells by replicating each record to all overlapping cells but the query processor has to employ a duplicate avoidance technique to account for replicated records.

III. SPATIAL CLOUD COMPUTING

Although computing hardware technologies, including a central processing unit (CPU), network, storage, RAM, and graphics processing unit (GPU), have been advanced greatly in past decades, many computing requirements for addressing scientific and application challenges, such as applications for big geospatial data processing, go beyond existing computing capabilities [4].

These challenges require the readiness of a computing infrastructure that can [20]: i) better support discovery, access and utilization of data and data processing so as to relieve scientists and engineers of IT tasks, allowing them to focus on scientific discoveries; ii) provide real-time IT resources to enable real-time applications, such as emergency response; iii) deal with access spikes; and iv) provide more reliable and scalable service for massive numbers of concurrent users to further public knowledge.

Cloud computing offers facilities to overcome the challenges of a big data environment, providing heightened computer power and vast storage. In the most used definition for cloud computing, NIST [4] indicates five essential characteristics, namely: on demand self-service, broad network access, resource pooling, rapid elasticity, and measured service.

However, other characteristics are relevant when providing a spatial cloud computing environment. Akdogan et al. [20] proposed a cost-efficient partitioning of spatial data in clouds. This partitioning method considers location-based services and optimizes the storage of spatial-temporal data by making it possible to turn-off idle servers, thereby reducing costs.

Yang et al. [20] defines Spatial Cloud Computing as the cloud computing paradigm that is driven by geospatial sciences, and optimized by spatiotemporal principles for enabling geospatial science discoveries and cloud computing within a distributed computing environment. The intention is to supply the computational needs for geospatial data intensity, computing intensity, concurrent access intensity and spatiotemporal intensity.

A. Public Cloud Providers

According to NIST [4], there are four deployment models for clouds, namely private, community, public and hybrid. Specifically to public clouds, [4] defines how the cloud infrastructure is provisioned for open use by the general public. In this model of cloud deployment, services are charged in a pay-per-use method at some level of abstraction appropriate to the type of service (e.g., storage, processing or bandwidth). When working with big geospatial data, the volume of data and the power of processing are always high and, consequently, expensive.

Amazon AWS is, according to the "Gartner Magic Quadrant for Cloud Infrastructure as a Service", the leading Public Cloud Provider [27]. It offers "Elastic Map Reduce" (EMR) that uses Hadoop fundamentals and is integrated with other services available from providers, such as storage, data mining, log file analysis, machine learning, scientific simulation, and data warehousing. Our tests were conducted in an Amazon AWS environment.

IV. SYSTEM ARCHITECTURE

To support the method proposed in this paper, an architecture composed of three layers, namely Web Interface, Storage and SpatialHadoop (Fig. 2), was put together.

The main characteristics of each layer are described below:

- *Web Interface Layer*: a user-friendly interface to receive inputs and show results. In this layer, the user selects an available dataset (or uploads one if it is new) and defines the following parameters for the application: queries and operations, indexing (Grid, R-Tree, R+-Tree) and stickiness.

 ISBN: 978-1-61208-539-5

- *Storage Layer*: this layer stores all datasets available to the application and saves the results after application execution.
- *SpatialHadoop Layer*: this is the core layer. It is responsible for provisioning the SpatialHadoop cluster with one master node and *n* data nodes. The quantity of data nodes is defined based on dataset size, as shown in Section V. After provisioning the cluster, this layer indexes the dataset (based on user choice in the Web Interface layer), processes queries and operations and saves the results file back in the Storage Layer.

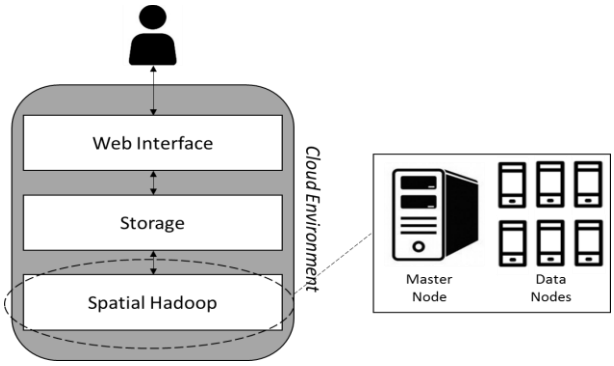

Figure 2. System Architecture Overview.

