On-the-fly Visualization of Scientific Geospatial Data Using Waveltes

Cyrus Shahabi, Farnoush Banaei-Kashani, Kai Song

Department of Computer Science, University of Southern California
Los Angeles, CA 90089-0781
{kaisong, banaeika, shahabi}@usc.edu

***Abstract*— Visualization of multi-dimensional scientific datasets often requires computationally-complex data analysis. Existing technologies perform offline data analysis in preparation for on-line visualization. In this paper, we propose a wavelet-based method to enable *on-the-fly* visualization of scientific geospatial data. We used SST (Sea Surface Temperature) datasets to demonstrate the effectiveness and efficiency of our proposed method.**

I. Introduction

With the emergence of sensors, numerous data readings are collected by the sensors. Due to the huge size of the data, visualization of the scientific datasets is a great challenge. The online map systems (e.g. Microsoft Virtual Earth, Google Map and Yahoo! Map) are the state of the art tools used for this purpose. With these tools, scientists can navigate the data and make research observations by adding various overlays. However, these overlays and tiles are all pre-generated beforehand; hence, two limitations in data visualization with these tools:

- *Data update*: because all the overlays and/or tiles are generated in advance, data changes will result in significant extra computational cost. Therefore, existing techologies fail to provide real-time visualization.
- *User-defined resolution*: for the purpose of multi-resolution display, existing technologies break overlays into tiles at different display resolutions, where the tile sizes are determined by the resolution levels. However, the levels themselves are pre-defined, which means the users can only view the overlays at some specific resolutions.

Motivated by the above observations, we propose a wavelet-based method, which supports on-the-fly visualization of scientific geospatial data. Our contributions are summarized as follows:

- We extend our prior technique [1], a wavetlet-based tool for efficient data analysis, and develop real-time data analysis techniques that enable *on-the-fly* visualization of scientific geospatial data.
- We develop a visualization system that supports various user-defined ranges and resolutions.

As far as we know, our paper is the first work that investigate the problem of on-the-fly visualization for large scientific data. We also developed a map-based system to demonstrate our proposed features.

II. Background

In our prior work, we have developed a system called ProDA [1] for efficient data analysis. With ProDA, we have proposed and developed a data analysis framework that utilizes the wavelet transformation of the multi-dimensional data to enable fast, efficient, and progressive query processing. The use of the wavelet transformation is justified by the fact that the query cost is reduced from the query size to the logarithm of the data size, which is a major benefit especially for large range queries. With ProDA, our general query formulation supports arbitrary high-order polynomial range-aggregate query, where all queries are efficiently processed in a progressive fashion. Therefore, ProDA is best utilized for range-aggregate queries, especially for large ranges and aggregate queries that can be formulated as polynomial functions (e.g., count, sum, average, variance, covariance, kurtosis). Using ProDA, an aggregate query on a dataset with d dimension of size n at each dimension only costs $O(log^d n)$.

III. On-the-fly Data Visualization

In this section, we discuss how we on-the-fly generate visualization as overlay and propose two optimization strategies to reduce the query time.

For scientific data, it is very important that the data analysis results can be displayed visually to the scientists for research purposes. In this paper, we visualize the results by generating a color-coded overlay on top of online maps (e.g., Google Map). Therefore, the key problem is how to color-code the overlay on-the-fly in various resolutions. Specifically, when resolution changes (e.g., zoom in/out), the range of the values of the results should alter accordingly. Therefore, the image overlay need to be adjusted based on the new range. Existing technologies pre-generate overlay tiles for every pre-defined resolution levels. Given a user's query containing the resolution and geographical information, the system just retrieves the corresponding tiles and displays them on top of a map. Although this method is efficient, it lacks the flexibility to be adaptive to users' requirements. i.e., if the pre-defined resolution levels and/or even the underlying data change, the pre-generated overlay tiles not the accurate answers to the query. In our method, image overlays are generated in real-time manner. For a given resolution level, we generate the overlay image by calculating the geographical area covered

by each pixel in the image. Such area is denoted as *coverage area* through out this paper. As illustrated in figure 1, every grid is covered by a pixel which is represented by a red dot. Each of the grids is called the cover area of that pixel. For each pixel, we then submit a range query, which is considered as the value of the pixel, to ProDA. By utilizing the computational capability of ProDA, query results can be obtained in an efficient fashion. Values for the pixels are then scaled and plotted to produce an image overlay as the visualization result.

