Dynamic Tiled Map Services: Supporting Query-Based Visualization of Large-scale Raster Geospatial Data

Jianting Zhang
Department of Computer Science
City College of New York
138 Convent Avenue,New York, NY 10031
jzhang@cs.ccny.cuny.edu

Simin You
Department of Computer Science
City University of New York Graduate Center
365 Fifth Avenue, New York, NY, 10006
syou@gc.cuny.edu

ABSTRACT

Query based visual explorations of raster geospatial data plays an important role in stimulating scientific hypothesis and subsequently seeking casual relationships. While it is desirable to enable visual explorations of large-scale raster geospatial data in a Web environment, improving the end-to-end performance between query backend and the client applications remains a challenging technical issue. Techniques for providing tiled map services that are adopted by major commercial Internet maps APIs have been successful in handling static geospatial data. Motivated by the practical needs of supporting query-based visual explorations in a Web environment, we have proposed a dynamic tiled map services approach that integrates and extends existing Web-based standards and best practices in serving tiled images for static raster geospatial data. The approach includes quadtree-based indexing and query processing at the server side and a middleware to efficiently convert quadrants of dynamic query results into tiled images. A prototype system has been developed to demonstrate the feasibility of the proposed approach. Experimental results have showed that the prototype system achieves an end-to-end performance in the order of sub-second for 1024*1024 pixels display area consisting of multiple tiles.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Spatial Databases and GIS; H.2.4 [**Systems**]

General Terms

Design, Experimentation

Keywords

geospatial data, Visual Exploration, Tiled Map, Web Services

1. INTRODUCTION

Advances in geospatial technologies, especially space-borne satellite remote sensing and large-scale environmental modeling outputs, have generated large amounts of raster geospatial data at the ever increasing spatial, temporal and thematic resolutions. For example, the next generation geostationary weather satellite GOES-R (whose first lunch is scheduled in 2015) will improve the current generation weather satellite by 3, 4 and 5 times with respect to spectral, spatial and temporal resolutions [1]. At a spatial resolution of 2 km which is roughly 1 arc-minute at equator, the number of raster cells for the global would amount to (360*60)*(180*60) = 233,280,000, i.e., nearly a quarter of a billion. The number of raster cells is 16 times larger, i.e., 4 billions, for its 0.6μm band where the spatial resolution is 4 times finer. With a temporal resolution of 5 minutes, there are 288 coverages everyday for each of its 16 bands. Numerous efforts have been carried out to store, access, disseminate and visualize such large-scale geospatial raster datasets, including NASA, NOAA and standardization organizations such as Open Geospatial Consortium (OGC[2]). More specifically, OGC Web services, such as Web Map Services (WMS[3]) and Web Coverage Services (WCS[4]) have been widely adopted for accessing and visualizing archived data. The recently approved OGC Web Coverage Process Services (WCPS) standard [5][6] has provided a comprehensive framework to process the data in a Web-based distributed computing environment. However, we argue that WMS and WCS are mostly suitable for archived static data where no further processing is needed except overlaying layers for WMS and sub-setting for WCS. In this study, we are interested in visual explorations of large-scale raster geospatial data in the Web environment that involves query-driven visualization. Our goal is to investigate how OGC standards can be extended and integrated to facilitate visual explorations. Visual explorations are becoming increasingly important in exploratory analysis of geo-referenced gridded environmental data which is often the first step towards stimulating sound scientific hypothesis and seeking causal relationships.

We observed that while WCS provides only limited query-related capabilities other than subsetting, as a standard, WCPS is expressive enough to allow virtually any processing of raster data, including finding Region of Interests, or ROI-finding queries. While the formal definition will be given in section 2, an example, such as finding regions that have temperature between [t1,t2) and precipitation are between [p1,p2), would be sufficient to understand the nature of the query. Since t1, t2 and p1, p2 are provided by users dynamically, the query results are dynamic as well. While it is not difficult to configure WCPS to handle such ROI-based queries, how to return the query results in a format that can be visualized at the client applications efficiently and effectively remains unspecified in neither WCS nor WCPS. An intuitive solution would be to output the query results for the whole spatial coverage as a binary image and then use WMS and its tiled extension [7] for the purpose. While a viable solution, there are several inherent disadvantages. First, evaluating a large number of cells could incur significant computation overheads and more importantly, writing out a large binary image representing a query result could incur con-

COM.Geo 2010, June 21-23, 2010 Washington, DC, USA

siderable disk I/Os. Both will contribute to a slow start in serving WMS requests. Second, different from archived static data, storing large amount of query results derived from different combinations of query parameters may require considerable disk spaces while the majority of the intermediate ad-hoc query results are useless to users.

