International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2017 IJSRCSEIT | Volume 2 | Issue 6 | ISSN : 2456-3307

Geospatial Enormous Information Processing in an Open Source Circulated Computing Environment

[1]N. Deshai, [2]Dr. I. Hemalatha, [2]Dr. G. P. Saradhi Varma
[1]Assistant Professor, Department Information Technology, S.R.K.R Engineering College, Bhimavaram, Andhra Pradesh, India
[2]Professor, Department of Information Technology, S.R.K.R Engineering College, Bhimavaram, Andhra Pradesh, India

ABSTRACT

Significant progression of spatial information sharing organization, it introduces appeal to comfort and expansibility of supporting system. In perspective of tremendous scale versatile server gathering, disseminated processing passes on needs to settle the current troublesome issues in the space of geospatial information advantage. In this paper, we imported disseminated figuring development including MapReduce model and Hadoop arrange into the space of Geographic Data Framework (GIS). Those key advancement issues in the utilization of GIS; for instance, spatial data accumulating, spatial rundown and spatial operation were delineated and inspected in detail. We surveyed the execution and viability of spatial operation in Hadoop attempt condition with this present reality educational gathering. It displays the fittingness of circulated registering advancement in handling heightened spatial applications.

Keywords : Geographic Information System, Hadoop, Cloud Computing, Map Reduce, Geospatial Service

I. INTRODUCTION

GIS is as of now expecting a basic part in various districts of present day city. Space information has transformed into the basic structure of modem propelled city and is the crucial bit of information advancement [1]. Generally, GIS limits, for instance, spatial examination is related with an extensive measure of vector data (centers, lines or polygons) and raster data (satellite or lifted pictures). This sort of data is irregularly made through unprecedented sensors, satellites or GPS contraptions. In view of the broad size of spatial stores and the complicacy of the geospatial models, it take on upper time multifaceted nature and require high figuring resources. As the significant headway of spatial information sharing organization, unborn synthetical GIS arrange require indicate more copious information, respond a considerable measure of parallel requests and fulfill sharp information getting ready. This sort of information advantage going for Web isn't exactly the same as prevenient intelligent enlisting and trade getting ready. It introduces appeal to convenience and expansibility of supporting structure and traditional first class preparing and database systems can't meet those requesting.

Appropriated registering is another term for a long-held dream of handling as an utility, which has starting late created as a business reality [2]. Appropriated processing give advantage over the Internet to customers in light of sweeping versatile figuring resources. It is a standout amongst the latest examples in the standard IT industry. Conveyed processing can give a tolerable making condition with geographic information development. It passes on might want to settle the current troublesome issues in the space of geospatial information advantage. Conveyed figuring can supply a strong IT establishment to GIS feasibly. In perspective of broad 978-1-4244-8515-4/10/$26.00 ©20 10 IEEE 275 Sheng Wang Beijing Easymap Information Development Co., Ltd Beijing, China wangsheng@easymap.com.cn scale server gathering, conveyed processing can upgrade structure execution, figuring and limit capacity unimaginably, and reduce programming and hardware cost enough.

Through executing virtualization organization with physical resources, cloud stages have awesome structure adaptability and can supply mass geospatial enlisting needs. In this paper, we import circulated figuring development including Hadoop stage and MapReduce parallel enlisting model into the territory of geospatial information organizations. We have thought about those key development issues including spatial data accumulating, spatial data record and spatial operation in the utilization of GIS. Going for the characteristics of spatial chairmen, we have laid out the strategy stream of spatial data. We survey their execution using real spatial enlightening list in perspective of the genuine utilization of these spatial counts on Hadoop. The examination happens show that MapReduce is important for enrolling heightened spatial applications.

Related Work

Activated by the necessities to process expansive scale spatial information, there is an expanding late enthusiasm for utilizing Hadoop to Bolster spatial operations. Existing work can be delegated either particular to a specific spatial operation or a framework for a suite of spatial operations. Spatial Hadoop has a place with the framework classification, yet, with many recognized attributes as was talked about.

