Time-series Databases

Minsam Kim, Jiho Park
Objective and Contents

Objective: concise theoretical/practical introduction

1. TS / TSDB
2. Use-cases
3. Characteristic Operations
4. NoSQL TSDB
5. Using OpenTSDB
Time-series Data

What is it?

Definition: a sequence of pairs of time stamp and a value:
{time1, value1}, {time2, value2}, {time3, value3}, ...
Time-series Data

Why use it?
- Dynamic World
- Big Data, KDD, Machine Learning
  - Trends, patterns, correlations, prediction, causal relationships
- Adding another dimension to the information
Time-series Database (TSDB)

Time-series seems good,
But do we really need a special database for it?

More Observations -> More Data
  A lot to insert, search, update, ...

More Analysis Techniques -> More Data
  Past: choose what to sample before collecting data
  Present: over-collect first
  OLAP databases
TSDB Use-cases: Representative Industry - Finance

- Chicago Mercentile Exchange
- 100M live contracts stored
- 14M contracts updated per day
- ~2M messages per second
- Trade volume peaks
TSDB Use-cases: Representative Industry - Sensors / Logistics

- Bottleneck Identification (e.g. Truck Companies)
- Quality Control in Manufacturing (e.g. Inferior Goods)
- Business data providers (e.g. Bloomberg)
- Smartware
TSDB Use-cases: Task - Monitoring

- Airplane Flight Path Scheduling
- All Black-boxes
- System Logs (Data Center)
  - Predictive Maintenance Scheduling
TSDB Use-cases: Task - Testing and Analytics (Pattern Discovery)

- Scientific Research (e.g. CERN)
- Testing Example: Check whether a hedge fund’s trading strategy would have worked well for previous 2 years.
  - What if the real-time data injection is already a tough job for the system?
- Analytics Example: Over-collecting any time-series data may turn into insight.
  - Unsupervised Learning (Clustering) of smartwear accelerometer time-series data
  - User-based customization with user’s history data
TSDB Characteristic Operations

- Insertion: usually the bottleneck, must be very fast
- Retrieval: must be as fast
- Update: happens rarely (cannot change the past)
- Join multiple time series data on time:
  - select NYMEX/Gold_Price * NYMEX/Gold_Volume
- Filter by time:
  - select onpeak(cellphone_usage)
- Interpolate and Regularize
- Compress Time-series Data
Time-series Data in Relational Database

- Naive Design w/ Star Schema

- 1 row = 1 data point
- Row insertion rate?

<table>
<thead>
<tr>
<th>Time</th>
<th>Time series ID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:51:00</td>
<td>101</td>
<td>0.01</td>
</tr>
<tr>
<td>15:51:03</td>
<td>102</td>
<td>1.16</td>
</tr>
<tr>
<td>15:52:07</td>
<td>101</td>
<td>0.04</td>
</tr>
<tr>
<td>15:52:11</td>
<td>101</td>
<td>0.08</td>
</tr>
<tr>
<td>15:53:17</td>
<td>103</td>
<td>4.18</td>
</tr>
</tbody>
</table>
NoSQL Database with Wide Tables

- Store multiple values per row
- # of columns are unbounded (100 - 1000 samples per row)
- Reduction of retrieval overhead

![Diagram of NoSQL Database with Wide Tables](image)
NoSQL Database with Hybrid Design

- Compress some data into one data structure (blob = binary large object)
- Makes retrieval even faster
- Compress if there is no new data or changes
NoSQL Database with Hybrid Design (cont.)
NoSQL Database with Direct Blob Insertion (cont.)
NoSQL Database with Hybrid Design (cont.)

- Since data is inserted in the uncompressed format, the arrival of each data point requires a row update operation to insert the value into the database.
- Direct blob insertion can increase insertion rate to 1,000-fold.
Section Summary

- RDBMS may not be suitable for TS Data (why not blob in RDBMS?)
- Various NoSQL methods (wide table, hybrid design) to handle billions of TS data points
- How to do blob compression is the key for significantly increasing insertion rate (about 1000 data points in one blob)
OpenTSDB

- Time Series Daemon (TSD)
- Multiple TSDs at the same time
- Return data by PNG or Json format
- Default time window is one-hour
- Or use Grafana (open source data visualization tool)
OpenTSDB (cont.)
OpenTSDB (cont.)

- Bulk loading of historic data (blob loader)
- Blob loader separated from TSD
- accelerates data ingestion by short-circuiting the normal path
- Use it only for testing & backfill because data may not show in DB immediately
Accessing OpenTSDB

1. **Selection** - The time series that you want are selected from others by giving the metric name and some number of tag/value pairs.
2. **Grouping** - The selected data can be grouped together.
3. **Down-sampling** - It is common for the time series data retrieved by a query to have been sampled at a much higher rate than is desired for display.
4. **Aggregation** - functions like average, sum, minimum of particular time windows
5. **Interpolation** - The time scale of the final results regularized at the end by interpolating as desired to particular standard intervals.

These queries can be done through either by REST API or by using packages in various languages like R, Go, or Ruby
Accessing OpenTSDB data with SQL-on-Hadoop

- If you need to analyze large volumes of time series data beyond what works with the REST interface, you can access directly via the HBase API.
- The wide table and blob formats in order to get high performance can make it more difficult to access this data using SQL-based tools.
- **SQL-on-Hadoop** is a class of analytical application tools that combine established SQL-style querying with newer Hadoop data framework elements.
- **Apache Spark SQL** is recommended to query data for Open TSDB data, because it is directly accessible from a full programming language Scala, which is suitable for manipulating TS Data.
Conclusion

- What is Time Series Data and their characteristics
- What are the real-life use cases
- How to handle them in an actual database
- What are the popular existing solutions