ST-Hadoop: A MapReduce Framework for Big Spatio-temporal Data

Louai Alarabi
PhD Student, University of Minnesota, Minneapolis, MN
louai@cs.umn.edu

1. PROBLEM AND MOTIVATION

The importance of processing spatiotemporal data is growing with the emerging and popularity of applications that create them in large-scale datasets. For example, space-telescopes weekly collect around 150 GB [15]. NASA spacecraft projects collect over 4 TB of data every day [21,22]. Medical magnetic resonance imaging (MRI) produce images at a rate of 50 PB per year [7]. New York City Taxi and Limousine Commission archive over 1.1 Billion trajectories [24]. Twitter has more than 500+ Million tweets every day, and 80% of its users are mobile [12, 29]. Facebook collect 3.2+ Billion interactions, and its active mobile users represent 67% [8]. A common characteristics of all collected data is that it has space and time information.

Domain experts who need to process spatiotemporal data can either use: (a) *General purpose* distributed frameworks such as Hadoop [13] or Spark [26], or (b) *Big spatial data systems* such as ESRI tools on Hadoop [6, 30], Parallel-Secondo [19], $\mathcal{MD}$-HBase [23], Hadoop-GIS [2], GeoTrellis [16], GeoSpark [31], and SpatialHadoop [4]. The former has been acceptable for typical analysis tasks as they organizes data as non-indexed heap files. However, using these systems as is will result in sub-performance for spatiotemporal applications that need indexing. The latter reveal their inefficiency for supporting time-varying of spatial objects because their indexes are mainly geared toward processing spatial queries. Executing spatiotemporal queries on either general purpose frameworks or spatial data systems will require scanning through irrelevant data that will result in bad performance.

In this paper, we present ST-Hadoop [27] as a novel system that acknowledges the fact that space and time play a crucial role in query processing. ST-Hadoop is an extension of a Hadoop framework that injects the spatiotemporal awareness in the code base of four layers inside SpatialHadoop, namely, language, indexing, MapReduce, and operations layers. The key point behind the performance gain of ST-Hadoop is the idea of indexing, where data are temporally loaded and divided across computation nodes.

2. BACKGROUND AND RELATED WORK

MapReduce frameworks, e.g., Hadoop [13], have been used extensively in different applications that include terabyte sorting [1], machine learning [11], and graph processing [10, 18, 25]. Current efforts to process spatiotemporal big data on MapReduce environment are either run on-top of a general purpose framework [3,9,14,17,20,28,30], or as an ad-hoc operation using spatial data systems [5, 6]. The spatial data systems are not well designed to efficiently process spatiotemporal queries, mainly because their indexes are equipped to only support spatial queries. All on-top approaches result in poor performance as the framework internals are unaware of the temporal locality of the data.

This work describes ST-Hadoop; a full-fledged MapReduce framework with native support for big spatiotemporal data. ST-Hadoop is a comprehensive extension to Hadoop that injects spatiotemporal data awareness inside SpatialHadoop layers.