Specific Spatial Operations

Existing work in this category has mainly focused on addressing a specific spatial operation. The idea is to develop *map* and *reduce* functions for the required operation, which will be executed *on-top* of existing Hadoop cluster. Examples of such work include: (1) *R-tree construction* [17], where an R-tree is constructed in Hadoop by partitioning records according to their Z-values, building a separate R-tree for each partition, and combining those Rtrees under a common root. (2) *Range query* [18], [19], where the input file is scanned, and each record is compared against the query range. (3) *kNN query* [19], [20], where a brute force approach calculates the distance to each point and selects the closest k points [19], while another approach partitions points using a Voronoi diagram and finds the answer in partitions close to the query point [20]. (4) *All NN (ANN) query* [21], where points are partitioned according to their Z-values to find the answer similar to *k*NN queries. (5) *Reverse NN (RNN) query* [20], where input data is partitioned by a Voronoi diagram to exploit its properties to answer RNN queries. (6) *Spatial join* [19], where the partition-based spatial-merge join [22] is ported to MapReduce. The map function partitions the data using a grid while the reduce function joins data in each grid cell. (7) *kNN join* [23], [24], where the purpose is to find for each point in a set R, its *k*NN points from set S.

Three methodologies were proposed to construct frameworks for a suite of spatial operations: (1) Hadoop-GIS [10] broaden Hive [11], an information distribution center foundation based over Hadoop, to help spatial information investigation systems. It expands Hive with uniform lattice record which is utilized to accelerate extend inquiry and self join. However, Hadoop-GIS does not alter anything in the basic Hadoop framework, and thus it stays constrained by the impediments of existing Hadoop frameworks. Likewise, conventional MapReduce programs that entrance Hadoop specifically can't make any utilization of Hadoop-GIS, and consequently its materialness is restricted. (2)MD-HBase [8] broadens HBase [9], a non-social database keeps running over Hadoop, to help multidimensional files which takes into account productive recovery of focuses utilizing reach and kNN inquiries. MD-HBase has indistinguishable downsides from Hadoop-GIS, where the hidden Hadoop framework stays in place, and conventional MapReduce projects won't profit by it. (3) Parallel-Secondo [7] is a parallel spatial DBMS those utilizations Hadoop as a circulated undertaking scheduler, while capacity and inquiry handling are finished by spatial DBMS examples running on group hubs.

System Architecture

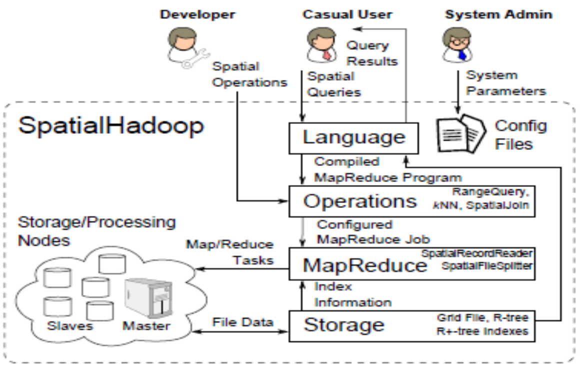

Figure 1: Spatial Hadoop system architecture

II. Implementation

Continuous Spatial Question Motor A central segment of Hadoop-GIS is its independent spatial inquiry motor. Porting a spatial database motor for such object isn't possible, because of its tight incorporation with RDBMS motor and multifaceted nature on setup and streamlining. We built up a Constant Spatial Inquiry Motor (RESQUE) to help spatial question handling. RESQUE exploits worldwide tile files and nearby on-request files to help productive spatial inquiries. What's more, RESQUE is completely improved, bolsters information pressure, and acquires low overhead on information stacking. In this way, RESQUE is an exceptionally proficient spatial inquiry motor contrasted with a customary SDBMS motor.

