A construction System of Lake-Warehouse Integration in the Electricity Industry Based on Hudi

* Peng Wang[a], Jianhao Zhang[b], Xiang Wang[c], Guangrui Peng[d], Mingli Chen[e], Tianfeng Shao[f], Xiaotong Tuo[g], Jian Hu[h]

*[a]wangpeng@sgepri.sgcc.com.cn, [b]zhangjianhao@sgepri.sgcc.com.cn, [c]wangxiang@sgepri.sgcc.com.cn, [d]pengguangrui@sgepri.sgcc.com.cn, [e]chenmingli@sgepri.sgcc.com.cn, [f]shaotianfang@sgepri.sgcc.com.cn, [g]tuoxiaotong@sgepri.sgcc.com.cn, [h]hujian@sgepri.sgcc.com.cn

NARI Group Corporation (State Grid Electric Power· Research lnstitute); China Realtime Database Co., Ltd.Nanjing, China

Abstract. This current data integration system utilizes various methods such as DataWorks_DI, Ogg+DataHub, etc., for large-scale data access. It generates shared layer and analytical layer data in a T+1 manner, supporting upper-level data applications through Restful services. Given the existing data architecture, there are issues including redundant data synchronization links, inadequate processing timeliness, slow application queries, and non-integrated quantitative measurement data. These issues result in situations where business users still encounter unavailable or unreliable data during data utilization. To address the overall requirement of improving the timeliness of the data centralization system, and to meet the business needs for various real-time common data sets, there is an urgent need to establish a data architecture system that includes real-time incremental and full-data merging, real-time data processing modeling, and real-time data services. This system aims to build a comprehensive data processing capability that integrates both streaming and batch processing, enhancing the timeliness of data centralization application support. The goal is to enable rapid perception, monitoring, alerting, and processing of data from production business systems, ultimately improving the stability of production system operations.

Keywords: data lake; Real-time computing; lake house; data management

1. Introduction

With the continuous development and updates of the company's information technology, the volume of business data, operational logs, and other data generated by various business systems has experienced exponential growth. To address issues related to data storage and computation, the company has introduced various products associated with the data hub, which are currently widely used in multiple scenarios. However, as the requirements for the timeliness of analytical data increase, the conventional T+1 analytical approach is no longer sufficient. To address this challenge, a new lake-warehouse integration construction system based on Hudi is proposed for the power grid scenario [1]. This system aims to provide a comprehensive solution to enhance data processing efficiency and meet the increasing demands for analytical data timeliness.

Based on the concept of a data lake, integrating open-source frameworks with the data centralization platform, we are constructing a comprehensive data development platform for

PMBDA 2023, December 15-17, Nanjing, People's Republic of China
Copyright © 2024 EAI
DOI 10.4108/eai.15-12-2023.2345365

real-time data warehousing technology. This platform facilitates the ingestion, analysis, and operation of data from multiple sources and different formats, elevating traditional big data analytics from a T+1 scenario to a minute-level responsiveness. This significantly enhances the timeliness and accuracy of data. The platform accommodates the parsing and transformation of data from various heterogeneous sources, conducting technical verification, design implementation, and functional testing of data lake components [2]. This effort aims to improve data timeliness, simplify data analysis complexity, balance the trade-off between data timeliness and computational resource utilization, and optimize the design of data lake layers based on the improved real-time capabilities.

This article, based on new big data components such as Hudi and the integrated construction concept of data lake and warehouse, proposes a comprehensive lake-warehouse construction system for the power grid scenario based on Hudi. It provides a detailed exposition of the construction system, covering platform architecture, platform design philosophy, and common scenarios. The discussion delves into the specifics of building a unified lake-warehouse system tailored for the power grid scenario using Hudi and similar cutting-edge big data components.

2. Integrated Lake-Warehouse Overall Architecture Design

The integrated lake-warehouse construction system for the power grid scenario based on Hudi primarily encompasses full-scenario construction, including data ingestion, data management, data development, computation and storage resource control, task scheduling, and more. The most crucial aspect of this architecture, in comparison to the company's previous big data frameworks, is the ability to ingest and query data at a minute-level granularity. To achieve this requirement, the system employs a combination of in-house developed tools and open-source technologies, designed around Hudi. It relies on S3 object storage and Kubernetes (k8s) for resource management, achieving effective resource isolation [3]. This comprehensive architectural framework covers the entire process from development to analysis and visualization.

