THE DESIGN OF A HIGH PERFORMANCE EARTH IMAGERY AND RASTER DATA MANAGEMENT AND PROCESSING PLATFORM

Qingyun (Jeffrey) Xie

Oracle Corporation, One Oracle Drive, Nashua, NH 03062, USA -
qingyun.xie@oracle.com

Commission IV, ICWG IV/II

KEYWORDS: Raster Database, Image Processing, Raster Analytics, HPC, Oracle GeoRaster, Platform

ABSTRACT:

This paper summarizes the general requirements and specific characteristics of both geospatial raster database management system and raster data processing platform from a domain-specific perspective as well as from a computing point of view. It also discusses the need of tight integration between the database system and the processing system. These requirements resulted in Oracle Spatial GeoRaster, a global scale and high performance earth imagery and raster data management and processing platform. The rationale, design, implementation, and benefits of Oracle Spatial GeoRaster are described. Basically, as a database management system, GeoRaster defines an integrated raster data model, supports image compression, data manipulation, general and spatial indices, content and context based queries and updates, versioning, concurrency, security, replication, standby, backup and recovery, multitenancy, and ETL. It provides high scalability using computer and storage clustering. As a raster data processing platform, GeoRaster provides basic operations, image processing, raster analytics, and data distribution featuring high performance computing (HPC). Specifically, HPC features include locality computing, concurrent processing, parallel processing, and in-memory computing. In addition, the APIs and the plug-in architecture are discussed.

1. INTRODUCTION

The traditional dichotomy is to classify spatial data into vector data and raster data. As a matter of fact, raster data (imagery and gridded data) is the dominant form of spatial information, which includes thematic maps, DEM/DSM, remote sensing imagery, photogrammetric photos, scanned maps, geophysical images, geological maps, etc. These raster types have very different data structure comparing to vector data types and are complex data types in comparison to structured and simple data types such as numbers and strings. To be efficiently managed, they require specialized indexing, querying, processing, and analyzing algorithms. They are generally huge in size and they are "big data" in nature. All these mean we have to build specialized management, processing, and analyzing engines for raster data types. Scalability and performance of such systems are two keys to success. Full support of modern computing architecture and enabling the development of a wide variety of internet or cloud based applications are essential.

This paper summarizes the general requirements and specific characteristics of both geospatial raster database management system and raster data processing platform from a domain-specific perspective as well as from a computing point of view. It also discusses the need of tight integration between the database system and the processing system. These requirements resulted in Oracle Spatial GeoRaster, a global scale and high performance earth imagery and raster data management and processing platform. This paper describes the rationale, design, and implementation of Oracle Spatial GeoRaster as well as the benefits and applications.

2. THE REQUIREMENTS

Understanding the requirements is the first step to success. We will discuss the general requirements in this section.

2.1 Raster Database Management

Geoimagery and raster gridded data are growing exponentially. Numerous remote sensors of different types on various platforms are collecting real time data about the Earth and our environment for different purposes on a daily basis. These images are processed to create products including rasters such as DEM, NDVI and orthophotos, which are then used to update SDI and GIS databases. As a result, it's critically important to effectively and efficiently archive, manage, and distribute all those raster data sets. In other words, truly scalable and really robust raster database management systems are required.

As a database, such management system needs to provide a schema to store, index, query, manipulate, manage, and deliver raster image and gridded data and its associated attribute data and metadata. From a domain perspective, an integrated data model should be designed to provide broad compatibility. Comprehensive ETL tools should be developed to load and export data in a variety of formats. Advanced indexing should include both spatial and non-spatial indices. Queries should support both context-based and content-based queries. Operations such as storage optimization, compressing, subsetting, layer merging, reprojection, image appending and mosaicking are required to accomplish basic manipulation and management tasks. Data versioning and lineage are important in many applications too.

