IoTDBProposal

IoTDB Proposal

v0.1.1

Abstract

IoTDB is a data store for managing large amounts of time series data such as timestamped data from IoT sensors in industrial applications.

Proposal

IoTDB is a database for managing large amount of time series data with columnar storage, data encoding, pre-computation, and index techniques. It has SQL-like interface to write millions of data points per second per node and is optimized to get query results in few seconds over trillions of data points. It can also be easily integrated with Apache Hadoop MapReduce and Apache Spark for analytics.

Background

A new class of data management system requirements is becoming increasingly important with the rise of the Internet of Things[1]. There are some database systems and technologies aimed at time series data management. For example, Gorilla and InfluxDB which are mainly built for data centers and monitoring application metrics. Other systems, for example, OpenTSDB and KairosDB, are built on Apache HBase and Apache Cassandra, respectively.

However, many applications for time series data management have more requirements especially in industrial applications as follows:

- Supporting time series data which has high data frequency. For example, a turbine engine may generate 1000 points per second (i.e., 1000Hz), while each CPU only reports 1 data points per 5 seconds in a data center monitoring application.
- Supporting scanning data multi-resolutionally. For example, aggregation operation is important for time series data.
- Supporting special queries for time series, such as pattern matching, time series segmentation, time-frequency transformation and frequency query.
- Supporting a large number of monitoring targets (i.e. time series). An excavator may report more than 1000 time series, for example, revolving speed of the motor-engine, the speed of the excavator, the accelerated speed, the temperature of the water tank and so on, while a CPU or an application monitor has much fewer time series.
- Optimization for out-of-order data points. In the industrial sector, it is common that equipment sends data using the UDP protocol rather than the TCP protocol. Sometimes, the network connect is unstable and parts of the data will be buffered for later sending.
- Supporting long-term storage. Historical data is precious for equipment manufacturers. Therefore, removing or unloading historical data is highly desired for most industrial applications. The database system must not only support fast retrieval of historical data, but also should guarantee that the historical data does not impact the processing speed for “hot” or current data.
- Supporting online transaction processing (OLTP) as well as complex analytics. It is obvious that supporting analyzing from the data files using Apache Spark/Apache Hadoop MapReduce directly is better than transforming data files to another file format for Big Data analytics.
- Flexible deployment either on premise or in the cloud. IoTDB is as simple and can be deployed on a Raspberry Pi handling hundreds of time series. Meanwhile, the system can be also deployed in the cloud so that it supports tens of millions ingestions per second, OLTP queries in milliseconds, and analytics using Apache Spark/Apache Hadoop MapReduce.
- * (1) If users deploy IoTDB on a device, such as a Raspberry Pi, a wind turbine, or a meteorological station, the deployment of the chosen database is designed to be simple. A device may have hundreds of time series (but less than a thousand time series) and the database needs to handle them.
- * (2) When deploying IoTDB in a data center, the computational resources (i.e., the hardware configuration of servers) is not a problem when compared to a Raspberry Pi. In this deployment, IoTDB can use more computation resources, and has the ability to handle more time seires (e.g., millions of time series).

Based on these requirements, we developed IoTDB, a new data store system for managing time series data.

IoTDB started as a Tsinghua University research project. IoTDB's developer community has also grown to include additional institutions, for example, universities (e.g., Fudan University), research labs (e.g, NEL-BDS lab), and corporations (e.g., K2Data, Tencent). Funding has been provided by various institutions including the National Natural Science Foundation of China, and industry sponsors, such as Lenovo and K2Data.

Rationale

Because there is no existed open-sourced time series databases covering all the above requirements, we developed IoTDB. As the system matures, we are seeking a long-term home for the project. We believe the Apache Software Foundation would be an ideal fit. Also joining Apache will help coordinate and improve the development effort of the growing number of organizations which contribute to IoTDB improving the diversity of our community.