The system includes in-house developed data ingestion tools that support the ingestion of data from dozens of internal data sources. It introduces a new data layering approach, breaking away from fixed relationships between traditional data centralization layers while still maintaining compatibility with traditional data centralization scheduled offline analysis. This ensures the timely acquisition of changes in the underlying analytical data for scenarios with high real-time change demands, facilitating the retrieval of the latest results. The specific technical architecture is illustrated in the diagram below Figure 1.

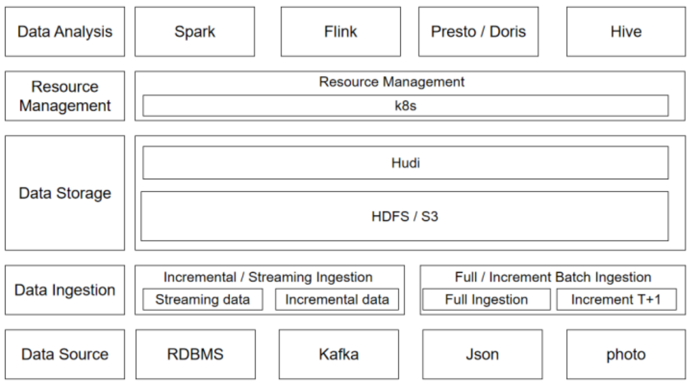

Fig 1. framework

3. Main Tasks of System Construction

3.1. Data Integration

In the aspect of data integration, we conducted separate tests on the performance and resource consumption of data ingestion using Spark, Flink, and the Java client approach. Both Spark and Flink require additional memory and resources to maintain the cluster, which is inefficient for data ingestion tasks [4]. Moreover, the data types supported by Spark and Flink are bound to their own types, making it challenging to integrate with the existing products in our company. Although the Java client approach provides more flexibility for data transformation and ingestion, its data write interface is not mature, and the existing functionality does not meet the requirements of our company's business.

To strike a balance between data writing efficiency and resource consumption, and to meet the later demand for real-time data ingestion monitoring, we performed source-code-level development and optimization of Hudi. See Figure 2, We utilized the Java client approach in conjunction with our existing SG_ETL tool to develop a Hudi plugin, enabling unified offline and real-time data ingestion. Compared to traditional methods, this approach significantly saves resources during data writing, and during transmission, it allows real-time monitoring of data writing based on the content of each batch of data ingested into the lake, thereby reducing write latency.

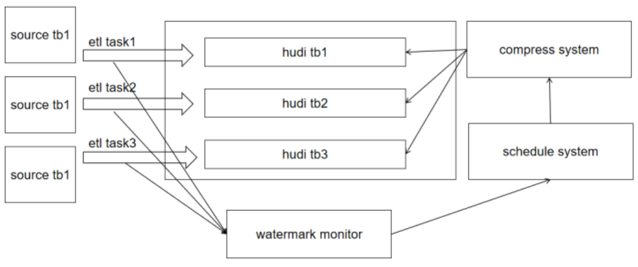

Fig 2. Data ingestion

3.2. Storage-Compute Separation

With the current 10 Gigabit network bandwidth, network transmission is no longer a performance bottleneck. The primary task of a data lake system is how to manage the relationship between the storage resources for a massive amount of platform data and the computing resources for various types of analysis tasks in different businesses. See Figure 3, Due to differences in storage data formats and importance levels, a unified storage-compute architecture cannot independently scale storage or computing resources. In response to practical scenarios, the system adopts a storage-compute separation approach. The system stores business data in the Hudi format in S3-like systems, ensuring data storage isolation between different layers and users based on their respective buckets. For computing, Flink and Doris are used based on different scenarios such as analysis and ad-hoc queries. Analytical tasks run on Kubernetes (k8s) [5], and multi-tenant resource allocation and isolation are ensured through Kubernetes' permission management. The storage structure is illustrated in the diagram below.

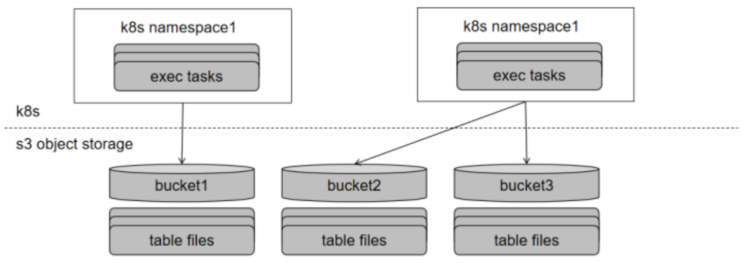

Fig 3. Storage-Compute Separation

3.3. Data Model

In traditional big data models, data is divided into source layer, shared layer, analytical layer, and other business logic. This model, when calculating data on a daily basis, allows multiple businesses to share the same calculated part, saving resources. However, in a data lake, the bottom layer of source data is updated in real-time. Using the traditional layering model cannot reflect the real-time data writing characteristics. Business logic can only meet the timeliness of processing logic by directly reading source layer data during computation.