Motivated by the practical needs to visualize query results in support of query-driven visualization in scientific explorations, we have proposed a dynamic tiled map services (or dynamic tiling) approach to address these issues. Instead of evaluating each individual cell, a quadtree data structure [9][10] is first built to index the original raster geospatial data. As tree data structures are good at the pruning search space, the query efficiency can be greatly improved. Instead of pre-generating query results for the whole image, we only query the quadtree on the tiles that are needed by client applications during the visual explorations. As detailed in Section 3, we can coordinate the client tile hierarchy with the quadtree data structure so that the tile images can be generated efficiently. The derived tiled images for a specific query result can be cached at the server side for efficiency purposes and the dynamic tiled map services can be used as normal tiled map services in a variety of client applications. To the best of our knowledge, we are the first to address the challenges of providing dynamic tiled map services to facilitate visualizing dynamic query results. Our specific technical contributions are as the following.

Our specific technical contributions can be summarized in three aspects:

- We have identified the practical needs to support visualizing dynamic query results of large-scale raster geospatial data in a Web environment and propose a framework combining OGC WMS and WCPS standards to provide a standard based solution.
- Second, built on our previous works on using a quadtree structure to index and query large-scale raster geospatial data, we have developed efficient techniques to query data and convert quadrants in query results into tiled images.
- Third, we have built an end-to-end system using a real dataset to demonstrate the feasibility of the proposed framework and the efficiency of the query and conversion algorithms.

The remainder of the paper is structured as follows. Section 2 introduces background and related works. Section 3 presents the prototype system architecture and the implementations of tile-based query and conversion algorithms as well as other components. Section 4 is the experiments to demonstrate the efficiency of the proposed approach and effectiveness of the system implementation. Finally, Section 5 concludes the paper and outlines future research directions.

2. BACKGROUND AND RELATED WORK

As the spectral, spatial and temporal resolutions are getting increasingly finer and the data volumes are getting increasingly larger, it is unrealistic to ask scientists to manually examine these datasets which is both time consuming and error-prone. It is necessary to develop indexing and query processing techniques to find subsets of the raster geospatial data efficiently based on certain query criteria in a way similar to relational databases. Quite a few previous works have been reported towards this direction [11][12][13][14][14][16]. However, most of existing works focus on data management and dynamic query processing inside databases without considering how the query results can be efficiently and effectively delivered to applications in distributed computing environments, especially in a Web environment. Visualizing query results of raster geospatial data in Web browsers is especially technical challenging since Web browsers usually support a limited number of data types natively and most of them are text and images.

Tiled map service techniques are gaining increasingly popular in delivering large-scale geospatial data over the Web [17]. Leading Internet based map providers, such as Google Map [18], Yahoo Map [19] and Microsoft Bing Map [20], are providing tiled maps so that they can be displayed in various client applications efficiently. In the GIS community, major commercial and open source software packages, such as ArcGIS Server [21] and MapServer[22]/TileCache[23] have provided such techniques to publish geospatial data as tiled maps. Existing tiled map service techniques require the underlying maps and images drawn with a certain legend/style exist before they are subdivided hierarchically into tiles. While generally this is not a problem when the tiled maps serve mostly for providing background information (i.e., base map), the techniques are not suitable for scientific explorations where scientists need to identify regions of interests based on certain domain criteria and then subsequently focus on identified regions for further investigations. Unlike consumer electronic maps that use only a limited number of colors to represent a limited number of land cover types, most scientific raster geospatial data are numeric and can have a large number of distinct values. Since human eyes can only effectively distinguish a limited number of colors at a time, it is inappropriate to generate static maps based on raster geospatial data using pre-defined coloring scales and mapping styles. Techniques that support tiled map services for dynamic query results, or dynamic tiled map services, are more suitable in applications such as query driven visual explorations.

We next formally define the ROI-based query and discuss the works on encoding/decoding bi-level images in the context of supporting Web-based scientific explorations of large-scale raster geospatial data. Given a set of rasters representing environmental variables $\{F_i|0 \leq i < n\}$ over a spatial domain D whose value ranges are $\{V_i^H\}$ and $\{V_i^L\}$ respectively, a ROI finding query Q identifies regions in D whose cells C_j satisfy the compound condition $\left\{C_j|V_{1j} \in [V_1^{QL}, V_{1j}^{QH}] \textbf{ op } ... \textbf{ op } V_{kj} \in [V_k^{QL}, V_k^{QH}]\right\}$ where op can be either conjunctive and disjunctive and $0 < k < n$. V_i^{QL} and V_i^{QH} represent the lower and high bounds of query Q for variable i. It is clear that when all the cells are evaluated individually, the results would be a bi-level image with each pixel being 1/0 to indicate whether the corresponding raster cell satisfies the compound query criteria or not. Quite some works have been reported on encoding and decoding bi-level images [24][25][26][27][28][29]. While some of them achieve higher compression rates than popular image formats such as JPEG and PNG, they are not supported by mainstream Web browsers directly. Since it requires considerable coding efforts to implement such encoders and decoders, we have decided not to use such specialized algorithms in our applications. In addition, we have observed that typically users can tolerant delays up to a few seconds from the time a query is issued to the server to the time query results are displayed in the Web browser. A decent enoder and network transmit rate will provide a reasonable performance in the whole process. That being said, developing more efficient encoding algorithms specifically for resulting binary

images representing query results and decoding plug-ins for Web browsers will certainly improve the system response times. They are left for our future work.