IoTDB contains multiple modules, which are classified into categories:

- **TsFile Format**: TsFile is a new columnar file format.
- **Adaptor for Analytics and Visualization**: Integrating TsFile with Apache Hadoop HDFS, Apache Hadoop MapReduce and Apache Spark. Examples of integrating IoTDB with Apache Kafka, Apache Storm and Grafana are also provided.
- **IoTDB Engine**: An engine which consists of SQL parser, query plan generator, memtable, authentication and authorization,write ahead log (WAL), crash recovery, out-of-order data handler, and index for aggregation and pattern matching. The engine stores system data in TsFile format.
- **IoTDB JDBC**: An implementation of Java Database Connectivity (JDBC) for clients to connect to IoTDB using Java.

TsFile Format

TsFile format is a columnar store, which is similar with Apache Parquet and Apache CarbonData. It has the concepts of Chunk Group, Column Chunk, Page and Footer. Comparing with Apache Parquet and Apache CarbonData, it is designed and optimized for time series:

Time Series Friendly Encoding

IoTDB currently supports run length encoding (RLE), delta-of-delta encoding, and Facebook's Gorilla encoding.

Lossy encoding methods (e.g., Piecewise Linear Approximation (PLA) and time-frequency transformation are works-in-progress.

Chunk Group

The data part of a TsFile consists of many Chunk Groups. Each Chunk Group stores the data of a device at a time interval. A Chunk Group is similar to the row group in Apache Parquet, while there are some constraints of the time dimension: For each device, the time intervals of different Chunk Groups are not overlapped and the latter Chunk Group always has a larger timestamp.

Given a TsFile and a query with a time range filter, the query process can terminate scanning data once it reads data points whose timestamp reaches the time limit of the filter. We call the feature *fast-return* and it makes the time range query in a TsFile very efficient.

Different Column Chunk Format (Unnecessary the Repetition (R) and Definition (D) Fields)

While Apache Parquet and Apache CarbonData support complex data types, e.g., nested data and sparse columns, TsFile is exclusively designed for time series whose data model is \<device_id, series_id, timestamp, value\>.

In a `Chunk Group`, each time series is a `Column Chunk`. Even though these time series belong to the same device, the data points in different time series are not aligned in the time dimension originally.

For example, if you have a device with 2 sensors on the same data collection frequencies, sensor 1 may collect data at time 1521622662000 while the other one collects data at time 1521622662001 (delta=1ms). Therefore, each Column Chunk has its timestamps and values, which is quite different from Apache Parquet and Apache CarbonData. Because we store the time column along with each value column instead of making different chunks share the same time column for the sake of diverse data frequency for different time series, we do not store any null value on disk to align across time series. Besides, we do not need to attach `repetition` (R) and `definition` (D) fields on each value. Therefore, the disk space is saved and the query latency is reduced (because we do not align data by calculating R and D fields).

Domain Specific Information in Each Page

Similar to Apache Parquet and Apache CarbonData, a `Column Chunk` consists of several `Pages`, and each `Page` has a `Page header`. The `Page header` is a summary of the data in the page.

Because TsFile is optimized for time series, the page header contains more domain specific information, such as the minimal and maximal value, the minimal and the maximal timestamp, the frequency and so on. TsFile can even store the histogram of values in the page header.

This header information helps IoTDB in speeding up queries by skipping unnecessary pages.

Adaptor for Analytics

The TsFile provides:

- InputFormat OutputFormat interfaces for Reading/Writing data.
- Deep integration with Apache Spark/Hadoop MapReduce including predicate push-down, column pruning, aggregation push down, etc. So users can use Apache Spark SQL/HiveQL to connect and query TsFiles.

IoTDB Engine

The IoTDB engine is a database engine, which uses TsFile as its storage file format. The IoTDB Engine supports SQL-like query plus many useful functions:

- Tree-based time series schema
- Log-Structured Merge (LSM)-based storage
- Overflow file for out-of-order data
- Scalable index framework
- Special queries for time series

Tree-based Time Series Schema

IoTDB manages all the time series definitions using a tree structure. A path from the root of the tree to a leaf node represents a time series. Therefore, the unique id of a time series is a path, e.g., `root.China.beijing.windFarm1.windTurbine1.speed`.