Based on the real-time characteristics, we have formulated the development idea of storing algorithms without storing data, with independent business logic. Nodes within a business are dependent on each other, while nodes outside the business are for analysis. Although this increases the computational cost, each business logic can achieve the optimal execution time. At the same time, to be compatible with traditional daily-level business operations, as Figure 4 show, we retain the layered model, recording the snapshot information of each table after the last update each day. Through the source layer, relevant calculations are executed to the shared analytical layer, achieving compatibility with the lake-warehouse and fully supporting the original business logic analyzed on a daily basis.

We have specified a data table creation standard, where the data lake tables follow the convention of using the business system name as a prefix + specific table name. and the prefix is used to automatically obtain information such as business ownership and responsibility when creating the data table [6]. The creator ensures the classification of the data table, and business analysts can quickly obtain the required data by querying the system prefix for analysis.

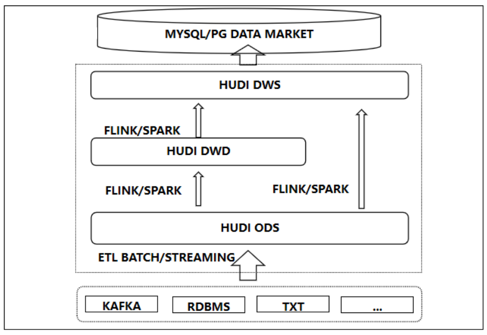

Fig 4. Data Model

3.4. Watermark

In traditional big data models, data is typically updated on a daily basis, and analysis can only be performed on the changes between the last snapshot of the previous day, with intermediate results being deleted during the merge process. This approach fails to meet the requirements for real-time and accurate analysis. The new data lake technology, leveraging the update and delete capabilities of components like Hudi, follows the 'one table' philosophy, synchronizing incremental and historical data in a single table, making data immediately available upon ingestion. However, the traditional approach of determining whether all required data has been written based on time intervals is no longer applicable in the absence of incremental and full tables. If the required data is not entirely ingested during the calculation, it may lead to inaccuracies and discrepancies in results from multiple calculations.

One approach to address this challenge is to save data at different times in a timeline and retrieve data at different time points through timeline backtracking, thus improving the accuracy of calculation results. However, since data is ingested in real-time, ensuring that the required data is fully available when a calculation task starts becomes a crucial aspect of the data lake system [7]. Simply relying on time and parent node dependencies cannot fully satisfy this requirement. To address this issue, the company, in its self-developed ETL data ingestion tool, referenced the watermark concept from streaming computing frameworks. It designed a watermark data detection system that considers multiple dimensions such as the time of data generation and the time of data ingestion. When the incoming data includes the time of data generation, the system can record the maximum and minimum values of the fields in each batch during ingestion through configuration. As the data generation time is incremental, and considering possible delays, a 'table-ready' node was designed in the scheduling process. The system determines that the required data is fully ingested by checking if the maximum time of the data needed for a calculation task is less than or equal to (the minimum time point in the latest batch ingested - acceptable delay time). Otherwise, the scheduling program continues monitoring and waiting to ensure the timeliness and accuracy of the scheduled tasks.

The specific process of publishing a computing task is as follows:

Configure the 'table-ready' node: In this node, configure the maximum time point (max_time) of the data needed for the tables involved in the analysis program to determine that all required data has been ingested.

Users save and submit the written analysis SQL, click 'run,' or configure the task scheduling time. The scheduling system is responsible for triggering the execution. When saving, the system automatically checks whether the user has read and write permissions for the analysis table, and if the SQL statement is written correctly. When the scheduled time is reached, the system sends the task to the execution program.

During program execution, the 'table-ready' node continuously checks whether the watermark of the table meets the requirements. If not, it enters a loop waiting state until the watermark >= max_time. Once the required data has been completely ingested, it starts executing the SQL logic.

After the execution of the business logic is completed, update the task instance status and notify relevant programs to display running data or execute the next node.

Once all nodes are completed, the process concludes.