In the proposed method, all three layers were designed to use a public cloud environment. It is also possible to allocate the Web Interface Layer and the Storage Layer to different environments. However, it is important to consider that public cloud providers usually charge users for each gigabyte stored and transferred, and this can affect the total cost.

V. METHODOLOGY

A common uncertainty for Hadoop environment administrators is how to define the cluster size infrastructure. In a static environment, like a private cloud, most of the time the computational resources are limited and big geospatial data grows faster, requiring ever more resources. On the other hand, in public cloud providers the computational resources are unlimited, but users are charged for them, so it is very important to define a cost-effective environment.

A twenty-node cluster can be necessary to process SpatialHadoop queries and operations on a 100Gb dataset, but it is overprovisioned to work on a dataset of only 5Gb. To solve this problem, a formula to calculate the quantity of data nodes based on dataset size is fundamental. Adapting the proposal by [6] and [23], the following formula can be used to determine the ideal number of data nodes in a SpatialHadoop environment on public cloud providers:

$$DN = \left\lceil \frac{T}{d} \right\rceil \qquad (1)$$

DN represents the total data nodes needed; *T* is the total amount of data and *d* is the disk size in each node.

It is necessary to calculate *T* because the total amount of data used in a SpatialHadoop application is not only the volume of the dataset. To calculate *T*, the following formula can be used:

$$T = \frac{C\ x\ R\ x\ S}{(1-i) * (1+w)} \qquad (2)$$

C represents the compression rate of the dataset, required because SpatialHadoop can work with compressed files. When no compression is used, the value must be 1. *R* is the number of replicas of data in HDFS and *S* represents the size of the dataset. The notation *i* refers to the intermediate working space dedicated to temporarily storing results of Map Tasks. Finally, *w* represents the percentage of space left (wasted) to HDFS.

To demonstrate the use of these formulas, let us consider a real Open Street Map (OSM) dataset of 96Gb of total size (2.7 billion records) [28]. Without compression ($C = 1$), without replication ($R = 1$), considering $i = 25\%$ and $w = 20\%$, the value obtained for *T* is 106.67. Considering that each data node has a disk with 32Gb ($d = 32$) it is possible to conclude that the ideal number of data nodes (*DN*) is 4.

Changing any other parameter value can affect the number of data nodes in cluster. For example, using the same values for parameters *C*, *R*, *S* and *w* ($C = 1$, $R = 1$, $S = 96$, $w = 20\%$), and changing the value for *i* to 40% the number of data nodes (*DN*) grows to 5. This result will affect application performance and, also the total cost of environment.

VI. TESTS AND RESULTS

A SpatialHadoop environment was built using Amazon AWS Elastic MapReduce to test the proposed method. Although all three layers of the system architecture – Web Interface, Storage and SpatialHadoop – were allocated on a cloud provider, the focus of performance and costs used in this test scenario were specifically on the SpatialHadoop layer.

Table I presents the instances configurations used to run the tests on Amazon AWS.

TABLE I. INSTANCES CONFIGURATIONS ON AMAZON AWS.

Function	vCPUs	Memory	Disk (SSD)	Price (US$)
Master	8	15	160 Gb	0.42 / hour
Data	4	7.5	80 Gb	0.21 / hour

The datasets used were extracted from Open Street Map and Tiger and are available to download on the SpatialHadoop site [28]. The clusters created for the tests were composed of one master node and the quantity of data nodes based on the formula shown in Section V, considering the following values to the others parameters: $C = 1$, $R = 3$, $i = 25\%$ and $w = 20\%$. Details about datasets and number of data nodes are described in Table II.

 ISBN: 978-1-61208-539-5

TABLE II. DATASETS AND DATANODES.