In the past few years, there have been significant developments in terms of architectures and standards that allow uniform accesses to large-scale geospatial data. OGC WMS, WCS and WCPS standards have been designed to visualize, access and process raster geospatial data. Fig. 1 provides an overview of OGC service stack and operations for these three services. While OGC standards are designed to handle geo-referenced data, the Open Source Project for Network Data Access Protocol (OPeNDAP[30]) standards describe the management of multi-dimensional array data that are not necessarily geo-referenced. OPeNDAP has been implemented in UCAR Unidata Thematic Realtime Environmental Distributed Data Services (THREDDS) Data Server (TDS [31]) and the Hyrax server ([32]) and have gained increasing popularity in the oceanographic and atmospheric communities[33][34]. In the recent releases of TDS, a WMS component has been provided to visualize remote OPeNDAP compliant data. Similar to WCS, while OPeNDAP support sub-setting multidimensional array through its DODS request, it does not support ROI-finding queries. Our work in providing dynamic tilted map services for ROI-finding query results extends static WMS services (including its tiled extension) and can be beneficial to applications based on WCS, WCPS and OPeNDAP DODS services for the following reason. It is common that users are only interested in part of a dataset that satisfy certain criteria. However, extensive query-based visual explorations are needed before the regions of interests can be identified. Our work helps users to identify such regions and subsequently to use WCS, WCPS and OPeNDAP DODS standards to retrieve data in these regions. In this sense, our work integrates visual display based standard (WMS) and raw data access based standard (WCS and OPeNDAP DODS) by efficiently realizing the WCPS standard for visual exploration based applications. The dynamic tiling approach avoids fully materializing query results and the system performances can thus be improved.

3. THE PROPOSED SOLUTION

In this section, we first introduce the architecture of the prototype system we have developed and we then introduce the implementation details of each individual component. The algorithm to convert query results into tile images is presented in Section 3.3 when the image generation middleware component is introduced.

3.1 System Architecture

The architecture of the prototype system is illustrated in Fig. 2 which includes three major components. The client module is responsible for interacting with users and formulating query strings representing desired image tiles and send the request to the middleware. The client module also assembles returned tile images and visualizes the base map and tiled images properly. The middleware forwards the query strings to the query processing server and convert the query results into tiled images before returning the images to the client. The middleware is also responsible for providing a caching mechanism so that the same tile image requests can be answered immediately from its cache without performing query and format conversion. We have used commercial ArcGIS server from ESRI to provide base map services. We also use ArcGIS Flex APIs to develop the client application. However, we note that we can easily replace ArcGIS server with any other commercial and open source software to provide dynamic (e.g. MapServer) and tiled (e.g., TileCache) map services. Similarly we can use Google Map API or OpenLayers [35] to visualize base maps and tiled images in Web browsers. We next introduce the implementation details of the components according to the order as numbered in Fig. 2.

3.2 Query Processing Server

We build a quadtree based index for each dataset offline. For the Binned Min-Max Quadtree (BMMQ-Tree) discussed in [8], we first re-scale the raw data into a certain number of bins. Each node of a BMMQ-Tree is associated with the minimum and maximum value of the pixels under the quadrant corresponding to the quadtree node. The purpose of binning is to reduce the index size so that it can fit into the main memory of the server for fast query processing. Binning is not necessary for server machines with large memory or if datasets are relatively small. The query processing server is implemented in C with socket programming. The procedure of processing a ROI query on a BMMQ-Tree for arbitrary spatial and value ranges are discussed in [8]. We next introduce the query processing algorithm for tile-based requests using the example illustrated in Fig. 3.

The algorithm consists of two consecutive steps. The first step is to locate the quadtree node representing the tile being requested based on the common spatial tessellation of both the quadtree and the tile hierarchy. For tile (x,y) at level L, we repetitively divide x and y by 2 for L times. The remainders of the divisions in the reverse orders tell which child node to follow. Assuming the size of the tile (typically the same across different levels) is $s = 2^k$, then starting from the quadtree node we just locate, we go down the quadtree up to k level to examine whether each leaf node under the located quadtree node satisfy the query value range. Note that the min/max values associated with the intermediate quadtree nodes can be used to prune the search space, i.e., if the query value range does not intersect with the range of the minimum/maximum value pair, then all the leaf nodes under the node can be safely pruned. The information associated with the resulting leaf quadtree nodes can be used to generate tile images in the middleware component to be detailed in the next subsection. Note that we could have combined the middleware and the query processing server and generate tile images directly. We choose to separate the two parts so that we can reuse the caching system available in the open source THREDDS Data Server (TDS) package [31] that is written in a different language (Java). In addition, the separation may also improve the scalability of the prototype system by allowing middleware to query multiple query processing servers if multiple copies of such servers are deployed on multiple machines.

