

A SUPPLEMENTARY

A.1 Implementations in IoTDB

All algorithms we presented in this paper have been fully implemented in Apache IoTDB, an open-source time series database system. The corresponding Java classes are Data allocation table [N7], Data partition table [N8], Greedy CopySet replication algorithm [N9] and Min cost flow distribution algorithm [N10].

A.2 Customers Scenarios

IoTDB has served over 1000 backbone and industrial leading enterprises [N11]. This section presents typical user scenarios.

A.2.1 Changan Automobile. Changan Automobile [N12] is one of the "Big Four" state-owned car manufacturers of China. Changan Automobile built a smart platform for querying massive amounts of connected vehicle data and remote diagnostics based on Apache IoTDB.

A.2.2 Mcc Cisd. Mcc Cisd [N13] is one of the core subsidiaries of MCC under China Minmetals, the Fortune Global 500. Mcc Cisd provides overall solutions for domestic and foreign steel industry customers. Mcc Cisd built the core part of the Water and Soil Cloud Industrial IoT Platform for data development governance base on Apache IoTDB.

A.2.3 AUTO AI. AUTO AI [N14] is a provider of technologies and integrated hard/software solutions in the field of intelligent cockpit. It uses Apache IoTDB to manage the driving behavior data of all service vehicles, with a total of 1.6 million online vehicles every day.

The real customer workload is shown at Table 3. Taking AUTO AI as an example, we collected its production cluster metric from May 2, 2024 to May 9, 2024 at Table 4. It can be seen that the IoTDB cluster is balanced in both computing and storage since the write throughput and the used disk space fluctuate very little, where the coefficient of variation is less than 4%.

A.3 Experiment Setup

We first conducted the simulated evaluation on a personal Mac computer, which is equipped with 10-core M1 Pro chip and 16 GB memory. Then we deployed an IoTDB cluster consisting of 11-nodes on Tencent Cloud for processing the IoTDB evaluation, which covers the typical cluster size as shown in Table 3. Each node is equipped with a 4-core CPU, 8 GB memory, 100 GB SSD storage and 1.5 GB ethernet. Of these 11 nodes, 8 are used to deploy DataNodes, 2 are used to deploy IoT-benchmark, and the remaining 1 is used to deploy ConfigNode.

A.3.1 IoT-Benchmark. IoT-benchmark [N15] is an open-source benchmark for TSDB in IoT scenario, which we used for generating periodic time series data according to the configuration and send the data batch by batch to IoTDB cluster. We configured IoT-benchmark to simulate sensors with different load, configured batch size to 100, and assembled the data generated by every 100 sensors into one write request, with the average size of each write request being 60 KB. Finally, the IoTDB cluster is written about 40 million data points per second. The benchmark load covers real customer scenarios as shown in Table 3.

Table 3: Customer workload

Enterprise	Changan Automobile	Mcc Cisd	AUTO AI
Cluster Size (DataNodes)	3	3	6
CPU core (per DataNode)	64	8	16
Memory (per DataNode)	256 GB	32 GB	128 GB
Disk (per DataNode)	10 TB	2 TB	1.2 TB
Time Series	1.2 billion	2 million	64.8 million
Write Frequency	Per 30 s	70% per 1 s 20% per 10 s 10% per 0.1 s	Per 3 s
Write Throughput (Data Points)	avg 6 m/s max 10 m/s	avg 3 m/s	avg 1.55 m/s max 3 m/s
TTL	3 months	7 days	5 days
Time Partition	7 days	7 days	7 days

Table 4: IoTDB cluster in AUTO AI

Load Metric	Write Throughput	Used Disk Space
DataNode 1	516113/s	231 GB
DataNode 2	516792/s	235 GB
DataNode 3	521633/s	238 GB
DataNode 4	523417/s	229 GB
DataNode 5	508919/s	244 GB
DataNode 6	510220/s	252 GB
Mean	516182.33	238.17
Standard Deviation	5843.49	8.61
Coefficient of Variation	1.13%	3.62%

A.3.2 Alternative Algorithms. We implemented the following DataRegion placement algorithms [N16] to compare with our GCR algorithm. The first one is a Greedy algorithm which always place the new DataRegion to the DataNode with the smallest number of DataRegion since it's a intuitive solution. We then implemented the Copyset [4] and Tiered Replication [3] algorithms. Because the optimization of GCR algorithm in disaster recovery capability is inspired by these two papers. For all DataRegion placement algorithms, we have them generate the same number of DataRegion placement schemes so that each DataNode ends up holding an average of 6 DataRegions, where the load factor 6 is a reasonable amount we tested beforehand to determine the appropriate number of DataRegions hosted by each DataNode.

