Ganos Aero: A Cloud-Native System for Big Raster Data Management and Processing

Fei Xiao
Alibaba Group
jibo.xf@alibaba-inc.com

Jiong Xie
Alibaba Group
xiejiong.xj@alibaba-inc.com

Zhida Chen
Alibaba Group
zhida.chen@alibaba-inc.com

Feifei Li
Alibaba Group
lifeifei@alibaba-inc.com

Zhen Chen
Alibaba Group
erchen.cz@alibaba-inc.com

Jianwei Liu
Alibaba Group
jerven.ljw@alibaba-inc.com

Yinpei Liu
Alibaba Group
yinpei.lyp@alibaba-inc.com

ABSTRACT

The development of Earth Observation technology contributes to the production of massive raster data. It is vital to manage and conduct analytical tasks on the raster data. Existing solutions employ dedicated systems for the raster data management and processing, respectively, incurring problems such as data redundancy, difficulty in updating, expensive data transferring and transformation, etc. To cope with these limitations, this demonstration presents Ganos Aero, a cloud-native system for big raster data management and processing. Ganos Aero proposes a unified raster data model for both the data management and processing, which stores a single copy of the raster data and without performing an expensive tiling procedure, and thus achieves significant improvement in the storage and updating efficiency. To enable efficient query and batch task processing, Ganos Aero implements an on-the-fly tile production mechanism, and optimizes its performance using the cloud features including decoupling compute from storage and pushing costly operations closer to the storage layer.

Since deployed in Alibaba Cloud in 2022, Ganos Aero has been playing a critical role in many real applications including the modern agriculture, environment monitoring and protection, et al.

PVLDB Reference Format:
Fei Xiao, Jiong Xie, Zhida Chen, Feifei Li, Zhen Chen, Jianwei Liu, and Yinpei Liu. Ganos Aero: A Cloud-Native System for Big Raster Data Management and Processing. PVLDB, 16(12): 3966 - 3969, 2023.
doi:10.14778/3611540.3611597

1 INTRODUCTION

The development of Earth Observation technology gives rise to an explosion of the raster data, e.g., NASA's climate observation data has reached 32PB since 2000 [6]. There exist extensive raster data applications such as evaluating the condition of a broad geospatial area, and simulating a geophysical phenomenon. A raster data object consists of a matrix of equal-sized pixels (or cells), where each pixel is associated with a geographic location and contains values such as the temperature and altitude. It is critical to manage and analyze the raster data, where an analytical task usually encapsulates many expensive operations on the large-scale raster data. Existing solutions employ dedicated systems for the raster data management and big raster data processing, respectively. Typically, the raster data is managed in a DBMS, and the raster data analytics is performed in a distributed computing framework. Such strategy has several defects. Firstly, it results in data redundancy and high storage costs. In a DBMS, the raster data is stored as a data format that is compact and is efficient for indexing, while in a raster data processing system, the raster data is partitioned into a vast number of small pieces called the tiles so that the processing can be easily parallelized for better efficiency. Secondly, it is unfriendly to update. When the raw raster data is updated, both systems need to perform the data updating accordingly, which is especially expensive for the raster data processing system because it must conduct the tiling procedure on the whole dataset. Thirdly, it has poor support for the analytical tasks that operate on the non-raster data as well, requiring the cooperation of the DBMS and the raster data processing system. Lastly, it is unfriendly to use. The user has to interact with entirely different systems for different applications. To solve these problems, we propose Ganos Aero, a novel system that integrates the raster data management and processing into one system. Before introducing Ganos Aero, we first briefly review existing works.

Many traditional relational DBMSs (e.g., PostGIS [3], Oracle Spatial [1]), and some array-based DBMSs (e.g., SciDB [2], Rasdaman [4]) have supported the raster data and developed indexing mechanisms to facilitate the query processing. However, the parallelism of a DBMS is mostly optimized for the parallel processing at the object level, while the raster data is usually large and differs greatly in size, which hinders the parallel processing and makes it not efficient for a complex raster data processing task. The other line of works (e.g., SpatialHadoop [7], GeoTrellis [5]) develop raster data processing systems based on a distributed computing framework. They perform a tiling procedure to divide the raster data into a vast number of tiles for the parallel processing. However, the tiling procedure is time-consuming and results in high storage cost. Also, they have limited support for processing the queries running on both the raster and non-raster data.