4. System usage example

4.1. Partial Field Update

In company frequently encounters scenarios where various ledger tables in business systems need to be transformed into a consolidated ledger table for use by the analysis system. The existing middleware framework often results in delayed data updates, leading to inaccurate analytical results that are unusable. Given that each ledger table in this scenario typically has a unified identity recognition component, a self-developed merging logic class based on Hudi's PartialUpdateAvroPayload class is employed in the data lake. See Figure 5, this class is responsible for merging the updated data obtained from the business systems into the data lake table.

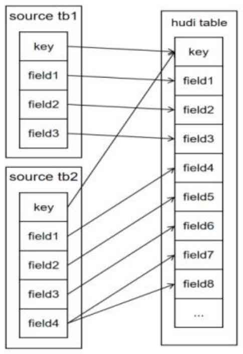

Fig 5. Partial Field Update

Merging Logic:

1. Table create: Aggregate all table fields based on the primary key. To prevent other identical fields in multiple tables, append the source table name to the field names in the wide table to avoid inadvertent modifications.

2. Insert: Directly insert when there is no data with the primary key, and set other fields in the wide table to null. If the data exists, update all fields involved in the wide table. When a field in the inserted data is null, set it uniformly to a value that won't occur in the business scenario based on the field type.

3. Update: Update all the fields involved in the table that are being updated, leaving other non-updated fields unchanged.

4. Delete: Set all the involved fields in the wide table to null. If all fields are null, delete the data.

4.2. Large-Scale Data Update and Merge

In the company's use of the data lake, the primary data scenario involves daily ingestion of billions of records into a single table, with a significant number of update operations. In the existing mid-tier system for this scenario, the data has to be first written to incremental tables, and a daily schedule is employed to merge and obtain the final results. As the Figure 6 described, With the adoption of data lake technology, data is directly written into the data table using a read-time copy mode to accelerate the writing speed. Hudi's write-merge logic consumes resources, and simultaneous writing and merging impact the writing speed. In a scenario where scheduling and merging are separated, the writing program generates log files in real-time, while the merging program is cyclically scheduled by the scheduling module.

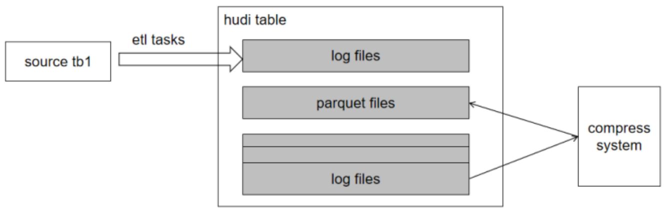

Fig 6. Large-Scale Data Update and Merge

5. Conclusions

The current integrated construction system for the power grid scenario based on Hudi involves the overall development of the data lake-warehouse system, encompassing data ingestion, storage, management, analysis, and utilization. The goal is to enhance data timeliness, reduce usage costs, and achieve dynamic scaling and resizing while ensuring data accuracy and stability. Built around Hudi as the core data storage component, the system implements storage-compute separation and supports dynamic scaling and resizing. It leverages big data computing frameworks such as Spark/Flink and MPP libraries like Doris to address various data analysis scenarios. This approach provides effective support for maintaining a balance between storage and computing resources for the company's big data tasks, as well as ensuring the timeliness of data analysis.

References

[1] Ait, E.S., Hajji, H., Ait, E.K.K., Badir, H. (2023) Spatial big data architecture: From Data Warehouses and Data Lakes to the LakeHouse. Journal of Parallel and Distributed Computing, 176: 70-79

[2] Idowu, E.A.A., Teo, J., Salih S., Valverde, J., Yeung J.A. (2023) Streams, rivers and data lakes: an introduction to understanding modern electronic healthcare records. Clinical medicine, 23: 409-409

[3] Jiménez, P., Roldán, J.C., Corchuelo, R. (2022) On exploring data lakes by finding compact, isolated clusters. Information Sciences, 591: 103-127

[4] Colleoni, C.J., Teixeira, B.O., Dubugras, R.D. (2022) Data integration in a Hadoop-based data lake: A bioinformatics case. International Journal of Data Mining & Knowledge Management Process, 12: 1-24

[5] Joe, M. (2021) The New Data Analytics: Riding on Data Lakes, Data Warehouses, and Clouds. Database Trends and Applications, 35: 4-8

[6] Kimia, A., Maryam, G. (2023) Big Data Analytics Capability and Firm Performance: Meta-Analysis. The Journal of Computer Information Systems, 63: 1477-149

[7] Ford, E.W. (2020) Data Streams, Data Lakes, and Information Pipelines: Enter the Chief Research Information Officer. Journal of Healthcare Management, 65: 379-381