UNIVERSITÀ DEGLI STUDI DI UDINE

Dipartimento di Scienze Matematiche, Informatiche e Fisiche

Corso di Laurea Triennale in Tecnologie Web e Multimediali

Tesi di Laurea

COMPRESSION ALGORITHMS FOR TIME SERIES DATA

Relatore:
Prof. NICOLA VITACOLONNA

Laureando:
UMBERTO D'OVIDIO

ANNO ACCADEMICO 2018-2019

Indice

Chapter 1

Introduction to Time Series Data

1.1 What is time series data?

Time series data can be thought of as a series of data points, measuring a certain variable over time, stored in time order. Time Series datasets most commonly share three characteristics [1] :

- Data points are almost always stored as a new record
- Data typically arrive in time order
- Time is a primary axis

For example, let us consider the S&P 500, a stock market index based on the market capitalization of the largest 500 companies in the USA. We can collect valuable information during a given time interval. For example, we could be interested in the high, low and closing price on a daily basis. The dataset is illustrated in figure 1.1.

The date column is the time index, while the other columns represent measurements of some specific metrics. This is a typical example of time series data, although not every dataset containing date/time information is necessarily a time series data set.

The rule of thumb is that with time series data sets changes over time are tracked by creating new records, rather than updating existing ones. If you have a record with a timestamp, and whenever something changes in the measurement you create a new record instead of updating the previous one, then you have a time series dataset.

Date	Open	High	Low	Close	Adj Close	Volume
2018-12-31	2498.939941	2509.239990	2482.820068	2506.850098	2506.850098	3442870000
2019-01-02	2476.959961	2519.489990	2467.469971	2510.030029	2510.030029	3733160000
2019-01-03	2491.919922	2493.139893	2443.959961	2447.889893	2447.889893	3822860000
2019-01-04	2474.330078	2538.070068	2474.330078	2531.939941	2531.939941	4213410000
2019-01-07	2535.610107	2566.159912	2524.560059	2549.689941	2549.689941	4104710000
2019-01-08	2568.110107	2579.820068	2547.560059	2574.409912	2574.409912	4083030000
2019-01-09	2580.000000	2595.320068	2568.889893	2584.959961	2584.959961	4052480000
2019-01-10	2573.510010	2597.820068	2562.020020	2596.639893	2596.639893	3704500000
2019-01-11	2588.110107	2596.270020	2577.399902	2596.260010	2596.260010	3434490000

Figure 1.1: Time series values for the S&P 500 aggregated on a daily basis

1.2 What is the importance of time series data?

Time series data is becoming increasingly important in our world, as one can infer from the growing popularity of time series databases in the period 2017-2019 (figure 1.2).

Many different businesses rely on the collection and the analysis of this type of data. DevOps monitoring, scientific measurements, financial markets are just a few business sectors heavily using time series data. In most of the applications relying on time series data, it is vital to collect values as often as possible, to create faster and more accurate monitoring systems, better models and simulations, more accurate forecasting predictions. As an example, consider a system that monitors the overall CPU usage of a distributed system. Whenever the CPU usage exceeds a certain threshold, a remediation action is triggered (for example spawning an additional node to reduce the load on the system). It is apparent that the usefulness of the system depends on how often the CPU usage values are collected (the more frequently values are collected, the faster the detection of some anomaly and the faster the remediation action can take place).

The need for finer temporal resolutions requires automatically storing larger quantities of data, up to the point where traditional relational database systems are not able to handle the scale. This is because relational database systems store data as a collection of fixed-size pages on disk. Data is usually indexed by data structures that minimize disk access, most commonly B-trees. When data sets become large to the point it is not possible to store indexes in memory anymore, inserting or updating a row might require swapping in-

