Time Series Data Models

Time series data is a set of data indexed by time. Simply put, this type of data describes the measurements of a measured subject at each time point of a time range.

Modeling of time series data includes three important parts: subject, time point, and measurements. Applying this model, you will find that you are in constant contact with this type of data in your daily work and life.

- If you are a stockholder, the stock price of a stock is a type of time series data. It records the stock price of the stock at each time point.
- If you are O&M personnel, the monitoring data is a type of time series data. For example, the monitoring data of the CPU records the actual usage of the CPU at each time point.

The world is made up of data, and every object in the world is producing data all the time. The exploitation and use of these types of data is silently changing people's lifestyles in this era. For example, the core of the personal health management feature of wearable devices is to continuously collect your personal health data. Such data includes heart rate and body temperature. After collecting such data, the device uses a model to calculate and evaluate your health status.

If you free your vision and your imagination, you will find that all data in your daily life can be exploited and used. Objects that generate data include your mobile phone, car, air conditioner, refrigerator, and so on. The core idea of the currently hot Internet of Things (IoT) technology is to build a network that collects data generated by all objects, and exploits the value of the collected data. Data collected by this network is typical time series data.

Time series data is used to describe the state change information of an object in the historical time dimension. The analysis of time series data is the process of trying to understand and master the rules of the changes. Time series data experiences explosive growth with the development of IoT, big data, and artificial intelligence (AI) technologies. To better support the storage and analysis of such data, a variety of database products have come into being and are available on the market. The invention of this kind of database products is to solve the shortcomings and drawbacks of the conventional relational databases in terms of time series data storage and analysis. These products are uniformly classified as time series databases (TSDBs).

As can be seen from the ranking of the most popular database management systems on DB-Engines, the popularity of TSDBs has maintained a high growth rate over the last two years.

Later on, I will write a few articles to analyze:

1. The basic concepts of time series data, including models, characteristics, basic queries, and processing operations.
2. The analysis of the underlying implementation of several popular open source TSDBs.

A data model of time series data mainly consists of the following parts:

- Subject: The subject to be measured. A subject may have attributes with multiple dimensions. Taking server status monitoring as an example, the measured subject is a server, and its attributes may include the cluster name, host name, and so on.
- Measurements: A subject may have one or more measurements, each corresponding to a specific metric. In the case of the server status monitoring, the measured metrics may include CPU usage and IOPS. The value of CPU usage is a percentage, and the value of IOPS is the number of I/Os during the measurement period.
- Timestamp: The measurement report is always attached with a timestamp attribute to indicate the time.

Currently, mainstream TSDBs use two modeling methods: modeling by data source and modeling by metrics. I will use two examples to illustrate the difference between these two methods.

Modeling by Data Source

Timestamp	cluster	hostname	cpu	Iops
2015-04-28 T 17 50：00 Z	Cluaster-A	Host-a	10	10
2015-04-28 T 17：50：10 Z	Cluster-A	Host-b	20	30
2015-04-28 T 17：50：20 Z	Cluster-A	Host-a	5	8
Timestamp	Subject dimenson		Metrics and measurements	

The above is an example of modeling by data source. Measurements of all metrics of the same data source at a certain time point are stored in the same row. This model is used by Druid and InfluxDB.

Modeling by Metrics

Metric	Timestamp	Cluster	Hostname	Metric value
Cpu	2015-04-28 T 17：50：00 Z	Cluster-A	Host-a	10
Cpu	2015-04-28 T 17：50：10 Z	Cluster-A	Host-b	20
Cpu	2015-04-28 T 17：50：20 Z	Cluster-A	Host-a	5
Iops	2015-04-28 T 17：50：00 Z	Cluster-A	Host-a	10
Iops	2015-04-28 T 17：50：10 Z	Cluster-A	Host-b	30
Iops	2015-04-28 T 17：50：20 Z	Cluster-A	Host-a	8
Metrics	Timestamp	Subject dimension		Measurements

The above is an example of modeling by metrics, where each row of data represents a measurement of a certain metric of a data source at a certain time point. This mode is used by OpenTSDB and KairosDB.

There is no clear distinction between these two models. If the underlying layer architecture adopts columnar storage and there is an index on each column, modeling by data source may be better. If the underlying layer architecture is similar to HBase or Cassandra, storing multiple metric values on the same row may affect the query or filter efficiency on one of the metrics. Therefore, we typically choose to model by metrics.

Processing of Time Series Data

This section mainly describes the processing of time series data. In addition to the basic data writing and storage, query and analysis are the most important features of a TSDB. The processing of time series data mainly includes filter, aggregation, GroupBy, and downsampling. To better support GroupBy queries, some TSDBs will pre-aggregate the data. Downsampling is done through rollups. To support faster and more real-time rollups, TSDBs usually support auto-rollups.

