A Demonstration of Shahed: A MapReduce-based System for Querying and Visualizing Satellite Data

Ahmed Eldawy[1#], Saif Alharthi[2§], Abdulhadi Alzaidy[3§], Anas Daghistani[4§]
Sohaib Ghani[5§], Saleh Basalamah[6§], Mohamed F. Mokbel[7#]

[#]*Deptartment of Computer Science and Engineering, University of Minnesota, Minneapolis, MN*
[§]*KACST GIS Technology Innovation Center, Umm Al-Qura University, Makkah, KSA*
{[1]eldawy,[7]mokbel}@cs.umn.edu, {[2]sharthi,[3]azaidy,[4]adaghistani,[5]sghani,[6]sbasalamah}@gistic.org

***Abstract*—Several space agencies such as NASA are continuously collecting datasets of earth dynamics—e.g., temperature, vegetation, and cloud coverage—through satellites. This data is stored in a publicly available archive for scientists and researchers and is very useful for studying climate, desertification, and land use change. The benefit of this data comes from its richness as it provides an archived history for over 15 years of satellite observations. Unfortunately, the use of such data is very limited due to the huge size of archives ($> 500TB$) and the limited capabilities of traditional applications. In this demo, we present Shahed, an interactive system which provides an efficient way to index, query, and visualize satellite datasets available in NASA archive. Shahed is composed of four main modules. The *uncertainty* module resolves data uncertainty imposed by the satellites. The *indexing* module organizes the data in a novel multi-resolution spatio-temporal index designed for satellite data. The *querying* module uses the indexes to answer both spatio-temporal selection and aggregate queries provided by the user. The *visualization* module generates images, videos, and multi-level images which gives an insight of data distribution and dynamics over time. This demo gives users a hands-on experience with Shahed through a map-based web interface in which users can browse the available datasets using the map, issue spatio-temporal queries, and visualize the results as images or videos.**

I. Introduction

Huge amounts of remote sensing data are collected by satellites and are made publicly available for use by scientists. For example, NASA provides the Land Process Distributed Active Archive Center (LP DAAC) [5] with more than 500TB of data that is increasing in a daily manner. The collected datasets measure several physical phenomena including land temperature, vegetation, and thermal anomalies. This data is useful in many applications and research areas such as land cover change, detection of desertification, and climate informatics. Data is collected at different temporal resolutions (e.g., daily and weekly) and spatial resolutions (e.g., 250 meters and 1 KM) and kept in one tremendously large archive.

Although the data in the archive is very rich, scientists are not able to make full use of it due to three challenges. (1) The huge size of this archive makes it too difficult to be processed by traditional applications designed for a single machine. Existing techniques either work on a downsized sample of the data (e.g., one day out of each year) and produce an approximate answer, or take too long time (e.g., hours or days) to produce a more accurate answer. (2) The LP DAAC archive provides only the raw data and does not support running spatio-temporal queries. A standard way to access the data is through a web interface (e.g., Reverb [1]) in which the user provides spatio-temporal criteria and is presented with a list of files to download. For example, to find all temperature values in a specific point over a period of one year, the web interface is accessed first to retrieve 365 files for all days in that year which contains hundreds of millions of points and with a total download size of around 2GB. These files are then processed to extract the 365 points in the answer. (3) The files downloaded from the archive usually have missing values due to satellites mis-alignment or due to clouds covering an area underneath. This requires a data cleaning step to recover missing values and to ensure that the query answer is correct.

In this demo, we present Shahed; a system that allows users to query and visualize satellite data efficiently with a user-friendly interface. The core of Shahed consists of four main components. (1) The **uncertainty** component processes newly downloaded data and uses a two-dimensional interpolation technique to estimate missing data. (2) The **indexing** component employs a novel multi-resolution spatio-temporal index which allows Shahed to answer spatio-temporal queries efficiently. (3) The **querying** component uses the constructed spatio-temporal indexes to answer both selection and aggregate spatio-temporal queries in a real time manner. (4) The **visualization** component allows users to export images, videos [1], and multi-level images which represent the distribution of the data over space and time as a heat map.

