

Optimizing Apache Horaedb for High-Cardinality Metrics at AntGroup

Jiacai Liu
July 27, 2024



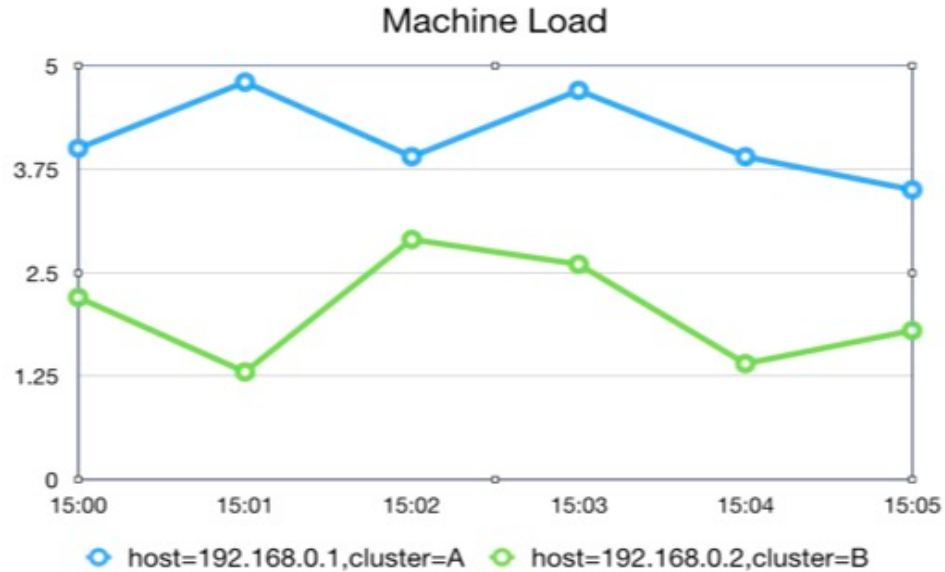
Table of Contents

- 1. What's time series
- 2. What's HoraeDB
 - 2.1. Core design
- 3. Write path optimization
- 4. Query path optimization
- 5. Looking Forward

About ME

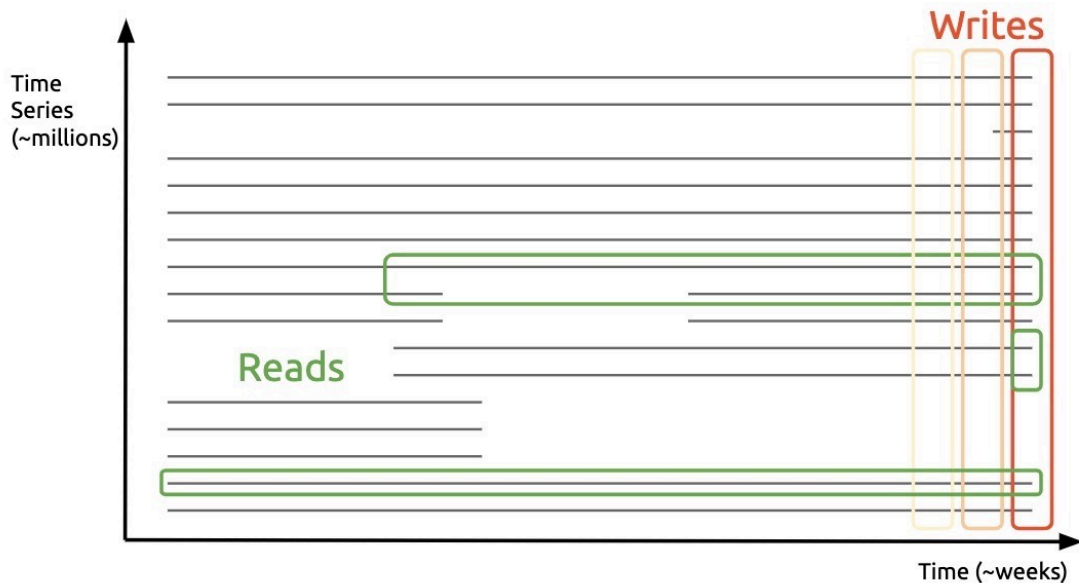
- Senior Engineer at AntGroup
- 5 years of experience with Rust
- [Apache HoraeDB](#) PPMC Member
-  Open source
 - Podcast: [RustTalk](#)
 - <https://github.com/jiacai2050>