3.3 Image Generation and Caching Middleware

The middleware component accepts client tile-based requests, formulate proper queries, send them to the query processing server and convert the resulting quadtree nodes into tiled images. Assuming the tile request (x,y,L) and the length of a tile image is $s = 2^k$, then the query processing should stop at level L+k as the quadrants at the level represent a single pixel in the tile image. In our implementation, k is passed to the query processing server as a parameter. The Java ImageIO package available in its development kit provides a convenient way to draw quadrants on a tile image. While both JPEG and PNG are supported by the Java ImageIO package, we choose to use PNG as it supports transparency.

Assuming that the quadtree nodes are given in the form of (r,c,l)

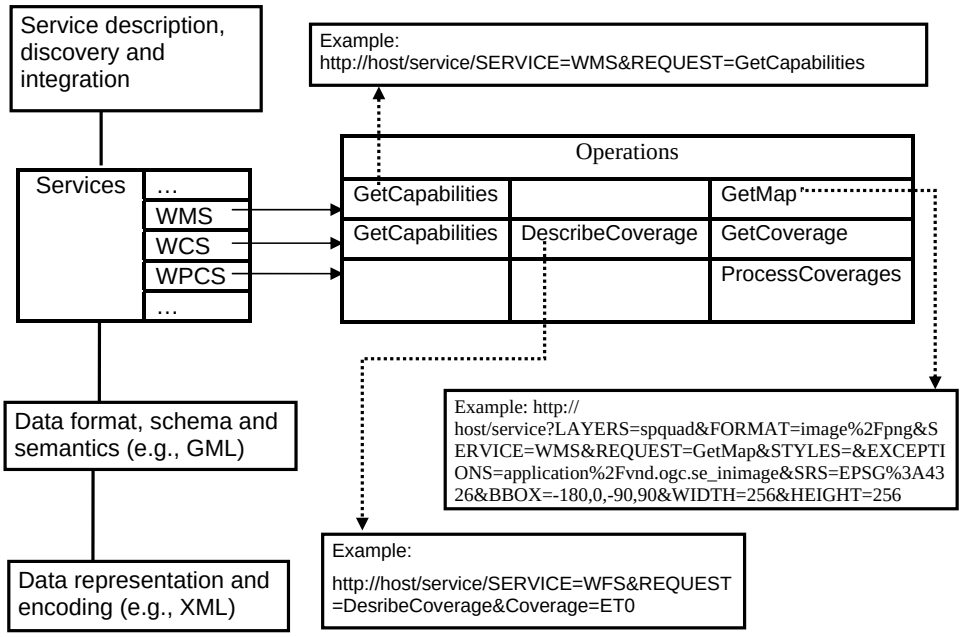

Figure 1: OGC Service Stack and Operations for WMS, WCS, and WCPS.

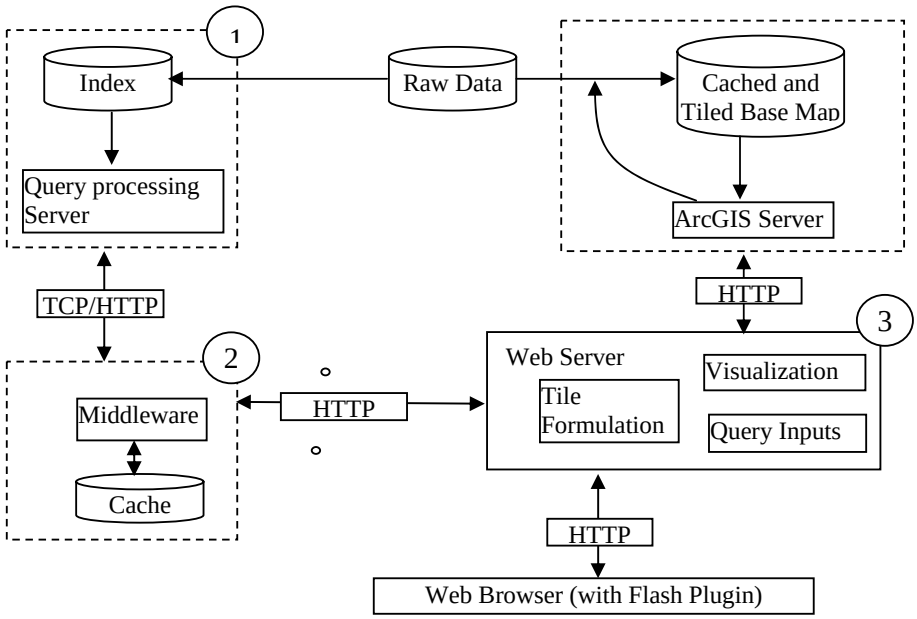

Figure 2: Prototype System Architecture.

triples, we can simply draw a square starting at (r,c) with length $2^{k-(l-L)}$ on the tile image. Since l<=k+L, the length is guaranteed to be equal or larger than 1. On the other hand, when k+L is larger than the maximum level of the quadtree, i.e., $k + L > max\_quad\_lev$, the resulting tile images with larger L values do not convey additional information. Instead, it is simply digital magnifications. As such, the largest tile level should be less than $max\_quad\_lev - k + c$ where c is a small number for digital magnification and can be as small as 0.