This study presents Ganos Aero, a unified system that integrates Alibaba's cloud-native DBMS PolarDB for PostgreSQL [8] (hereafter

Proceedings of the VLDB Endowment, Vol. 16, No. 12 ISSN 2150-8097.
doi:10.14778/3611540.3611597

simplified as PolarDB) and Apache Spark for big raster data management and processing. Many challenges exist for the integration, such as how to break through the barrier between a DBMS and a batch computing framework, how to reduce the system deployment cost, and how to store the massive raster data in database without compromising the batch processing performance. To solve the problems arisen, Ganos Aero leverages the cloud features of decoupling compute from storage and the flexible storage mode, which stores a single copy of the raster data in Object Storage Service (OSS), and employs multiple high-performance computing instances to enable efficient computations on the raster data. Besides, by providing the raster data as a native data type, Ganos Aero implements a transparent yet efficient raster data accessing mechanism, and develops optimization techniques including pushing costly operations to be performed in the DBMS. Our contributions are summarized below.

- This study presents Ganos Aero, a novel cloud-native big raster data management and processing system. Ganos Aero is a full-fledged system having the features of large-scale raster data management, and being capable of processing both the interactive queries and complex batch tasks.
- Ganos Aero adopts a unified raster data model for both the data management and processing, which stores a single copy of the raster data and without building and maintaining a vast number of tiles. Therefore, Ganos Aero achieves significantly better storage and updating efficiency than existing solutions. It greatly saves the system deployment cost for TBs or even PBs of raster data.
- Ganos Aero implements an on-the-fly tile production mechanism, and leverages the cloud features to optimize its efficiency including decoupling compute from storage and pushing costly operations closer to the storage layer, thus enabling efficient interactive query and batch processing.
- Since deployed in Alibaba Cloud in 2022, Ganos Aero has been playing a critical role in many real applications. We showcase the features of Ganos Aero on real datasets.

2 SYSTEM ARCHITECTURE

Figure 1 shows the architecture of Ganos Aero, composed of the management layer, the computing layer, and the user interface.

2.1 Management Layer

The management layer, i.e., PolarDB, is responsible for the raster data management, including storing, indexing, and optimizing the operations on the raster data. PolarDB is cloud-native and provides features such as high-performance data accessing, resource autoscaling, and fault tolerance. Ganos Aero is developed on top of Ganos [9], the spatial and temporal database engine of PolarDB, and adds supports for the raster data management and processing. To reduce the system deployment cost, Ganos Aero can store the data in OSS, which is much cheaper than storing in a table. A transparent yet efficient data accessing mechanism is implemented so that the user can access the data without noticeable difference.

The user can decide for each raster object whether to build a pyramid structure. The pyramid consists of the data chunks having different resolutions, which facilitates the online tile production and the visualization procedure. The user can decide between using

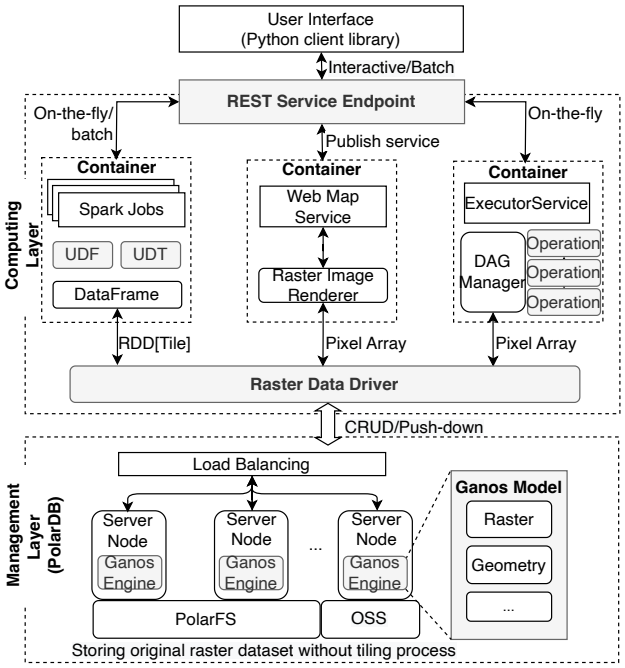

Figure 1: Ganos Aero architecture.

the raster data or the pyramid for a query based on the user's demands w.r.t. the accuracy and efficiency. This is an advantage over a tile-based system that can only operate on the tiles.