1. What's time series



- A collection of time-based data points that can be connected into (time) lines.
- Use tags to differentiate between lines

Characteristics



- Vertical writes, horizontal(-ish) reads

Scenarios

- IoT
- APM (Application Performance Monitoring)
- Weather Forecasting
- Stock Market Analysis

Time series database

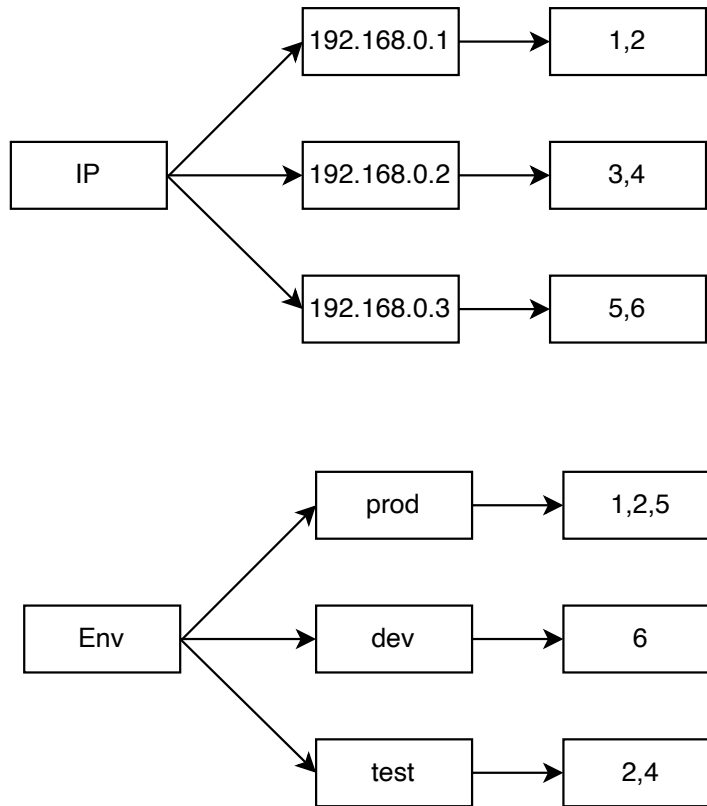
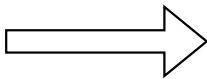
Specialized database that efficiently stores and retrieves time-stamped data

- Prometheus
- InfluxDB
- TimescaleDB
- Apache HoraeDB
-

Challenge

- High write throughput
- High-Cardinality tags, lead to BIG index
- Real-time OLAP like query pattern

TSID	Time	IP	Env	Value
1	10:00	192.168.0.1	prod	50.1
2	10:00	192.168.0.1	test	10.2
3	10:00	192.168.0.2	prod	29.5
4	10:00	192.168.0.2	test	4.5
5	10:00	192.168.0.3	prod	43.1
6	10:00	192.168.0.3	dev	20.2