This kind of schema can express `group by` naturally. For example, `root.China.beijing.windFarm1.*.speed` represents the speed of all the wind turbines in wind farm 1 in Beijing, China.

Log-Structured Merge (LSM)-based Storage

In a time series, the data points should be ordered by their timestamps. In IoTDB, we use Log-Structured Merge (LSM) based mechanism. Therefore, a part of the data is stored in memory first and can be called as `memtable`. At this time, if data points come out-of-order, we resort them in memory. When this part of data exceeds the configured memory limit, we flush it on disk as a `Chunk Group` into an unclosed TsFile. Finally, a TsFile may contain several Chunk Groups, for reducing the number of small data files, which is helpful to reduce the I/O load of the storage system and reduces the execution time of a file-merge in LSM. Notice that the data is time-ordered in one Chunk Group on disk, and this layout is helpful for fast filtering in one Chunk Group for a query.

Rule 1: In a TsFile, the Chunk Groups of one device are ordered by timestamp (Rule 1), and it is helpful for fast filtering among Chunk Groups for a query.

Rule 2: When the size of the unclosed TsFile reaches the threshold defined in the configuration file, we close the file and generate a new one to store new arriving data spanning the entire data set. Like many systems which use LSM-based storage, we never modify a TsFile which has been closed except for the file-merge process (Rule 2).

Rule 3: To reduce the number of TsFiles involved in a query process, we guarantee that the data points in different TsFiles are not overlapping on the time dimension after file mergence (Rule 3).

Overflow File for Out-of-order Data

When a part of data is flushed on disk (and will form a `Chunk Group` in a TsFile), the newly arriving data points whose timestamps are smaller than the largest timestamp in the Tsfile are `out-of-order`.

To store the out-of-order data, we organize all the troublesome `out-of-order` data point insertions into a special TsFile, named `UnSequenceTsFile`. In an UnSequenceTsFile, the Chunk Groups of one device may be overlapping in the time dimension, which violates the Rule 1 and costs additional time compared to a normal TsFile for query filtering.

There is another special operation: updating all the data points in a time range, e.g., `update all the speed values of device1 as 0 where the data time is in [1521622000000, 1521622662000]`. The operation is called when: (1) a sensor malfunctions and the database receives wrong data for a period; (2) we may want to reset all the records. Many NoSQL time series databases do not support such an operation. To support the operation in IoTDB, we use a tree-based structure, Treap, to store this part of operations and store them as `Overflow` files.

Therefore, there are 3 kinds of data files: TsFiles, UnSequenceTsFiles and Overflow files. TsFiles should store most of the data. The volume of UnSequenceTsFiles depends on the workload: if there are too many out-of-order and the time span of out-of-order is huge, the volume will be large. Overflow files handle fewest data operations but will depend on the use of the special operations.

LSM-tree

Normally, LSM-based storage engines merge data files level by level so that it looks like a tree structure. In this way, data is well organized. The disadvantage is that data will be read and written several times. If the tree has 4 levels, each data point will be rewritten at least 4 times.

Currently, we do not merge all the TsFiles into one because (1) the number of TsFiles is kept lower than many LSM storage engines because a memtable is mapped to several Chunk Groups rather than a file; (2) different TsFiles are not overlapping with each other in the time dimension (because of Rule 3).

As mentioned before, TsFile supports *fast-return* to accelerate queries. However, UnSequenceTsFile and Overflow files do not allow this feature. The time spans of UnSequenceTsFile, Overflow file andTsFile may be overlapped, which leads to more files involved in the query process. To accelerate these queries, there is a merging process to reorganize files in the background. All the three kinds of files: TsFiles, UnSequenceTsFiles and Overflow files, are involved in the merging process. The merging process is implemented using multi-threading, while each thread is responsible for a series family. After merging, only TsFiles are left. These files have non-overlapping time spans and support the *fast-return* feature.

