TsQuality: Measuring Time Series Data Quality in Apache IoTDB

Yuanhui Qiu
Tsinghua University
qiuyh21@mails.tsinghua.edu.cn

Chenguang Fang
Tsinghua University
fcg19@mails.tsinghua.edu.cn

Shaoxu Song
BNRist, Tsinghua University
sxsong@tsinghua.edu.cn

Xiangdong Huang
Timecho Ltd
hxd@timecho.com

Chen Wang
Timecho Ltd
wangchen@timecho.com

Jianmin Wang
Tsinghua University
jimwang@tsinghua.edu.cn

ABSTRACT

Time series has been found with various data quality issues, e.g., owing to sensor failure or network transmission errors in the Internet of Things (IoT). It is highly demanded to have an overview of the data quality issues on the millions of time series stored in a database. In this demo, we design and implement TsQuality, a system for measuring the data quality in Apache IoTDB. Four time series data quality measures, completeness, consistency, timeliness, and validity, are implemented as functions in Apache IoTDB or operators in Apache Spark. These data quality measures are also interpreted by navigating dirty points in different granularity. It is also well-integrated with the big data eco-system, connecting to Apache Zeppelin for SQL query, and Apache Superset for an overview of data quality.

PVLDB Reference Format:
Yuanhui Qiu, Chenguang Fang, Shaoxu Song, Xiangdong Huang, Chen Wang, and Jianmin Wang. TsQuality: Measuring Time Series Data Quality in Apache IoTDB. PVLDB, 16(12): XXX-XXX, 2023.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://thssdb.github.io/TsQuality/.

1 INTRODUCTION

Time series data are often found with various data quality issues, such as completeness, consistency, and validity, especially in the scenarios of IoT [5]. In the process of time series data management, from being collected to being stored in time series databases, any issue like sensor failure or network transmission errors, may lead to data quality problems. Analysis upon dirty data without prior assessment of data quality may yield misleading results.

Existing systems, such as Cleanits [1] and cleanTS [3], propose to clean the dirty data in individual time series. However, commodity databases often store millions of time series, for thousands of devices [7]. There is still a lack of overall assessment for all the data stored in the database. Moreover, such works are not well integrated with the eco-system, from storage to analysis.

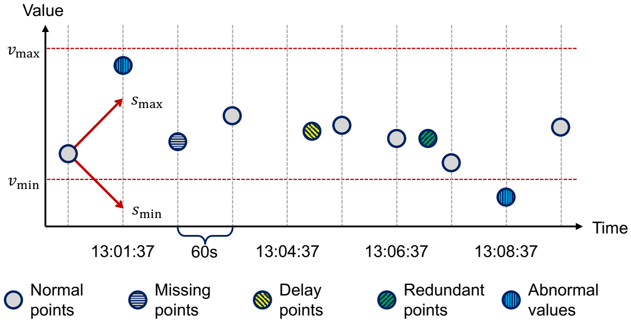


Figure 1: Time series data quality example

In this demo, we present TsQuality to measure data quality in Apache IoTDB[1], an open-source time series database developed based on our preliminary study [7]. Each time series consists of a time column and a value column in the database. For the timestamp issues, we consider missing, redundant and delay points [2]. For the value column, we investigate abnormal values w.r.t. range, variation, speed [6], and acceleration [4].

Example 1.1. Figure 1 presents a segment of time series with four types of data quality issues. As shown, the data points are usually collected every minute, a preset frequency of sensors. A point, however, is missing at time 13:02:37, and leads to *completeness* issue. In contrast, the point at 13:06:37 is re-transmitted, resulting in a redundant one, known as *consistency* issue. Moreover, a point could also be delayed, e.g., the one that should appear at time 13:04:37 but not until 30 seconds later. Such an issue is measured by *timeliness*.

The *validity* measure is evaluated w.r.t. a set of constraints on both time and value. For instance, the two horizontal red lines, $v_{\min}$ and $v_{\max}$, denote the valid range of values. The point at time 13:08:37 has an abnormal value smaller than the minimum. In addition, the two red arrows, $s_{\min}$ and $s_{\max}$, specify the speed of maximum and minimum value fluctuation over time. The data point at time 13:01:37 has a speed of $\frac{250-115}{60} = 2.25 > 2 = s_{\max}$, and thus has abnormal value as well.