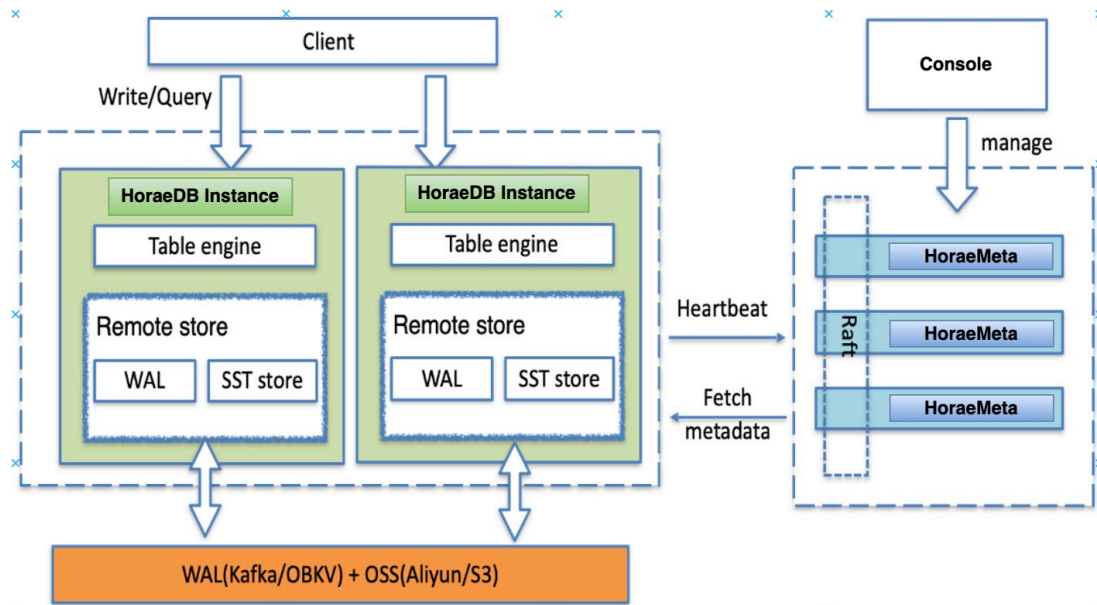
Inverted Index

2. What's HoraeDB

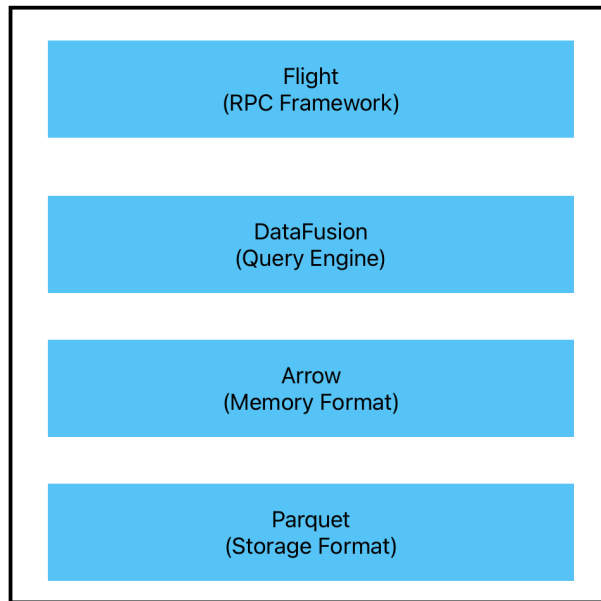
Distributed, Enhanced Analytical Ability for Time Series

2.1. Core design

- Separating compute from storage with object storage
- **FDAP** stack
- **BRIN**(Block range index)
 - Min/Max Index
 - **Xor Filters**