Shahed employs SpatialHadoop [3], a MapReduce framework for spatial data, as a backbone to handle the huge amounts of data. SpatialHadoop uses MapReduce as the main programming paradigm for spatial data processing. Since MapReduce is designed for offline batch processing, Shahed uses SpatialHadoop for efficient index construction and visualization as both are offline jobs. For interactive spatio-temporal queries, Shahed employs a separate query engine that utilizes the spatio-temporal indexes to provide real time query answers. Shahed also provides a simple web interface which allows users to access all of its features easily.

[0#]This work was done while these two authors were visiting the GIS Technology Innovation Center in Umm AlQura University and is supported by the center under project GISTIC-13-05

[1]Please refer to an example at http://youtu.be/hHrOSVAaak8

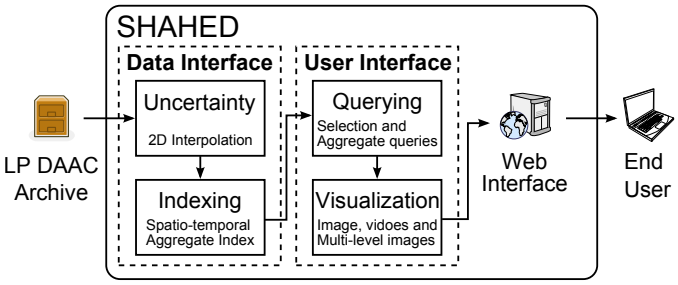
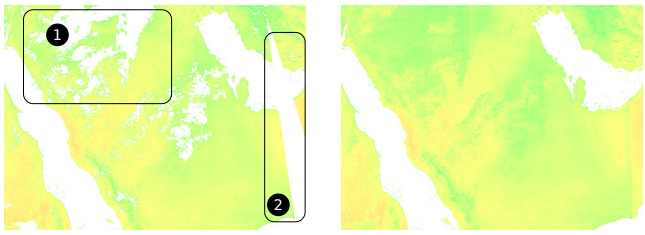


Fig. 1. SHAHED Overview



(a) Types of missing data (b) Missing data recovered

Fig. 2. Recovering missing data in a heat map

II. Overview of Shahed

Figure 1 gives an overview of Shahed. The input data from the LP DAAC archive goes through a *data interface* layer which regularly reads new datasets and stores them in spatio-temporal indexes. The user interacts with the system through a *user interface* layer which answers spatio-temporal queries and visualizes the results. The system contains four main modules described briefly below.

The **uncertainty module** is triggered on newly downloaded datasets to recover missing values using a two dimensional interpolation function. The **indexing module** employs a novel spatio-temporal index structure based on the quad-tree [6] which keeps data organized spatially and temporally in a hierarchical index structure and annotates it with partial aggregates to answer both selection and aggregate spatio-temporal queries efficiently. The **querying module** uses the aggregate spatio-temporal indexes to answer both spatio-temporal selection and aggregate queries in real-time manner by employing early pruning techniques and using partial aggregates available in the index. The **visualization module** allows users to visualize the answers returned by the querying module as heat maps. Shahed supports visualization of heat maps as still images, videos, and multi-level images, all generated efficiently using MapReduce programs running in SpatialHadoop.

Shahed makes all system features available through a map-based web interface that is easy to use. Users can navigate to any area and see the heat map of a selected dataset (e.g., temperature). Users can also issue spatio-temporal queries and get the answer in real-time or if query contains a large selected area over a period of time, the results are delivered through email.

III. Uncertainty

The way in which data is collected by satellites imposes a level of uncertainty in the data. Figure 2(a) depicts an example of missing data in the land temperature values in the area of Saudi Arabia. There are two types of missing data shown in the figure. (1) The white random area is caused by clouds that blocked the satellites sensors at the moment the satellite image was taken. (2) The sharp triangle-like white area is caused by satellites mis-alignment that leaves a blind spot not covered by any of the satellites. Depending on the application, these missing values might affect the quality or correctness of a query answer. Using historical data to estimate missing value is not possible due to the high variance of some datasets (e.g., temperature).