Fig. 1. Cover area

Let h and w denote the dimension of the visualization map, i.e., the height and width of the map are h and w pixels respectively. Moreover, assume the geographical area covered by the map are x degree in longitude and y degree in latitude, which are represented by n and m data points respectively. Note that n and m are also determined by *number of data points per degree*, which is a property of the raw data when they are sampled. Therefore, each pixel covers an area with x/w by y/h degree, and also n/w by m/h data points. For each pixel, We take the average value of the n/w by m/h data points as its representative value of the pixel and then submit an "average" query to ProDA. Recall that an aggregate query on a dataset with size of n only cost $O(logn)$ and for multi-dimensional cases, the cost is the product of the cost of each dimension. Therefore, each query will take $O(log(n)log(m))$, and it takes $O(whlog(n)log(m))$ to complete the whole visualization process, while the traditional method takes $O(nm)$.

It could be argued that when $w = n$ and $h = m$, ProDA actually takes $O(mnlog(n)log(m))$, and this number is even bigger than nm, which is the total number of the coefficients. This observation implies that if each pixel is processed separately, some coefficients are retrieved redundantly. To handle this, we treat the queries on all the pixels as a batch query and introduce cache mechanism to reduce the redundancy. If the size of the cache is larger than $O(mn)$, which can hold all the coefficients, every coefficient is retrieved at most once during the entire visualization process; otherwise, the cache can ensure that every efficient is retrieved as few times as possible under the specific cache size. By exploiting the cache mechanism, system performance can be optimized.

Another way to accelerate the query process is to take advantage of the wavelet's multi-resolution and approximation properties. In previous work, we proposed *split-shift* [2] which are operations directly performed in the wavelet domain. For a specific frequently submitted query type, we improve the system response time of ProDA to adapt it to the required high-performance data analysis and visualization by further extending *split-shift*. Specifically, we utilized the multi-resolution property of wavelet to provide query and visualization of a different granularity [3]. For rough queries under low resolution, only few wavelet coefficients are necessary. As resolution becomes high, more coefficients are retrieved incrementally to refine the results. The intuition behind the method is that only a small number of wavelet coefficients is needed to capture the trends, which is known as best B-term approximation property or small-B approximation property[4][5][6].

Applying these two optimization techniques, the average response time is improved by two orders of magnitude, which is within 3 seconds.

IV. Experiments

In this section, we discuss the implementation of a map-based on-the-fly visualization system to demonstrate our research results. All the experiments were performed on a PC with 3.2GHz CPU, 2G Memory and Windows XP SP2.

A. Data Repository

We conducted our experiments with the SST (Sea Surface Temperature) dataset. The SST raw data consists of 5 netCDF [7] files, each of which contains only one measure attribute "sea surface temperature" and two dimension attributes: latitute and longitude. The SST dataset is a daily-sampled data that describes sea surface temperature in a global area with the resolution 1km*1km (i.e., every data point represents a 1km*1km area). Therefore, for each file, there are 36000*18000 data points in total, and the size of each file is 2GB. Note that about 1/3 of data in each file consists of null values, which implies that the corresponding areas are land and no data of sea surface temperature is sampled there. The SST dataset will be expanded and updated in a daily fashion; therefore, instead of transforming all data into one wavelet file, we transform each raw data file into one wavelet data file. Once new data files arrive, new wavelet files are generated accordingly.

B. System Implementation

The system is developed as a mashup on top of Google Map. Figure 2 shows the SST visualization result in the North Atlantic Ocean area. For each pixel, we average all the SST values (i.e., submitting *average* queries to ProDA) within the *coverage area* of that pixel and consider the average as its value. The temperature in this area ranges from 285.0K to 303.9K. When we zoom in to see more details in a small area (see Figure 3.a), exsiting system just show the zoomed-in display without re-scaling the results(see Figure 3.b). With our system, we update the visualization overlay by re-calculating the *coverage areas*, re-submitting the aggregate queries and