The generated tile images can be written to a file cache system before sending the byte stream to the client. In our implementation, we have extracted relevant code from TDS source tree and use it to cache the generated tile images. While the part of code in TDS was originally designed to cache WCS and OPeNDAP requests, we have successfully modified the code to make it cache tile images. By reusing the code, we are able to log quite a few types of events that are relevant to generating and serving tile images. The code also allows automatically clean up the outdated files in the cache system. The well designed, implemented and tested code has been proven to be efficient and effective which has saved us considerable time in developing the prototype system. We have also investigated the way that MapServer generates images for WMS requests and we found that the Java ImageIO package is easier to use than the GD package that MapServer relies on. Nevertheless, as discussed above, we believe that it is technically feasible to combine the query processing server with the middleware and serve tile images to client applications directly. The obvious advantage is reducing data communication times between the two systems. In our implementation, we co-locate the query processing server and the middleware on a same machine to minimize the data communication overheads between the two systems.

3.4 Visualization Client

The client is responsible to interact with users to accept query inputs and formulate query strings before sending request to the middleware. When a binned quadtree is used, the client is also responsible to quantify the input minimum and maximum query values into the corresponding bins. In this study, we use Adobe Flex programming API [38] and its ArcGIS Flex API [37] to develop the client component. Flex API, as a Rich Internet Application (RIA,[36]) programming platform, has provided rich built-in graphics user interfaces to interact with users. The free ArcGIS Flex API [37], as an extension to Adobe Flex API [38], has provided sophisticated functions to manipulate vector and raster geographical data. While we only use ArcGIS Flex API to display base maps in the image form, as reported in [8], quadrants in a query result can actually be visualized as vector geographical data. We choose to visualize the resulting quadrants as tiled images mostly for performance considerations as it is not efficient to process a large number of vector objects in the Flash plugin. Instead, performance can be greatly enhanced by handling only a handful tiled images. We extend the TiledMapServiceLayer class in ArcGIS Flex API so that the dynamic tiled images for the query results can be visualized in the client just like static tiled images. We overwrite the getTileURL function by appending the query value range to a query string to be sent to the middleware. We also overwrite the buildTileInfo function to set the correct resolution and scale for each of the tile levels so that the map scale bar can be shown correctly. Fig. 4 shows a snapshot of mosaicing the dynmaic tiled images of a query result overlaying with the raw data using the January global precipitation data as detailed in the next section.

4. EXPERIMENTS AND EVALUATION

The January global precipitation data from the WorldClim [39][40] is used for demonstration purposes. The value range of the dataset is [0,1003] in millimeters. The dataset has a spatial resolution of 30 arc-seconds which is approximately 1km at the equator. The fine resolution leads to 432000*216000 raster cells for the dataset. We have set the maximum level of the BMMQ-Tree to 16 as $2^{16} = 65536$ is larger than 432000. In the experiments, k is set to 8 which makes a 256 pixels by 256 pixels tile image. Studies reported in [8] also showed that visualizing resulting quadrants as squares(vector) often incur serious performance issues at the client side when the number of quadrants are beyond the order of hundreds. The work reported in this paper serves as a client side performance improvement to the work reported in [8]. We will show that the proposed dynamic tiled map services approach improves the client side performances significantly by reporting the measured statistics of end-to-end average query response times, tiled image generation times and tiled image sizes for all non-blank tiled images at the different levels under two realistic query value ranges. A Dell Precision T5400 workstation with 16G main memory is used to host both the query processing server and the middleware. GNU g++ 4.1.2 is used to compile the query processing server without further optimization. SUN JDK1.6.0 is used to compile and run the middleware in a batch mode for all non-blank tiled images.

For a quadtree of level L and a tile image of size $s = 2^k$, it is not difficult to calculate that the number of tiles at the lowest tile level is $N = 2^{(L-k)} * 2^{(L-K)} = 2^{(2L-2k)}$. As L is set to 16 and k is set to 8 in our experiments, even without considering the magnification factor (i.e., c is set to 0) , N would be $2^{(32-16)} = 2^{16} = 65536$. Based on this number, the total number of tile images is $N + \frac{N}{4} + \frac{N}{4^2} + \frac{N}{4^3} + ... + 1 = \frac{4N}{3}$ which is more than 87,000, a considerably large number. However, we note that the actual data occupies only a fraction of the space covered by the quadtree. More precisely, the space occupancy rate is $(43200/65536) * (21600/65536) = 21.7\%$. As the dataset has no data on the Earth surface covered by oceans and the percentage is roughly 71%, the data space occupancy rate is further reduced to about $P = 21.7\% * 29\% = 6.3\%$. The low space occupancy makes quadtree data structures much more efficient than array representations with respect to memory utilization. The unoccupied space will result no quadrants for any query range values and the tiles for the no-data regions can be represented as a generic blank image. For queries with high selectivity, we expect the numbers of needed tiles are even lower.