Shahed employs a two-dimensional interpolation technique that estimates missing data based on nearby points. Figure 2 shows an example of how the answer looks like after employing the uncertainty module in Shahed to recover missing data. The basic idea of the two-dimensional interpolation technique is to provide two estimates for each points, one for each dimensions, and then compute a final estimate as the average of these two estimates. The x-estimate is computed using a linear interpolation function based on the two closest points on the same horizontal line. Similarly, the y-estimate is computed using the two closest points on the same vertical line. If no points are available on the same horizontal or vertical line, the corresponding estimate is not computed and the other one is used as the final estimate.

IV. Spatio-temporal Indexing

The indexing module in Shahed supports a novel multi-resolution spatio-temporal index that is designed for satellite data. Figure 3 gives a high level overview of the index which is organized in two orthogonal hierarchies, *temporal* and *spatial*. In the *temporal hierarchy*, the index is organized in three temporal layers, each one contains a copy of all the data partitioned by a different temporal resolution, namely, yearly, monthly, and daily. A temporal partition is created only after the corresponding time frame is concluded. For example, a partition has been created for 2013 as that year has finished while no partition has been created yet for 2014. In the *spatial hierarchy*, each temporal partition (e.g., 2012) is further indexed using an aggregate quad tree which is similar to a quad tree [6] with augmented aggregate values in each node summarizing the subtree under that node. The aggregate functions supported by Shahed are minimum, maximum, count, and sum. More aggregate functions can be derived such as *range* and *average*.

As new datasets are added to the LP DAAC archive, a daily job in Shahed is triggered at midnight to download them. New data is always added to the archive as daily snapshots where each snapshot is indexed using an aggregate quad tree in the daily layer. To construct an aggregate quad tree, point are first sorted using their Z-order values, a quad tree is constructed on top of the sorted index using the properties of the Z-curve [2], and finally the aggregate values are computed in each node. To compute aggregate values, we start by the leaf nodes and compute aggregate values by scanning points under each one. The aggregate values of leaf nodes are further aggregated to compute the aggregate values of their parents until reaching the root.

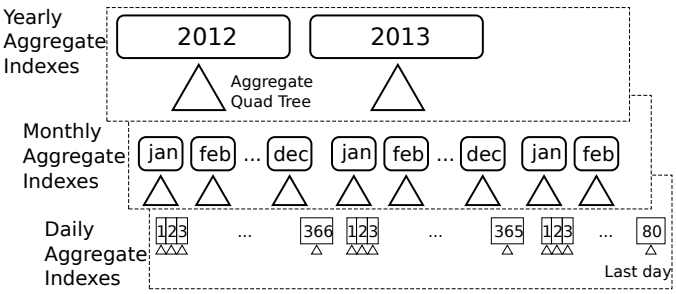


Fig. 3. Multi-resolution spatio-temporal index

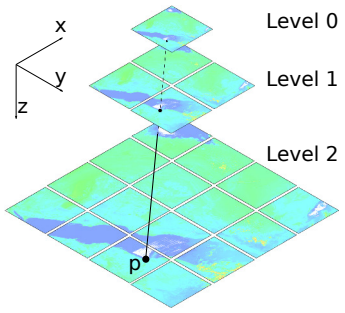


Fig. 4. Multi-level heat map images

Once all daily partitions in one month are created, they are merged to form one partition in the monthly layer. In the merge process, the input is a list of aggregate trees each corresponding to one tile and the output is another aggregate tree corresponding to the same tile but at a higher level in the temporal hierarchy. The output tree has the same size as input trees in terms of number of points and nodes. The difference is the *cardinality* which is the number of values stored at each point. In the output tree, all values stored at the same spatial point are stored as a contiguous range in the output file sorted by time. This allows a query asking about all values in a specific point over a large time frame to be answered with one disk access. In addition, aggregate values in each node in the output tree are computed by aggregating all values of corresponding nodes in all input trees.

V. Query Processing

In this section, we describe how the spatio-temporal indexes constructed by the indexing module are used to answer spatio-temporal *selection* and *aggregate* queries.