Dataset	Size	Records	Data nodes
LinearWater	9.0 Gb	8.4 million	1
Roads	7.7 Gb	20 million	1
Buildings	26.0 Gb	115 million	2
Lakes	9.0 Gb	8.4 million	1

Once parameters were defined in the Web Interface Layer and the dataset was stored in the Storage Layer, the SpatialHadoop Layer was configured to execute the following steps:

- *Provisioning Cluster:* a defined request is sent to a cloud provider, with the number and type of master node, and data nodes.
- *Transfer Dataset:* copy an existing dataset from Storage Layer to data nodes.
- *Index Dataset:* apply the user-defined index type to dataset.
- *Queries and Operations:* executes the user-defined queries and operations.
- *Save Results:* saves the result file – usually a text file – on Storage Layer to be accessed by the user.
- *Turn-off Cluster:* to avoid wasting of computational resources and increasing financial costs, all the clusters (master node and data nodes) are turned off, unless some stickiness parameter was defined by the user.

Table III presents the runtime of each task in a test environment. The values present an average of 3 execution for the smallest (Roads) and the biggest (Buildngs) datasets. The queries – KNN and Range Query – and the indexing type *Grid* were chosen randomly, and could be changed by any query or operation and indexing type.

TABLE III. TIME MEASURED IN EACH TASK.

Task	Smallest Dataset (seconds)	Biggest Dataset (seconds)
Provisioning Cluster	300	420
Transfer Dataset	60	120
Index Dataset	600	3540
KNN	10	8
Range Query	8	6
Save Results	2	2
Turn-off Cluster	100	164
TOTAL Time	1080	4260

The indexing task is very important to ensure SpatialHadoop is high performing. Note that the majority of time is spent on the index process, but once it is finished, the queries are done very quickly. A comparassion of the runtime of the indexing task for the Buildings Dataset, using a cluster with 2 datanodes, is presented in Fig. 3.

Since the cluster to support the Smallest Dataset (1 master node and 1 data node) costs US$ 0.63/hour, the total cost to process these two queries was US$ 0.19. The cost of the cluster to support the large dataset (composed of 1 master node and 2 data nodes) is US$ 0.84/hr, so the cost of processing these queries was US$ 0,99.

If this cluster had been created without considering the dataset´s size – and other parameters defined in the formula – it would had been necessary to consider the largest dataset available to ensure that any query or operation could be executed in this cluster. Considering all datasets available to download on the SpatialHadoop webpage [28], the largest dataset – an OSM file with 137Gb of size and 717M records about road networks represented as individual road segments – would require a cluster composed of 1 master node and 6 data nodes. The total cost of this cluster would be US$ 1.68 per hour and running the small dataset (18 minutes) would cost US$ 0.50, costing 263% more than was really needed

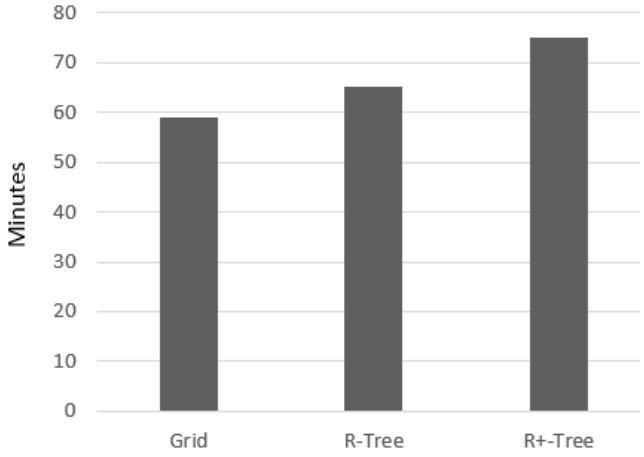

Figure 3. Index task runtime for Buildings Dataset.

Analyzing all datasets availabe on the SpatialHadoop webpage [28], and considering the scenario and parameters defined in our test environment (C = 1, R = 3, i = 25% and w = 20%), only 7 from a total of 33 datasets needed more than 1 data node to be executed. On the other extreme, only 1 dataset needed a 6-node cluster. Processing any other datasets wastes computational resources if the proposed formula is not applied.

TABLE IV. TOTAL COST OF CLUSTER.

Number of Data Nodes	Total cluster lifecycle time	Total Cost
1 (defined by formula)	20 minutes	US$ 0.21
2	19 minutes	US$ 0.27
4	17 minutes	US$ 0.36

Table IV presents the total cluster lifecycle time and cost to process the queries KNN, Spatial Join and Range Query using the Lakes dataset using the number of instances proposed by the formula (1), and also using 2 and 4 instances. However, even though the total time of execution is higher, the cost is lower. This occurs because there is a low reduction in the indexing task runtime (only 1 minute per core added).