SIGMOD'17 Student Research Competition May 14-19 2017, Chicago, IL, USA

ACM ISBN 978-1-4503-4199-8/17/05.
DOI: http://dx.doi.org/10.1145/3055167.3055181

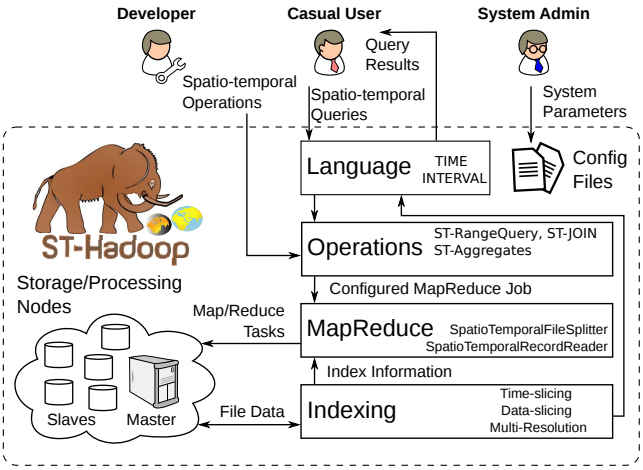

Figure 1: ST-Hadoop system architecture

3. APPROACH AND UNIQUENESS

Figure 1 gives the high level architecture of our ST-Hadoop; with a *built-in* support for spatiotemporal data types and operations. ST-Hadoop cluster contains one master node that breaks a map-reduce job into smaller tasks, carried out by slave nodes. Three types of users interact with ST-Hadoop: (1) *Casual users* who access ST-Hadoop through its language to process their datasets. (2) *Developers*, who have a deeper understanding of the system and can implement new operations, and (3) *Administrators*, who can tune up the system through adjusting system parameters in the configuration files provided with the ST-Hadoop installation. ST-Hadoop adopts a layered design of four main layers, namely, *language*, *Indexing*, *MapReduce*, and *operations* layers, described briefly below:

Language Layer: This layer provides a simple high-level SQL-like language that supports spatiotemporal data types (i.e., TIME and INTERVAL) and operations (e.g., OVERLAP, and JOIN).

Indexing Layer: ST-Hadoop employs a two levels index structure

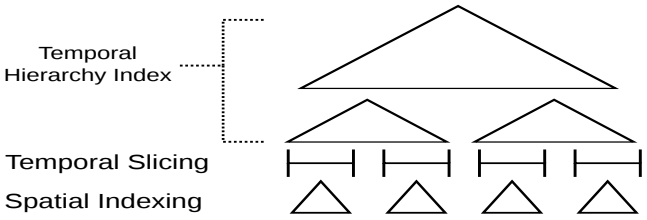

Figure 2: Indexing in ST-Hadoop

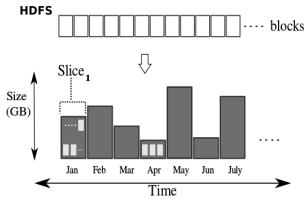

Figure 3: Time-Slice

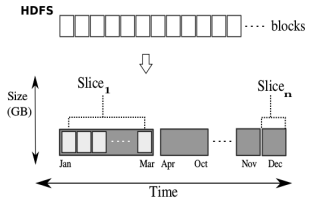

Figure 4: Data-Slice

of *global* and *local* indexing. The *global* index partitions the data across the computation nodes, while the *local* index organizes the data inside each node. Space and time are taken into consideration inside the two levels index structures.

MapReduce Layer: ST-Hadoop adds two new implementations to MapReduce layer, namely, *SpatioTemporalFileSplitter*, and *SpatioTemporalRecordReader*, which they exploit the global and local indexes, respectively, for data pruning.

Operations Layer: This layer encapsulates the implementation of two common spatiotemporal operations, namely, spatiotemporal range query, and spatiotemporal join queries. More operations can be added to this layer by ST-Hadoop *developers*.

4. INDEXING

Figure 2 Illustrates the abstract idea of indexing in ST-Hadoop, which can be described in three consecutive phases:

(1) **Temporal slicing**. In this phase, we slice the input file into multiple intervals, to efficiently support a fast random access to a sequence of objects bounded by the same time interval. ST-Hadoop employs two slicing techniques:

- *Time-Based.* This slicing technique slices the data loaded into the HDFS into multiple splits, that are uniformly on their time intervals. Figure 3 shows the general idea of this type of slicing, where ST-Hadoop splits a year of data into the interval of one month. While the time interval of the slices is fixed, the size of slices varies according to the time distribution of the data.

- *Data-Based.* In this technique the data sliced to the degree that all sub-splits are uniformly in their data size. All slices hold the same number of data blocks, while their time intervals are disjointed. Figure 4 depicts the key concept such that a $\texttt{slice}_1$ and $\texttt{slice}_n$ are equally in size, but they differs in their interval coverage.

(2) **Spatial Indexing**. This phase has two main tasks: (a) Determining the spatial boundries of each temporal slice, and (b) Physically partitioning data within a slice spatially across computation nodes. ST-Hadoop takes the advantages of applying different types of spatial partitioning techniques already equipped in SpatialHadoop such as Grid, R-tree-like, Quad-tree, and Kd-tree.

