Relational or Non-Relational? A Comparative Evaluation of Database Solutions for Energy Consumption Data

Poster Abstract

Ambreen Zaina
Georg-August-Universität Göttingen
Göttingen, Germany
ambreen.zaina@cs.uni-goettingen.de

Andreas Reinhardt
Technische Universität Clausthal
Clausthal-Zellerfeld, Germany
reinhardt@ieee.org

Jana Huchtkoetter
Technische Universität Clausthal
Clausthal-Zellerfeld, Germany
jana.huchtkoetter@tu-clausthal.de

ABSTRACT

The significance of energy consumption analysis has been growing almost exponentially over the course of the last decade. In order to get analysis results of high accuracy, many algorithms require large volumes of available input data. Different kinds of databases are available today and offer a common interface to store and retrieve such time series data. *Relational* databases, such as MySQL, are well-established solutions for the storage of key-value pairs (e.g., timestamp of collection and sampled value). However, the constant growth of consumption data poses a serious challenge for the scalability of relational databases, and has led to the emergence of non-relational (*NoSQL*) database solutions. In this work, we compare a relational and a non-relational database with regard to their performance when used to store energy consumption time series. The insights gained shall serve as a recommendation for the informed database choice for energy data.

CCS CONCEPTS

• **Information systems → Data management systems**;

1 INTRODUCTION

Due to the strong rise in the number of smart meters deployed worldwide, the storage of energy consumption data is becoming a research question of increasing importance. Even when only one sample is being collected every second, a total of 86,400 samples result per meter per day, i.e., 31.5 million readings per year. When comparing existing collections of energy data, different ways can be found how their data are being stored. Some data sets, such as BLOND [6], rely on the use of the HDF5 format [5]. Others simply use comma-separated value (CSV) files, such as tracebase [7] and ECO [2]. Full database access is available for data collected as part of the Pecan Street Dataport project[1], yet little detail is published about the underlying database technology.

Databases are widely used in other data warehousing systems. It thus surprises that their usage is not very widespread in energy data analytics research, given the benefits they provide. The retrieval of data over a unified interface, guaranteed data consistency, less redundancies, and the possibility to run database services on computer clusters are only a few of their advantages. Particularly the recent emergence of non-relational databases has sparked a lot of interest in the research community. Many large corporations have meanwhile moved to such solutions (e.g., Google's BigTable [3]), mainly for performance and scalability reasons. We thus conduct a comparative assessment of the database performance for energy data storage in this work, in order to provide a starting point for the wider use of such databases in practice.

[1]Available at https://www.dataport.cloud

2 CHOICE OF DATABASES

We have considered the following database solutions for our analysis, based on their ranking amongst the most popular relational and non-relational databases [1].

PostgreSQL[2] is an open-source object-relational database management system, fully compliant to the ACID principles and the structured query language (SQL). We have used PostgreSQL v9.6.1.

InfluxDB[3] is an open-source time series database, written in Go and featuring a query language similar to SQL. InfluxDB is a schemaless database allowing users to add new data (measurements, tags, or fields) at any time. We have tested on InfluxDB v1.3.5.

Both PostgreSQL and InfluxDB support all elementary data types required for power consumption data: Date fields to indicate the date and time of collection as well as integer or floating-point values to indicate the consumption at that time.

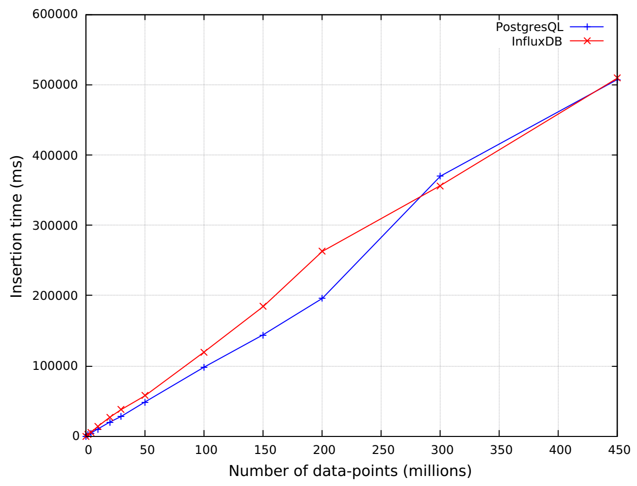
3 EVALUATION

The objective of the conducted evaluation was to determine insertion times and memory demand for each of the chosen database solutions, given that these are important indicators for the scalability of energy consumption databases.

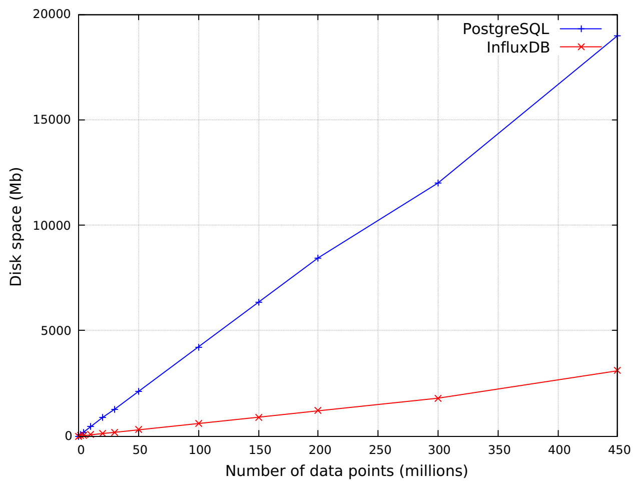
3.1 Evaluation setup

Experiments were performed on a 64-bit Ubuntu 16.04 LTS host with Intel Core i5-6600 processor (4x3.3 GHz) and 16 GB of RAM. The aggregate electrical power consumption of an individual household, sourced from [4], was used in all conducted tests. Measurements were collected at a frequency of one reading per minute between December 2006 and November 2010; the data set thus contains slightly more than 2 million samples. The data are represented as an univariate series with timestamp and corresponding power

[2]Available at https://www.postgresql.org
[3]Available at https://www.influxdata.com/time-series-platform/influxdb/



(a) Insertion performance of PostgreSQL and InfluxDB.