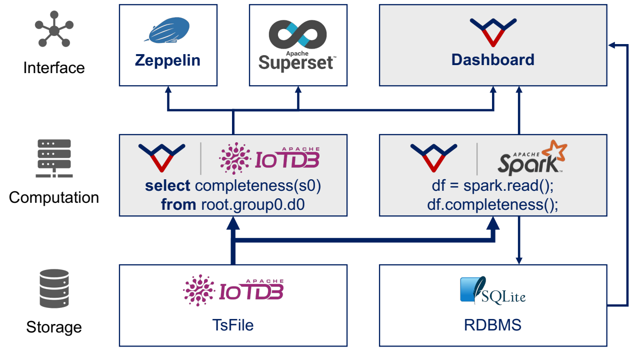
The major features of TsQuality are as follows.

(1) **Well-integrated eco-system.** The data quality measures have been implemented as database-native functions in Apache

Proceedings of the VLDB Endowment, Vol. 16, No. 12 ISSN 2150-8097.
doi:XX.XX/XXX.XX

Shaoxu Song (https://sxsong.github.io/) is the corresponding author.

[1] https://iotdb.apache.org/



```
select completeness(s0)
from root.group0.d0
```

```
df = spark.read();
df.completeness();
```

Figure 2: System architecture

IoTDB as well as data-intensive operators in Apache Spark. Moreover, it may also connect to Apache Zeppelin for SQL query and Apache Superset for an overview of data quality.

(2) **Interpreted results.** In addition to the overview of all data quality measures in the entire database, TsQuality Dashboard provides an interpretation of each dirty point. Users can thus navigate individual time series for data quality issues in various granularity.

(3) **Customized measures**: TsQuality is able to accommodate different data quality definitions for various scenarios by extending the ER diagram and writing IoTDB UDFs.

The document of four data quality functions, completeness, consistency, timeliness, and validity, is also available on the product website of Apache IoTDB.[2] The corresponding code is included in the GitHub repository of the system.[3]

2 SYSTEM DESCRIPTION

In this section, we first overview TsQuality and introduce the system architecture. Then we describe the storage design for storing IoTDB time-series-related statistics in SQLite. Finally, we close this section by introducing data quality evaluation in TsQuality and its ability to adapt to different definitions of data quality.

2.1 System Architecture

Figure 2 illustrates the system overview and the data flow between different components. The whole system follows a three-tier architecture model: *Storage, Computation, and Interface*. The interface layer is responsible for direct interaction with the user, it receives SQL queries regarding data quality and visualizes the results in the form of graphs. Specifically, users can choose TsQuality Dashboard, the native visualization tool of IoTDB, which gives an overview of data quality at the time series level, as well as a detailed analysis of abnormal values to locate and fix data quality problems in the data. TsQuality is also tightly integrated with the open-source ecosystem by interfacing with Apache Zeppelin for custom SQL queries, and Superset for the overall overview of data quality.

We store the time series statistics in a relational database, in order to read and visualize them in TsQuality Dashboard. For this purpose, we devise and implement the computation layer, which reads the original time series data from IoTDB, calculates the number of all kinds of data quality issues in each time series, and stores the statistics in SQLite. In particular, we have developed two computing methods to suit the needs of different scenarios. We first implement a series of functions in IoTDB to perform the computation, as shown in the left part of the computation layer in Figure 2. This approach allows users to monitor data changes in real-time through IoTDB's triggers and take different measures according to actual demands. Meanwhile, considering the possible performance bottleneck of this approach when facing large amounts of data, we leverage Apache Spark to cope with large data volume, as shown in the right part of the computation layer in Figure 2. While losing the ability to monitor data changes, this method calculates statistics much faster than the previous one.

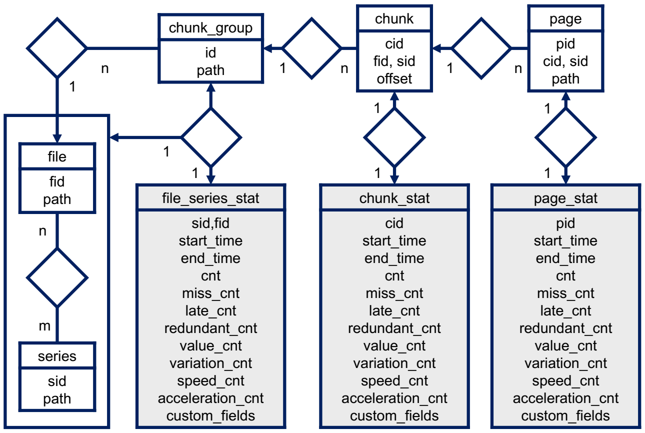


Figure 3: ER diagram of time series statistics

The bottom layer in Figure 2 is the storage layer. IoTDB stores the time series data in the form of TsFile and provides data upwards for the computation layer as well as the three tools in the interface layer. Considering the relatively simple data storage schema (number of various abnormal values), we use SQLite, a lightweight relational database, to store the results of the computation layer and provide information for the interface layer.

