Real-time Calculation Method of Big Data Index Based on Storm

Shuang-zhou GUO1,∗, Jin-lan LIANG2 and Jian-yun YU2

1Department of Information and Intelligent Engineering NingBo City College of Vocational Technology, Zhejiang Ningbo China
2Ningbo City College of Vocational Technology, Zhejiang Ningbo China

∗Corresponding author

Keywords: Big data, Storm, Statistics, Indexes, Flow processing.

Abstract. The real-time computing method of big data index based on stream processing is composed of logs collection, message management, coordination management, real-time processing, and other parts. This paper presents the discussion of the technical framework, implementation method and path of the real-time computing method of big data index based on Storm, and provides the algorithm verification process and result analysis.

Introduction

Big data's processing system can probably be divided into two categories, that is, batch processing and flow processing system[1]. Batch processing big data system (represented by Hadoop) needs to "aggregate the data in batches" first, after batch pretreatment, it can be loaded into analytical data warehouse, which can be used for off-line "real-time query" with high performance. Although this batch processing system can realize efficient real-time query to complete big data set, there is no way to query incremental real-time on-line data, which has the problem of data delay. Compared with batch processing big data system, stream processing is a large number. According to the typical application of real-time processing technology[2], it is an infinitely growing, borderless set of dynamic data, and the stream processing system represented by Spark Streaming, Storm, and Flink does not need to store big data. The data can be pre-processed on-line in real-time and efficiently, and loaded into the high-performance memory database for query in an all-round and piece-by-piece manner.

This paper studies an index real-time computing model based on database log analysis, stream processing, memory computing and distributed technology, which is a real-time index calculation method based on stream processing technology.

Overview of Flow Processing Technology

The flow processing system can meet the need of real-time computation of the data entering the system. Compared with the batch processing system such as Hadoop, Spark, the processing mode is very different. Stream processing is more like a general model for Map Reduce computing, except that its response time can reach the order of seconds or even milliseconds. Stream processing does not need to compute complete data samples, but only for each data item through the system. A flow processing system can theoretically handle an infinite amount of data, but at the same time, it can handle only one piece of data. (True stream processing) or a small amount of data (micro batch processing), leaving only a minimal amount of data between different records. Stream processing is a combination of data enrichment, continuous processing, and analysis of unbounded data in a general sense. The flow processing mode is suitable for tasks with near-real-time processing requirements, such as real-time product recommendation based on website user behavior, real-time calculation of business indicators, customer credit audit, and business audit, anti-fraud and so on.
**Real-time Calculation Scheme of Indexes Based on Storm**

The real-time calculation of indexes based on flow processing technology is the real-time parsing of log and database operation instructions by monitoring and capturing database logs in real-time, and using stream processing technology to analyze the log and database operation instructions in real-time. And real-time analysis results are used in the index calculation of big data processing model.

Flink is an emerging project, and Storm has matured over the years, with lower processing latency than Spark Streaming, or even milliseconds. We can completely update the need for real-time updates with the full index, so we choose Storm is the core technology of real-time processing.

Hadoop is a batch processing system. It has many advantages, such as large data throughput, automatic fault tolerance and so on, so it is widely used in the processing of massive heterogeneous big data. However, Hadoop is suitable for off-line processing of mass data and is not good at real-time computing because it is originally developed for batch processing, which is also the consensus of all of us. However, Hadoop can be used as the basic framework platform for running components such as Storm.