 ISBN: 978-1-61208-539-5

VII. RELATED WORK

SpatialHadoop was presented in 2013 by Eldawy and Mokbel [14] as the first fully-fledged MapReduce framework with native support for spatial data. In this article, the authors used a demonstration scenario created on an Amazon AWS, with a 20 node cluster to compare SpatialHadoop and traditional Hadoop in three operations, namely, range query, knn and spatial join.

Also in 2013, Eldawy et al. [12] presented CG_Hadoop, which is a suite of scalable and efficient MapReduce algorithms for various fundamental computational geometry problems - polygon union, skyline, convex hull, farthest pair, and closest pair – comparing the performance of these computational geometry operations on traditional Hadoop and SpatialHadoop and concluded that SpatialHadoop algorithms significantly outperform Hadoop algorithms as they take advantage of the spatial indexing and components within SpatialHadoop.

In recent years, some articles have been published about improvements to SpatialHadoop. Mokbel et al. [15] proposed a web-based road-network, traffic generator, called, MNTG. Alarabi et al. [16] created TAREEG, a MapReduce-based web service that uses SpatialHadoop fundamentals for extracting spatial data from OpenStreetMap. Eldawy et al. [17] used SpatialHadoop to query and visualize spatio-temporal satellite data in an application called SHAHED. Eldawy et al. [18] created HadoopViz, a MapReduce framework for extensible visualization of Big Spatial Data. All of these studies were developed in static and dedicated clusters.

A modular software architecture for processing big geospatial data in the cloud was presented by [1]. Since the proposed framework is not affected by whether the cloud environment is private or public, a third-party tool – Ansible – was used to execute provisioning scripts.

Finally, in 2016, Das et al. [19] proposed a geospatial query resolution framework using an orchestration engine for clouds. However, the cloud environment used was private and no dynamic allocation of computational resources was performed.

None of these works present a method to optimize the use of computational resources and reduce financial costs on public cloud providers when using SpatialHadoop to process big geospatial data.

This paper presents a cost-efficient method to process geospatial data on public cloud providers, optimizing the number of data nodes in a SpatialHadoop cluster according to dataset size.

VIII. CONCLUSION AND FUTURE WORKS

SpatialHadoop is a MapReduce framework for big geospatial data that has high performance but requires a computational infrastructure that can be expensive. When working on public cloud providers, in which each computational resource is charged for, it is necessary to look for a cost-effective solution.

The method proposed in this paper achieves the goal of supporting a SpatialHadoop environment on public cloud providers, while avoiding the waste of computational resources. The formula to define the number of data nodes was validated in a test scenario, resulting in a cost savings of approximately 263%.

As future works we suggest optimizations on performance that can be obtained using task nodes – for job processing only - and data nodes together. In this way, it is possible to apply scalability in SpatialHadoop applications based on user-defined threads. Formulas to calculate other computational resources – CPU and memory – based on datasets and queries or operations can also be defined.

REFERENCES

[1] M. Krämer and I. Senner, "A modular software architecture for processing of big geospatial data in the cloud." In Computers & Graphics, pp. 69-81, 2015.

[2] S. Seref and S. Duygu, "Big data: A review." In International Conference on Collaboration Technologies and Systems (CTS), pp. 42-47, 2013.

[3] J. S. Ward and A. Barker, "Undefined by data: a survey of big data definitions." arXiv:1309.5821, 2013.

[4] P. Mell and T. Grance, “Draft NIST working definition of cloud computing,” [Online]. Available from: http://csrc.nist.gov/groups/SNS/cloud-computing/index.html [acessed: 2016-11-01]

[5] N. R. Herbst, S. Kounev, and R. Reussner, "Elasticity in cloud computing: What it is, and what it is not." Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13), pp. 23-27, 2013.

[6] Hadoop Online Tutorial. Formula to calculate NDFS nodes storage. [Online]. Avilable from: http://hadooptutorial.info/formula-to-calculate-hdfs-nodes-storage/ [acessed: 2016-11-03]

[7] J. Pramila, "Cloud Architecture for Big Data." International Journal of Engineering and Computer Science, pp. 12757 – 12765, June, 2015.

