Time-series based solution using InfluxDB

Julian Ganz
Department of Computer Science
Furtwangen University
Furtwangen, Germany
Julian.Ganz@hs-furtwangen.de

Matthias Beyer
Department of Computer Science
Furtwangen University
Furtwangen, Germany
Matthias.Beyer@hs-furtwangen.de

Christian Plotzky
Department of Computer Science
Furtwangen University
Furtwangen, Germany
Christian.Plotzky@hs-furtwangen.de

***Abstract*—In this paper, we will describe the implementation of two queries using a time series database. Time series are data structures optimized for storing records along with a time stamp, as well as allowing fast access to records in a given time range. We will discuss a time series based data model and our implementation of the two queries, to show the advantages in both storage size and access latency. We show that one of the queries can be optimized using an aggregation.**

***Index Terms*—databases, time series**

I. Introduction

We were given the task to implement two semi-complex queries with timestamp based data. It was suggested to use the time series capable database IBM® Informix®. However, implementing a solution with IBM® Informix® proved to be unnecessarily complicated. We took a different approach and switched to a newer time series based database called InfluxDB. After a brief introduction to time series, we will propose a time series based data model. Furthermore, we will provide an implementation of the aforementioned queries.

II. Time series

Collecting data and events like server metrics, application monitoring, network load, sensor data, trades, etc. over time is done with time series. Time series can occur in regular and irregular form. Regular time series data is used for data which is accumulated in fixed intervals. Irregular time series are data points of events triggered by users or external services [1, p. 2 f.].

Time series data is thus sequential data with a timestamp attached, indicating when the value was recorded [2, p. 25].

Traditional relational database systems do not scale well with large amounts of time series data, especially if, for example, one billion data points per day are stored, aggregated and queried. This is why there are so called time series databases, that are optimized to work with this form of data [2].

A time series database, which is inherently also a NoSQL database,

> can (i) store a record that consists of a timestamp, value, and optional tags, (ii) store multiple records grouped together, (iii) can query for records and (iv) can contain time ranges in a query.
> [3, p. 25]

Also, a time series database can store multiple values per timestamp.

III. Data model

For both queries, we require fast access to the word counts for a specific word in a specific time range. Hence, we define a "words" table mapping each word to a time series containing the word counts for each document at some point in time. Naturally, the words are chosen as primary keys, implicitly making them unique among the table. Since we are only interested in ranges of dates, one day is sufficient as the temporal resolution for the time series.

Compared to both the timestamp and the count, storing a document URL requires a relatively large amount of data. On top of that, URLs are expected to be redundant, since a document will typically contain many words and thus its URL will end up in multiple time series. Nonetheless, in order to keep the approach as simple as possible, we chose to not outsource the URLs but store them alongside the time series data.

IV. Importing data

Importing the time series data from the input schema displayed in listing 1 into InfluxDB involved a short script (listing 2) to prepare the data accordingly to the InfluxDB import format.

Source Code 1: Input Data

```
word|date|url|count|
...
```

Source Code 2: Converter Script

```
require 'csv'
CSV.foreach(ARGV[0], col_sep: '|')
do |row|
  w = "word=\"#{row[0]}\""
  d = Date.parse(row[1]).strftime("%s")
  u = "url=\"#{row[2]}\""
  v = "value=#{row[3]}"
  puts "ms,#{w} #{u},#{v} #{d}"
end
```

V. Queries

The first of the two queries to implement returns the overall occurrences of a specific, given word for each day in a time

range. This data can be easily retrieved from the data model described in section III.

Since the time series implementation is presumably optimized for this kind of query [1], the query is naturally fast.

Source Code 3: Query 1

```
SELECT SUM(value) AS count
  FROM wcuret.ms
 WHERE word='google'
   AND time > 1409702400000000000
   AND time < 1440028800000000000
```

The second query returns the top k URLs for a given time range where a specific word occurs. This can be implemented using the "TOP()" aggregate function in InfluxDB, which returns the top N values of a $field$ where the $field$ must be of type "int64" or "float64".

Source Code 4: Query 2

```
SELECT url,word,TOP(value, 10)
  FROM wcuret.ms
 WHERE time > 1409702400000000000
   AND time < 1440028800000000000
```

Similar as in the first query, the relevant time series and the specified range of data points can be queried in a short amount of time. As it does not introduce large overhead, we can also return the word for each data point in this query.

Note that the returned list of rows is not sorted. If a sorted list is required, application code must sort the result of the query, as InfluxDB is not capable of performing this.

VI. Aggregation

In query 1, an aggregation could be made for the word counts for each word per day. In this case, one is neither interested in the URLs queried that day nor in a single word count.

Hence, an aggregation in advance would speed up the query using an additional table holding the pre-aggregated data. That table would be nearly identical in structure to the table described in section III, although lacking the "url" field. The data for this table would be obtained by summing up all word counts for a specific word and day.

Performing query 1 is then a simple matter of selecting the specified range in the time series corresponding to the specified word. Using a table of this schema less data needs to be read due to the fact that, for a specific day, at most one entry exists and the "documents" field would not be present.

As a result, query 1 would be expected to complete in a shorter amount of time. However, the aggregation means additional effort beforehand. Also, in a scenario where both queries are performed, the use of an additional table could hurt performance due to reduced effectiveness of the buffer pool or cache misses. Either way, ad-hoc measurements show that query 1 can be executed on a small machine (4 virtual cores, 4 GiB RAM) in almost no time (0.012 sec, where the starting of the commandline application for querying the database takes approximately 0.010 sec).

[1] e.g. by ensuring the data points are stored in sequence on disk

VII. Conclusion

Our implementation of the two queries makes use of the time series concept and presents an example of how to work with time series.

We have explained why the concept of time series theoretically has two advantages: Firstly the given data consumes less disk space and secondly it allows for faster querying of time series data than relational or non-time-series databases. For further practical analysis of how effective InfluxDB does this, one would have to run similar queries against relational or other non-time-series databases and compare the results.

Time series are not a new scientific achievement, but rather a concept which has been used for decades in statistics and other realms [4, S. 1]. Time series problems require time series database capabilities, which InfluxDB does provide.

Acknowledgment

We want to thank Prof. Dr. L. Piepmeyer for his patience and help during our evaluation of IBM Informix.

References

[1] P. Dix, "Why time-series matters for metrics, real-time and sensor data," jun 2016. [Online]. Available: https://www.influxdata.com/wp-content/uploads/2016/05/Time-Series-Tech-Paper-6-6.pdf

[2] T. Dunning and E. Friedman, "Time series databases," 2015.

[3] A. Bader, "Comparison of time series databases," Diplomarbeit, University of Stuttgart, Universitätsstraße 38, D–70569 Stuttgart, jan 2016.

[4] R. S. Tsay, "Time series and forecasting: Brief history and future research," *Journal of the American Statistical Association*, vol. 95, no. 450, pp. 638–643, 2000. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/01621459.2000.10474241