2.2 Computing Layer

Ganos Aero provides two processing modes, namely the interactive mode and the batch mode. The interactive mode performs operations on the raster data in a record-by-record manner. It is suitable for the query that runs on a small raster data subset. By contrast, the batch mode performs operations on the raster data in a batch manner, suitable for the task that encapsulates a pipeline of complex computations on the large-scale raster data. A background map service is responsible for transforming the results into the images that can be displayed by the user interface. The raster data driver bridges the computing and the management layers, which reads and writes the raster data between the two layers.

In an interactive processing instance, ExecutorService produces a directed-acyclic-graph (DAG) workflow by parsing the query request. Then, the DAG manager encodes the DAG into a data structure called a template and calls the management layer to store the template in the database. Subsequently, the interactive processing instance communicates with the management layer to obtain the raster objects of interest. Instead of directly sending back the objects, the management layer reads the template and determines the operations to be performed in the DBMS. Next, it performs the operations on those objects accordingly and sends the intermediate results to the interactive processing instance. The interactive processing instance performs the remaining operations on the intermediate results one by one and sends the result to the user interface

for display. By pushing some operations to the DBMS, the query optimizer helps improve the query plan, and reduce the network costs, e.g., by conducting a cropping operation in the DBMS, the cropped raster objects instead of the original ones are transferred.

Similarly, a batch processing instance parses the query request and interacts with the management layer to push a subset of the operations to the DBMS. The sent-back objects are packed into RDDs, transformed into a DataFrame, i.e., the concept of a batch of data in Spark, and are processed in parallel by Spark. Ganos Aero extends the DataFrame model to support the raster data by developing a set of user-defined functions (UDFs). In contrast to other Spark-based raster data processing frameworks, e.g., GeoTrellis, Ganos Aero can query the raster data stored in the database via SQL, and produce the tiles on-the-fly to achieve high efficiency in the subsequent operations, thereby avoiding the extra storage cost of the tiles.

3 DATA ORGANIZATION

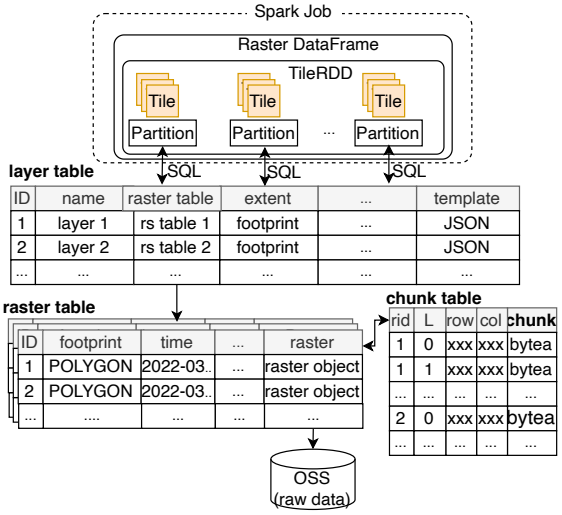

Figure 2: Ganos Aero data organization.

Figure 2 shows the raster data model and the data organization approach in Ganos Aero. The raw raster data is stored in OSS. The raster table stores the metadata and the address of a raster object in OSS. The pyramid table stores the raster data chunks for online tile production. The layer table stores the layers, each of which refers to a subset of raster objects that are specified by a query request.

Raster Data Type. Ganos Aero provides a native raster data type, composed of the metadata and the address of the raw data in OSS. The metadata is not stored as the columns of the table because the raster data from different sources differ in the metadata, making it difficult and memory-inefficient to predefine the table schema. The footprint and timestamp of the raster data are stored as the columns, which are used for indexing. When importing a raster object into the raster table, the user can specify whether to build a pyramid based on that object. The pyramid is composed of many data chunks, each of which stores an image of the raster object at a specific zoom scale. The pyramid is similar to the tiles returned by a tiling procedure, so it can facilitate the on-the-fly tile production. The key difference is that the pyramid is associated with a single raster object, hence it can be updated efficiently when that object is updated. To reduce the storage cost, the pyramid only stores the data chunks at small zoom scales.