(3) **Temporal Hierarchy Indexing**. For an efficient retrieval of spatiotemporal objects, ST-Hadoop introduces a temporal hierarchy for spatial indexes. This index is organized into multiple levels; each corresponds to specific *time resolution*. In each level, the whole dataset is replicated and spatiotemporally partitioned according to the resolution of that level. ST-Hadoop sacrifices storage to achieve more efficient performance in supporting both spatiotemporal time point and interval queries. In fact, the temporal hierarchy of indexing in ST-Hadoop is the key point in its superior performance over Hadoop or SpatialHadoop.

5. INDEX UPDATE AND MAINTENANCE

This module maintains the *Temporal Hierarchy Index* on a regular basis, such as every day, to reflect newly received data on the index. A single map-reduce job is responsible for appending the incoming data as following. First, it constructs a new indexed temporal-slice in the highest resolution layer inside the *temporal hierarchy index*. Then, we check if the newly created slice would contribute to the creation of a lower resolution level. If this is not the case, then it will be carried out for a next maintenance call. During the maintenance, if any indexed temporal-slices contribute to the creation of a lower layer, then the data of these slices will be merged, and a new bigger temporal-slice will be created in the lower resolution.

6. RESULTS AND CONTRIBUTIONS

The indexing in ST-Hadoop is the key point in its superior performance over Hadoop or SpatialHadoop. Extensive experiments conducted on 10 TB of Twitter dataset [29], to show that ST-Hadoop achieves orders of magnitude higher job throughput. In this paper, we present two case studies of operations that utilize the ST-Hadoop indexing, namely, *spatiotemporal range query* and *join* queries.

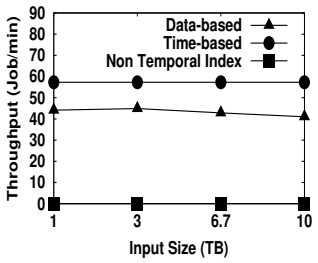

Figure 5: Range query

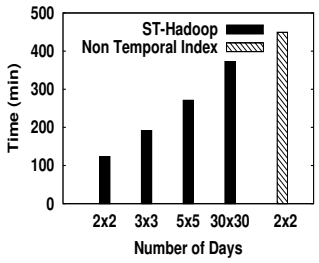

Figure 6: join

In figure 5, we increase the size of input file from 1 TB to 10 TB, while measuring the throughput of SpatialHadoop and ST-Hadoop. The throughput performance metric indicates the number of map-reduce jobs finished per minute, which was calculated on a batch of 20 randomly submitted queries that are selected with a spatial area ratio of 0.001% and a temporal window of 24 hours. For all file sizes, ST-Hadoop index has more than two orders of magnitude higher throughput consistently, due to the early pruning employed by the spatiotemporal index that results in hitting exact partitions within the given query parameters. Meanwhile, SpatialHadoop needs to scan all partitions that overlap with the spatial area, its throughput decreases with the increasing size of the input file.

Figure 6 gives the result of spatiotemporal join experiments, where we compare our *distributed* join algorithm for ST-Hadoop with spatial join implementation [4]. As shown in the figure, the x-axis represent the join query on a number of days$\times$days in ascending order.ST-Hadoop significantly decreases the processing time, as only partitions that overlap with both space and time property are considered in the distributed join algorithm. Meanwhile, SpatialHadoop distributed join considers all spatially overlapping partitions, which result in very poor performance for joining spatiotemporal data. In all cases, *distributed* join in ST-Hadoop significantly outperforms SpatialHadoop algorithm with triple performance.

7. REFERENCES

[1] Sort benchmark, July 2016. http://sortbenchmark.org.

[2] 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.

[3] K. M. Al-Naami, S. E. Seker, and L. Khan. GISQF: An Efficient Spatial Query Processing System. In *CLOUDCOM*, 2014.

[4] A. Eldawy and M. F. Mokbel. SpatialHadoop: A MapReduce Framework for Spatial Data. In *ICDE*, 2015.