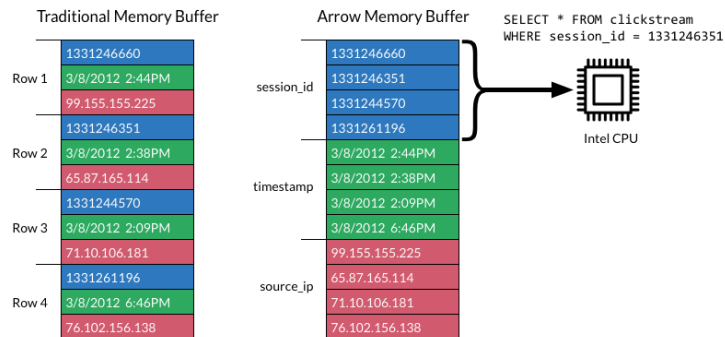
Overall Architecture



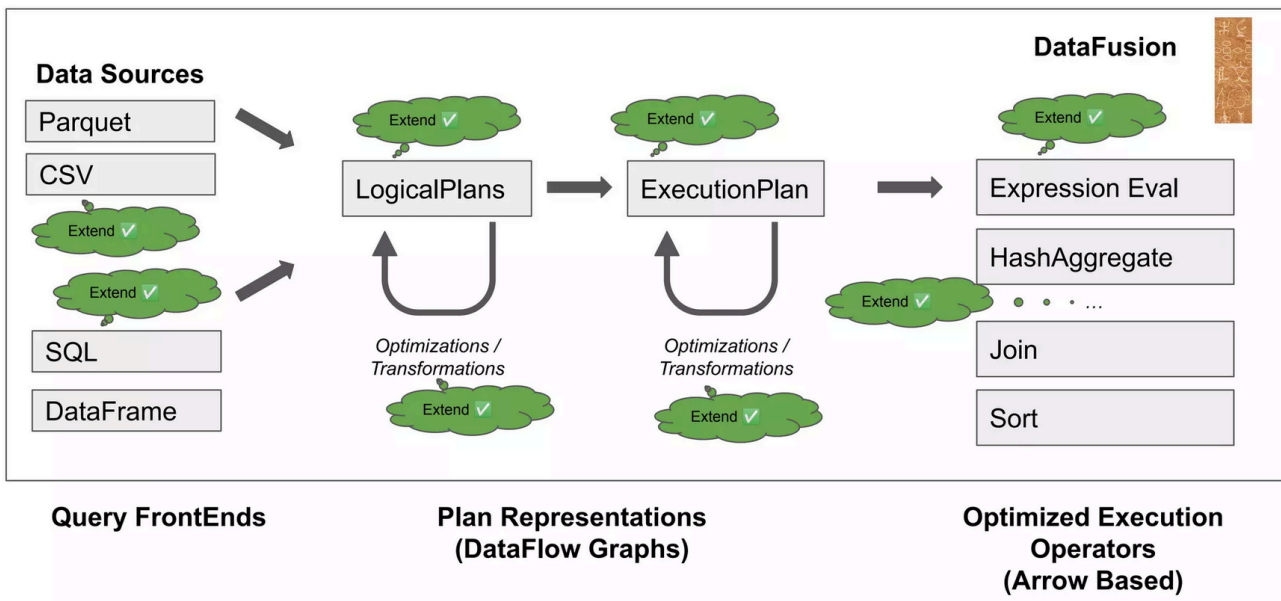
FDAP stack

How it works for time series

	session_id	timestamp	source_ip
Row 1	1331246660	3/8/2012 2:44PM	99.155.155.225
Row 2	1331246351	3/8/2012 2:38PM	65.87.165.114
Row 3	1331244570	3/8/2012 2:09PM	71.10.106.181
Row 4	1331261196	3/8/2012 6:46PM	76.102.156.138