Layer. When a user submits a query request, the computing layer parses the query request into a DAG workflow. The workflow is then encoded into a data structure called the template, which stores a set of constraints specifying the raster objects of interest. Next, Ganos Aero adds a new tuple representing the new layer into the layer table. The management layer loads the corresponding set of raster objects after reading the layer table, and decides the subset of operations defined in the template that are better to be performed in the DBMS. After performing the operations, the management layer sends the intermediate results to the computing layer, and the computing layer performs the remaining operations.

TileRDD and Raster DataFrame. Ganos Aero extends the RDD and DataFrame model of Spark for the raster data, allowing Ganos Aero to leverage the rich features of Spark for various tasks.

4 DEMONSTRATION SCENARIO

User Interface. Figure 3 shows the front-end interface of Ganos Aero. Users can write code or SQL in the workspace of the Jupyter notebook, and the map UI will display the running result.

Data Management and Interactive Query Processing. As shown in the first picture of Figure 3, the user can input SQL to fulfill the raster data management tasks and to check the running results of the small queries in an interactive manner. Step one shows the procedure of establishing a connection to the database and querying the objects in the raster data table. The tiles of the objects are produced on-the-fly so that the user can see the query result in the map UI. The part highlighted by the rectangle labeled one in the map UI shows the only raster object in the table. When the user performs common map operations such as zoom in/out, and panning, the map UI will send the tile requests to the backend process, and Ganos Aero will produce the corresponding tiles on-the-fly and send them back for visualization. In the second step, two raster objects are inserted into the table, and the part highlighted by the ellipse labeled two in the map UI shows the inserted raster objects, whose tiles are produced on-the-fly as well.

Step three calls the function **ST_NormalizedDifference** on a raster data table containing 9,190 objects. The function computes the normalized difference vegetation index (NDVI) of the raster objects, which is an indicator of quantifying vegetation greenness and is commonly used in accessing the vegetation density of a geospatial area. To save space, we zoom out the visualization result and put it on the bottom right of the first picture. As we can see, the user can easily recognize the geospatial areas having denser vegetation, i.e., the greener parts. It takes only a few seconds to compute and visualize the query result, giving the user an interactive experience. This feature allows the user to conduct trial and error on a small dataset before raising a complex task request using the batch mode.

Batch Processing using Spark. The second picture of Figure 3 showcases the capability of Ganos Aero leveraging Spark to conduct a cumbersome task. The task is composed of two sub-tasks. The first sub-task performs a spatial join on the global raster data and a

