An Introduction to OpenTSDB

Kayee Jiang

**1. What is TSDB?**

Time series database (TDSB) is a data management system that provides efficient access to time series data and statistical analysis functions. It is widely used in Internet of Things (IoT) equipment monitoring systems, enterprise energy management systems (EMS), production safety monitoring systems, and power detection systems.

**2.What is OpenTSDB?**

OpenTSDB uses HBase to store all the time series (without sampling) to build a distributed and scalable time series database (TSDB). It supports second-level data collection of all metrics and permanent storage, can also do capacity planning, and is easily integrated into the existing alarm system. OpenTSDB can obtain corresponding metrics from large-scale clusters (including network devices, operating systems, and applications in the cluster) and store, index, and serve them, making these data easier to understand for humans, such as web-based and graphical Wait.

OpenTSDB is a scalable distributed time series database that relies on HBase at the bottom. As a typical representative of time series databases developed based on general storage, it started relatively early and has a relatively high degree of recognition in the time series market. It is a layer of data read and write services built on Hbase.

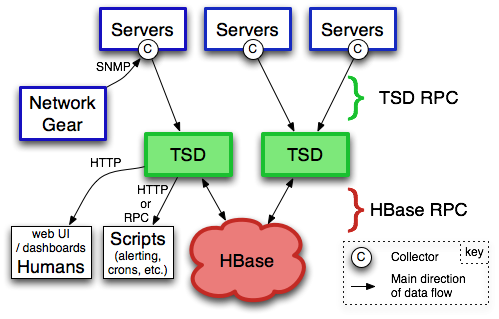
OpenTSDB was written to meet a common need: to store, index and provide metrics systems, applications collected from computer systems (network devices, operating systems), and to make this data easily accessible and crawlable.

For operation and maintenance engineers, OpenTSDB can obtain real-time status information of infrastructure and services, and display various software and hardware errors, performance changes and performance bottlenecks of the cluster. For managers, OpenTSDB can measure system SLAs, understand interactions between complex systems, and show resource consumption. The overall operation of the cluster can be used to assist in budgeting and coordination of cluster resources. For developers, OpenTSDB can show the main performance bottlenecks of the cluster, frequently encountered errors, so that they can focus on solving important problems.

OpenTSDB uses the LGPLv2.1+ open source protocol, and the current version is 2.X.

**3.How does OpenTSDB work?**

OpenTSDB consists of a Time Series Daemon (TSD) as well as set of command line utilities. Interaction with OpenTSDB is primarily achieved by running one or more of the TSDs. Each TSD is independent. There is no master, no shared state so you can run as many TSDs as required to handle any load you throw at it. Each TSD uses the open sources database [HBase](http://hbase.org/) or hosted Google Bigtable service to store and retrieve time-series data. The data schema is highly optimized for fast aggregations of similar time series to minimize storage space. Users of the TSD never need to access the underlying store directly. You can communicate with the TSD via a simple telnet-style protocol, an HTTP API or a simple built-in GUI. All communications happen on the same port (the TSD figures out the protocol of the client by looking at the first few bytes it receives).

OpenTSDB framework:

Servers：It is the server, C refers to Collector, collect data through Collector, push data；

TSD：TSD is a stateless service for external communication. Collector can push monitoring data through TSD's simple RPC protocol；

TSD also provides a webUI page for data query; in addition, you can query monitoring data through scripts, and alarm the monitoring data;

HBase：After TSD receives the monitoring data, it writes the data to HBase through the AsyncHbase library; AsyncHbase is a completely asynchronous, non-blocking, thread-safe HBase client that uses fewer threads, locks and memory, and can provide higher Throughput, especially for a large number of write operations.

**3.1 Writing**

The first step in using OpenTSDB is to send time series data to the TSDs. A number of [tools](http://opentsdb.net/docs/build/html/resources.html#clients) exist to pull data from various sources into OpenTSDB. If you can't find a tool for your needs, you may need to write scripts that collect data from your systems (e. g. by reading interesting metrics from /proc on Linux, collecting counters from your network gear via SNMP, or other interesting data from your applications, via JMX for instance for Java applications) and push data points to one of the TSDs periodically.

[StumbleUpon](http://www.stumbleupon.com/) wrote a Python framework called [tcollector](https://github.com/OpenTSDB/tcollector) that is used to collect thousands of metrics from Linux 2.6, Apache's HTTPd, MySQL, HBase, memcached, Varnish and more. This low-impact framework includes a number useful collectors and the community is constantly providing more. Alternative frameworks with OpenTSDB support include Collectd, Statsd and the Coda Hale metrics emitter.