Scalable Index Framework

We allow users to implement indexes for faster queries. We currently support an index for pattern matching query (KV-Match index, ICDE 2019). Another index for fast aggregation (PISA index, CIKM 2016) is a work-in-progress.

Special Queries

We currently support `group by time interval` aggregation queries and `Fill by` operations, which are similar to those of InfluxDB. Time series segmentation operations and frequency queries are work-in-progress.

Initial Goals

The initial goals are to be open sourced and to integrate with the Apache development process. Furthermore, we plan for incremental development, and releases along with the Apache guidelines.

Current Status

We have developed the system for more than 2 years. There are currently 13k lines of code, some of which are generated by Antlr3 and Thrift. There are 230 issues which have been solved and more than 1500 commits.

The system has been deployed in the staging environment of the State Grid Corporation of China to handle ~3 million time series (i.e, ~30,000 power generation assembly * ~100 sensors) and an equipment service company in China managing ~2 million time series (i.e, ~20k devices * 100 sensors). The insertion speed reaches ~2 million points/second/node, which is faster than InfluxDB, OpenTSDB and Apache Cassandra in our environment.

There are many new features in the works including those mentioned herein. We will add more analytics functions, improve the data file merge process, and finish the first released version of IoTDB.

Meritocracy

The IoTDB project operates on meritocratic principles. Developers who submit more code with higher quality earn more merit. We have used `Issues` and `Pull Requests` modules on Github for collecting users' suggestions and patches. Users who submit issues, pull requests, documents and help the community management are welcomed and encouraged to become committers.

Community

The IoTDB project users communicate on Github (https://github.com/thulab/tsfile) . Developers make the communication on a website which is similar with JIRA (Currently, only registered users can apply to access the project for communication, url: https://tower.im/projects/36de8571a0ff4833ae9d7f1c5c400c22/). We have also introduced IoTDB at many technical conferences. Next, we will build the mailing list for more convenience, broader communication and archived discussions.

If IoTDB is accepted for incubation at the Apache Software Foundation, the primary goal is to build a larger community. We believe that IoTDB will become a key project for time series data management, and so, we will rely on a large community of users and developers.

TODO: IoTDB is currently on a private Github repository (https://github.com/thulab/iotdb), while its subproject TsFile (a file format for storing time series data) is open sourced on Github (https://github.com/thulab/tsfile).

Core Developers

IoTDB was initially developed by 2 dozen of students and teachers at Tsinghua University. Now, more and more developers have joined coming from other universities: Fudan University, Northwestern Polytechnical University and Harbin Institute of Technology in China. Other developers come from business companies such as Lenovo and Microsoft. We will be working to bring more and more developers into the project making contributions to IoTDB.

Relationships with Other Apache Products

IoTDB requires some Apache products (Apache Thrift, commons, collections, httpclient).

IoTDB-Spark-connector and IoTDB-Hadoop-connector have been developed for supporting analysing time series data by using Apache Spark and MapReduce.

Overall, IoTDB is designed as an open architecture, and it can be integrated with many other systems in the future.

As mentioned before, in the IoTDB project, we designed a new columnar file format, called TsFile, which is similar to Apache Parquet. However, the new file format is optimized for time series data.

Known Risks

Orphaned Products

Given the current level of investment in IoTDB, the risk of the project being abandoned is minimal. Time series data is more and more important and there are several constituents who are highly inspired to continue development. Tsinghua and NEL-BDS Lab relies on IoTDB as a platform for a large number of long-term research projects. We have deployed IoTDB in some company's staging environments for future applications.

Inexperience with Open Source

Students and researchers in Tsinghua University have been developing and using open source software for a long time. It is wonderful to be guided to join a formal open-source process for students. Some of our committers have experiences contributing to open source, for example:

- druid: https://github.com/druid-io/druid/commit/f18cc5df97e5826c2dd8ffafba9fcb69d10a4d44
- druid: https://github.com/druid-io/druid/commit/aa7aee53ce524b7887b218333166941654788794
- YCSB: https://github.com/brianfrankcooper/YCSB/pull/776

Additionally, several ASF veterans and industry veterans have agreed to mentor the project and are listed in this proposal. The project will rely on their guidance and collective wisdom to quickly transition the entire team of initial committers towards practicing the Apache Way.