[5] A. Eldawy, M. F. Mokbel, S. Alharthi, A. Alzaidy, K. Tarek, and S. Ghani. SHAHED: A MapReduce-based System for Querying and Visualizing Spatio-temporal Satellite Data. In *ICDE*, 2015.

[6] GIS Tools for Hadoop, Feb. 2014. http://esri.github.io/gis-tools-for-hadoop/.

[7] European XFEL: The Data Challenge, Sept. 2012. http://www.eiroforum.org/activities/scientific_highlights/201209_XFEL/index.html.

[8] The Top 20 Valuable Facebook Statistics, 2016. https://zephoria.com/top-15-valuable-facebook-statistics/.

[9] S. Fries, B. Boden, G. Stepien, and T. Seidl. Phidj: Parallel similarity self-join for high-dimensional vector data with mapreduce. In *ICDE*, 2014.

[10] J. Gao, J. Zhou, C. Zhou, and J. X. Yu. Glog: A high level graph analysis system using mapreduce. In *ICDE*, 2014.

[11] A. Ghoting, R. Krishnamurthy, E. Pednault, B. Reinwald, V. Sindhwani, S. Tatikonda, Y. Tian, and S. Vaithyanathan. SystemML: Declarative Machine Learning on MapReduce. In *ICDE*, 2011.

[12] 2015. https://blog.gnip.com/tag/geotagged-tweets/.

[13] Apache. Hadoop. http://hadoop.apache.org/.

[14] W. Han, J. Kim, B. S. Lee, Y. Tao, R. Rantzau, and V. Markl. Cost-based predictive spatiotemporal join. 2009.

[15] Telescope Hubbel site:, 2015. http://hubblesite.org/the_telescope/hubble_essentials/quick_facts.php.

[16] A. Kini and R. Emanuele. Geotrellis: Adding Geospatial Capabilities to Spark, 2014. http://spark-summit.org/2014/talk/geotrellis-adding-geospatial-capabilities-to-spark.

[17] Z. Li, F. Hu, J. L. Schnase, D. Q. Duffy, T. Lee, M. K. Bowen, and C. Yang. A spatiotemporal indexing approach for efficient processing of big array-based climate data with mapreduce. *IJGIS*, 2016.

[18] W. Lin, X. Xiao, and G. Ghinita. Large-scale frequent subgraph mining in mapreduce. In *ICDE*, 2014.

[19] J. Lu and R. H. Guting. Parallel Secondo: Boosting Database Engines with Hadoop. In *ICPADS*, 2012.

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

[21] Land Process Distributed Active Archive Center, Mar. 2015. https://lpdaac.usgs.gov/about.

[22] Data from NASA's Missions, Research, and Activities, 2016. http://www.nasa.gov/open/data.html.

[23] S. Nishimura, S. Das, D. Agrawal, and A. El Abbadi. $\mathcal{MD}$-HBase: Design and Implementation of an Elastic Data Infrastructure for Cloud-scale Location Services. *DAPD*, 2013.

[24] NYC Taxi and Limousine Commission, 2016. http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml.

[25] L. Qin, J. X. Yu, L. Chang, H. Cheng, C. Zhang, and X. Lin. Scalable big graph processing in mapreduce. In *SIGMOD*, 2014.

[26] Apache. Spark. http://spark.apache.org/.

[27] ST-Hadoop website. http://st-hadoop.cs.umn.edu/.

[28] H. Tan, W. Luo, and L. M. Ni. Clost: a hadoop-based storage system for big spatio-temporal data analytics. In *CIKM*, 2012.

[29] 2016. https://about.twitter.com/company.

[30] R. T. Whitman, M. B. Park, S. A. Ambrose, and E. G. Hoel. Spatial Indexing and Analytics on Hadoop. In *SIGSPATIAL*, 2014.

[31] J. Yu, J. Wu, and M. Sarwat. GeoSpark: A Cluster Computing Framework for Processing Large-Scale Spatial Data. In *SIGSPATIAL*, 2015.