In a **selection query**, the user specifies a dataset (e.g., temperature), a spatial range as a rectangle, a temporal range as start and end dates, and the query answer is all values contained in this range. The query runs in two steps, *temporal filter* and *spatial filter*. In the *temporal filter* step, the yearly temporal partitions are examined first and partitions that are completely contained in the temporal range are selected. After this, the monthly and daily partitions are examined in the same manner to cover the entire input range. Examining the partitions in that order ensures that the number of matched partitions is minimum which increases the query performance. In the *spatial filter* step, the aggregate quad tree in each partition is processed with a standard spatial range query and all matching values are returned. To query an aggregate quad tree, a traditional range query is run against the associated stock tree starting at the root and going deeper as needed until the leaves. The values contained under each matching node are retrieved from the aggregate quad tree stored on disk. Notice that all points contained under one node are guaranteed to be in a contiguous range on disk as the points are kept sorted by their Z-order values [2].

Similar to a selection query, in an **aggregate query** the user specifies a dataset, and a spatial and temporal ranges. However, the answer is a set of aggregate values for all matching points, namely, minimum, maximum, count, and sum. The query runs in two steps, *temporal filter* and *aggregate calculation*. The first step is exactly the same as in the selection query. In the *aggregate calculation* step, each matching aggregate tree is queried to compute the aggregate values of the part covered by the query range. These aggregate values are further aggregated to compute the final answer. Running an aggregate query on a quad tree is similar to a selection query with two differences. First, instead of returning matching records, their aggregate values are computed and returned instead. Second, if a node is completely contained in the query range, the aggregate values cached in it are returned without going deeper in the tree.

VI. Visualization

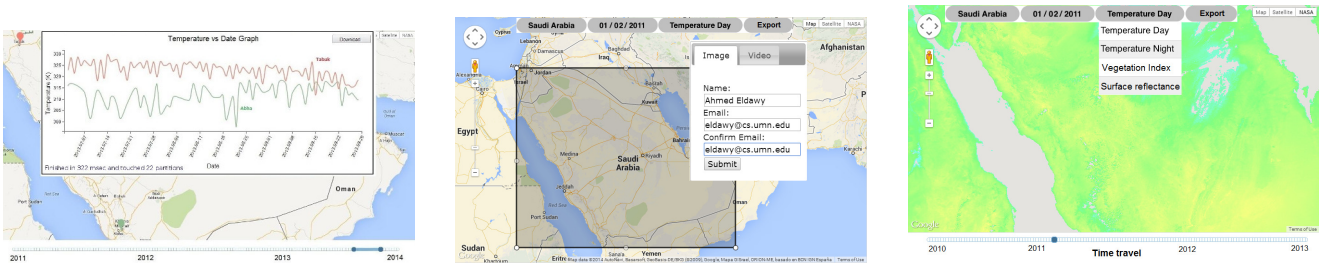
Shahed supports three visualization options, *images*, *videos*, and multi-level images which use MapReduce programming to visualize query answers as heat maps.

A. Image and Videos

A query answer can be visualized as a heat map that shows the distribution of values in the selected area and time range. A still image is generated for each day while a video is generated for a time range by combining a series of images. A single image is generated using a MapReduce program in which the map function partitions the data using a uniform grid and the reduce function plots a heat map for each tile. A heat map for a single cell is generated by scanning all points in that cell, mapping each one to a pixel which is colored according to the value of the point; blue maps to the minimum value and red maps to the maximum value. If more than one point map to the same pixel, their average is used to color that pixel. Once all images are generated, a final *stitch* step puts all generated images together to form the final picture.

B. Multi-level images

An alternative option for visualization is the multi-level images in which a set of images is generated for different regions and zoom levels. Figure 4 gives an example of a three-level heat map image for temperature. In level 0, the whole area is represented as one image of size 256×256 pixels. In zoom level 1, the same area is represented in higher resolution by splitting it into four images, each of size 256×256 pixels. To handle the exponentially increasing number of tiles/images per zoom level, we employ a MapReduce program that runs in two steps, *partition* and *plot*. In the *partition* step, the map function replicates each data point to all overlapping tiles. For example, the point p in Figure 4 is replicated to three tiles, one in each zoom level. In the *plot* step, the reduce function



(a) Spatio-temporal point queries (b) Heat map generation (c) Browsing multi-level heat maps

Fig. 5. Screenshots of Shahed

takes all points in each tile, and generates a heat map for them as an image of size 256×256 pixels. No stitch step is required as these images are kept separate on disk.

