The Course Report of the Design and Application of Time-Series Database

Topic: Why do we need a time-series database?
2019141460481 Liu Yuguo

1. INTRODUCTION

With a time-series database (TSDB), it's possible to add, process, and track massive quantities of time series data. They do this efficiently and continuously, with lightning speed and precision. Other types of databases also work for these workloads, and have indeed been used in the past, TSDBs haven specific algorithms and architecture to meet the requirements of speed and high volumes.

A time-series database stores data as pairs of time(s) and value(s). Storing data this way makes it easy to analyze sequences of points recorded in order over time. A TSDB can handle concurrent series, measuring many different variables or metrics in parallel.

Early time-series databases were mostly used for processing volatile financial data and streamlining securities trading. The world's changed a lot since they were first introduced, and many new use cases have emerged as technology has continued to evolve.

For example, the Internet of Things (IoT) concept employs sensors that constantly collect and stream data. IoT technology is used for multiple purposes, from powering industrial applications to predicting sales demand, from weather monitoring to wearable fitness devices. The amount of data they produce is staggering.

Well, it can be staggering for more traditional databases, which were designed for different purposes. Luckily, there are time series databases, and they are getting better and better at dealing with the growing demand of this data type.

The rest of this article will discuss the topic as follow:

- The brief introduction to time-series database.
- The benefit/strength of time-series database comparing to the traditional.
- The examples of time-series database.
- The future of time-series database

2. INTRODUCTION TO TIME-SERIES DATABASE

A time series database (TSDB) is a database optimized for time-stamped or time series data. Time series data are simply measurements or events that are tracked, monitored, downsampled, and aggregated over time. This could be server metrics, application performance monitoring, network data, sensor data, events, clicks, trades in a market, and many other types of analytics data.

A time series database is built specifically for handling metrics and events or measurements that are time-stamped. A TSDB is optimized for measuring change over time. Properties that make time series data very different than other data workloads are data lifecycle management, summarization, and large range scans of many records.

Time series databases are not new, but the first-generation time series databases were primarily focused on looking at financial data, the volatility of stock trading, and systems built to

solve trading. But financial data is hardly the only application of time series data anymore — in fact, it's only one among numerous applications across various industries. The fundamental conditions of computing have changed dramatically over the last decade. Everything has become compartmentalized. Monolithic mainframes have vanished, replaced by serverless servers, microservers, and containers.

Today, everything that can be a component is a component. In addition, we are witnessing the instrumentation of every available surface in the material world — streets, cars, factories, power grids, ice caps, satellites, clothing, phones, microwaves, milk containers, planets, human bodies. Everything has, or will have, a sensor. So now, everything inside and outside the company is emitting a relentless stream of metrics and events or time series data.

This means that the underlying platforms need to evolve to support these new workloads — more data points, more data sources, more monitoring, more controls. What we're witnessing, and what the times demand, is a paradigmatic shift in how we approach our data infrastructure and how we approach building, monitoring, controlling, and managing systems. What we need is a performant, scalable, purpose-built time series database.

3. THE BENEFIT OF TIME-SERIES DATABASE COMPARING TO THE TRADITIONAL

The era of big data has come for many years, big data solutions are basically mature, and the Hadoop cluster processing solution has basically become a best practice for processing big data. The data he processes includes structured, semi-structured, and unstructured data, collects data through Sqoop, Flume, kafka, stores data through hbase, hdfs, calculates data through mapreduce, sparkstreaming, etc., and finally uses hive as a data warehouse for applications Layers provide the required data.

This is a set of general-purpose, comprehensive big data solutions.

If we subdivide the data types, how should we optimize the storage for a large amount of time series data?

For time series data, we summarize the following characteristics:

1. Data characteristics: The amount of data is large, the data grows with time, the same dimension is repeatedly taken values, and the indicators change smoothly (the track coordinates of a vehicle that are uploaded by a certain device to smoothly change).

2. Writing features: high concurrent writing, and will not be updated (the trajectory will not be updated).

3. Query characteristics: Statistical analysis is performed on indicators according to different dimensions. There are obvious hot and cold data. Generally, only recent data will be queried (generally, we will only care about recent trajectory data).

Features can be improved as follows:

The first feature is that the amount of data is large, and the same dimensions are repeated values. We can compress and store these same dimensions (because they are repeated) to reduce storage costs. For example, it is good to store only one copy of the repeated host and port. .