[8] E. S. A. Ahmed, and A. S. Rashid, "A Survey of Big Data Cloud Computing Security." International Journal of Computer Science and Software Engineering (IJCSSE), pp. 78-85, 2014.

[9] A. Eldawy and M. F. Mokbel, "Spatialhadoop: A mapreduce framework for spatial data." In 2015 IEEE 31st International Conference on Data Engineering, pp. 1352-1363, 2015.

[10] A. Eldawy, "SpatialHadoop: towards flexible and scalable spatial processing using mapreduce." Proceedings of the 2014 SIGMOD PhD symposium, pp. 46-50, 2014.

[11] A. Eldawy and M. F. Mokbel, "Pigeon: A spatial mapreduce language". In 2014 IEEE 30th International Conference on Data Engineering, pp. 1242-1245, 2014.

[12] A. Eldawy, Y. Li, M. F. Mokbel, and R. Janardan, “CG_Hadoop: computational geometry in MapReduce.” The 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 294-303, June, 2014.

[13] A. Eldawy, A. Alarabi, and M. F. Mokbel, "Spatial partitioning techniques in SpatialHadoop." Proceedings of the VLDB Endowment, pp. 1602-1605, 2015.

[14] A. Eldawy and M. F. Mokbel, "A demonstration of SpatialHadoop: an efficient mapreduce framework for spatial data." Proceedings of the VLDB Endowment, pp. 1230-1233, 2013.

[15] M. F. Mokbel et al., "A demonstration of MNTG-A web-based road network traffic generator," In 2014 IEEE 30th International Conference on Data Engineering, pp. 1246-1249, March, 2014.

 ISBN: 978-1-61208-539-5

[16] L. Alarabi, A. Eldawy, R. Alghamdi, and M. F. Mokbel, “TAREEG: a MapReduce-based web service for extracting spatial data from OpenStreetMap”. In 2014 ACM SIGMOD International Conference on Management of data, pp. 897-900, 2014.

[17] A. Eldawy et al., “Shahed: A mapreduce-based system for querying and visualizing spatio-temporal satellite data”. In 2015 IEEE 31st International Conference on Data Engineering, pp. 1585-1596, April, 2015.

[18] A. Eldawy, M. Mokbel, and C. Jonathan, "HadoopViz: A MapReduce framework for extensible visualization of big spatial data." In 2016 IEEE International. Conference on Data Engineering (ICDE), pp. 601-612, 2016.

[19] J. D. Das, A. Ghosh, and R. A. Buyya, “Geospatial Orchestration Framework on Cloud for Processing User Queries”. In 2016 IEEE International Conference on Cloud Computing for Emerging Markets, pp. 19-21, 2016.

[20] A. Akdogan, “Cost-efficient partitioning of spatial data on cloud”. In 2015 IEEE International Conference on Big Data, pp. 501-506, 2015

[21] C. Yang et al., “Spatial cloud computing: how can the geospatial sciences use and help shape cloud computing?” International Journal of Digital Earth, pp. 305-329, 2011.

[22] C. Qu, R. N. Calheiros, and R. Buyya, "Auto-scaling Web Applications in Clouds: A Taxonomy and Survey." arXiv:1609.09224. 2016.

[23] Distributed System Archicteture. Hadoop cluster size. [Online]. Available from: https://0x0fff.com/hadoop-cluster-sizing/ [acessed: 2016-10-26]

[24] Amazon AWS. [Online]. Available from: https://aws.amazon.com [acessed: 2016-11-04]

[25] Microsoft Azure. [Online] Available from: https://azure.microsoft.com [acessed: 2016-11-04]

[26] The Google Cloud Provider. [Online] Available from: https://cloud.google.com [acessed: 2016-11-04]

[27] Magic Quadrant for Cloud Infrastructure as a Service, Worldwide. [Online]. Available from: https://www.gartner.com/doc/ reprints?id=1-2G2O5FC&ct=150519. [acessed: 2016-11-02]

[28] SpatialHadoop Datasets. [Online]. Available from: http://spatialhadoop.cs.umn.edu/datasets.html [acessed : 2016-11-03]

 ISBN: 978-1-61208-539-5