In OpenTSDB, a time series data point consists of:

* A metric name.
* A UNIX timestamp (seconds or millisecinds since [Epoch](http://en.wikipedia.org/wiki/Unix_epoch)).
* A value (64-bit integer or single-precision floating point value), a JSON formatted event or a histogram/digest.
* A set of tags (key-value pairs) that describe the time series the point belongs to.

{

"metric":"temperature",

"timestamp":1567675709879,

"value":20.5,

"tags":{

"host":"device-1"

}

}

Tags allow you to separate similar data points from different sources or related entities, so you can easily graph them individually or in groups. One common use case for tags consists in annotating data points with the name of the machine that produced it as well as name of the cluster or pool the machine belongs to. This allows you to easily make dashboards that show the state of your service on a per-server basis as well as dashboards that show an aggregated state across logical pools of servers.

mysql.bytes\_received 1287333217 327810227706 schema=foo host=db1  
mysql.bytes\_sent 1287333217 6604859181710 schema=foo host=db1  
mysql.bytes\_received 1287333232 327812421706 schema=foo host=db1  
mysql.bytes\_sent 1287333232 6604901075387 schema=foo host=db1  
mysql.bytes\_received 1287333321 340899533915 schema=foo host=db2  
mysql.bytes\_sent 1287333321 5506469130707 schema=foo host=db2

This example contains 6 data points that belong to 4 different time series. Each different combination of metric and tags makes up a different time series. All of the 4 time series are for one of two metrics mysql.bytes\_received or mysql.bytes\_sent. A data point must have at least one tag and every time series for a metric should have the same number of tags. It is not recommended to have more than 6-7 tags per data point, as the cost associated with storing new data points quickly becomes dominated by the number of tags beyond that point.

With the tags in the example above, it will be easy to create graphs and dashboards that show the network activity of MySQL on a per host and/or per schema basis. New to OpenTSDB 2.0 is the ability to store non-numeric annotations along with data points for tracking meta-data, quality metrics or other types of information.

**3.2 Reading**

Time series data is usually consumed in the format of a line graph, thus OpenTSDB offers a built-in, simple user interface for selecting one or more metrics and tags to generate a graph as an image, alternatively an HTTP API is available to tie OpenTSDB into external systems such as monitoring frameworks, dashboards, statistics packages or automation tools.

**3.3 Data Query**

In the monitoring scenario, we can define a monitoring indicator like this:

Indicator name: sys.cpu.user

Label: host=10.101.168.111, cpu = 0

Indicator value:0.5

The query scenarios supported by OpenTSDB are:

Specify the indicator name and time range, and give one or more label names and label values as conditions to query all data.

Taking the above example as an example, we can query:

sys.cpu.user (host=,cpu=)(1465920000<= timestamp < 1465923600)

Query the user-mode CPU consumption on all CPU cores of all machines between 0:00 a.m. and 1:00 a.m.

sys.cpu.user (host=10.101.168.111,cpu=\*)(1465920000<= timestamp < 1465923600)

Query the user-mode CPU consumption on all CPU cores of a machine between 0:00 a.m. and 1:00 a.m.

sys.cpu.user (host=10.101.168.111,cpu=0)(1465920000<= timestamp < 1465923600)

Query the user-mode CPU consumption on the 0th CPU core of a machine between 0:00 a.m. and 1:00 a.m.

**3.4 Storage Optimization**

Rowkey uses a combination of metricname + timestamp + tags to uniquely determine a metric value.

The strategy adopted by OpenTSDB is to assign a UID to each metric, tag key and tagvalue, and the UID is a fixed length of three bytes.

The length of Rowley is greatly shortened, the benefits are as below:

Save storage space;

Improve query efficiency: reduce key matching search time;

Improve transmission efficiency: not only saves the bandwidth of reading from the file system, but also saves the bandwidth occupied by data return, and improves the speed of data writing and reading;

Alleviate memory pressure: The metric name, tagkey or tag value stored in String can now be replaced with a 3-byte byte array, which greatly saves memory usage.

Other storage optimization strategy:

1. Reduce the number of Key-Value: Combine rows and columns to further reduce storage.
2. Concurrent Write Optimization: Pre-partition buckets to avoid write hotspots.

**4.Other features of OpenTSDB**

Metadata：

OpenTSDB is mainly used to store time series data, and it can easily perform various operations on the data, but it can also tell us what kind of data is stored in it, and provide us with some data context. You can see the official documentation for details.

Tree：

Version 2.0 proposes the concept of tree, which must be used with metadata. It is probably to convert the information in the metadata into a tree structure according to various rules, which is convenient for users to view, similar to the data file directory in the computer. You can see the official documentation for details.

**Reference**

1. The official documentation of Opentsdb:<http://opentsdb.net/docs/build/html/index.html>