The second feature, high concurrent writes, like hbase, we can use LSM instead of B-tree

The third feature is aggregation and hot and cold data. We can store cold data with reduced precision, that is, aggregate historical data to save storage space.

There is no problem at first glance, but after I carefully studied the principle of time series

database, I have a new understanding of this.

First of all, for the first improvement, we can put these same dimensions into traditional databases such as mysql, and then generate a unique id for each dimension, and then store the changed indicator data and unique id in hbase, so that no There is a problem of repeated storage of the same dimension. At most, it is just to check mysql once more when querying. In the case of few dimensions, it is completely acceptable.

Not to mention the second improvement, hbase is fully satisfied.

The third improvement is just an optimization point and cannot be a decisive factor.

So what are the essential differences between time series databases and traditional big data storage solutions?

I think the most important difference is structured data.

1. It stores structured data. We all know that the data to be stored in traditional big data solutions includes structured, semi-structured, and unstructured data, which determines that we cannot decide which fields and the data types that define each field, such as hbase through the byte type. Unified storage, that is to say, the data placed in hbase are all byte arrays. We need to do it ourselves to convert from ordinary types to byte arrays. We don't know how to convert them into byte arrays, and their storage efficiency will be higher. However, the data generated by time series data is structured data. We can define the fields and types of the data in advance, and let the database system choose the optimal compression method according to different field types, which greatly improves the storage utilization.

2. Analysis and aggregation is structured data. Since the analysis and aggregation is structured data, then we do not need to use complex computing tools such as map-educe, nor do we generally need data warehouses such as hive, but only need to cohere at the database storage level, similar to sum and avg in this calculation. The tools are enough, and even some simple stream computing can be done, which provides the basis for 'hyper-convergence' (hyper-convergence means that multiple components similar to the previous big data processing solutions are merged into one component, mainly because Structured data is too simple, and collection and calculation are relatively simple, which is also the development trend of subsequent time series databases, reducing system complexity).

4. THE EXAMPLES OF TIME-SERIES DATABASE

4.1 Influxdb

Influxdb is a popular time series database in the industry, especially in the fields of IOT and monitoring. It is developed using the go language, and its outstanding feature is performance.

Features:

- Efficient time series data write performance. Custom TSM engine, fast data writing and efficient data compression.
- No additional storage dependencies.
- Simple, high-performance HTTP query and write API.
- Plug-in support for data ingestion of many different protocols, such as: graphite, collected, and OpenTSDB
- SQL-like query language that simplifies query and aggregation operations.
- Index Tags, support fast and efficient query time series.
- Retention policies effectively remove outdated data.

- Continuous queries automatically compute aggregated data, making frequent queries more efficient.

Influxdb has converted the distributed version to closed source. Therefore, it is a weakness in the distributed cluster and needs to be implemented by yourself.

4.2 OpenTSDB

The Scalable Time Series Database. Open the official website of OpenTSDB, and the first thing you see is this sentence. It has Scalable as its important feature. OpenTSDB runs on Hadoop and HBase, which takes full advantage of HBase's features. The service is provided through a standalone Time Series Demon (TSD), so it can be easily scaled up or down by adding or removing service nodes.

- Opentsdb is a time series database based on Hbase (the new version also supports Cassandra). Its distributed column storage feature based on Hbase realizes the features of high data availability and high-performance writing. Limited by Hbase, the storage space is large and the compression is insufficient. Depends on the whole set of HBase, ZooKeeper
- Use a schemaless tagset data structure (sys.cpu.user 1436333416 23 host=web01 user=10001). Simple structure, multi-value query is not friendly
- HTTP-DSL query

OpenTSDB's table design and RowKey design for TSDB on HBase is a feature worthy of our in-depth study. Interested students can find some detailed information to study.

4.3 Druid

Druid is a real-time online analysis system (LOAP). Its architecture integrates the characteristics of real-time online data analysis, full-text retrieval system and time series system, so that it can meet the data storage requirements of different usage scenarios.

- Columnar storage: supports efficient scanning and aggregation, and is easy to compress data.
- Scalable distributed system: Druid itself implements a scalable, fault-tolerant distributed cluster architecture. Deployment is simple.
- Powerful parallel capability: Druid cluster nodes can provide query services in parallel.
- Real-time and batch data ingestion: Druid can ingest data in real-time, such as through Kafka. Data can also be ingested in batches, such as by importing data through Hadoop.
- Self-healing, self-balancing, and easy operation and maintenance: Druid's own architecture achieves fault tolerance and high availability. Different service nodes can add or subtract nodes according to the needs of the response.
- Fault-tolerant architecture to ensure that data is not lost: Druid data can retain multiple copies. In addition, HDFS can be used as deep storage to ensure that data is not lost.
- Index: Druid implements reverse encoding and Bitmap index for String columns, so it supports efficient filter and groupby.
- Time-based partitioning: Druid partitions the original data based on time, so Druid will be more efficient for time-based range queries.
- Automatic pre-aggregation: Druid supports pre-aggregation of data during data ingestion.