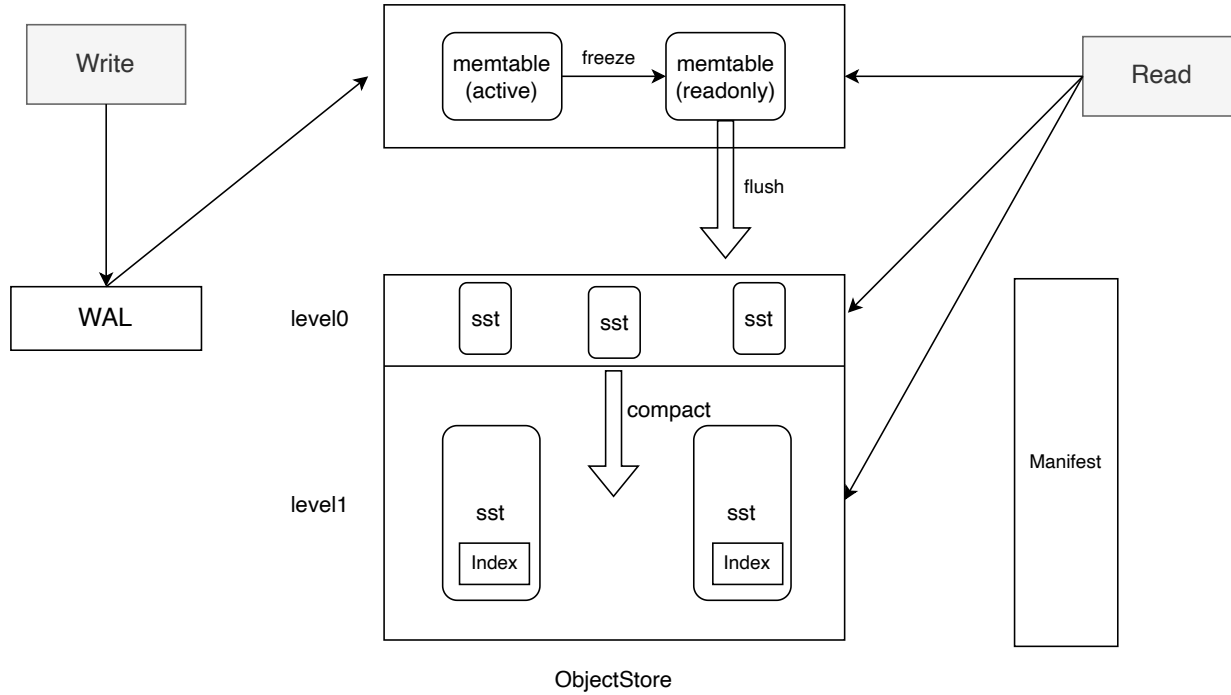
Arrow



DataFusion: LLVM-like Infrastructure for Databases

3. Write path optimization

- Metrics are sharded with partitioned table
 - Hash
 - Range
 - Round-robin
- Reduce IO as possible as we can
 - Group commit for WAL
 - Skip building the index for recently-written metrics