Therefore, the whole real-time computing system is based on Hadoop platform, which is composed of log collection, message management (Kafka), coordination management (Zookeeper), real-time processing (Storm), memory database (Redis) and other parts.[3]

The log collection module (plug-in developed by Shell, C, Java and other scripts) monitors the database log and pushes the log down to the Kafka distributed message management system in real time. The message in Kafka is consumed through Storm system, and the consumption record is managed by Zookeeper at the same time, the log is analyzed and processed by Storm according to the logic of index calculation, and output to Redis memory database. Finally, the results in the Redis are read by the application and displayed to the user, or transferred to the database for persistent storage.[4] From the technical framework, from the top down, first is the source-side database, the next layer is the log collection part, you can collect multiple databases at the same time, under the log collection is the Kafka message management system. Under the Kafka message management system layer is the Storm real-time processing layer, the coordination management between Kafka and Storm is undertaken by Zookeeper, under the Storm stream processing layer is Redis in-memory database, and the lowest level is Web or The App application can also include a database for persistence (such as a column database such as Hbase), as shown in figure 1.[5]

![Figure 1. Technical Architecture of Real-time Analysis system based on Storm.](image)

Log Collection

Log collection programs or tools are available in a variety of ways. Three kinds of script are listed here: one is to write Shell, C or Java script program, self-made script is lightweight and completely autonomous and controllable, the pressure on the server is relatively small, but it needs a certain amount of independent development; the second is the use of third-party framework technology for
direct collection, such as Flume. Flume is a distributed, efficient log collection system, which can collect massive log files distributed on different servers into a centralized storage resource, but Flume Configuration is not simple, the relationship of Source, Channel, Sink is intertwined in the configuration file, very difficult to manage; Another way is to use CDC (Change Data Capture) products to capture the source-side database logs, but CDC usually needs to be purchased separately and software is installed on both the source-side database and the target-side data at a higher cost. Server performance requirements are also high.

In this scheme, the self-developed script program is used to collect and directly deploy the collection script in the source-side database server to monitor and read the log files in the disk device in real-time. This approach has three advantages over the other two:

First, the deployment is simple, it can be directly deployed in the source-side database server, there is no need to build a synchronization server, Flume and CDC need a number of servers to deploy;

Second, fast response speed, real-time capture incremental log can be achieved, and CDC real-time synchronization interval is usually several seconds, cannot meet the requirements;

Thirdly, only disk file reading is involved, and the CPU and memory resources of the source-side server are less than that of the source-side server, which has little effect on the operation of the server.

Message Management (Kafka)

Kafka is a message system based on log file. The message can be persisted to the hard disk and the data is not easily lost. Kafka can save the progress and location of the message. For the user, it can also define the starting point of the consumption, and realize the repeated and multiple consumption of the message. Kafka has two kinds of message consumption patterns: queue and publish/subscribe, which can ensure that messages in message queue can be consumed in order and fit well with Storm. In addition, Kafka's Consumer is pull-based. Ed model, which can relieve the pressure that the log production speed is faster than the consumption speed, and make the consumption speed match the production speed reasonably. Placing the Kafka messaging system in the middle of the log collection and the Storm module is to prevent blowout growth in the event of sudden, high concurrency, because logging may occur. If the consumption speed of Storm can not be faster than that of log at this time, it will lead to a large number of message processing lag, and then lead to loss, so the Kafka message system is added as a data buffer.

Real-time Processing (Storm)

Storm can implement the programming and extension of complex real-time computation relatively simple. Real-time processing of database data will use Storm, just as Hadoop is often used for offline batch processing, and Storm guarantees that every message can be processed without omission and faster.

Log Resolution. Different types of database products have different log encoding rules and storage logic. Parsing logs needs to first study the rules of database log encoding and storage, which is one of the prerequisites for the whole computing mode to function properly. Otherwise, logs cannot be parsed and converted into easy-to-identify information. After receiving the log messages, the log resolver will automatically split the logs according to the log rules of the database, identify the log types, eliminate the rollback and so on, and do not change the log types of the data, leaving only the additions and deletions. Change the log generated by the operation and convert the contents of each page of that part of the log from hexadecimal encoding Switch to Standard and available data for next instruction parsing. The whole log parsing process can be divided into four parts: capture, segmentation, identification and transformation, as shown in figure 2.
**Instruction Parse.** Based on the results of the log parsing section, the instruction parsing part will select all the operational information involved in the index calculation from the log in accordance with the algorithm requirements of the big data index to be counted, and parse each addition and deletion. The data table and field information affected by the instructions involved in the modification operation are extracted, and the information used for data filtering and calculation, especially the corresponding incremental change information of the data, is extracted. To put it simply, all the information needed to calculate the index is obtained by instruction parsing, and then the result is pushed to the memory program for the next step. The instruction parsing process can be divided into identification, screening, Parse and push four parts, as shown in figure 3.