From a computing perspective, security is one important requirement, particularly for intelligence and defense industries. Multi-user concurrency and multi-tasking are basic needs nowadays for large scale systems. Replication, backup and recovery are fundamental management tasks. For mission critical and real time systems, high availability without down time is required. Multitenancy at the database level is desired in the cloud computing era.

At the end, two of the most important tests are scalability and performance. Given the "big data" nature of raster data types, the raster database management system must be truly scalable and provide high performance in data storage, manipulation, and management.

2.2 Raster Data Processing and Analysis

Data archiving and management is not necessarily the end goal. Ultimately, data systems should support specific applications and serve decision making. So, raster data processing, analysis, and modeling are required and should be taken into consideration from the beginning in the design of any such systems.

From a domain perspective, a raster data processing system must provide image preprocessing, basic data manipulation operations, image enhancements, pattern recognition, raster algebra, cartographic modeling, and many raster analytics. It should also be tightly integrated with vector data types to offer more advanced spatial analytics and location intelligence.

From a computing perspective, high performance computing is critical due to the sheer data volume and the complexity of raster processing. And the functionalities must be scalable to handle virtually unlimited data size too. Specifically, locality computing, concurrent processing, parallel processing, and in-memory computing should be implemented or leveraged.

In addition, the platform should allow plug-ins of user's own algorithms or third-party software packages.

2.3 Requirement for an Integrated Platform

From the above requirement analysis, it's not difficult to conclude that on the one hand, building a scalable and easy-to-use raster database will create a great foundation for data processing and analysis, and on the other hand, building a fast processing and analysis solution is important for the application of the image and raster database as well. In other words, both raster database management platform and raster data processing platform are required.

For geospatial image and raster data processing and analyzing, many advanced and highly efficient desktop systems such as ERDAS Imagine and PCI Geomatica and server-based engines such as ArcGIS are readily available. When a large-scale enterprise spatial database is built, such desktop and server-based systems generally can connect to it and then retrieve the imagery and raster data out of the database and process them in the client or another server. However, moving large volume of data between the database and the processing engine is costly given the speed and bandwidth limitations of computer networks.

A typical geoimage database has tens or hundreds of terabytes of data. Petabytes of data is not uncommon. Data has "weight" and geospatial image and raster data sets are particularly "heavy". Given that the processing and analysis are data intensive, data locality should always be an important factor in our design and implementation strategy. So we conclude that building an in-database analytics engine should be a good strategy. It moves the data processing closer to the data instead of moving the data to the processing, which helps achieve greater performance by overcoming the bottleneck of computer networks.

In summary, an integrated platform for both database management and data processing would be ideal.

3. THE DESIGN OF AN INTEGRATED PLATFORM

Oracle Spatial GeoRaster is designed and implemented to meet the aforementioned requirements. It fully leverages the Oracle RDBMS server technology. It is an integrated platform for both raster database management and raster data processing. And it fits well into modern multitier architecture and cloud computing architecture for application development.

3.1 Background and Approach

Traditional imagery and raster data management systems are built on file systems directly. Some take a hybrid approach, which stores metadata and attributes in an RDBMS system while storing images as flat files. However, most file formats allow limited image size and have rigid structures, which prohibit flexible and scalable storage, fast querying and complex manipulations. Such file system does not offer good enough security, reliability, availability, and manageability either. Similarly, another approach is to take advantages of a standard RDBMS system's large object (LOB) data types by loading the raster files into database Binary LOB (BLOB) objects so that both metadata and raster data are stored inside the database. Since raster data are still in some specific file formats, the aforementioned disadvantages of file formats are automatically inherited in such a database system.

Another approach is the so called middleware approach. It stores all data inside a standard RDBMS system and processing the data in a middleware or client software package. Most RDBMS's don't have geospatial image data types defined. So this approach requires a relational database schema to be designed to store the imagery inside RDBMS. However, a fixed set of relational tables specified in such an application schema doesn't offer good flexibility when it comes to integrate geospatial raster datasets with other enterprise datasets. The middleware acts as a query and processing engine, so performance and data security are concerns with this approach. The other downside is either the lack of standard database SQL interface or the decoupling of its interface from the RDBMS system, which significantly limits the usability and enterprise integration efforts.