Reliance on Salaried Developers

Most of current developers are students and researchers/professors in universities, and their researches focus on big data management and analytics. It is unlikely that they will change their research focus away from big data management. We will work to ensure that the ability for the project to continuously be stewarded and to proceed forward independent of salaried developers is continued.

An Excessive Fascination with the Apache Brand

Most of the initial developers come from Tsinghua University with no intent to use the Apache brand for profit. We have no plans for making use of Apache brand in press releases nor posting billboards advertising acceptance of IoTDB into Apache Incubator.

Initial Source

IoTDB's github address and some required dependencies:

- The storage file format: https://github.com/thulab/tsfile
- Adaptor for Apache Hadoop MapReduce: https://github.com/thulab/tsfile-hadoop-connector
- Adaptor for Apache Spark: https://github.com/thulab/tsfile-spark-connector
- Adaptor for Grafana: https://github.com/thulab/iotdb-grafana
- The database engine: https://github.com/thulab/iotdb (private project up to now)
- The client driver: https://github.com/thulab/iotdb-jdbc

External Dependencies

To the best of our knowledge, all dependencies of IoTDB are distributed under Apache compatible licenses. Upon acceptance to the incubator, we would begin a thorough analysis of all transitive dependencies to verify this fact and introduce license checking into the build and release process.

Documentation

- Documentation for TsFile: https://github.com/thulab/tsfile/wiki
- Documentation for IoTDB and its JDBC: http://tsfile.org/document (Chinese only. An English version is in progress.)

Required Resources

Mailing Lists

- private@iotdb.incubator.apache.org
- dev@iotdb.incubator.apache.org
- commits@iotdb.incubator.apache.org

Git Repositories

- https://git-wip-us.apache.org/repos/asf/incubator-iotdb.git

Issue Tracking

- JIRA IoTDB (We currently use the issue management provided by Github to track issues.)

Initial Committers

Tsinghua University, K2Data Company, Lenovo, Fundan University, Microsoft

Jianmin Wang (jimwang at tsinghua dot edu dot cn)

Xiangdong Huang (sainthxd at gmail dot com)

Jun Yuan (richard_yuan16 at 163 dot com)

Chen Wang (wang_chen at tsinghua dot edu dot cn)

Jialin Qiao (qjl16 at mails dot tsinghua dot edu dot cn)

Jinrui Zhang (jinrzhan at microsoft dot com)

Rong Kang (kr11 at mails dot tsinghua dot edu dot cn)

Tian Jiangjiangtia18 at mails dot tsinghua dot edu dot cn

Shuo Zhang (zhangshuo at k2data dot com dot cn)

Lei Rui (rl18 at mails dot tsinghua dot edu dot cn)

Rui Liu (liur17 at mails dot tsinghua dot edu dot cn)

Kun Liu (liukun16 at mails dot tsinghua dot edu dot cn)

Gaofei Cao (cgf16 at mails dot tsinghua dot edu dot cn)

Xinyi Zhao (xyzhao16 at mails dot tsinghua dot edu dot cn)

Yi Xu(x-y16 at mails dot tsinghua dot edu dot cn)

Dongfang Mao (maodf17 at mails dot tsinghua dot edu dot cn)

Tianan Li(lta18 at mails dot tsinghua dot edu dot cn)

Yue Su (suy18 at mails dot tsinghua dot edu dot cn)

Hui Dai (daihui_iot at lenovo dot com , yuct_iot at lenovo dot com)

Sponsors

Champion

Kevin A. McGrail (kmcgrail@apache.org)

Nominated Mentors

Justin Mclean (justin at classsoftware dot com)

Christofer Dutz (christofer.dutz at c-ware dot de)

Willem Ning Jiang (willem.jiang at gmail dot com)

Joe Witt (joe.witt at gmail dot com)

References:
[1] https://en.wikipedia.org/wiki/Internet_of_things