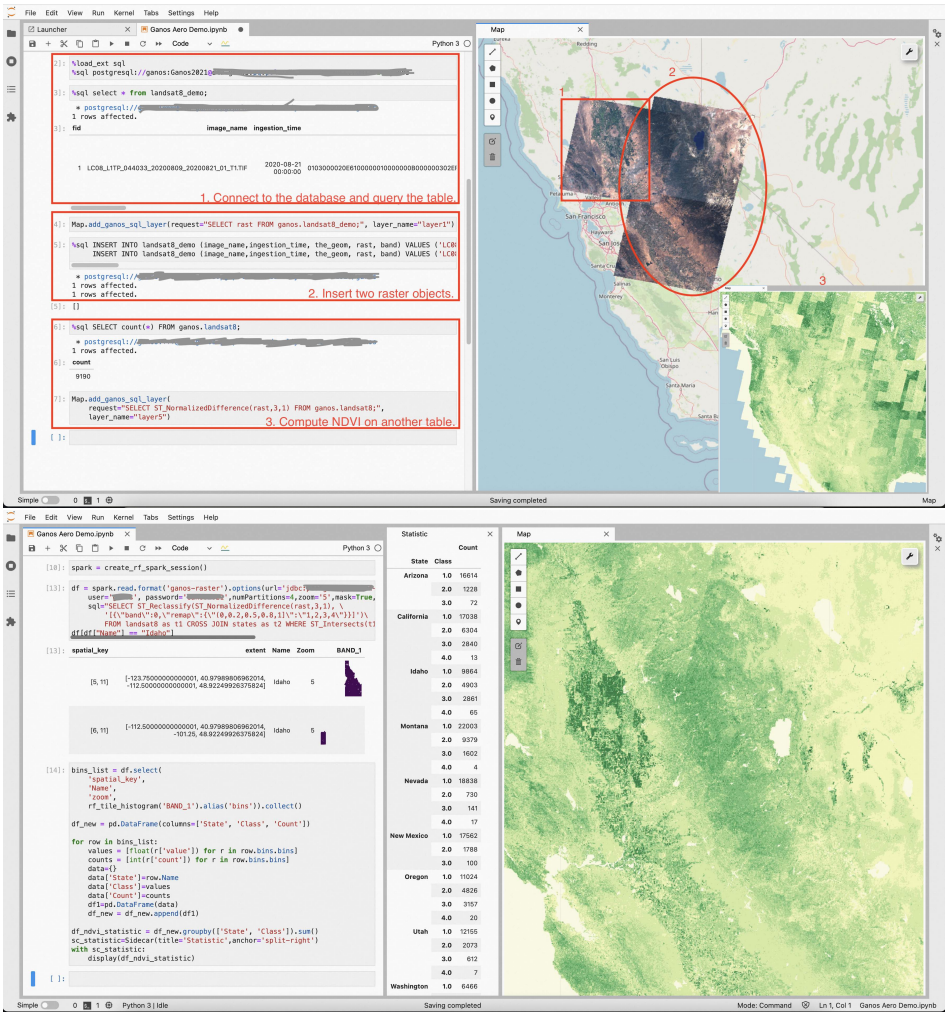

Figure 3: Demonstration scenarios.

table storing the geospatial areas of the administrative districts in the USA, computes the NDVI for the raster data in each state of the USA, and then calls the function **ST_Reclassify** to classify the NDVI values of the pixels into 1, 2, 3, or 4. The four categories represent different levels of the vegetation density, where the category 4 represents the densest vegetation. The second sub-task is to sum up the number of pixels belonging to each category in each state. This task is useful for analyzing the vegetation status of a country.

This demo scenario showcases the merits of Ganos Aero: With the on-the-fly tile production mechanism, Ganos Aero is efficient in handling a complex analytics task running on massive raster data with small storage cost; As a DBMS, it is convenient to conduct analytics tasks involving other non-raster data in Ganos Aero.

REFERENCES

[1] Oracle [n.d.]. *Oracle's Spatial Database.* Oracle. Retrieved January 4, 2023 from https://www.oracle.com/database/spatial/

[2] Carnegie Mellon Database Group [n.d.]. *SciDB.* Carnegie Mellon Database Group. Retrieved January 4, 2023 from https://dbdb.io/db/scidb

[3] PostGIS Project Steering Committee (PSC) [n.d.]. *Spatial and Geographic objects for PostgreSQL.* PostGIS Project Steering Committee (PSC). Retrieved January 4, 2023 from https://postgis.net/

[4] Edgewall Software [n.d.]. *Welcome to rasdaman - the world's most flexible and scalable Array / Datacube Engine.* Edgewall Software. Retrieved January 4, 2023 from http://www.rasdaman.org/

[5] GeoTrellis [n.d.]. *What is GeoTrellis?* GeoTrellis. Retrieved January 4, 2023 from https://geotrellis.readthedocs.io/en/latest/

[6] Peter Baumann, Dimitar Misev, Vlad Merticariu, and Bang Pham Huu. 2021. Array databases: concepts, standards, implementations. *Journal of Big Data* 8, 1 (2021), 1–61.

[7] Ahmed Eldawy and Mohamed F Mokbel. 2015. Spatialhadoop: A mapreduce framework for spatial data. In *ICDE*. IEEE, 1352–1363.

[8] Feifei Li. 2019. Cloud-native database systems at Alibaba: Opportunities and challenges. *PVLDB* 12, 12 (2019), 2263–2272.

[9] Jiong Xie, Zhen Chen, Jianwei Liu, Fang Wang, Feifei Li, Zhida Chen, Yinpei Liu, Songlu Cai, Zhenhua Fan, Fei Xiao, and Yue Chen. 2022. Ganos: A Multidimensional, Dynamic, and Scene-Oriented Cloud-Native Spatial Database Engine. *PVLDB* 15, 12 (sep 2022), 3483–3495.