To verify these calculations, we have performed two of experiments. The first experiment uses the full value range of the dataset, i.e., [0,1003] as the query value range and the second experiment uses the [90,300] query value range which is the same as used in our previous study reported in [8]. As the binning mechanism is used in our query processing sever, these two value ranges translate to bin ranges of [0,25] and [19,24], respectively. Table 1 lists the numbers of non-blank tiled images, the minimum, maximum and average of the corresponding query response times , tiled image generation times and tiled image sizes (S, in bytes) at the different levels under the two query value ranges. Note that the query response times are measured in an end-to-end manner. We have collocated the middleware with the query processing server to minimize the effect of network traffic variations and thus the data communication costs between the two systems are negligible. Data communication overheads need to take into consideration when the middleware and the query processing server are distributed.

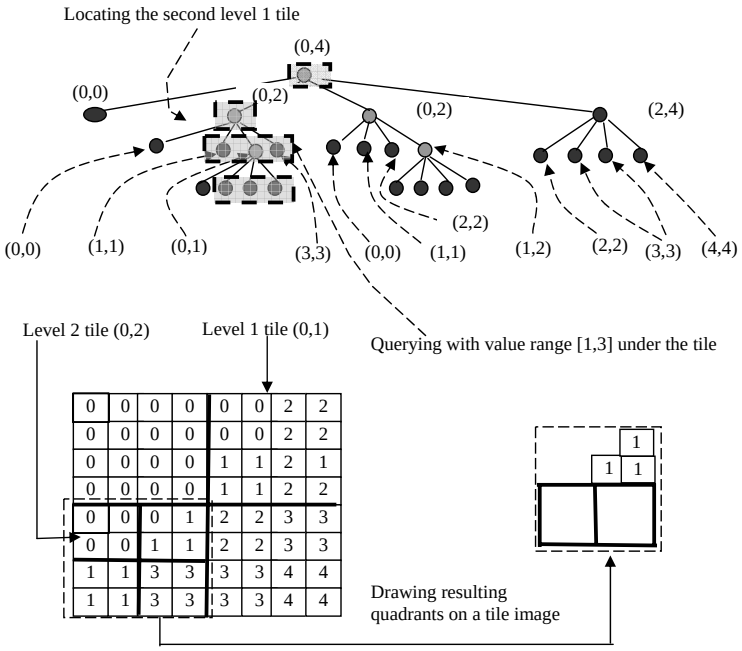

Figure 3: Processing Tile-based ROI-Finding Query and Generating Tiled Images Using Quadtree