Typical workflow of spatial query processing on MapReduce

1. Data/space partitioning;
2. Data storage of partitioned data on HDFS;
3. Pre-query processing (optional);
4. For *tile* in *input collection* do Index building for objects in the tile; Tile based spatial querying processing;
5. Boundary object handling;
6. Post-query processing (optional);
7. Data aggregation;
8. Result storage on HDFS;

RESQUE is gathered as a mutual library which can be effectively sent in a bunch domain. Hadoop-GIS exploits spatial access techniques for inquiry preparing with two methodologies. At the larger amount, Hadoop-GIS makes worldwide area based spatial records of apportioned tiles for HDFS document split sifting. Thusly, for some spatial inquiries, for example, control questions, the framework can effectively channel most insignificant tiles through this worldwide area list. The worldwide area record is little and can be put away in HDFS and shared crosswise over group hubs through Hadoop dispersed store system. At the tile level, RESQUE backings an ordering on request approach by building tile construct spatial files in light of the fly, chiefly for inquiry handling reason, and putting away file documents in the primary memory. Since the tile measure is moderately little, file expanding on a solitary tile is quick and essentially enhances spatial inquiry handling execution. Our trials demonstrate that file building devours little portion of general question preparing expense, and it is unimportant for process and information serious inquiries, for example, cross-coordinating.

Spatial Data Partitioning

Geospatial information has a tendency to be vigorously skewed. For instance, if OpenStreetMap is apportioned into 1000 x 1000 settled size tiles, the quantity of items contained in the most skewed tile is about three requests of extent more than the one out of a normal tile. Such expansive skewed tiles could altogether build the reaction time in a parallel processing condition because of the straggling tiles. Along these lines powerful and productive spatial information apportioning is fundamental for adaptable spatial inquiries running in MapReduce. Spatial apportioning approaches produce limit protests that cross different segments, in this way abusing the segment freedom. Spatial question preparing calculations get around the limit issue by utilizing an imitate and-channel approach in which limit objects are reproduced to different spatial parcels, and reactions of such replication is cured by sifting the copies toward the finish of the inquiry handling stage. This procedure includes additional inquiry handling overhead relative to the quantity of limit objects. Thusly, a great spatial apportioning methodology ought to limit the quantity of limit objects.

Here create SATO a powerful and adaptable dividing structure which produces adjusted areas while limiting the quantity of limit objects. The dividing strategies are intended for adaptability, which can be effortlessly parallelized for superior. SATO remains for four primary strides in the apportioning pipeline: Test, Break down, Tear, and Enhance. Initial, a little portion of the dataset is inspected to recognize general worldwide information conveyance with potential thick districts. Next, the inspected information is broke down to deliver a coarse parcel conspire in which each segment area is required to contain generally rise to measures of spatial items. At that point these coarse parcel locales are passed to the dividing segment that tears the areas into more granular allotments fulfilling the segment necessities. At long last, the created parcels are broke down to deliver multi-level segment lists and extra segment measurements which can be utilized for advancing spatial inquiries. SATO coordinates various dividing calculations that can deal with assorted datasets, and each of the calculation has its own benefits. SATO likewise gives MapReduce based execution of the spatial apportioning techniques through two option approaches: top-down approach with locale level parallelization, and base up approach with question level parallelization.

MapReduce Based Spatial Query Processing

RESQUE gives the center question motor to help spatial inquiries, which empowers us to build up an expansive scale spatial inquiry handling system in light of MapReduce. Our approach depends on spatial information parceling, tile based spatial inquiry handling with MapReduce, and result standardization for tile limit objects.