We implemented the following primary replica selection algorithms [N17] to compare with our CFD algorithm. The first one is a Greedy algorithm which always select the DataRegion on the DataNode that holds the fewest number of primary replica currently as the primary one to act as an intuitive solution. The second one is a Random algorithm which select a random DataRegion as the primary one. We want to simulate the situation where the DataRegionGroup elects the primary replica by itself, and each

DataRegion has the same weight to be elected as the primary one by using this algorithm.

A.4 Supplementary experiments

We will redo the Section 6.2 IoTDB evaluation, which we hope can answer the following questions that reviewers are concerned about:

- R2D1: Add the comparison of time series dimension. The number of time series will be increased proportionally after the cluster is expanded in expansion experiment.
- R3O2: Broader deployment and evaluation. As shown in Table 3, most of our customers can meet their business requirement through deploying an IoTDB cluster with not exceeding six DataNodes. Thus, to better demonstrate the expansion capabilities that benefit from our algorithms. We will set the number of DataNodes to 16 and deploy a 19-nodes IoTDB cluster for expansion experiment.
- R1D6, R3D1: Create Unbalanced partitions, using real world dataset. We have configured IoT-benchmark to generate sensors with different load previously. However, since real production environments tend to be more complex, we'll use IoT-benchmark to read a real dataset and then write to the IoTDB cluster.

To sum up, we will redo the IoTDB evaluation mentioned at Section 6.2 by reconfiguring IoT-benchmark to read a real dataset and then write to the IoTDB cluster. Furthermore, we'll start from an 8-DataNode IoTDB cluster and extends it to 16 DataNodes, exponentially increase the number of time series after expansion for the expansion experiment.

[N7] Data allocation table. <https://github.com/apache/iotdb/blob/master/iotdb-core/confignode/src/main/java/org/apache/iotdb/confignode/manager/load/balancer/partition/DataPartitionPolicyTable.java>

[N8] Data partition table. <https://github.com/apache/iotdb/blob/master/iotdb-core/node-commons/src/main/java/org/apache/iotdb/commons/partition/DataPartitionTable.java>

[N9] Greedy CopySet replication algorithm. <https://github.com/apache/iotdb/blob/master/iotdb-core/confignode/src/main/java/org/apache/iotdb/confignode/manager/load/balancer/region/GreedyCopySetRegionGroupAllocator.java>

[N10] Min cost flow distribution algorithm. <https://github.com/apache/iotdb/blob/master/iotdb-core/confignode/src/main/java/org/apache/iotdb/confignode/manager/load/balancer/router/leader/MinCostFlowLeaderBalancer.java>

[N11] User cases. <https://www.timecho-global.com/product>

[N12] Changan Automobile. https://en.wikipedia.org/wiki/Changan_Automobile

[N13] Mcc Cisi. <http://www.cisdigroup.com.cn/html/2/about>

[N14] AUTO AI. <https://www.autoai.com/en>

[N15] IoT-benchmark. <https://github.com/thulab/iot-benchmark>

[N16] Alternatives of replica placement algorithms. <https://github.com/CRZbulabula/iotdb/tree/elastic-storage/iotdb-core/confignode/src/main/java/org/apache/iotdb/confignode/manager/load/balancer/region>

[N17] Alternatives of primary replica selection algorithms. <https://github.com/CRZbulabula/iotdb/tree/elastic-storage/iotdb-core/confignode/src/main/java/org/apache/iotdb/confignode/manager/load/balancer/router/leader>