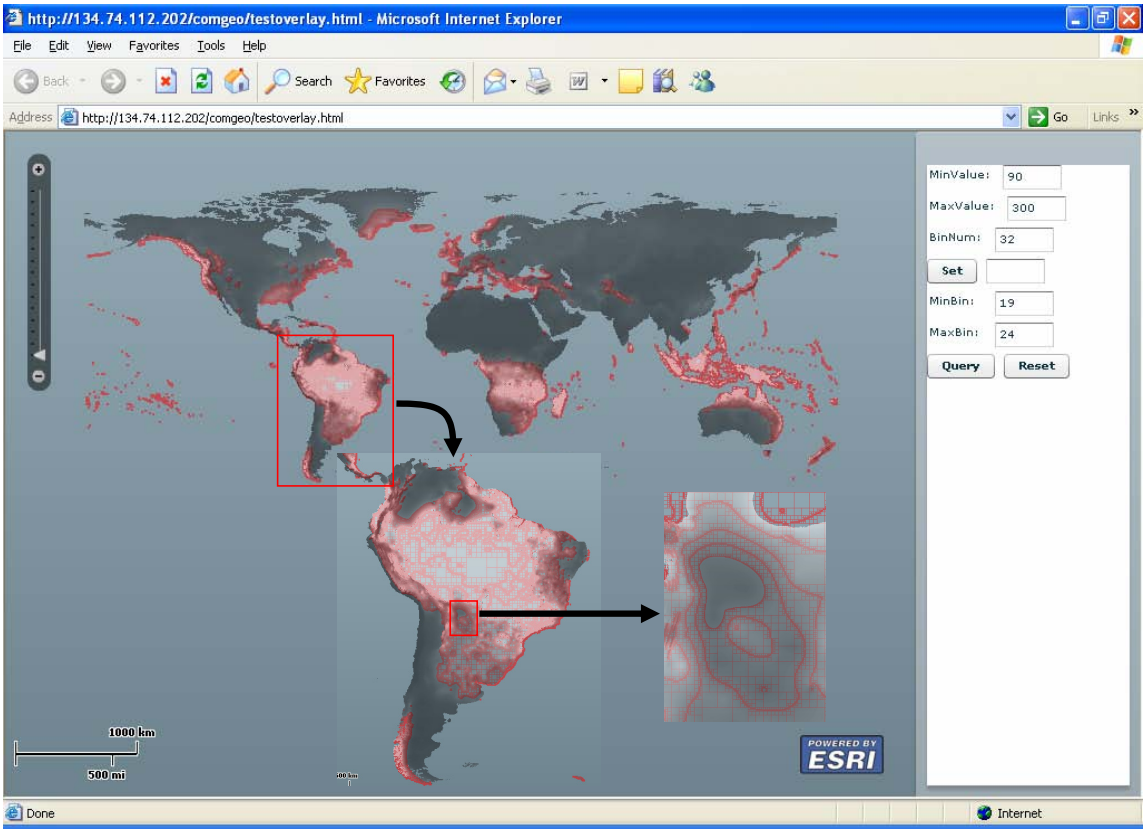

Figure 4: A Snapshot Showing the Dynmaic Tiled Images of a Query Result Overlaying with the Raw Data

Table 1: Statistics on the Query Processing Times, Tiled Image Generation Times and Tiled Image Sizes for the Two Tests

	Test1 [0,1003]			Test2 [90,300]		
	#of Tiles=6846			#of Tiles=1197		
Measurement Type	Min	Max	Avg	Min	Max	Avg
Query Response Time (QT,in milliseconds)	46	107	55.01	46	81	51.80
Tile Generation Time (GT,in milliseconds)	0	199	11.46	0	142	5.10
Tile Image Size (S, in bytes)	339	8275	3076	336	8275	2071

From Table 1 we can see that both the average query processing times and the tiled image generation times are around 50 milliseconds on both tests. The sizes of the generated image tiles vary from a few hundred bytes to a few thousands bytes with averages around 2-3 kilobytes for both tests. Assuming that the data communication rates between the middleware and clients are in the order of a few hundreds kilobytes to a few megabytes, then the image tile can be transmitted to clients (i.e.,image transmission time, TT) in the order of 1-10 milliseconds. We conservatively assume TT=10 milliseconds in typical network traffic scenarios. Also assuming the client display area is 1024*1024 pixels, then the number of active tiles is about 4*4=16. This lead to the total response times in the order of (QT+GT+TT)*16 = (50+10+10)*16=1120 milliseconds. Considering that users usually focus on the tiles centered at the display area, the response time for requesting key tiles should be less than half a second. We thus conclude that the dynamic tiled map services implemented in our prototype system has achieved the desired performance for online interactive visual exploration based applications.

5. CONCLUSIONS AND ONGOING WORK

In this study, we address the technical challenges of facilitating visualization of dynamic query results on large-scale raster geospatial data in a distributed computing environment by integrating open Web-based standards, including OGC WMS and its tile extension, OGC WCS and OPeNDAP DDS, from an architectural view. We have designed algorithms to efficiently perform tile-based queries on quadtrees and to convert quadrant-based query results into tiled images. An end-to-end prototype has been developed to demonstrate the feasibility of the proposed dynamic tiled map services approach. Experiments results have showed that the prototype system achieves an end-to-end performance in the order of sub-second for $1024 * 1024$ pixels display area using 16 tiles.

The reported work leads to several future work directions. First, while our query processing server implements a typical but specialized coverage operation (i.e., ROI-finding), we did not follow WCPS syntax strictly. We would like to revise our query processing server by following the WCPS standard to achieve better interoperability. This will not only standardize the communications between the middleware and the query processing server to serve tiled images, but also allows other applications to use our WCPS-compliant query processing server in an interoperable manner. Second, in this study we used a main-memory based quadtree to speed query processing. While the binning strategy significantly reduced memory consumption, it also brings false positives. Although this may not necessarily be a disadvantage as quite often aggregated and generalized information are easier to interpret, we would like to indicate the uncertainty information in the tiled images. The new improvement is likely to increase end-to-end response times and proper tradeoffs might be made. Finally we would like to address the technical challenges of visual scientific explorations of the same large-scale raster geospatial data on mobile handheld devices using the same dynamic tiled map services strategy but accommodate issues such as low computation power, small screen, short battery life and perhaps more importantly, low-rate and unreliable wireless connections.

6. REFERENCES

[1] Schmit, T. J., J. Li, et al.: The GOES-R Advanced Baseline Imager and the Continuation of Current Sounder Products. Journal of Applied Meteorology and Climatology 47(10): 2696-2711, 2008.

[2] Open Geospatial Consortium Inc. . http://www.openGeospatial.org

[3] OpenGIS Web Map Server Implementation Specification http://www.openGeospatial.org/standards/wms .

[4] Web Coverage Service (WCS) http://www.openGeospatial.org/standards/wcs .

[5] Web Coverage Processing Service (WCPS) http://www.openGeospatial.org/standards/wcps .