Spatial Join with MapReduce

Spatial join is among the most as often as possible utilized and exorbitant questions in numerous spatial applications. Next, we talk about how to outline join inquiries into the MapReduce registering model. We initially demonstrate a case spatial join inquiry for spatial cross-coordinating in SQL, This question discovers all converging polygon matches between two arrangements of articles created from a picture by two distinct calculations, and figures the cover proportions (convergence to-union proportions) and centroid separations of the sets. The table markup polygon speaks to the limit as polygon, calculation UID as algorithm uid. The SQL punctuation accompanies spatial augmentations, for example, spatial relationship administrator ST Crosses, spatial protest administrators ST Convergence and ST UNION, and spatial estimation capacities ST CENTROID, ST Separation, and ST Territory. For straightforwardness, we initially show how to process the spatial join above with MapReduce disregarding limit objects, at that point we come back to talk about limit taking care of. Information datasets are divided into tiles amid the information stacking stage, and each record is allotted a remarkable segment id. The spatial join question is actualized as a MapReduce inquiry administrator prepared in following three stages: I) Guide step: the information datasets are checked for Guide administrator, and every mapper, in the wake of applying client characterized capacity or channel operation, transmits the records.

```
1: SELECT
2:
ST_AREA(ST_INTERSECTION(ta.polygon,tb.polygo
n))/
3: ST_AREA(ST_UNION(ta.polygon,tb.polygon)) AS
ratio,
4: ST_DISTANCE(ST_CENTROID(tb.polygon),
5: ST_CENTROID(ta.polygon)) AS distance,
6: FROM markup_polygon ta JOIN markup_polygon
tb ON
8: ST_INTERSECTS(ta.polygon, tb.polygon) = TRUE
9: WHERE ta.algrithm_uid='A1' AND
tb.algrithm_uid='A2' ;
```

Boundary Handling

In segment based spatial question handling, some spatial items may lie on parcel limits. As the parcel estimate gets littler, the level of limit objects increments. When all is said in done, the division of limit objects is contrarily corresponding to the extent of the segment. Limit objects represent the test that they have a place consistently with numerous disjoint parcels and would produce copy comes about. Hadoop-GIS cures the limit issue in a basic however compelling way. On the off chance that an inquiry requires to return finish question result, Hadoop-GIS creates an inquiry arrange for which contains a pre-handling assignment and a post-preparing undertaking. In the pre-handling undertaking, the limit objects are copied and doled out to various converging parcels (numerous task). At the point when each segment is prepared freely amid question execution, the outcomes are not yet remedy because of the copies. In the post-handling step, comes about because of various segments will be standardized, e.g., to dispose of copy records by checking the question uids, which are inside allocated and all around exceptional. In the post-handling step, items will experience a separating procedure that dispenses with copy records. Naturally, such approach would acquire additional inquiry preparing cost because of the replication and copy disposal steps. Be that as it may, this extra cost is little and immaterial contrasted with the general inquiry handling time.

III. Experiments Results

This section provides an extensive experimental study for the performance of Spatial Hadoop compared to standard Hadoop. We decided to compare with standard Hadoop and not other parallel spatial DBMSs for two reasons. First, as our contributions are all about spatial data support in Hadoop, the experiments are designed to show the effect of these additions or the overhead imposed by the new features compared to a traditional Hadoop. Second, the different architectures of spatial DBMSs have great influence on their respective performance, which are out of the scope of this paper. Interested readers can refer to a previous study [38] which has been established to compare different large scale data analysis architectures. Meanwhile, we could not compare with MD-HBase [8] or Hadoop-GIS [10] as they support much limited functionality than what we have in Spatial Hadoop. Also, they rely on the existence of HBase or Hive layers, respectively, which we do not currently have in Spatial Hadoop. Spatial Hadoop (source code is available at: [12]) is implemented inside Hadoop 1.2.1 on Java 1.6. All experiments are conducted on an Amazon EC2 [39] cluster of up to 100 nodes. The default cluster size is 20 nodes of 'small' instances.

Datasets

Here utilized the accompanying genuine and manufactured datasets to test different execution viewpoints for SpatialHadoop: (1) TIGER: A genuine dataset which speaks to spatial highlights in the US, for example, lanes and waterways [40]. It contains 70M line fragments with an aggregate size of 60 GB. (2) OSM: A genuine dataset separated from OpenStreetMap [35] which speaks to outline from the entire world. It contains 164M polygons with an aggregate size of 60 GB. (3) NASA: Remote detecting information which speaks to vegetation lists for the entire world more than 14 years. It contains 120 Billion focuses with an aggregate size of 4.6 TB. (4) SYNTH: A manufactured dataset produced in a region of 1M × 1M units, where each record is a rectangle of most extreme size d × d; d is set to 100 naturally. The area and size of each record are both produced in view of a uniform distribution. I create up to 2 Billion rectangles of aggregate size 128 GB. To enable scientists to rehash the tests, we make the initial two datasets accessible on SpatialHadoop site.