(b) On-disk storage of PostgreSQL and InfluxDB.

consumption. We have intentionally augmented the data set (by repeating its values periodically) beyond a size that still fits into the computer's RAM to see whether a performance bottleneck exists at the point where changes need to be written to the disk.

3.2 Data insertion

The test setup consisted of the manual creation of databases in PostgreSQL and InfluxDB. In PostgreSQL, a table has been created for the time series consumption data, with one column corresponding to the timestamp of data collection, and a second column containing the consumption measurement. We have used PostgreSQL's batch import function to subsequently insert the testing data into the database. Through the `timing` command of the PostgreSQL server, the total time taken for the data insertion was measured.

Likewise, a schema-less table has been created in InfluxDB to accommodate the testing data. In order to import the data into InfluxDB, they were converted to its *line protocol* input format, where each line in a file represents a single data point. In InfluxDB, the data set was chunked into batches of 5,000 data points each, which were iteratively inserted by means of its batch import feature, and clocked by summing up the execution times of all insertions. The necessity to split the data set into chunks became apparent during our tests, where data files containing more than 5,000 points were repeatedly rejected by InfluxDB.

Figure 1a shows the insertion performance of PostgreSQL and InfluxDB, measured as the time required to add a given number of elements into the database. Up to 450 million points of input data were used to determine the insertion time; experiments were run ten times each, with average values plotted in the diagram (in milliseconds). At the maximum input size of 450 million data points, the insertion took approximately 8 minutes in real-time for both databases. Only minor performance differences were observed when fewer input data points were being used. At 200 million data points a significant difference (about 20%) is visible, although this difference is already equalized from 300 million data points onwards. To sum up, both of the databases have performed quite closely when performing data insertion.

3.3 On-disk storage

PostgreSQL offers the possibility to determine the disk usage of databases, tables, and indexes. InfluxDB does not provide such commands by design, yet allows for the monitoring of its database directory on the disk. Thus, both solutions offer a way to determine their on-disk storage demand. Figure 1b visualizes this value for PostgreSQL and InfluxDB, depending on the number of inserted data points. Both databases were populated with up to 450 million entries, summing up to a total of 12 GB input file size when represented in CSV format. Note that this does not necessarily reflect the on-disk storage demand of the database.

While the observed insertion performances were largely similar, a huge performance difference can be noted in terms of storage demand. While PostgreSQL takes 19 GB of disk space (i.e., it requires an overhead in excess of 58%) to store its data points, InfluxDB requires only 3 GB of disk space. A possible reason for this insight might be the application of data encoding or compression, yet no statement regarding the used compression algorithm (if any) could be found in the documentation. We were equally unable to tell whether any advantages result from PostgreSQL's overhead during the testing we have conducted; an in-depth investigation of this research question is considered as future work.

4 CONCLUSIONS

Scalable, reliable, and efficient energy data storage is expected to become increasingly important for future applications that rely on energy consumption data. By evaluating the performance of PostgreSQL and InfluxDB for their suitability to store energy consumption data, we have found that both approaches work and show similar performance when inserting data. However, the on-disk storage strongly deviates between the two approaches. PostgreSQL incurs a significant overhead (58%) when writing data to disk, while InfluxDB actually reduces their size by 75% when writing them to its on-disk storage. Its specific design for time-series data makes it an interesting candidate to consider for future research.

ACKNOWLEDGMENTS

This research was supported by Deutsche Forschungsgemeinschaft (DFG) grant no. RE 3857/2-1. We thank the Simulationswissenschaftliches Zentrum Clausthal-Göttingen (SWZ) for financial support.

REFERENCES

[1] Andreas Bader, Oliver Kopp, and Michael Falkenthal. 2017. Survey and Comparison of Open Source Time Series Databases. In *Proceedings of the 17th Conference on Database Systems for Business, Technology, and Web (BTW)*.
[2] Christian Beckel, Wilhelm Kleiminger, Romano Cicchetti, Thorsten Staake, and Silvia Santini. 2014. The ECO Data Set and the Performance of Non-intrusive Load Monitoring Algorithms. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings (BuildSys)*.
[3] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. 2006. Bigtable: A Distributed Storage System for Structured Data. In *Proceedings of the 7th Symposium on Operating System Design and Implementation (OSDI)*.
[4] Dua Dheeru and Efi Karra Taniskidou. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml
[5] The HDF Group. 2012. Hierarchical Data Format (HDF) Version 5. Online: https://www.hdfgroup.org/HDF5/.
[6] Thomas Kriechbaumer and Hans-Arno Jacobsen. 2018. BLOND, a Building-level Office Environment Dataset of Typical Electrical Appliances. *Scientific Data* 180048 (2018).
[7] Andreas Reinhardt, Paul Baumann, Daniel Burgstahler, Matthias Hollick, Hristo Chonov, Marc Werner, and Ralf Steinmetz. 2012. On the Accuracy of Appliance Identification Based on Distributed Load Metering Data. In *Proceedings of the 2nd IFIP Conference on Sustainable Internet and ICT for Sustainability (SustainIT)*.