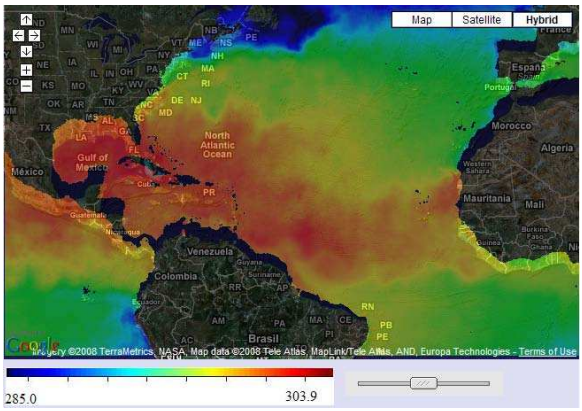

Fig. 2. Visualization result in North Atlantic Ocean

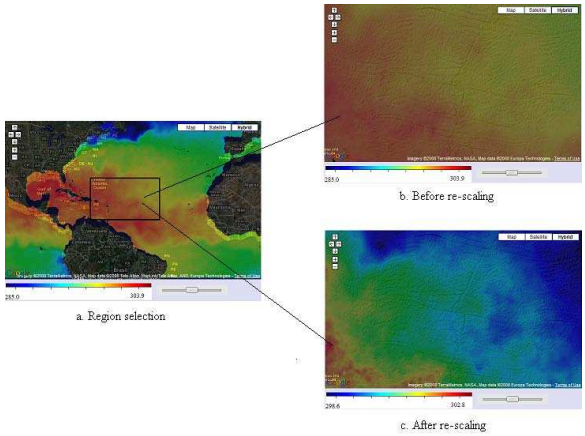

Fig. 3. Visualization result in North Atlantic Ocean

re-scaling the queries results, whose range has change from 285.0K-303.9K to 298.6K-302.8K in this example.

We also randomly selected 10 regions for performance test. Caching and approximation optimization are also incorporated into the system. For cache optimization, we use number of coefficients as the unit of cache size, and we set the size of the cache to 1000; For approximation optimization, we always choose the coefficients whose sum contributes more than 90% of the total energy. Figure 4 shows the reponse time for the 10 cases. From the graphs, we can tell that either cache or approximation can bring significant enhance for response time. Combining the two can further achieve a slight enhancement. However, in the high-resolution cases, approximation may result in higher errors since the visualization requires better precision results.

V. Conclusions and Future Work

In this paper, we propose a method that can analyze and geospatially visualize scientific data on-the-fly at any user-defined resolution level. Specifically, a 2D map is divided into small areas, each of which is represented by a pixel. By submitting a range query for each area, we receive the results from ProDA and color code them into an image overlay.

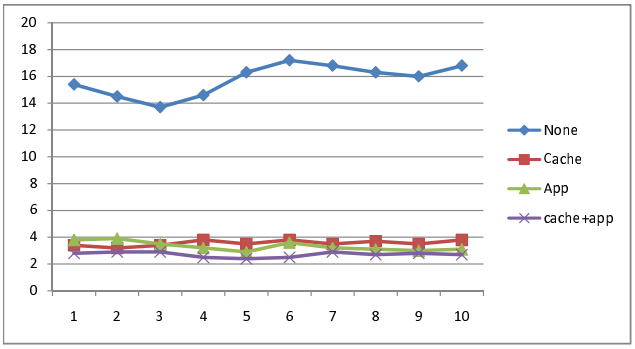

Fig. 4. Response time under different methods

Experiments on two datasets demonstrate the efficiency and effectiveness of our proposed method.

In future work, we expect to extend our system to handle 3D datasets for analysis and visualization in a 3D environment (e.g. Google Earth). Meanwhile, we plan to investigate I/O sharing problems (e.g., how the size of cache will affect the system performance and how many coeffcients are enough to obtain an acceptable visualization results) to further improve the system performance.

Acknowledgment

We appreciate Dr. Mehrdad Jahangiri's help on ProDA and wavelet. This research has been funded in part by NSF grants IIS-0238560 (PECASE), IIS-0324955 (ITR), IIS-0534761 and IIS-0742811 (SGER), and in part under JPL SURP program. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] M. Jahangiri and C. Shahabi, "Proda: a suite of web-services for progressive data analysis," in *SIGMOD Conference*, 2005, pp. 894–896.

[2] M. Jahangiri, D. Sacharidis, and C. Shahabi, "Shift-split: I/o efficient maintenance of wavelet-transformed multidimensional data," in *SIGMOD Conference*, 2005, pp. 275–286.

[3] M. Jahangiri and C. Shahabi, "Plot query processing with wavelets," in *SSDBM*, 2008, pp. 473–490.

[4] G. Das, D. Gunopulos, N. Koudas, and N. Sarkas, "Ad-hoc top-k query answering for data streams," in *VLDB*, 2007, pp. 183–194.

[5] G. Cormode, M. N. Garofalakis, and D. Sacharidis, "Fast approximate wavelet tracking on streams," in *EDBT*, 2006, pp. 4–22.

[6] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss, "Optimal and approximate computation of summary statistics for range aggregates," in *PODS 2001, Proceedings of the Twenteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 2001, pp. 228–237.

[7] "http://www.unidata.ucar.edu/software/netcdf/docs/."