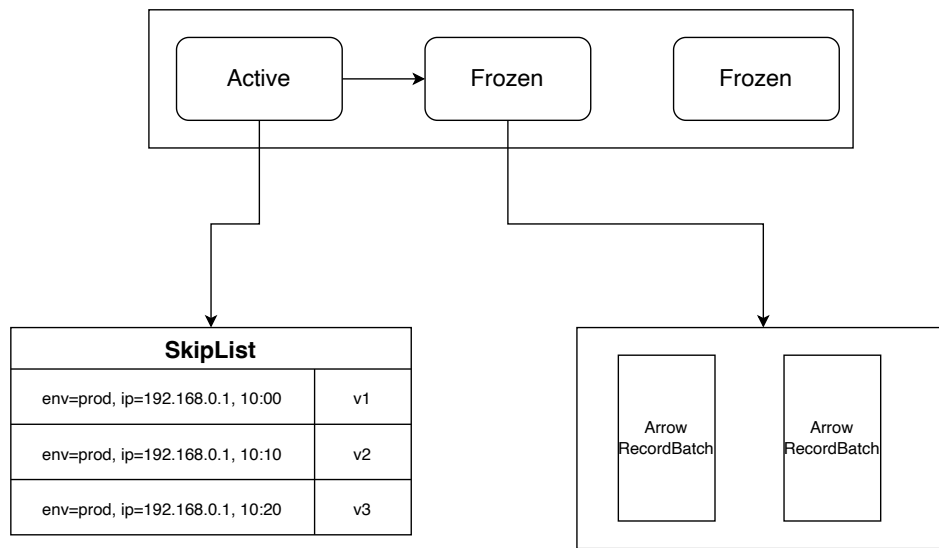


LSM-like engine

4. Query path optimization

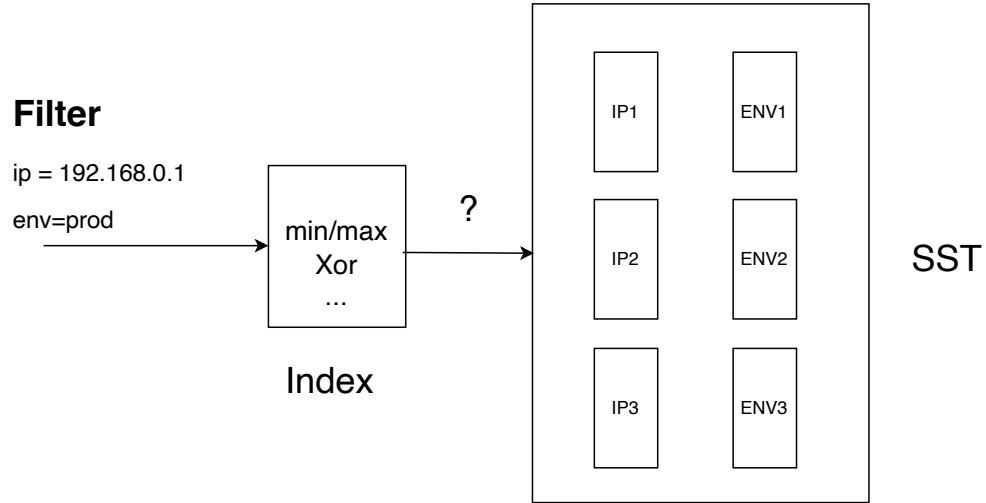
- Reduced IO without inverted index
 - Memtable
 - SST
- Distributed query
 - Partitioned table routing
 - Distributed execution