One trend is to leverage Big Data platforms such as Hadoop and NoSQL. These systems have a shared nothing architecture and use clustering of low end servers. Besides the benefit of lower costs, it offers the best scalability and is best used for static data sets whose subsets are stored, accessed, processed, and analyzed independently. MapReduce and Parallelism provide great performance advantage in reading and processing such data sets. A great example is to store preprocessed and static Earth imagery and map tiles for web visualization purposes, as with most online web mapping services. However, it's not yet proven to be an efficient approach for large datasets that require aggregation of their smaller subsets, which are typically stored on different computing nodes in the Hadoop system, before they are processed. This is because data moving cost among distributed storage nodes is very high. Generally, large scale spatial data types, including raster data types, are such datasets, which require frequent aggregation for processing and analysis. Examples may include band merging, mosaicking, multilayer raster algebra operations, and time

series analysis, to name a few. In addition, there are limited security and managing capabilities in such systems comparing with enterprise RDBMS systems.

Most specialized data warehousing or OLAP systems use a shared nothing architecture too. Data moving cost among distributed storage nodes is a concern as well.

In contrast, enterprise RDBMS systems provide best security and management capabilities for both OLTP and OLAP databases. They also offer best hardware platforms for large-scale data processing and analysis using computer clustering and engineered systems. They typically use a shared everything architecture, which avoids or minimizes data moving cost across different computing nodes by using advanced storage management technologies, such as Oracle Automatic Storage Management (ASM). So, in summary, enterprise RDBMS is the best platform to combine raster database management with raster data processing if it's enhanced to support spatial data types, while other platforms can be complementary to this RDBMS centric approach.

3.1 GeoRaster Database Management System

Oracle Spatial GeoRaster is built to take full advantage of the Oracle enterprise RDBMS system. It makes raster data storage independent of any raster file formats. Basically, the Oracle database system is internally enhanced to provide a new native GeoRaster data type (SDO_GEORASTER) to store and manage raster datasets with virtually no size limitations (Xie, 2008a. Xie, 2008b).

The design of the SDO_GEORASTER data type is one of the keys. It allows the database treat imagery and raster data as if they are simple types such as numbers and arrays. Using standard SQL language, a user can define any table and create one or more columns inside that table using the SDO_GEORASTER type. In the Oracle database, a GeoRaster table is any user-defined table that has at least one data column of type SDO_GEORASTER. It could have any number of additional columns of any other SQL data types. From a user perspective, a GeoRaster database is basically a list of GeoRaster tables, in which each image or raster grid is stored as a GeoRaster object in one row. It can contain unlimited number of GeoRaster objects in one or more schemas and each object can be terabytes in size (Xie, 2006).

Once a GeoRaster table is created and the data is loaded, users can build appropriate indexes on various columns of the GeoRaster tables such as a spatial R-tree index on the GeoRaster column and B-tree indexes on other columns so that queries and other operations on the tables can be supported efficiently.

GeoRaster provides over 190 raster and metadata operations through a PL/SQL API to optimally manage and manipulate the GeoRaster database. Examples include listing and validating all GeoRaster objects in a schema or in the database, deleting GeoRaster objects and dropping GeoRaster tables using standard SQL, adjusting the internal raster blocking size to optimize the storage, generating pyramids, compressing or decompressing GeoRaster objects, editing and updating both raster metadata and cell data, cropping rasters, georeferencing rasters, generating statistics and histograms, combining and appending rasters, reprojecting and mosaicking rasters. GeoRaster also provides ETL tools for loading and exporting raster data.

Because GeoRaster is natively built inside the Oracle database server, almost all enterprise database management features are readily available to GeoRaster users. These include long transaction and rollback, general and spatial indices, data manipulation with parallelism, versioning, multi-tasking, multi-user concurrency, high security, partitioning, advanced replication, physical and logical standby, backup and recovery, and multitenancy. Besides using ETL, Oracle utilities and transportable tablespace can be used for data transfer or migration among databases. In addition, the database is highly scalability using computer and storage clustering technology.