The third dataset is as of now made accessible by NASA [5]. The generator is dispatched as a component of SpatialHadoop and can be utilized as portrayed in its documentation. In our analyses, we look at the execution of the range inquiry, kNN, and circulated join calculations in Spatial-Hadoop proposed in Area VII to their customary usage in Hadoop [19], [37]. For run inquiry and kNN, we utilize framework throughput as the execution metric, which shows the quantity of MapReduce employments completed every moment. To figure the throughput, a bunch of 20 questions is submitted to the framework to guarantee full use and the throughput is ascertained by separating 20 over the aggregate time to answer every one of the inquiries. For spatial go along with, we utilize the preparing time of one question as the execution metric as one inquiry is generally enough to keep all machines occupied. The test comes about forgo inquiries, kNN questions, and spatial join are accounted for in Areas VIII-A, VIII-B, and VIII-C, separately, while Segment VIII-D ponders the execution of record creation.

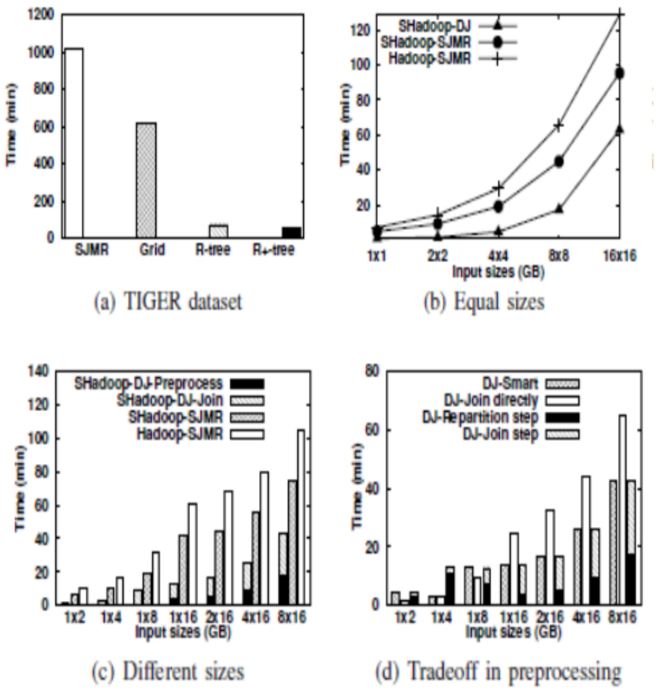

Figure 2 : Performance of spatial join algorithms

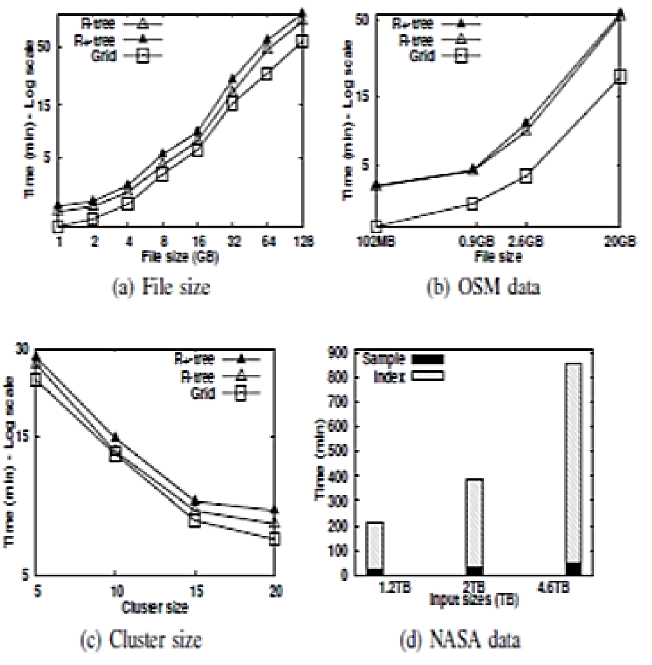

Figure 3 : Index Creation

IV. Performance Evaluation

In the accompanying examinations, we concentrate on a miniaturized scale seat check contrasting the execution times for single administrators with various settings. The benchmarks are executed on our group of 16 hubs with an Intel Center i5 processor and 16 GB Slam on every hub. The hubs are interconnected with a 1 Gbit/s organize. The Start employments are executed with 32 agents and 2 centers for every agent. The information generator, test programs, settings, and more trials and results are accessible on GitHub2 . To run our tests, we initially needed to _x issues in GeoSpark3. The most vital issue was that operations that utilization a list for questioning restored the applicant set returned by the record (R-tree). We added the hopeful pruning advance to acquire the right outcomes. Additionally needed to utilize variant 0.3 as