Reduced IO – Memtable

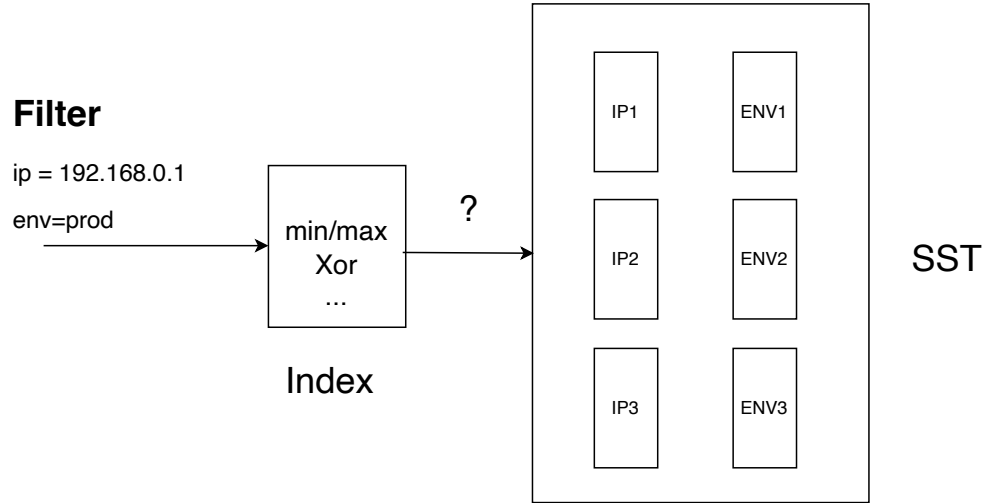


- Active: Write optimized
- Frozen: Read optimized

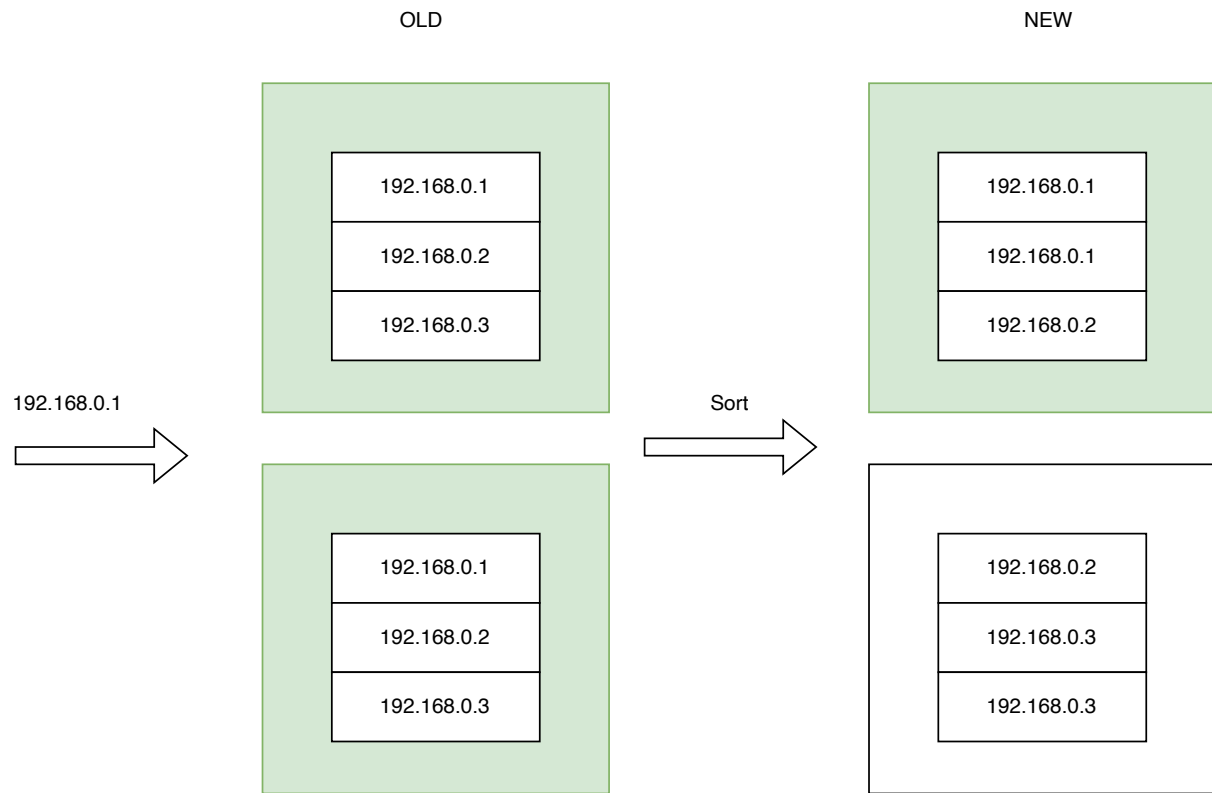
Reduced IO – SST