2.2 Storage Design

Figure 3 presents the entity-relationship model of the data in SQLite, where the white entities are existing concepts in IoTDB and the gray ones are new information in TsQuality. The entities *page, chunk, chunk_group, file* correspond to the hierarchical storage structure of IoTDB, sorted in order of storage granularity from fine to coarse. Defined according to the tree data model of IoTDB, entity *series* records the logical path of a time series. In IoTDB, all time series will be sliced and stored into multiple data files partitioned by time. The composite entity *file-series* thus corresponds to one partition of a time series.

The three entities *page_stat, chunk_stat, file_series_stat* store the statistical information of the corresponding storage level, including the start and end timestamps, the total data count and the number of all kinds of abnormal values introduced in Section 1. In addition,

[2] https://iotdb.apache.org/UserGuide/Master/Operators-Functions/Data-Quality.html

[3] https://github.com/apache/iotdb/tree/master/library-udf/src/main/java/org/apache/iotdb/library/dquality

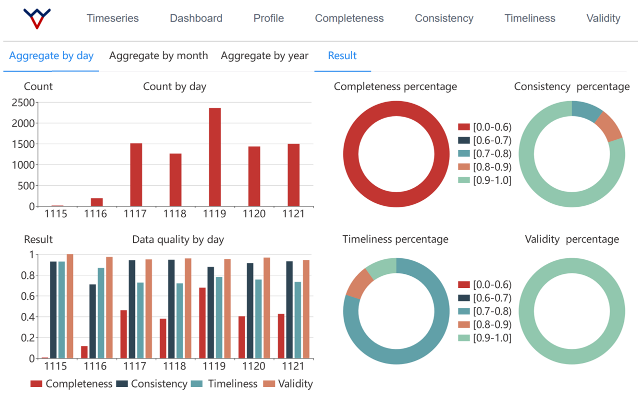


Figure 4: Data quality overview of time series in TsQuality

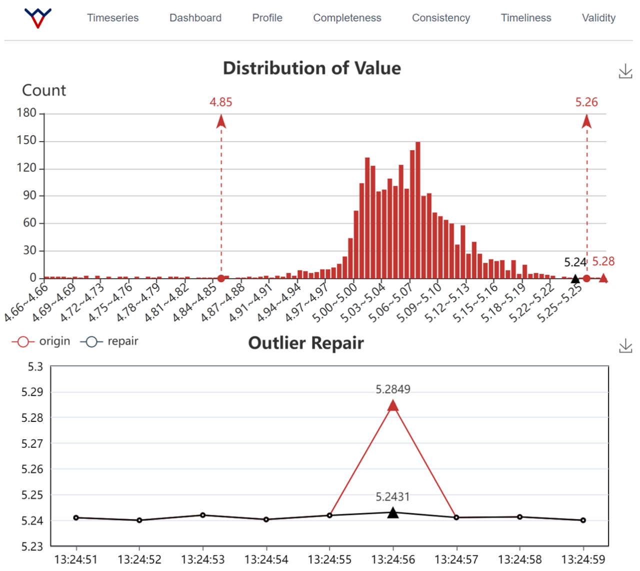


Figure 5: Data quality explanation of time series in TsQuality

the ER diagram can be extended with custom fields to accommodate different data quality metric definitions in real-world scenarios.

2.3 Data Quality Evaluation

In this section, we introduce how to measure data quality in IoTDB with TsQuality. TsQuality supports two forms of data quality evaluation. In addition to calculating data quality metrics based on the statistic information stored in SQLite, we also implement a series of functions in IoTDB to perform data quality-related queries. The data quality functions currently supported by IoTDB are listed below:

(1) *Completeness* measures the ratio of data that is not missing.
(2) *Consistency* measures the ratio of data that is not redundant.
(3) *Timeliness* measures the ratio of data that is not delayed.
(4) *Validity* measures the ratio of data that meets constraints.