3.2 GeoRaster Data Processing and Analysis

Once the data is stored and managed by the database, they can be indexed, queried and retrieved to serve clients or applications. However, the majority of preprocessing, processing, and analytical operations should be done where the data is stored. There are several benefits of doing that. Firstly, it offers true security for the data because the data no longer needs to be retrieved and loaded into a middleware or client through an insecure network and processed in an unmanaged computer memory. Secondly and most importantly, the processing is closest to the data so it runs faster by avoiding data transferring cost. Thirdly, since modern RDBMS systems such as Oracle support large-scale computer clusters and offer strong computing power, the processes can be run concurrently and deployed onto many powerful servers to reduce the burden on the desktop processing systems. Finally, the processing engine can be coupled with middleware and client-side processing systems to fully leverage the power of enterprise distributed computing systems.

With these, GeoRaster is designed to support powerful raster processing and analysis inside the database server. As a platform, it supports application development and allows plug-ins of existing algorithms. To summarize, there are two major types of processing capabilities as follows.

First, GeoRaster provides a PL/SQL-based raster algebra engine enabling fast cell data searching, raster data analysis, and cartographic modeling. The GeoRaster raster algebra language is an extension to the PL/SQL language, which includes specific algebraic expressions and functions. The algebraic expressions support general arithmetic, logical, relational and casting operations. The raster algebra functions include major procedures to support arithmetic operation, logical operation, statistical analysis, conditional queries, raster segmentation, and cell value-based updates or edits (Xie, 2012).

Second, GeoRaster provides a list of advanced image processing and image serving capabilities. These include GCP georeferencing, reprojection, rectification, orthorectification, image scaling, image enhancements, image masking, image segmentation, NDVI computation, Tasseled Cap Transformation, image appending, band merging, large-scale advanced image mosaicking, and virtual mosaic. The operations described here are most commonly used to process and serve geospatial images, particularly raw satellite imagery and airborne photographs. However, those operations, just like the GeoRaster raster algebra, apply to all raster data types stored in the GeoRaster database (Xie, 2013).

Currently, the list of functions provided by GeoRaster could meet many application requirements. However, it's also worth to note that much more are needed to meet broader raster data

processing requirements. The key take away from our work is we have proved that all image processing and raster analysis functionalities can be efficiently and effectively implemented inside the database, regardless how big the rasters and how complex the processing operations are. In addition, users have the option of plugging in their own algorithms to meet special requirements.

3.3 GeoRaster High Performance Computing

Geospatial images and raster data are complex data and big data, thus geospatial raster processing is not only computationally complex but also I/O intensive. Real-time or near real-time performance should be one of the top considerations at the beginning in the design of any such modern image management and processing solutions.

Oracle database runs on or supports high end computing servers, computer clusters, storage clusters, and engineered database machines. These high end infrastructures make the development of high performance computing possible for GeoRaster. We took a comprehensive approach for HPC by taking into consideration of CPU, memory, storage, as well as computer clustering. Specifically, locality computing, concurrent processing, parallel processing, and in-memory computing are either implemented or experimented in GeoRaster.

For GeoRaster, locality computing is also called in-database processing, which refers to the integration of data processing functionalities into the databases. Given that the processing and analysis are data intensive, data locality should always be an important factor in our design and implementation strategy. The basic idea is to eliminate the overhead of moving large data sets from the enterprise databases to separate processing and analytical software applications. The aforementioned raster functionalities are all developed inside the database so that better performance can be achieved by overcoming the bottleneck of computer networks and also improve scalability and security (Xie, 2008a. Xie, 2012. Xie, 2013).

