Advances in Engineering Research, volume 119
2nd International Conference on Automatic Control and Information Engineering (ICACIE 2017)

Design and Implementation Method of a High Performance Web Image Browsing Server

Genyuan Zhang
School of Electronic Information
Zhejiang University of Media and Communications
Hangzhou, China

Yongjie Pan
Zhejiang Radio and Television Group
Hangzhou, China

Yuebo Ying
Zhejiang Radio and Television Group
Hangzhou, China

***Abstract*—It is an important how to provide effective cost, high performance image block query and image block analysis for the design of WEB image server. To solve the above problems, this paper proposes a novel method to design and implement WEB image server. Like most modern WEB image server, the server stores the pre rendered image blocks with different zoom scale on a web server, and then the image blocks is loaded in the client browser. At the same time, this paper describes in details the key issues of web server design and the methods of image block query, image block prediction and image block analysis. The prototype of the above method is implemented, and the results of the benchmark results also show the effectiveness of the method.**

Keywords—WEB image server, rasterization, image block analysis, image block prediction

I. INTRODUCTION

In recent years, WEB image sever has been used widely. The network has become a major platform for accessing, processing image block. Classifying information according to some query criteria (including image block prediction) is the essence for the image server. Services derived from modern WEB image server products are cpu-intensive and require memory types[1,2,3,4]. For WEB image server design, how to provide effective cost, high performance image block query and image block analysis implementation is an important problem[5,6,7]. In this paper, we propose a novel grating method to design and implement the WEB image server. At the same time, this paper describes in details the key issues of web server design and the methods of image block query, image block prediction and image block analysis. The prototype of the above method is implemented, and the results of the benchmark results also show the effectiveness of the method.

II. SYSTEM STRUCTURE AND FRAMEWORK

The diagram below (Fig.1) shows the design of the framework and the network image server. On the client side, the Flex client enters a Rich Internet Application (RIA) scenario by creating a programmatic approach to make flash applications to use its ubiquitous cross-platform flash player. This programmatic approach uses the core language of Flex: XML's template language (MXML) and its scripting language. Flex integrates J2EE by using an additional server-side layer called real-time loop data service, called the BlazeDS, which is deployed on the application server. This extra layer on the server can help Flex applications call Java classes directly and communicate with Java classes, so that they can be invoked and accessed by Flex applications. The requests being preprocessed and passed by the small service program are processed by the corresponding Java modules (image block identifiers, image block queries, and so on) that are backed up by c + + implementation components. The choice of c + + language as the implementation language is due to its platform independence and high performance.

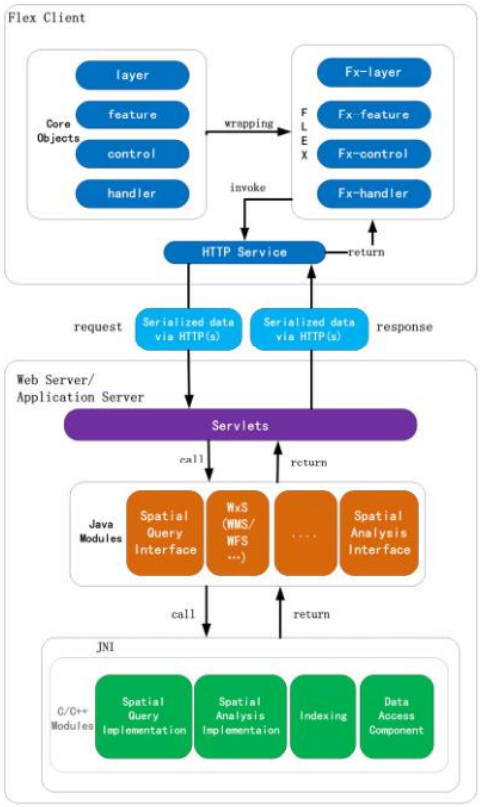

Fig. 1. System framework and component

Similar to now most of the image server is implemented, we put forward the system in the network is also stored on the server different scaling the size of the pretreatment of image block (and corresponding index files, the function of index files in the back), then the image block will be added to the client browser together. The image can be composed of a single layer or some combination layer used to form a base image or background image. The image cache is used to obtain complex image symbols and better drawing quality. Prerendering is also very useful for improving performance on the server by sacrificing some disk image blocks. An interesting and meaningful finding is that the cached image blocks can be not only the image of vector data, but also the rasterization of specific geometric shapes.