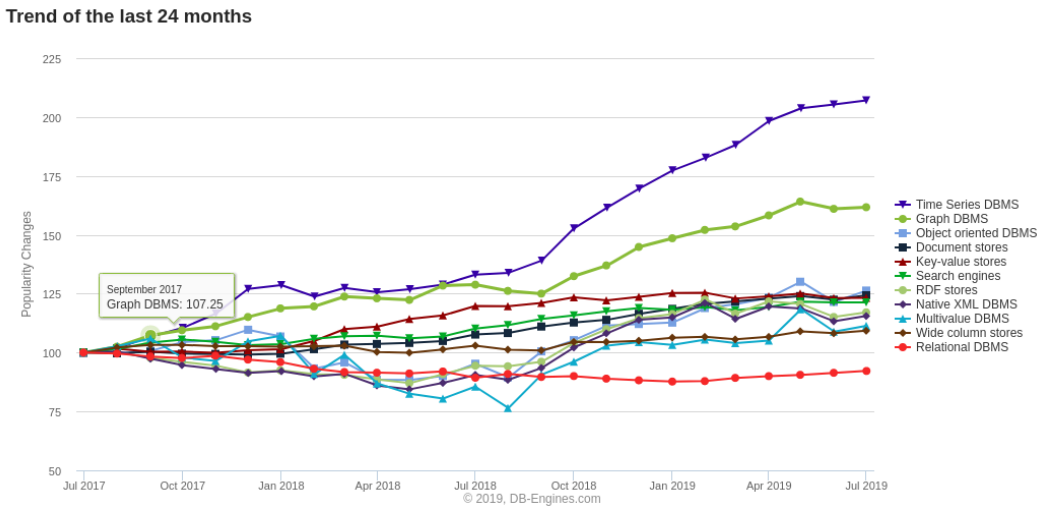

Figure 1.2: Database popularity in the period 2017-2019 according to `https://db-engines.com`, July 2019

dex pages from memory to disk, which drastically decreases performance [2]. For this reason, new databases that specifically target time series data have been created. These databases have higher ingestion rates, faster queries at scale and better data compression as compared to traditional RDBMS. In this dissertation, I will analyze the problem of data compression in time series data.

1.3 Why compressing time series?

Finding a good compression algorithm to store time series can have multiple benefits depending on the adopted technique and the use case. Intuitively, reducing the storage needs of an application allows the system to increase the temporal resolution at which the data is recorded. The storage requirements may be reduced up to the point the whole data set can fit in memory, reducing writing and reading latency by an order of magnitude or more. If the application is network bound, compressing data means reducing the amount of data to be transferred from server to client or the other way around. In some IoT applications, the compression can be done at the device level, so that less data need to be transferred over the network. This is usually more power efficient, as the energy required to compress the data is much less compared to the energy required to transmit it.

1.4 Four approaches for time series compression

We can identify four different approaches for compressing time series data.

1. **Storing aggregates**. In most applications, we are mostly interested in recent data as compared to older data. Thus, older data can be "aged out" , that is, stored in aggregates over a longer temporal interval. Consider the previous example of the S&P Index stock price. Stock prices of the current year can be aggregated over a one-hour interval, while older data can be aggregated over days, weeks, and even months. One problem with this approach is that aggregates remove fluctuations and outliers that are sometimes important when analyzing historical data.

2. **Lossless compression**. Time series data is characterized by some properties that can be exploited to obtain good compression ratios. For example, most of the time data comes at regular intervals, so it does not make sense to store the whole timestamp value for each record, but rather go with a delta encoding approach, that is storing only the difference between sequential timestamps.

3. **Lossy compression.** Many applications are concerned with the trends that can be extracted from time series data, and do not require a high level of accuracy. For this type of applications, lossy compression can be considered, which usually results in much better compression rates as compared to lossless algorithms.

4. **General-purpose algorithms**. Some time series databases do not offer compression out of the box. In these cases, one can consider running the databases on ZFS or other file systems that offer data compression [3]. Other benchmarks [4] have indicated that general-purpose algorithms such as LZ4, achieve good compression ratios and may be more desirable to custom solutions given their highly efficient implementations.

Chapter 2

Compression techniques for time series data

Bibliography

[1] A. Kulkarni, *What the heck is time-series data (and why do I need a time-series database)?*, 2018. [Online]. Available: https://blog.timescale.com/blog/what-the-heck-is-time-series-data-and-why-do-i-need-a-time-series-database-dcf3b1b18563/ (visited on 09/21/2019).

[2] M. Freedman, *Time-series data: Why (and how) to use a relational database instead of NoSQL*, 2017. [Online]. Available: http://www.surfline.com/surf-news/maldives-surf-access-controversy-update_75296/.

[3] *TimescaleDB Developer Docs*, 2019. [Online]. Available: http://www.surfline.com/surf-news/maldives-surf-access-controversy-update_75296/.

[4] J. Danjou, *Timeseries storage and data compression*, 2016. [Online]. Available: http://www.surfline.com/surf-news/maldives-surf-access-controversy-update_75296/.