Concurrency and multi-tasking are readily available in all enterprise RDBMS systems, including Oracle. Implementing raster processing inside the database enables us to leverage Oracle concurrent processing infrastructure directly. We fine-tune memory usage through our internal GeoRaster memory management system so that each concurrent process would not use too much memory resulting in better scalability. Concurrent processing is available to all GeoRaster functions, regardless the database is on a single machine or on a computer cluster. Concurrent processing drastically improves massive raster processing performance, database scalability and overall throughput (Xie, 2006. Xie, 2013).

Given that modern computers are mostly multicore or have multiple CPUs, parallel processing should be implemented in any modern geospatial and image processing solutions. GeoRaster supports two types of parallel processing: parallel execution of SQL statements and parallelized GeoRaster procedures. Parallel execution of SQL statements applies directly to all GeoRaster read-only functions such as metadata-related query operations and all single cell queries. Parallelized GeoRaster procedures include most of the raster processing and analysis operations such as compression, pyramiding, rectification, mosaicking, and raster algebra. The implementation of parallelism dramatically improves raster data processing performance inside the database. Parallelism is a key feature of the GeoRaster data processing and analysis platform (Xie, 2012. Xie, 2013).

Finally, in-memory computing is designed and experimented to leverage large memory size coming with modern computing servers. GeoRaster is designed to support small size of memory so that it can process any size of rasters on any hardware configurations and better support concurrency. For example, it implemented a sophisticated memory management system in the processing engine so that it can process images of terabyte in size on an average PC. The idea of in-memory computing is the opposite, which loads larger chucks of raster data or the whole raster into memory so that the overhead of I/O and data swapping can be minimized to improve performance. Our initial experiments have shown great performance improvement by leveraging larger memory in each process. Unlike other products, allowing users to control how much memory to use is one of the key in-memory computing features in the design of GeoRaster.

3.4 GeoRaster Application Development

Applications are diverse and may have very different levels of complexity. On the one hand, GeoRaster provides a large number of database management operations and data processing capabilities, which are readily available for applications to use. On the other hand, special algorithms and operations may be needed to serve special purposes in different applications. To support application development and customization, the GeoRaster platform provides API's as well as a plug-in architecture.

GeoRaster has a PL/SQL API and a Java API, which are provided for database creation, database administration, data manipulation, raster analysis, image processing, and raster data serving and delivering. They are the foundation for application development. They also help in the integration of existing applications with GeoRaster. Users can use PL/SQL, Java, C or C++ to leverage the PL/SQL API and the Java API or directly access the binary data of the open GeoRaster data model. Many of the functions listed in this paper are extended, augmented or leveraged by partner technologies delivered as ETL tools, comprehensive raster analysis and image processing client tools, or in the form of visualization engines.

The other key feature is allowing users to plug in their algorithms into the GeoRaster platform. In other words, existing algorithms can be plugged into the database to work directly on GeoRaster objects. For example, if the algorithm is implemented in Java, you can leverage the GeoRaster Java API and adopt the algorithm into a Java Stored Procedure, which runs inside the database like any other GeoRaster procedures. For any such user defined Java Stored Procedure, concurrent users can invoke it simultaneously and multi-tasking is immediately available, thus you can quickly take advantage of your powerful hardware platform. If the algorithm is implemented in C or C++, you can use the Oracle Call Interface (OCI) to adopt it and create an External Procedure to execute it on one or more GeoRaster objects. An External Procedure runs like other PL/SQL procedures so that concurrency and multi-tasking are available too. The GeoRaster Java API is designed to deal with GeoRaster objects specifically while OCI is a generic database C interface, so the major difference is that it's easier to plug in Java programs than C programs currently.

In addition, Oracle Fusion Middleware MapViewer is part of the GeoRaster stack. It fully supports GeoRaster data types and

is the web-based mapping and visualization application platform for GeoRaster. MapViewer also has a map tile server, which is a map image caching engine that fetches, caches, and serves pre-generated, fixed-size map image tiles. You can leverage it to cache GeoRaster images in the middle tier to speed up applications. It provides XML and HTML5 API's for application development. Spatial web services such as OGC WMS are provided in the middleware for internet based application development as well.