Druid architecture is quite complex. It subdivides the entire system into multiple services according to functions. The systems with different responsibilities of query, data, and master are deployed independently to provide unified storage and query services to the outside world. It

provides an underlying data storage service in the form of distributed cluster services.

4.4 TDengine

TDengine is an innovative big data processing product launched by Taosi Data in the face of the rapid growth of the IoT big data market and technical challenges. It does not rely on any third-party software, nor does it optimize or package an open source database or stream computing. It is a self-developed product after absorbing the advantages of many traditional relational databases, NoSQL databases, streaming computing engines, message queues and other software. It has its own unique advantages in the processing of time series space big data.

One of the modules of TDengine is the time series database. But in addition, in order to reduce the complexity of research and development and the difficulty of system maintenance, TDengine also provides functions such as caching, message queue, subscription, streaming computing, etc., providing a full-stack technical solution for the processing of big data in the Internet of Things and Industrial Internet , is an efficient and easy-to-use IoT big data platform. Compared with typical big data platforms such as Hadoop, it has the following distinctive features:

- **More than 10 times performance improvement**: An innovative data storage structure is defined. A single core can process at least 20,000 requests per second, insert millions of data points, and read more than 10 million data points, which is faster than existing general-purpose databases. more than ten times.
- **Hardware or cloud service costs are reduced to 1/5**: due to super performance, computing resources are less than 1/5 of general big data solutions; through columnar storage and advanced compression algorithms, storage space is less than 1/10 that of general-purpose databases.
- **Full-stack time series data processing engine**: integrates database, message queue, cache, stream computing and other functions, and applications do not need to integrate Kafka/Redis/HBase/Spark/HDFS and other software, greatly reducing the complexity and cost of application development and maintenance.
- **Powerful analysis functions**: Whether it is data ten years ago or one second ago, you can query it by specifying a time range. Data can be aggregated on a timeline or across multiple devices. Ad-hoc queries can be performed anytime through Shell, Python, R, Matlab.
- **Seamless connection with third-party tools**: Integrate with Telegraf, Grafana, EMQ, HiveMQ, Prometheus, Matlab, R, etc. without a single line of code. In the future, OPC, Hadoop, Spark, etc. will be supported, and BI tools will also be seamlessly connected.
- **Zero operation and maintenance cost and zero learning cost**: The cluster installation is simple and fast, no need for sub-database and sub-table, and real-time backup. Similar to standard SQL, supports RESTful, supports Python/Java/C/C++/C#/Go/Node.js, similar to MySQL, zero learning cost.

With TDengine, the total cost of ownership of typical IoT, Internet of Vehicles, and Industrial Internet big data platforms can be greatly reduced. However, it should be pointed out that due to the full use of the characteristics of IoT time series data, it cannot be used to process general-purpose data such as web crawlers, Weibo, WeChat, e-commerce, ERP, and CRM.

5. THE FUTURE OF TIME-SERIES DATABASE

Time series database is in the stage of rapid development, and time series data technology is gradually becoming mature, but this is by no means the end. Time series data technology also faces various new requirements and challenges. While improving the performance of time series databases, major manufacturers are proposing more solutions for new requirements:

(1) Cloud services. In addition to the stand-alone version, many manufacturers have also released distributed versions and cloud service versions, especially cloud services, which have become an inevitable development trend.

(2) Visualization services. With the advent of the Internet of Everything, users' demand for comprehensive information is increasing, and the visualization of time series data has become a major trend, which puts forward higher requirements for the query ability of time series databases.

(3) Edge computing services. In the era of the Internet of Everything, the huge amount of data brought by more sensors is unbearable for centralized processing, which makes data computing develop to the edge. The device processes and analyzes the data in real time through the edge device and then stores it centrally. , which can improve the real-time response capability of the device and enhance the value of time-sensitive data. Therefore, the support of the time series database for edge computing will become an important function of it.

In the face of these challenges and opportunities, I believe that time series databases will have a deeper development, and the future can be expected.