A drawback in this algorithm is that it incurs an extremely high overhead in tiles at the higher levels of the pyramid as they cover larger areas. For example, the single tile at the first level of the pyramid might cover the whole world which contains hundreds of millions of points. This huge number of points are not really needed to generate the heat map for two reasons. (1) Tiles at higher level do not need to be accurate as they give a high level overview. (2) Number of pixels in the generated image is $256^2 \approx 64K$ is way smaller than number of points overlapping the tile. To improve the algorithm, we employ a novel *adaptive sampling* technique which adjusts the number of points in each level to be equal to number of pixels in the generated image. The basic idea is to replicate a point to a tile at level i with a probability p_i which is adjusted such that the expected number of sampled points is equal to number of pixels in one image. This technique is shown to be much more efficient without sacrificing much of the quality.

VII. Demonstration Scenario

During the demonstration, we run a prototype of Shahed on an Amazon EC2 cluster of twenty nodes with SpatialHadoop 2.2 installed. The cluster is preloaded with the temperature, vegetation, and surface reflectance datasets for the last three years to make the demo self contained. The attendee will be able to access the system from a web interface accessible through a laptop while the processing will be done by Shahed on the cluster. As shown in Figure 5, the main interface displays an interactive map based on Google Maps [4]. On the top right, there is a map selector where the user can switch between map view, satellite view, and heat map view. On the top, there is a toolbar with a search box, date selector, and dataset selector. There is also a button that exports either an image or video. The details of the demonstration scenario for the three main functionalities in Shahed are described below.

A. Spatio-temporal Queries

A basic feature in Shahed is to run spatio-temporal selection and aggregate queries. Figure 5(a) shows a simple selection query which selects all values in two distinct points over a period of three months. The answer is displayed as a chart which allows users to easily compare and contrast the temperatures at the two selected points. The download button allows the attendee to download the answer as a CSV file to be used in another application. In addition to point queries, users can also specify spatial ranges where Shahed returns minimum, maximum, and average temperature in the given area for each day in the selected time period or for the one average for the whole selected range. Shahed also displays some statistic about the query such as total running time and number of partitions processed to answer the query.

B. Image/Video Generation

Figure 5(b) shows an example of generating heat map images or videos. The attendee will be able to specify a range on the map, a dataset and either a specific date for image, or a start and end dates for video. In addition, the user needs to specify an email address to which the generated image or video will be sent to. As the submit button is clicked, MapReduce jobs are sent to SpatialHadoop to generate the required images. The attendee will have an access to the admin interface of SpatialHadoop to see the progress and details of the running jobs. Once the jobs are finished, an email is sent to the user-provided email address with a link to download either the image or the video. In addition, a KML file provide to preview the generated image on Goggle Earth or a similar application.

C. Multi-level Images Browsing

To make it easier for users to explore the data, Shahed provides a heat map mode which displays an interactive heat map for the selected date and dataset as shown in Figure 5(c). The heat map is based on Google Maps and it provides the same navigation experience such as pan and zoom. As the visible area or the data on the top change, the web page loads the corresponding set of images from the generated pyramid. The dataset selector on the top allows users to view the heat map of different datasets such as vegetation or surface reflectance.

References

[1] Reverb - The Next Generation Earth Science Discovery Tool. http://reverb.echo.nasa.gov/reverb/.
[2] M. Bern, D. Eppstein, and S.-H. Teng. Parallel Construction of Quadtrees and Quality Triangulations. *IJCGA*, 9(6):517–532, 1999.
[3] A. Eldawy and M. F. Mokbel. A Demonstration of SpatialHadoop: An Efficient MapReduce Framework for Spatial Data. In *VLDB*, pages 1230–1233, 2013.
[4] Google Maps. http://maps.google.com.
[5] MODIS Land Products Quality Assurance Tutorial: Part:1, 2012. https://lpdaac.usgs.gov/sites/default/files/public/modis/docs/MODIS_LP_QA_Tutorial-1.pdf.
[6] H. Samet. The Quadtree and Related Hierarchical Data Structures. *ACM Computing Surveys*, 16(2):187–260, 1984.