It is not difficult to understand that Rasterization approximation is more similar to the image block boundary and shape feature than the MBR. At the same time, Rasterization approximation can better judge the relationship of image block. For example, if a grid that is represented by a matrix with feature 1 and the same grid that is represented by a matrix with feature 2overlap, so the Rasterization approximation can easily determine the overlap of these two grids. The overlapping grid is the grating approximation of feature intersection. In order to vectorize the Rasterization approach; this method can obtain the operation such as intersection and erasure. These operations become the basis of image block analysis in the network. The details of this algorithm are explained in section 4.

III. INDEXING TECHNIQUES

To get a block image, we used an Anti-Grain Geometry (AGG). AGG is an open source graphics library writed by industrial standard C+ +. Basically, AGG can be seen as a platform independent, high-performance, lightweight rendering engine. This rendering engine can generate pixel images in memory based on vector data. This rendering engine can render an anti-aliasing arbitrary polygon or line, and can guarantee the accuracy of the sub-pixel.

The rasterization method of traditional computer graphics is limited to geometrical coordinates and the characteristic information cannot be stored in the render image. If we save the feature ID in the index file for the feature ID of each pixel of the rendering image, we will be able to estimate the mapping between image pixels and feature attributes.

We can implement this idea by using the feature ID as the idea to fill the color drawing polygon. The main idea is that for each geographic feature, the ID value should be converted from decimal to the binary which the lowest 8 digits stored in the blue component。 If the binary value is more than 8 digits; its middle 8 digits should be stored in the green component. The other digits can be stored in other components in the same way. Let's take the feature id 24202 for an example to explain the process of parsing. Convert the feature ID from the decimal to the binary number (101111010001010), which has the lowest 8 digits, 8 digits, and the highest 8 digits, which should be stored in blue, green, and red components. When the whole parsing process is completed, the color of the feature ID is red: 0, green: 94, blue: 138.

In order to generate shadow images, the AGG rendering engine needs to be modified. For each render buffer unit, the feature ID is saved next to the coverage value, which records the area of the unit covered by the polygon (per percent). When the image block filter is obtained in the network image service protocol and the image block filter in the network feature protocol, the shadow image bears the role of the index file. If a user wants to know the feature attributes of the image that he/she is clicking on in the browser, the server will index the pixel location where the tap is located. This query can be completed in a shadow image on a constant time.

We did not retrieve the R tree example (down from the root node), the retrieval process in the new method is based on the grid system. It is also important to note that the generated index file allows the retrieval algorithm to eliminate the uncorrelated areas of the index image block, and only check the grid cells of the retrieved area that the user has made.

Our idea is to first assign width_tile * height_tile * 3 bytes of memory to the current index image (in this case, width_tile and height_tile are the width and height of the index image). For the point query, the coordinates of the query point need to be converted to the rows and columns in the index image (through the scan line to specify the grid cell in the memory). Then the color of the image ID point is resolved. For the region query, the specified line in the memory image should be scanned after the region row and column are determined based on the user query requirements. At the same time, several different RGB values can be calculated. So some ids that are needed can be easily converted from these color values.

IV. PREDICTION AND IMAGE COLORING ANALYSIS

Image block filters provide a traditional way to retrieve features from the server side by specifying some image block predicates(separation/intersecting, equal, external/external, high light, cross, internal/inclusive, overlap, surround box, etc.).Extended relational database management system (such as oracle image block) or geometric libraries (such as administrators and Terralib) often use CPU intensive and computational geometry that need a lot of memory to complete the image block to predict and image color. Extended relational database management system (such as oracle image block) or geometric libraries (such as administrators and Terralib) often use CPU intensive and computational geometry that need a lot of memory to complete the image block to predict and image color. We continue to use visible rasterization in image block prediction and image shading. In fact, in a modern WEB image server, the image blocks are pre-rendered or dynamically rendered when the initial client request is requested. In such a scenario, the cached image block is not only the visualization of vector data, but also the grating approximation of the specific geometry. Such images can have more accurate approximation precision than the minimum surrounding rectangle (MBR), and can also serve as the basis for image block prediction and image shading implementation.