Filter

Metric	Timestamp	Cluster	Hostname	Metric value
Cpu	2015-04-28 T 17：50：00 Z	Cluster-A	Host-a	10
Cpu	2015-04-28 T 17：50：00 Z	Cluster-A	Host-b	20
Cpu	2015-04-28 T 17：50：10 Z	Cluster-A	Host-a	6
Cpu	2015-04-28 T 17：50：10 Z	Cluster-A	Host-b	10
Cpu	2015-04-28 T 17：50：20 Z	Cluster-A	Host-a	30
Cpu	2015-04-28 T 17：50：20 Z	Cluster-A	Host-b	8

Select cpu from table where cluster = “cluster-a” and hostname = “host-a”

Metric	Timestamp	Cluster	Hostname	Metric value
Cpu	2015-04-28 T 17：50：00 Z	Cluster-A	Host-a	10
Cpu	2015-04-28 T	Cluster-A	Host-a	6

	17：50：10 Z			
Cpu	2015-04-28 T 17：50：20 Z	Cluster-A	Host-a	30

The above is a simple filter process. Simply put, it queries for all data that meets the given conditions of different dimensions. In the scenario of time series data analysis, filter usually starts from a high dimension, and then performs more detailed query and processing of data based on more-refined dimensional conditions.

Aggregation

Aggregation is the most basic function of time series data query and analysis. Time series data records the original state change information. However, when doing time series data query and analysis, we usually do not need the original information. Instead, we need the statistics based on the original information. Aggregation involves some basic computations for statistics. The most common computations are SUM, AVG, Max, and TopN. For example, when analyzing the server traffic, you would care about the average amount of traffic, the total amount of traffic, or the peak traffic.

GroupBy and Pre-Aggregation

Metric	Timestamp	Cluster	Hostname	Metric value
Cpu	2015-04-28 T 17：50：00 Z	Cluster-A	Host-a	10
Cpu	2015-04-28 T 17：50：00 Z	Cluster-A	Host-b	20
Cpu	2015-04-28 T 17：50：10 Z	Cluster-A	Host-a	6
Cpu	2015-04-28 T 17：50：10 Z	Cluster-A	Host-b	10
Cpu	2015-04-28 T	Cluster-A	Host-a	30

	17：50：20 Z			
Cpu	2015-04-28 T 17：50：20 Z	Cluster-A	Host-b	8

Select avg(cpu) from table group by Cluster

Metric	Timestamp	Cluster	Metric value(avg)
Cpu	2015-04-28 T 17：50：00 Z	Cluster-A	15
Cpu	2015-04-28 T 17：50：10 Z	Cluster-A	8
cpu	2015-04-28 T 17：50：20 Z	Cluster-A	9

GroupBy is the process of converting low-dimensional time series data into high-dimensional statistics. The above is a simple example of GroupBy. GroupBy is usually performed during query. After the original data is queried, we obtain the result through real-time computation. This process may be very slow, depending on the size of the originally queried data. Mainstream TSDBs optimize this process through pre-aggregation. After the data is written in real time, it will be pre-aggregated to generate the results after GroupBy according to the given rules. This allows us to directly query the results without re-computation.

Downsampling, Rollup and Auto-Rollup

Downsampling is the process of converting high-resolution time series data into low-resolution time series data. This process is called rollup. It is similar to GroupBy, but they are different. GroupBy is to aggregate data of different dimensions at the same time level based on the same time granularity. The time granularity of the converted data remains the same, but the dimension becomes higher. Downsampling is to aggregate data of the same dimension at different time levels. The time granularity of the converted data becomes coarser, but the dimension remains the same.

Metric	Timestamp	Cluster	Hostname	Metric value
Cpu	2015-04-28 T 17：50：00 Z	Cluster-A	Host-a	10
Cpu	2015-04-28 T 17：50：10 Z	Cluster-A	Host-a	20
Cpu	2015-04-28 T 17：50：20 Z	Cluster-A	Host-a	6
Cpu	2015-04-28 T 17：50：30 Z	Cluster-A	Host-a	10
Cpu	2015-04-28 T 17：50：40 Z	Cluster-A	Host-a	30
Cpu	2015-04-28 T 17：50：50 Z	Cluster-A	Host-a	8

Downsmapling from interval(10) to interval(30)

Metric	Timestamp	Cluster	Hostname	Metric value(avg)
Cpu	2015-04-28 T 17：50：10 Z	Cluster-A	Host-a	12
Cpu	2015-04-28 T 17：50：30 Z	Cluster-A	Host-a	16

The above is a simple example of downsampling, which aggregates the 10-second resolution data to 30-second resolution data, to obtain the statistical average.

Downsampling is divided into storage downsampling and query downsampling. Storage downsampling is to reduce storage costs of data, especially historical data. The query downsampling is mainly for queries with a larger time range to reduce the returned data points. Auto-rollup is required for both storage downsampling and query downsampling. Auto-rollup automatically performs a data rollup, rather than when it's waiting for a query. Similar to pre-aggregation, this process can effectively improve query efficiency. It is also a feature that has been or plans to be designed for the currently mainstream TSDBs. Currently, Druid, InfluxDB, and KairosDB support auto-rollup. OpenTSDB does not support auto-rollup, but it provides an API to support the import of results after auto-rollup is performed externally.