For example, the following SQL statement is used to partition the data into windows of 15 data points and query the validity of the time series *root.test.d1.s1* in IoTDB before January 1, 2023.

```
SELECT consistency(s1,"window"="15")
FROM root.test.d1 WHERE time <= 2023-01-01
```

Considering that the definition of data quality may vary in practical scenarios, TsQuality has two extension mechanisms to accommodate different demands which correspond to the two data quality evaluation forms respectively. First, based on the statistical information in SQLite, users can extend the ER diagram by adding custom fields as introduced in the previous section. In addition, users are also able to handle complex data quality analysis with TsQuality by writing IoTDB UDFs.

3 DEMONSTRATION PLAN

In this section, we demonstrate TsQuality using the three tools in Figure 2 to measure the data quality in IoTDB as follows.

3.1 TsQuality Dashboard

As the native visualization tool for IoTDB, TsQuality Dashboard[4] not only provides an overview of data quality at the series level but also marks the abnormal values and gives the possible repair in the original data distribution. Figure 4 gives an overview of data quality for a single time series from November 15th to November 21st. The two bar charts on the left give the total amount of data and the data quality aggregated by day, respectively. In addition to aggregation by day, users can also select the corresponding time range for aggregation by clicking on the two buttons at the top of the bar chart, *Aggregate by month* and *Aggregate by year*. The four pie charts on the right side of the figure show the distribution of the four data quality metrics of different dates. It can be seen that among the four metrics, completeness is the worst, all in the range of [0.0-0.6), followed by timeliness and consistency. Validity is the best, all in the range of [0.9-1.0]. Users can enter the detail pages of the four data quality metrics through the navigation bar at the top of the page.

After learning about the data quality of a time series aggregated by different time ranges, users may be interested in the lower data quality metrics and want to know the specific reasons why that data quality problem occurs. To this end, TsQuality Dashboard also gives the explanation of the data quality issue in the form of an outlier list and their possible repairs in the original time series, as shown in Figure 5. The histogram gives the value distribution of this time series. The horizontal axis indicates all value ranges in the data and the vertical axis represents the number of data points whose value is in the corresponding range. The two dashed red arrows in the figure give the minimum and maximum value constraints of the series, which are 4.85 and 5.26 respectively. All data points with values outside this range are considered validity outliers. The red triangle in the graph represents the validity outlier, and the black triangle represents the value of the outlier after repair.

The line chart below shows a possible repair scenario for the outlier point above. The red line gives the distribution of part of the original time series, and the black line gives the possible repair of this segment of time series. As we can see, the value of the outlier is modified to 5.24 because its original value 5.28 exceeds the maximum limit 5.26.

[4] https://thssdb.github.io/TsQuality/

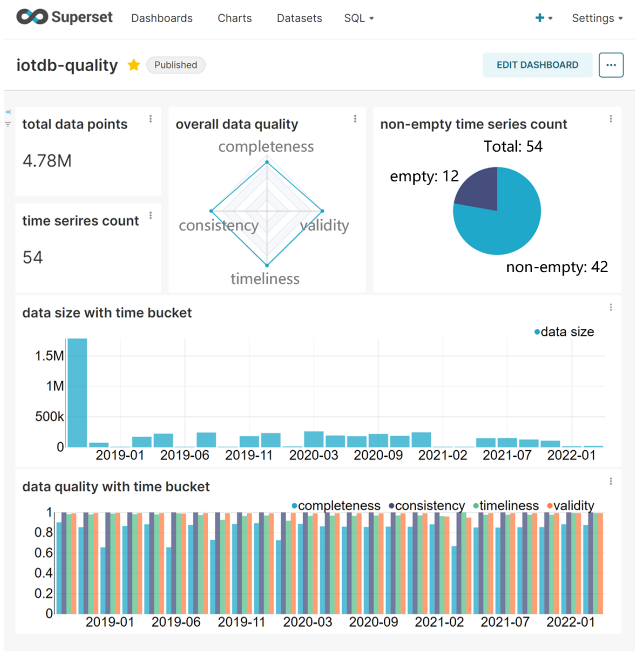


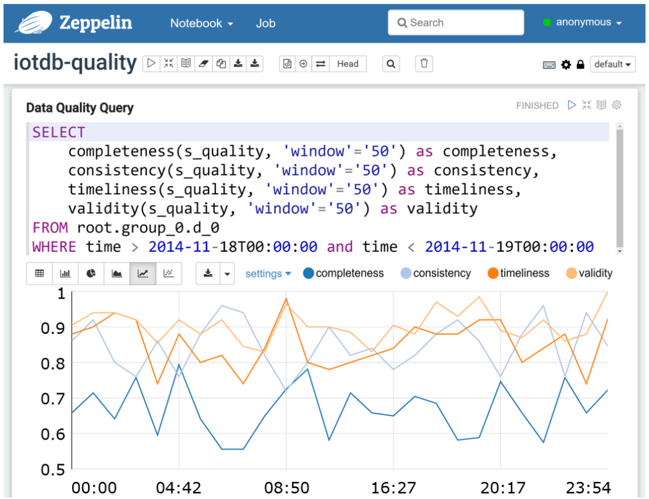
Figure 6: Quality overview of the entire database in Superset

