**Why do we need a time-series database?**

 In the context of big data, if we subdivide the data types, each subdivision type of data will have a certain storage optimization space, It mainly depends on whether the optimization has added a set of subdivision type data processing scheme to make it more valuable (of course, if our system only has data of a certain subdivision type, it will be much better to select the processing scheme of subdivision type).

 ​  Time series data is the subdivision of a data type under the background of big data. With the advent of the Internet of things era, the total amount of time-series data is increasing. Special processing schemes for time-series data types are constantly emerging. They are called time-series databases. Next, we will begin to uncover the mystery of time-series databases!

the era of big data has come for many years, big data solutions are basically mature, and Hadoop cluster processing solution has basically become a best practice for handling big data. The data he processes includes structured, semi-structured and unstructured data,Data are collected by sqoop, flume, Kafka, HBase and HDFS are used to store data, MapReduce and spark processing are used to calculate data. Finally, hive is used as a data warehouse to provide data for application layer .

 this is a general, comprehensive big data solution. If we subdivide the data type, how can we optimize the storage for a large number of time series data?

First of all, what is the timing?

 simply put, time series data is the data indexed by time dimension, such as vehicle trajectory data, sensor temperature data. With the advent of the Internet of things era, the amount of time series data is blowout, so the optimization of storage for this data segmentation is becoming more and more important.

What are the characteristics of time series data? For these characteristics, how can we optimize storage?

for time series data, we summarize the following characteristics:

 ​  1. Data characteristics: large amount of data, data growth, repeated values of the same dimension, smooth change of indicators (track coordinates of smooth changes uploaded by some equipment of a certain vehicle).

 ​  2. Write characteristics: high concurrent write, and will not update (trajectory will not be updated).

 ​  3. Query characteristics: according to different dimensions of statistical analysis of indicators, there are obvious cold and hot data, generally only recent data will be queried (generally, we only care about the recent trajectory data).

It can be improved as follows:

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

the second feature is high concurrent write. Like HBase, we can use LSM instead of B-tree.

the third feature is aggregation and hot and cold data. We can reduce the precision storage of cold data, that is, aggregate historical data to save storage space.

For the first time, we can store the same index in MySQL database, and then store the same data in MySQL database for the first time, In the case of few dimensions, it is completely acceptable. for the second improvement, not to mention HBase is completely satisfied. the third improvement is only an optimization point and cannot be a decisive factor.

So what are the essential differences between sequential database and traditional big data storage solutions?

I think the most important difference is structured data.

 ​  1. Structured data is stored. We all know that the data to be stored in the traditional big data scheme includes structured, semi-structured and unstructured data, so we can't decide which fields to have and define the data types of each field. For example, HBase is uniformly stored by byte type, that is, the data put into HBase is byte array, Converting from a normal type to a byte array requires us to do it ourselves. We don't know how to convert it to a byte array. Its storage efficiency will be higher. However, the data generated by time series data are structured data. We can define the fields and types of data in advance, so that the database system can choose the optimal compression method according to different field types, which greatly improves the utilization of storage.

 ​  2. Analysis and aggregation is structured data. Since analysis aggregation is structured data, we don't need to use MapReduce or hive data warehouse. Instead, we only need to cohere sum and AVG in the database storage level, and even do some simple flow calculation, It provides the foundation for "super fusion" (hyper fusion means that many components similar to the previous big data processing scheme are integrated into one component, mainly because the structured data is too simple, and the collection and calculation are relatively simple. This is also the development trend of the subsequent time series database, reducing the system complexity).

let's paste an introduction to the data model in the tdengine official document:

   data model:

Because the collected data are generally structured data, and in order to reduce the learning threshold, tdengine uses the traditional relational database model to manage the data. Therefore, users need to create a library first and then create a table before inserting or querying data.

Yes, you need to define data types first.

Besides, there are some general design ideas of time series database Let's take tdengine as an example.

 1. One device is a table, that is to say, the data of a device is stored in one place, so that continuous query can be conducted during query, and the query speed is accelerated. Of course, this table can be created dynamically (equivalent to HBase mandatory clustering by device, which is not a technical innovation, but a design innovation).

 2. There is a super table that stores the label attribute of the device, which is the storage that we mentioned in that can be replaced by mysql. There are two advantages of using super table: first, if there is no other requirement, you can not use Mysql to increase the system complexity; second, we can move some aggregation operations we do in the business system to the database system, Reduce the development difficulty (for example, for the above data, we need to query the data whose host is host4, then in the traditional process, we need to find out the devices with host 4 in the database, and then query the data in HBase one by one, and then aggregate. Now, if we look at the above data, the sequential database can help us complete all this).

What is the difference between influxdb and tdengine in usage scenarios, and why is there a big difference in performance?

Core difference: tdengine believes that all data should be submitted in an orderly manner, and there are no updates and deletions. Infixdb can support data reporting in reverse order, and infixdb is like crud and supports UD, but UD is not recommended.

mainly because of this idea, tdengine has the advantages of storage and query.

First of all, the memory consumption of lseng is greatly accelerated when the memory is not updated from 95% to 95% of the disk.

secondly, when querying, we do not need to consider the update and deletion statements in memory. For example, in the LSM of HBase mentioned in the previous article, we do not need to query all storefiles, which greatly speeds up the query speed.

finally, in the aggregate query, because our data is invariant, we can pre aggregate the invariant data files. For example, if we save a vehicle track file on February 3, we can know the maximum and minimum value of the file longitude and latitude in advance, which will greatly speed up the aggregate query.

 there are also some small differences:

 ​  1. If the design of tdengine is not the same, it is necessary to collect multiple sets of sequences. This means that multiple columns of each table can share the same timestamp, instead of having one timestamp for each column. The design idea of infixdb is that multiple sets of collection data can be in the same table, and each collection quantity has a timestamp mark, which will increase the storage. For example, in the vehicle network system, the track and temperature of vehicles are reported separately. In the design idea of tdengine, two tables need to be established for the same equipment, and the data in the same row in each table uses the same timestamp (which will cause multiple tables). In infixdb, trace and temperature are in the same table, because they are all of the same device, then each attribute in the same row needs a timestamp to identify.

 ​  2. In order to speed up the query, tdengine puts the super table in the memory. However, if the super table is too large (the probability of occurrence is not large, because the number of devices is not large), the memory will not be able to hold it, and tdengine will have problems. However, infixdb stores the measurement (similar to Super Table) on disk, and uses inverted index to query, so the capacity is larger.

 ​  4. Tdengine will move a large number of calculations to the client, which will increase the burden of the client and increase the complexity of the client. It is not lightweight and needs to install the client. But infixdb is calculated on the server side (similar to tdengine restful)