In summary, the API's and plug-in architecture enable unlimited customization and secondary development making GeoRaster a very power platform for applications.

4. THE BENEFITS AND APPLICATIONS

The design of GeoRaster, an integrated raster database management and raster data processing platform, is based upon large-scale enterprise RDBMS for data management as well as modern hardware and software infrastructure for high performance computing. The native GeoRaster data type and database schema is uniquely designed to meet all database management requirements. The implementation of raster manipulation operations, image processing, raster algebra, API's, and plug-in architecture enables a variety of applications and unlimited secondary development.

There are many benefits of this design. To give out a few examples, all metadata and cell data of GeoRaster objects are open to users and can be accessed at bit level as long as the users are granted the access privilege. This enables shared third-party application developments. It breaks the size barriers. The internal tuning tools and compressions can be used to optimize the raster blocking and other storage options to best fit into the needs and performance requirements of different applications. The native data type approach provides great flexibility to store raster data in regular relational tables. Images or rasters in different projections and locations can be stored in the same table and users can easily build global databases with whole-earth spatial index and other indices enabling fast queries and manipulations of rasters located anywhere on the Earth. GeoRaster features in-database processing, optimized multi-tasking, parallel processing, and in-memory computing. The in-database data manipulations, raster algebra and image processing capabilities allow data to be processed where it is stored. Coupled with multi-tasking and parallel processing, this provides great performance and true security. The standard PL/SQL and JAVA API's are flexible, powerful and easy-to-use. They help speed up enterprise integrations and broaden geospatial application development.

Application of this platform is unlimited. For example, it can be used to support image visualization, metadata management and image archiving, raster data processing, cartographic modeling, web services, data distribution, spatial data cloud, and application cloud. It offers a single database and platform yet it enables multiple applications to be easily built on top of it for large organizations.

5. CONCLUSION

Imagery and raster gridded data are complex data types and "big data". Specialized database management system and data processing system are required. An integrated platform for both raster database management and raster data processing is desired. Oracle Spatial GeoRaster is designed to meet those requirements and provides unlimited application potential. It offers the best database security, manageability, scalability, and availability by leveraging the best commercial RDBMS in the market. It provides high performance computing features enabling massive raster data processing and analysis and removing the overhead of data movement. As a platform, it provides API's for developing applications and allows plug-ins of existing algorithms so that a broad variety of applications can be built and the computing power of modern infrastructure can be easily and fully leveraged. The platform will continue to be enhanced and future directions may include adding more raster data processing and analyzing capabilities and doing more research and development on in-memory computing, plug-in architecture and related API's, simplification, and easy-to-use, to name a few.

6. REFERENCES

Xie, Q., Z. Li, and W. Xu, 2006. Using Enterprise Grid Computing Technologies to Manage Large-Scale Geoimage And Raster Databases. In: *the Proceedings of ASPRS 2006 Annual Conference*, Reno, Nevada, May 1 – 5, 2006.

Xie, Q., S. Ravada, W. Xu, and Z. Zhang, 2008a. An Enterprise Database-centric Approach for Geospatial Image Management and Processing. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXVII-B4.2008*, XXI ISPRS Congress, Beijing, China,

Xie, Q., 2008b. Oracle Spatial, Raster Data. *Encyclopedia of GIS*, Shashi Shekhar and Hui Xiong (editors), Springer. pp. 826 - 832.

Xie, Q., Zhang, Z., and S., Ravada, 2012. In-Database Raster Analytics: Map Algebra and Parallel Processing In Oracle Spatial GeoRaster. In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XXXIX-B4, 2012*, XXII ISPRS Congress, Melbourne, Australia.

Xie, Q., F., Chen, Z., Zhang, I., Lucena, 2013. In-database Image Processing in Oracle Spatial GeoRaster. In: *the Proceedings of ASPRS 2013 Annual Conference*, Baltimore, Maryland, March 24-28, 2013