Figure 3. Instruction parsing process diagram.

**Program advantage**

The above-mentioned real-time index calculation scheme based on flow processing technology has adopted more mature mainstream technologies and tools, and can realize real-time, on-line and continuous processing of big data. It can meet the real-time requirement of business and management decision-making. Compared with traditional data warehouse, BI, data acquisition (such as CDC, etc.), this scheme has five advantages.

First, the processing efficiency, from capturing the database log to the completion of the real-time calculation of indicators to achieve milliseconds;

Second, it has little influence on the source-side database server. Because of the direct reading of the server disk log files, it does not involve the management and interaction of the database system level, so it basically does not occupy the CPU and memory resources of the source-side server.

Third, high reliability, message queue (Kafka) and coordination system (Zookeeper) ensure that the log can be processed one by one, and the whole cluster can recover quickly after downtime.

Fourth, low cost, stream processing cluster based on X86 architecture server construction, low price, procurement, maintenance and upgrade simple;

Fifth, the scalability is strong, the whole system adopts Hadoop distributed cluster architecture, the linear improvement of processing ability can be realized by adding hardware equipment.

**Major Problems**

At present, there are two relatively big problems in the real-time calculation method of big data index based on Storm. One is the database product upgrade may bring about log format changes. The database product manual does not specify the coding rules for logs, the types of logs, and other information that needs to be studied by itself, if a change in the version of the database results in a change in the log encoding or storage rules. In this case, it is necessary to re-examine the log rules before upgrading, and then adjust the log parsing algorithm accordingly. Second, the adjustment of index caliber may lead to the change of system processing logic. If the caliber is adjusted, the logic of the algorithm changes. Transformation, such as statistical fields or data filtering conditions, requires the adjustment of instruction parsing and real-time computations. In both cases, especially the second problem, if it happens frequently, it may take a lot of time and manpower to complete the corresponding transformation work.

**Summary**

The real-time calculation method of big data index based on Storm is still in the research stage, it still needs further test and optimization, the stability needs to be further improved, and the processing ability also has the space of mining. As far as the practical effect of application is concerned, there is basically no problem in real-time calculation of a small number of indicators for enterprises with moderate data size, without requiring a large amount of capital input. However, the research of data log rules and the parsing efficiency of a single database log are still constraints to
the implementation of mass index calculation, so the foundation of large-scale application still needs to be further tamped down. But With the rapid progress of big data's technology, more mature and powerful components or products will emerge, and the stream processing ability will be more and more powerful through continuous upgrading and adjustment of this technology in the future.

References

[1] Jones MT. Process real-time big data with Twitter Stom [J]. IBM Technical Library 2018.

[2] CSDN. Technical selection reference for large data flow processing platform. https://blog.csdn.net/uxiAD7442KMy1X86DtM3/article/details/79554077, 2018-03-14.

[3] Leibiusky J, Eisbruch G, Simonassi D. Getting started with storm[M], “O’ Reilly Media, Inc.”, 2018.

[4] Stancevic D. Zero copy l: user-mode perspective[J]. Linux Journal 2018, 2018(105):3.

[5] CSDN. Technical selection reference for large data flow processing platform. https://blog.csdn.net/uxiAD7442KMy1X86DtM3/article/details/79554077. 2018-03-14

[6] Leibiusky J, Eisbruch G, Simonassi D. Getting started with storm[M], ”O’ Reilly Media, Inc.”, 2018.

[7] Stancevic D. Zero copy l: user-mode perspective[J]. Linux Journal 2018, 2018(105):3.

[8] Hatley D, Pirbhai l. Strategies for real-time system specification[M]. Addison-Wesley, 2018.