It is not difficult to understand the two steps of image block prediction and image coloring of the raster approach. (1) Using the method mentioned in the previous section to make the grating of vector data. (2) Based on the color operation of grating and image block relation judgment.

Empty (no cross grid cell and polygon), weak (grid and polygon intersection less than or equal to 50%), strong (grid and polygon intersection is more than 50%, less than 100%), with 100% (grid and polygon intersection). Only strong × strong is certain conditions, so weak×weak, strong×weak and weak × weak are all uncertain and still require further calculation. When calculating approximate area and confidence interval, it uses mathematical expectation and probability formula to estimate which may not be the proper real number. Mentioned in the last section of the rendering engine accurately to the border of the sub pixel accuracy to record grid cell coverage area, so it can pass judgment corresponding grid cell coverage area to determine whether two polygons overlap. Known, for example, in the first layer of a cell coverage is 49%, in the same grid position of the second layer of coverage is 52%, it can determine the two layers overlap on the grid unit, because the sum of coverage of more than 100%. But if you use 4CRS, you're not sure because it can't handle weak ×weak conditions(not talk about weak × weak conditions). In terms of side effects, the rendering engine can store feature attribute information in a cell structure that provides more useful leads for image overlap (multi-deformation ids, etc.)We know that there are two input feature sets A and B to determine whether these two layers overlap. Returns the corresponding polygon ID if it is true (the overlap may be treated as a filter condition). For image overlay analysis, the majority of the requirements mentioned above (the feature number of the result layer, the ID and area of each feature of the resulting layer) can be satisfied. But if the user wants to get the geometry of each feature, we need tovectorize the rasterization of the result layer.

V. SERVER-SIDE CACHING SYSTEM

The server-side caching system has a different way of organizing the cached image data. This way of organization first is described in this section, and then we'll introduce a memory-resident data structure and an index structure, through these structures can accelerate the cache file image retrieval process.

In this new cache system, each zoom level image block represents four image blocks at the next zoom level. To reduce the number of file data in the cache. Instead of storing each image file in a separate file, we store all the image blocks of the same layer in the same file. Therefore, for there are 16 levels and 16 zoom level image service, only have 16 files in the cache, instead of 2^{36} files, if every image file stored separately in each file, according to the formula (1), when p is 1.Each image file in the cache is given an identifier to indicate it is in this layer. The length of this identification number can be obtained by formula (3).

$$L_{Tid}=\lg\left(\sum_{i=p}^{p+n-1} 4^i\right) \quad (1)$$

In formula (1), L_{Tid} represents the shortest bit length of the identification number. N represents the number of zoom levels, p represents the first scaling level. The identification number is generated based on the zoom level and the image block sequence number at the corresponding zoom level. Fig. 2 describes how to determine the identification number of each image block in the cache.

Fig. 2 (a) describes an image server that provides two zoom levels, with 4 image blocks in the zoom level 1, and 16 image blocks in zoom level 2. At the lower zoom level of the image block (zoom level 1) the higher image block (zoom level 2) has a smaller identification number. In the same zoom level image block, the identification number of each image block is given in the z-axis order.

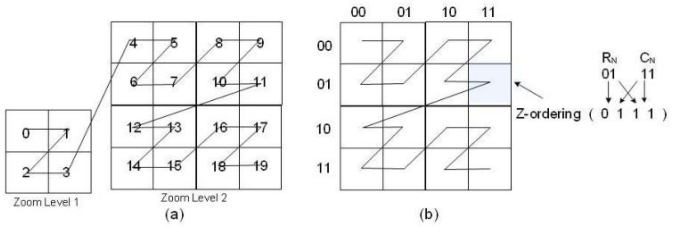

Fig. 2. Identifying number for image tiles

In the entire image at a scaling level, given the number of rows and columns of animage block, then the identification number of this image block can be obtained by formula (2).

