SPATIO-TEMPORAL INDEXING OF THE QUIKSCAT WIND DATA

Félix R. Rodríguez, Manuel Barrena

Department of Engineering of Informatic and Telematic Systems, University of Extremadura, Cáceres, Spain. {felixr, barrena}@unex.es

ABSTRACT
Global wind data are increasingly used in many applications and research fields, mainly because it provides global coverage of ocean winds with an unprecedented view, and data are available at a sufficiently large spatial scale and high temporal resolution. Global wind data provided by QuikSCAT is highly variable and needs a completely adapted index to enhance the retrieval process of the desired data. A novel indexing method has been developed, named Q-full-tree, integrating two existing structures: Q-tree, as the spatial index, and LV-tree, as the temporal index. Also, a particularized bulk load process for the QuikSCAT data has been designed. Q-full-tree remains completely accessible all wind data from the QuikSCAT since the satellite was launched, at any granularity, any spatial query, and any time interval.

Index Terms—Spatio-temporal indexing, Spatio-temporal databases, GIS, Geospatial data, Geocomputation

1. INTRODUCTION
Global wind data are increasingly used in many applications and research fields, such as environmental sciences, biology, meteorology, and climate sciences. Satellite altimetry provides a unique opportunity to understand the dynamics of sea winds. Currently, the most widely data source used to work with global spatiotemporal wind information (e.g., an area covered by an evolving storm) is provided from the SeaWinds/QuikSCAT (QuikSCAT) satellite [1], an Earth-orbiting NASA scatterometer (microwave radar). Mainly because it provides global coverage of ocean winds with an unprecedented view, and data are available at a sufficiently large spatial scale and high temporal resolution. A clear example of the utility of these data is a recent work of great impact that use the spatiotemporal component of winds (spatial characteristics of wind change over time) to infer that there are wind highways allowing a large number of species share earth areas separate thousands of kilometers, and, also, that those highways follows a very precise route, accessible during specific periods of time [2].

If we need to query about any isolated point, to select and query any particular wind area, to run complex queries, to make easier the update of existing data, to facilitate access to data at different resolution levels, to provide concurrency, data log, or to supply temporal evolution of data, it is extremely useful to maintain indexed all these global wind data with a powerful index structure. Based on the previous requirements, we have indexed, by means of a multidimensional structure and a temporal one, the huge amount of wind data coming from QuikSCAT. In this paper we present the results of our research on spatio-temporal indexing. Specifically, we propose a highly efficient way to store and index such amounts of data so that users can benefit from the above-mentioned advantages.

A plethora of spatio-temporal indexes have been proposed in literature [3, 4], but this type of very bulky and extremely variable data (wind data have a different value every time) needs a completely adapted index. Each observation captured by the satellite, and saved in a record file, is a moving object that does not follow any pattern or constant behavior with respect to the same point in the past or future, or with respect the other observations. Thus, we have developed a novel indexing method particularized to the QuikSCAT data that efficiently stores the whole wind dataset since the satellite was launched on 1999, and that loads the new acquired daily data. In addition, the proposed index is a member of the space partitioning indices family (literature usually proposes data partitioning indices), and this constitutes a novelty in our proposal.

The remainder of this paper is organized as follows: Section 2 presents the bulk load process of the QuikSCAT data into the Q-full-tree; Section 3 gives a brief description of the Q-full-tree, introducing the two structures that builds it: Q-tree and LV-tree. And finally, Section 4 includes the conclusions.

2. INDEXING WIND DATA
The SeaWinds on QuikSCAT mission daily provides data at a 25-km, with a typical accuracy of 1 m/s in speed, and 15° in direction (12.5-km resolution is also available). The data follows the NASA specific HDF4 format [5], laborious to handle if we want use them combined with other data sources, as
they might be from the Terra satellite.

To face the task of indexing as large an amount of variable data, we have made the following processes, all written in the Java programming language (ver. 1.4). Wind data are daily downloaded via FTP from the QuikSCAT site (ftp://podaac.jpl.nasa.gov/ocean_wind/quikscat/L2B). After that, wind data are unzipped and locally stored in disk. Daily data are transformed from the HDF format into the appropriate tuples (records) to be massively loaded into the index structure. The specifications for interpreting these HDF data are provided in [1]. Our system stores the tuples with the form: (timestamp, latitude, longitude, wind-speed, wind-direction, rain-ratio). The bulk load process begins now. To massively load the daily wind data, initially we follow the same method we have use for indexing the spatial Shuttle Radar Topographic Mission (SRTM) data [6], but adding two variations. Firstly, as data are not equally situated in latitude and longitude as the SRTM data, but may be located in any position, or not located, the process of spatial division is modified to fit rectangular regions which contain a number of observations (tuples) close to, or equal to, the maximum number that a data page (data block to disk) can accommodate. Secondly, data pages are not saved into the Q-tree. An identifier is assigned to each rectangular region, and this identifier along with its timestamp is the information loaded into the spatial Q-tree structure. In turn, all the tuples (records, observations) belonging to this rectangular region are moving to the data page to be carried to an external data file. The region identifier (key field), its timestamp, and the page identifier are related using the temporal structure, the LV-tree. Figure 3 shows the general outline of the bulk load process of wind data. The Figure illustrates the data loading of the tuples belonging the spatial region identifies with the number 1 in this case. The identifier is stored in the last level of the Q-tree (the leaves), and the data page with all the tuples of this region is related with its region identifier and the timestamp into the LV-tree.