3.2 Apache Superset

Apache Superset[5] is able to give the overall data quality of all time series in IoTDB in the form of a dashboard as illustrated in Figure 6. The top left corner gives the total number of data points as well as the total number of time series and to the right side of which is a radar chart showing the overall data quality of the data in IoTDB. The top right corner of the figure shows the percentage of non-empty time series in the database in a pie chart, where blue and purple represent non-empty and empty time series, respectively. The bar chart in the middle shows the amount of data in the database at different time periods. The time axis is not strictly divided by month; instead, it is determined by the actual time distribution of the data. At the bottom of the figure is the data quality of different time periods. Each of the four bars for each time period corresponds to the four data quality metrics.

3.3 Apache Zeppelin

Apache Zeppelin[6] allows users to input custom SQL query statements and visualize the results in the form of bar charts, scatter plots, and so on. Figure 7 presents an example of IoTDB data quality query in Zeppelin. The first parameter of the function is the name of the sensor in the time series path, and the second parameter *window* specifies the window size when reading the original time series data with a sliding window for calculation. The query results for the four data quality metrics are given at the bottom of the figure in the format of a line chart. The user can hover the cursor over the line chart to examine the specific values of the four data quality metrics. Also, the chart can be zoomed in by dragging the mouse over the horizontal axis to select the time period of interest.



```
SELECT
    completeness(s_quality, 'window'='50') as completeness,
    consistency(s_quality, 'window'='50') as consistency,
    timeliness(s_quality, 'window'='50') as timeliness,
    validity(s_quality, 'window'='50') as validity
FROM root.group_0.d_0
WHERE time > 2014-11-18T00:00:00 and time < 2014-11-19T00:00:00
```

Figure 7: Data quality query in Zeppelin

4 CONCLUSION

We demonstrate TsQuality, a system to measure the data quality in Apache IoTDB. Among the three tools in the interface layer, TsQuality Dashboard provides an overview of data quality for time series, explains why data quality issues occur, and gives the possible repair, while Apache Superset has the advantage of giving the overall data quality of the whole database through aggregate queries. Also, for interactive analysis, users can interact with Apache Zeppelin by executing custom queries and visualizing the results.

ACKNOWLEDGMENTS

This work is supported in part by the National Key Research and Development Plan (2021YFB3300500), the National Natural Science Foundation of China (62072265, 62021002, 62232005), Beijing National Research Center for Information Science and Technology (BNR2022RC01011), and Alibaba Group through Alibaba Innovative Research (AIR) Program. Shaoxu Song (https://sxsong.github.io/) is the corresponding author.

REFERENCES

[1] X. Ding, H. Wang, J. Su, Z. Li, J. Li, and H. Gao. Cleanits: A data cleaning system for industrial time series. *Proc. VLDB Endow.*, 12(12):1786–1789, 2019.
[2] C. Fang, S. Song, and Y. Mei. On repairing timestamps for regular interval time series. *Proc. VLDB Endow.*, 15(9):1848–1860, 2022.
[3] M. K. Shende, A. E. Feijóo-Lorenzo, and N. D. Bokde. cleants: Automated (automl) tool to clean univariate time series at microscales. *Neurocomputing*, 500:155–176, 2022.
[4] S. Song, F. Gao, A. Zhang, J. Wang, and P. S. Yu. Stream data cleaning under speed and acceleration constraints. *ACM Trans. Database Syst.*, 46(3):10:1–10:44, 2021.
[5] S. Song and A. Zhang. Iot data quality. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, pages 3517–3518. ACM, 2020.
[6] Y. Su, Y. Gong, and S. Song. Time series data validity. In *ACM SIGMOD International Conference on Management of Data, SIGMOD*, 2023.
[7] C. Wang, J. Qiao, X. Huang, S. Song, H. Hou, T. Jiang, L. Rui, J. Wang, and J. Sun. Apache IoTDB: A time series database for IoT applications. In *ACM SIGMOD International Conference on Management of Data, SIGMOD*, 2023.

[5] https://superset.apache.org/
[6] https://zeppelin.apache.org/