[6] Baumann, P: Designing A Geoscientific Request Language - A Database Approach. Scientific and Statistical Database Management Conference(SSDBM) 2009, New Orleans, USA, June 2-4, 2009

[7] OpenGIS Tiled WMS Discussion Paper http://portal.opengeospatial.org/files/?artifact_id=23206.

[8] Zhang J., You S., Supporting Web-based Visual Exploration of Large-Scale Raster Geospatial Data Using Binned Min-Max Quadtree To appear in the Proceedings of Scientific and Statistical Database Management Conference(SSDBM)'2010, Heidelberg,Germany, June 29-July 2, 2010.Lecture Notes in Computer Science (LNCS),Springer.

[9] Gaede, V., Gunther, O.: Multidimensional access methods. ACM Computing Surveys **30**(2) (1998) 170–231

[10] Samet, H.: Foundations of Multidimensional and Metric Data Structures. Morgan Kaufmann Publishers Inc. (2005)

[11] Wu, K., Koegler, W., Chen, J., Shoshani, A.: Using bitmap index for interactive exploration of large datasets. In: Scientific and Statistical Database Management Conference(SSDBM)'03. (2003) 65–74

[12] Stockinger, K., Shalf, J., Wu, K., Bethel, E.W.: Query-driven visualization of large data sets. In: IEEE Visualization. (2005) 22

[13] Glatter, M., Mollenhour, C., Huang, J., Gao, J.Z.: Scalable data servers for large multivariate volume visualization. IEEE Transactions on Visualization and Computer Graphics (TVCG) **12**(5) (2006) 1291–1298

[14] Kendall, W., Glatter, M., et al.: Terascale data organization for discovering multivariate climatic trends. In: SuperComputing'09. (2009) 1–12

[15] Fuchs, R., Hauser, H.: Visualization of multi-variate scientific data. Computer Graphics Forum **28**(6) (2009) 1670–1690

[16] Sinha, R.R., Winslett, M., Wu, K.: Finding regions of interest in large scientific datasets. In: Scientific and Statistical Database Management Conference(SSDBM)'09. (2009) 130–147
[17] Chow, T. E. The Potential of Maps APIs for Internet GIS Applications. Transactions in GIS , 12(2): 179-191,2008
[18] Google Map http://maps.google.com
[19] Yahoo Map http://maps.yahoo.com/
[20] Microsoft Bing Map http://www.bing.com/maps/
[21] ESRI ArcGIS Server http://www.esri.com/software/arcgis/arcgisserver
[22] MapServer http://mapserver.org/
[23] TileCache http://tilecache.org/
[24] Yuen, H. and L. Hanzo : Block-run run-length coding of handwriting and bilevel graphics based on quadtree segmentation. Pattern Recognition Letters 18(2): 187-191, 1997.
[25] Ageenko, E. and P. Franti : Lossless compression of large binary images in digital spatial libraries. Computers and Graphics-Uk 24(1): 91-98, 2000.
[26] Reavy, M. D. and C. G. Boncelet : An algorithm for compression of bilevel images. IEEE Transactions on Image Processing 10(5): 669-676, 2001.
[27] Tsai, Y.-C., M. S. Lee, et al.: A Quad-Tree Decomposition Approach to Cartoon Image Compression. 2006 IEEE 8th Workshop on Multimedia Signal Processing.
[28] Ryan, O.: Runlength-Based Processing Methods for Low Bit-depth Images. IEEE Transactions on Image Processing 18(9): 2048-2058, 2009.
[29] Raguram, R., M. W. Marcellin, et al: Improved resolution scalability for bilevel image data in JPEG2000. IEEE Transactions on Image Processing 18(4): 774-82, 2009.
[30] OPeNDAP: Open-source Project for a Network Data Access Protocol http://opendap.org/
[31] THREDDS Data Server http://www.unidata.ucar.edu/projects/THREDDS/
[32] Hyrax Server http://opendap.org/download/hyrax.html
[33] Woolf, A., Haines, K.,Liu, C. L., : A web service model for climate data access on the grid. International Journal of High Performance Computing Applications, 17(3), 281-295,2003.
[34] Rutledge, G. K., Alpert, J.,Ebisuzaki, W., : Nomads - A climate and weather model archive at the National Oceanic and Atmospheric Administration. Bulletin of the American Meteorological Society, 87(3), 327-341, 2006.
[35] OpenLayers http://openlayers.org/
[36] Rich Internet Application (RIA) http://en.wikipedia.org/wiki/Rich_Internet_application
[37] ArcGIS Flex API http://resources.esri.com/arcgisserver/apis/flex/
[38] Adobe Flex http://livedocs.adobe.com/flex/
[39] WorldClim Global Climate Data http://www.worldclim.org/current
[40] Hijmans, R.J., Cameron, S.E., et al: Very high resolution interpolated climate surfaces for global land areas. Int.J. of Climatology 25(15), 1965-1978, 2005