Finally, to check the quality of results for the spatio-temporal queries on the Q-full-tree, we have linked the diverse query responses data with the Unidata IDV (Integrated Data Viewer) [7], adjusting this viewer tool to show both wind and rain-rate data. Figure 2 shows a snapshot of one of these tests.

3. THE Q-FULL-TREE

Our index prototype, the Q-full-tree, combines both the Q-tree [8] and the LV-tree [9] to build a new spatiotemporal index structure well suited to the extremely variable data that QuikSCAT provides.

The Q-tree is a multidimensional indexing structure in the family of space partitioning kd-tree [10] based indexing methods. It is a hierarchical multi-purpose, dynamic, balanced and paginated (blocks of data are used) multidimensional index structure with concurrency and log file mechanisms. Index and data are clearly separated and every node is homogeneous in structure. All the nodes of the tree correspond to disk pages, and the structure is designed in such a way that

Fig. 1. General outline of the bulk load process of QuikSCAT data onto the Q-full-tree. It shows the daily partial load of one wind data page. A spatial region is selected (identifier=1) according to a page size previously established. The spatial region identifier and the data page are inserted into the Q-full-tree.
the number of nodes visited during the search and insert processes is minimized. The Q-tree is used as the spatial index.

Figure 3 gives a general perspective of a Q-tree, illustrating how the nodes of the Q-tree (called Qnodes) are organized in a tree structure. At the lowest level of the tree, there are data Qnodes (called containers) where the records (points of the space) are stored. The rest of the nodes are index Qnodes; those at the level above the leaves are called regions ($R_1$, $R_2$, $R_3$), and the index Qnodes at higher levels are called meta-regions ($S$). Every Qnode is stored as a disk page and hence has a fixed size. Thus, when there is no free space on a page to store new information, the corresponding Qnode suffers a splitting process. From a geometrical point of view, every Qnode covers both disjoint and connected zones of the data space. Containers represent compact rectangles while index Qnodes represent rectangles what have likely suffered the extraction of rectangular regions inside. Therefore, index Qnodes can be either compact or holey rectangles.

Data containers store the d-dimensional points, and, according to the kd-tree basis, they are split by using a (d-1)-dimensional hyperplane (the splitting hyperplane) that divides the space into two disjoint regions based on load balance criteria. Should the distributions of point’s value inside the overloaded data container not render an even division of points into two new containers, we set a cut-off point in the splitting hyperplane by means of a unique attribute that can identify each d-dimensional point. The splitting of an overfull container enables the new border to be added to the parent Qnode. Index Qnodes contain a kd-tree, called the local tree, which stores the hyperplanes used to separate the children Qnodes. Because data page splitting involves promoting information to the parent Qnode index, index pages can also become overfull. When an index Qnode becomes overfull, it is split by means of the extraction of a properly selected subtree, which is moved to a new index Qnode. The details about the splitting algorithm can be seen in [8].
Combining Q-tree and LV-tree, and slightly altering the structure of Q-tree, we build Q-full-tree, taking advantage of both indexes. QuikSCAT data are well defined spatially, and Q-tree works extremely well for this kind of spatially delimited data. On the other hand, each daily QuikSCAT dataset is a meteorological version, and LV-tree easily separates historical versions based on a key. Thus, i) if we associate each spatial region (determined by the Q-tree) to an identifier, ii) we store the data page with all observations that fall within the spatial region into an external repository file, and, iii) we take the region identifier as the key that LV-tree indexes in a given timestamp, adding the data page identifier to the information that every record has in the LV-tree, then, we have the essence of the Q-full-tree. For this purpose, and summarizing, the changes made to form the Q-full-tree are:

- Containers are removed completely from the Q-tree, storing the data pages in a separate file. Instead of containers, the Q-tree leaves stores the identifier of the current region (alive) and a list of changes (log) if the region is divided.
- LV-tree stores the relationship between region identifier, timestamp, and page identifier. Each new version of a region remains in a natural way by The LV-tree.
- If a region changes because is spatially divided (the number of QuikSCAT observations increases beyond the capacity established for the data page to stores these records), the identifier of each new spatial region and the old spatial region identifier are both together save in the corresponding leave of the Q-tree using a log list. Thus, the search of old regions (not alive) is allowed.
- If a spatial region, in a certain timestamp, contains less than the minimum capacity to host a data page (usually half), these data are stored on the data page of the same region with a previous timestamp, if there is space inside the data page; else, but underloaded, the data page is saved, allowing future underloaded regions share data pages.

4. CONCLUSIONS

A completely adapted index structure for the QuikSCAT wind data is presented: the Q-full-tree. The spatiotemporal index integrates two existing structures: i) the Q-tree, as the spatial index, and ii), the LV-tree, as the temporal index. Wind data are daily bulk loading onto the Q-full-tree, facilitating spatial (point and window/range) queries, temporal (time slice, and interval time) queries, and spatiotemporal queries by combining both spatial and temporal. In order to make faster the bulk load process, we consider spatial regions of wind data that completely fill a data page with a single inserting operation. The Q-tree stores spatial region identifiers (as keys) that fully accommodate all the data involved in each data page, and the timestamp where this spatial region is selected. Thus, the data pages, their spatial region identifiers, and their timestamps, are stored into the LV-tree, updating the version of every data page inserted. The Q-full-tree index provides a unique and useful way to link wind data with other data, or be embedded in an open GIS as it has been done with the SRTM data [6].

5. REFERENCES