the more current rendition 0.3.2 smashed without-of-memory mistakes for similar settings contained By (the operands were swapped). The principal explore executes a spatial channel administrator over a 50,000,000 polygon informational index (880 GB, uniform dissemination) with a contains predicate to discover those polygons that contain a given inquiry point. We utilized all accessible spatial specialists of every structure and executed the operation without ordering and in addition with live (on-the-y) file creation, if conceivable.

In this examination, STARK performed best with just 47 (BSP, live record) or 52 seconds (BSP, no list). Spatial Start is extremely constrained in its ease of use as a spatial apportioning isn't permitted in blend with a channel. The keep running without spatial apportioning and ordering took 3866 seconds (over 60 minutes). GeoSpark required 1237 seconds (20 minutes) without apportioning however was not ready to process this informational index at all with a spatial applicant and slammed following a few hours for every specialist. While examining this issue we executed the program on a littler polygon informational collection with 1,000,000 passages (17,6 GB). In the best case, it brought 54 seconds with Hilbert apportioning. That is a similar time that STARK expected to process 880 GB. For Spatial Hadoop (as an agent of the Hadoop based frameworks) we utilized their order line program, however were not ready to get a right outcome: The program completed following 39 minutes with zero outcomes. The issue is that a point question choice isn't accessible thus we gave a point as inquiry extend. A perception of the execution times alongside a more nitty gritty examination that incorporates the overhead of the apportioning can be found in our GitHub repository2.

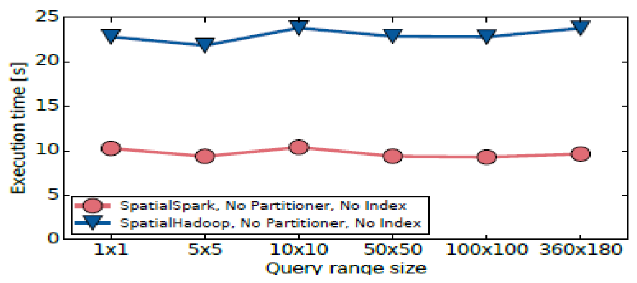

Figure 4 : Exec. times for different range query sizes for SpatialHadoop & SpatialSpark

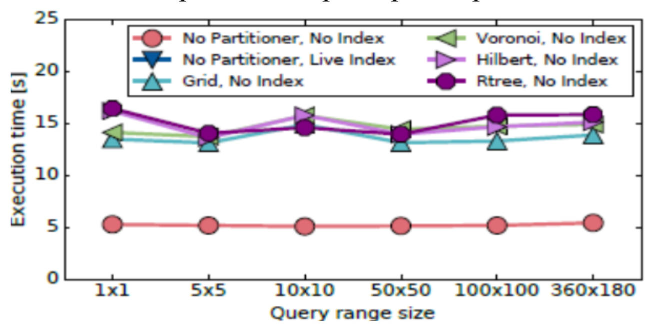

Figure 5 : Exec. times for di_erent range query sizes for GeoSpark. No Partitioner, Live Index is out of range (40 sec) for all queries.