$$TID(Z_L,R_N,C_N) = \sum_{i=p-1}^{p+Z_L-2} 4^i + Bit\_Shuffling(R_N,C_N)-1 \quad (2)$$

In formula (2), Z_L,R_N,C_N respectively represent the scale level, number of rows and columns. The TID is a function of the image block identification number that calculates a given zoom level, row number, column number, and Bit__shuffling is a function that gets the z-axis sequence code based on the number of rows and columns. As depicted in Fig.2 (b), the Bit__shuffling function intersected the number of rows and the number of columns in order to get the binary encoding of the z-axis sequence of the image block.

To improve the performance of the network image service, especially when the customer wants to switch from one zoom level to another lower zoom level. We store a pointer to each image block that does not belong to the lowest scaling level and to the next lower zoom level containing four corresponding block of images. The domain name of TN is the number of blocks of images stored in the block of files. A domain named ID is the[L_{Tid}]bit length representation of the block of images stored in the file block. A man named Pos domain ID is 2 bytes said referring to the Pos before the image of the location of the block of data, one is called the PN domain is said to contain the next level 1 byte length corresponding to the image block file pointer number, the location of the image block by Pos domain in PN domain before said. The data portion of the image block is divided into two parts. The front is the pointer to the next level image block, and the image block itself.

VI. BENCHMARK AND EXPERIMENTAL RESULTS

The operating environment is the IBM notebook T43 model, the pentium M760 (2.0 GHZ) processor, and 2GB of ram (DDR2). The speed of the hard disk is 5400RPM. The operating system is Windows xp (service pack 3).

In this experiment, the first data set D is a random polygon. As a million operations (overlap and inclusion), the comparison of the average time between a transaction of the method and the corresponding module in the geometry engine operating system. The experiment and data analysis show that the

efficiency of the image block prediction operation is not improved greatly for the sparse dataset, but for the dense graph, the time cost can be reduced by 60%.

The result of the point query performance indicates that, regardless of which data set is selected, the method proposed by us shows better performance than the R tree method. When retrieving an object that is contained to a given point, our algorithm finds the rows and columns on the index image based on its geographic coordinates, and can parse the color of the grid of the real ID. Constant in the computational complexity. By contrast, the R tree method starts to query the data from the root, which reduces the performance of the query because there are many redundant query paths.

VII. Summary and Prospect

This paper presents a rasterization approach based on high performance WEB image server design and implementation. The key problems of web server design and image block query, image block prediction, solution method of image block analysis and server side caching mechanism are described in detail. We have also proposed a prototype framework for using the above methods, and the relevant baseline result details also demonstrate the effectiveness of our approach. This paper is

supported by the project 'Research and development of real-time rendering algorithms for large 3d data for mobile terminal **(project No**: 2017C31088)'from Zhejiang science and technology department.

References

[1] P. A. Longley, M. F. Goodchild, D. J. Maguire and D. W. Rhind,Geographical Information Systems: Principles, Techniques,Management, and Applications, 2nd ed., New York: Wiley, 2005,pp.97-99.

[2] T. Ahmed, J. Hirschi and F. Abid. Flex3 in Action. Manning.2009.

[3] http://www.antigrain.com/doc/index.html.

[4] J. D. Foley, A. V. Dam, S. K. Feiner and J. F. Hughes, ComputerGraphics: Principles and Practice in C, 2nd ed., Addison-WesleyProfessional, 1996, pp.67-72.

[5] L. G. Azevedo, G. Zimbrão and J. M. Souza, "Approximate QueryProcessing in Spatial Databases Using Raster Signatures," VIIIBrazilian Symposium on GeoInformatics, Campos do Jordão, Brazil,November 19-22, 2006, INPE, p.3-17.

[6] Hui Dong, Zhenlin Cheng and Jinyun Fang, "One rasterization approachalgorithm for high performance map overlay," The 17 InternationalConference on Geoinformatics 2009, 12-14 Aug. 2009.

[7] http://wiki.woodpecker.org.cn/moin/lilin/geos-introduce.