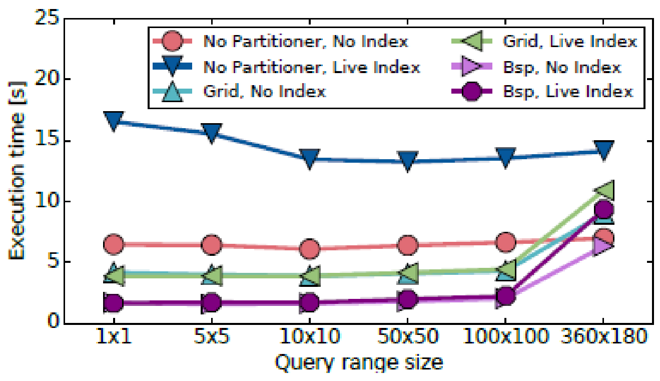

Figure 6 : Exec. times for di_erent range query sizes for STARK.

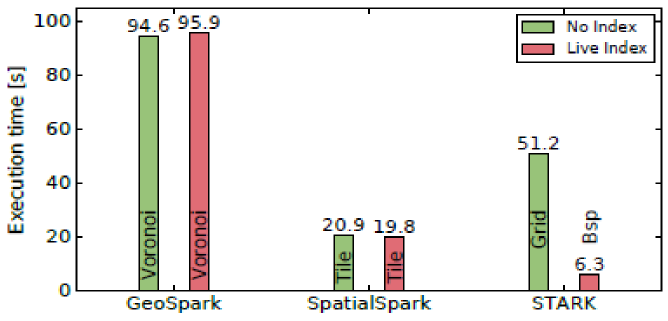

Figure 7 : Exec. times for spatial join queries.

V. CONCLUSION

This paper presents SpatialHadoop, an undeniable MapReduce system with local help for spatial information accessible as free open-source. SpatialHadoop is an exhaustive augmentation to Hadoop that infuses spatial information mindfulness in each Hadoop layer, in particular, the dialect, stockpiling, MapReduce, and operations layers. In the dialect layer, SpatialHadoop includes a straightforward and expressive abnormal state dialect with worked in help for spatial information sorts and operations. In the capacity layer, SpatialHadoop adjusts conventional spatial record structures, Network, R-tree, and R+-tree, to frame a two-level spatial list for MapReduce situations. In the MapReduce layer, SpatialHadoop advances Hadoop with two new parts, SpatialFileSplitter and SpatialRecordReader, for productive and adaptable spatial information handling. In the operations layer, SpatialHadoop is now furnished with three essential spatial operations, go question, kNN, and spatial join, as contextual investigations for actualizing spatial operations. Other spatial operations can likewise be included after a comparable approach. Broad analyses, in light of a genuine framework model and vast scale genuine datasets of up to 4.6TB, demonstrate that SpatialHadoop accomplishes requests of greatness higher throughput than Hadoop for range and k-closest neighbor questions and triple execution for spatial joins.

VI. REFERENCES

[1]. A. Ghoting, "et. al. SystemML: Declarative Machine Learning on MapReduce," in ICDE, 2011.

[2]. http://giraph.apache.org/.

[3]. G. Wang, M. Salles, B. Sowell, X. Wang, T. Cao, A. Demers, J. Gehrke, and W. White, "Behavioral Simulations in MapReduce," PVLDB, 2010.

[4]. J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Communications of ACM, vol. 51, 2008.

[5]. http://aws.amazon.com/blogs/aws/process-earth-science-data-on-aws-with-nasa-nex/.

[6]. http://esri.github.io/gis-tools-for-hadoop/.

[7]. J. Lu and R. H. Guting, "Parallel Secondo: Boosting Database Engines with Hadoop," in ICPADS, 2012.

[8]. S. Nishimura, S. Das, D. Agrawal, and A. El Abbadi, "MD-HBase: Design and Implementation of an Elastic Data Infrastructure for Cloudscale Location Services," DAPD, vol. 31, no. 2, pp. 289–319, 2013.

[9]. "HBase," 2012, http://hbase.apache.org/.

[10]. A. Aji, F. Wang, H. Vo, R. Lee, Q. Liu, X. Zhang, and J. Saltz, "Hadoop-GIS: A High Performance Spatial Data Warehousing System over MapReduce," in VLDB, 2013.

[11]. A. Thusoo, "et. al. Hive: A Warehousing Solution over a Map-Reduce Framework," PVLDB, 2009.

[12]. http://spatialhadoop.cs.umn.edu/.

[13]. A. Eldawy, Y. Li, M. F. Mokbel, and R. Janardan, "CG Hadoop: Computational Geometry in MapReduce," in SIGSPATIAL, 2013.

[14]. C. Olston, "et. al. Pig Latin: A Not-so-foreign Language for Data Processing," in SIGMOD, 2008.

[15]. A. Eldawy and M. F. Mokbel, "Pigeon: A Spatial MapReduce Language," in ICDE, 2014.

[16]. http://www.opengeospatial.org/.

[17]. A. Cary, Z. Sun, V. Hristidis, and N. Rishe, "Experiences on Processing Spatial Data with MapReduce," in SSDBM, 2009.

[18]. Q. Ma, B. Yang, W. Qian, and A. Zhou, "Query Processing of Massive Trajectory Data Based on MapReduce," in CLOUDDB, 2009.

[19]. S. Zhang, J. Han, Z. Liu, K. Wang, and S. Feng, "Spatial Queries Evaluation with MapReduce," in GCC, 2009.

[20]. A. Akdogan, U. Demiryurek, F. Banaei-Kashani, and C. Shahabi, "Voronoi-based Geospatial Query Processing with MapReduce," in CLOUDCOM, 2010.

[21]. K. Wang, "et. al. Accelerating Spatial Data Processing with MapReduce," in ICPADS, 2010.

[22]. J. Patel and D. DeWitt, "Partition Based Spatial-Merge Join," in SIG-MOD, 1996.

[23]. W. Lu, Y. Shen, S. Chen, and B. C. Ooi, "Efficient Processing of k Nearest Neighbor Joins using MapReduce," PVLDB, 2012.

[24]. C. Zhang, F. Li, and J. Jestes, "Efficient Parallel kNN Joins for Large Data in MapReduce," in EDBT, 2012.

[25]. J. Zhou, "et. al. SCOPE: Parallel Databases Meet MapReduce," PVLDB, 2012.

[26]. R. Lee, T. Luo, Y. Huai, F. Wang, Y. He, and X. Zhang, "Ysmart: Yet another sql-to-mapreduce translator," in ICDCS, 2011.

[27]. I. Kamel and C. Faloutsos, "Parallel R-trees," in SIGMOD, 1992.

[28]. S. Leutenegger and D. Nicol, "Efficient Bulk-Loading of Gridfiles," TKDE, vol. 9, no. 3, 1997.

[29]. I. Kamel and C. Faloutsos, "Hilbert R-tree: An Improved R-tree using Fractals," in VLDB, 1994.

[30]. S. Leutenegger, M. Lopez, and J. Edgington, "STR: A Simple and Efficient Algorithm for R-Tree Packing," in ICDE, 1997.

[31]. H. Liao, J. Han, and J. Fang, "Multi-dimensional Index on Hadoop Distributed File System," ICNAS, vol. 0, 2010.

[32]. J. Nievergelt, H. Hinterberger, and K. Sevcik, "The Grid File: An Adaptable, Symmetric Multikey File Structure," TODS, vol. 9, no. 1, 1984.

[33]. A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," in SIGMOD, 1984.

[34]. T. K. Sellis, N. Roussopoulos, and C. Faloutsos, "The R+-Tree: A Dynamic Index for Multi-Dimensional Objects," in VLDB, 1987.

[35]. http://www.openstreetmap.org/.

[36]. J.-P. Dittrich and B. Seeger, "Data Redundancy and Duplicate Detection in Spatial Join Processing," in ICDE, 2000.

[37]. S. Zhang, J. Han, Z. Liu, K. Wang, and Z. Xu, "SJMR: Parallelizing spatial join with MapReduce on clusters," in CLUSTER, 2009.