Reduced IO – SST

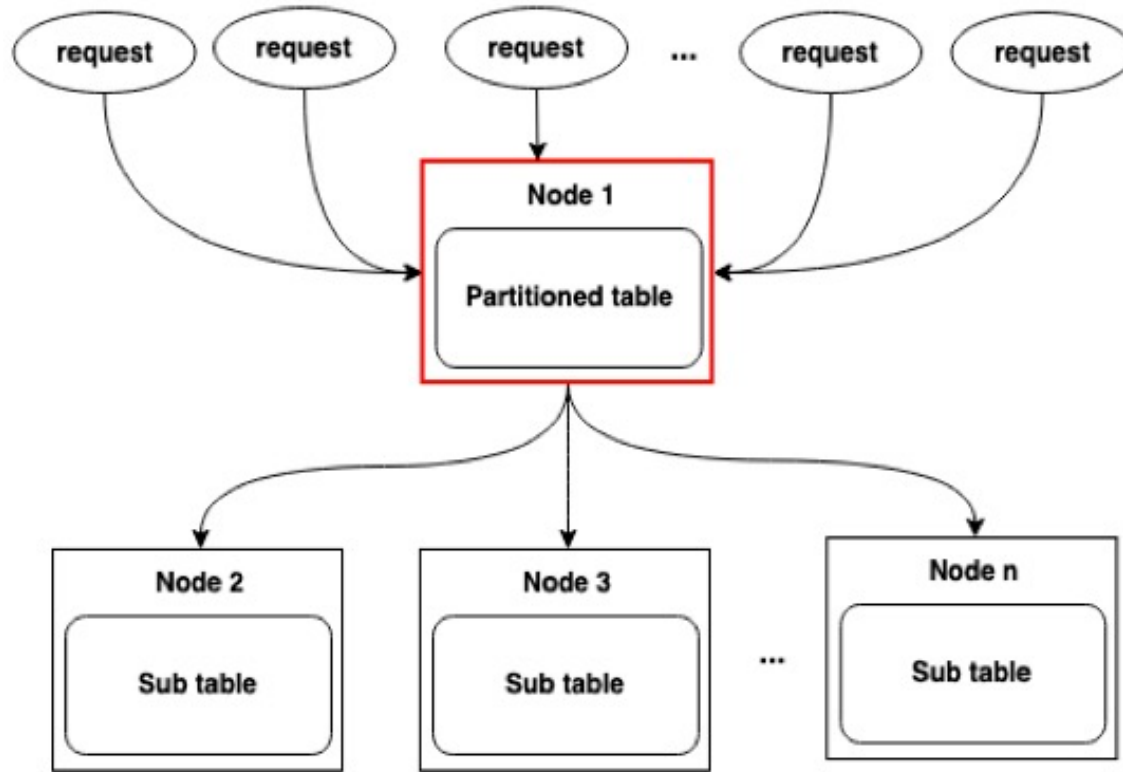


Order is important!

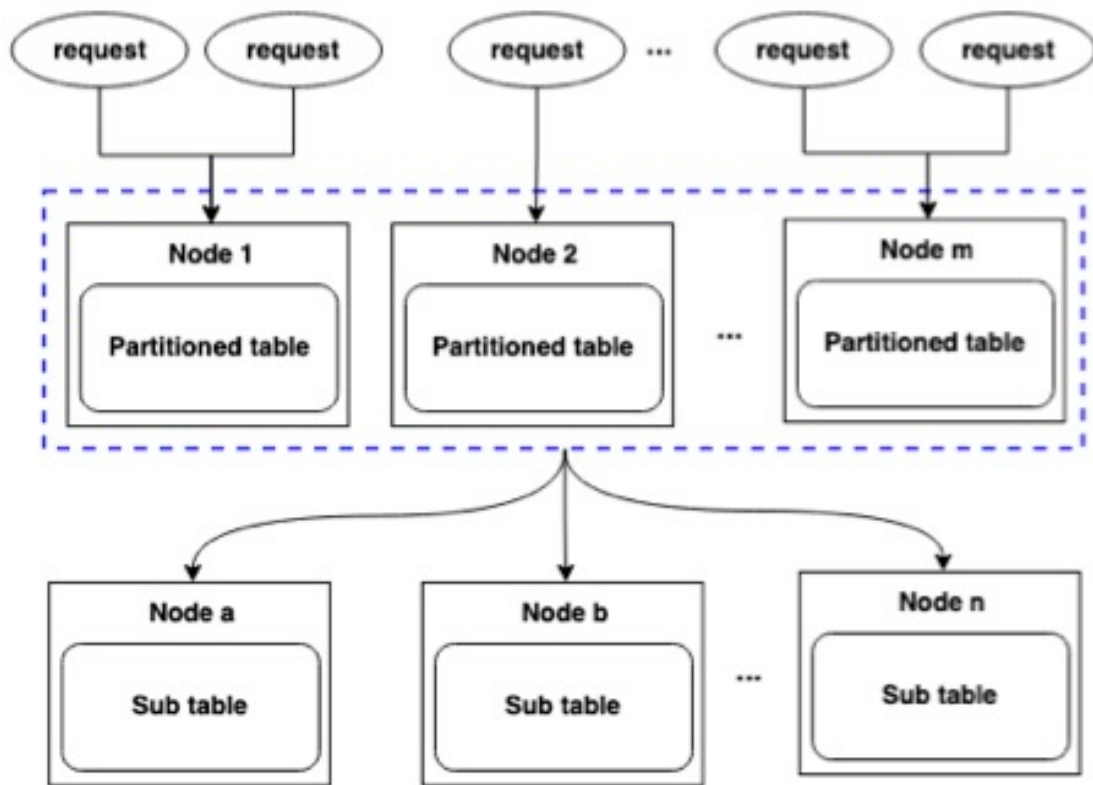


Automatic clustering based on history queries

Distributed



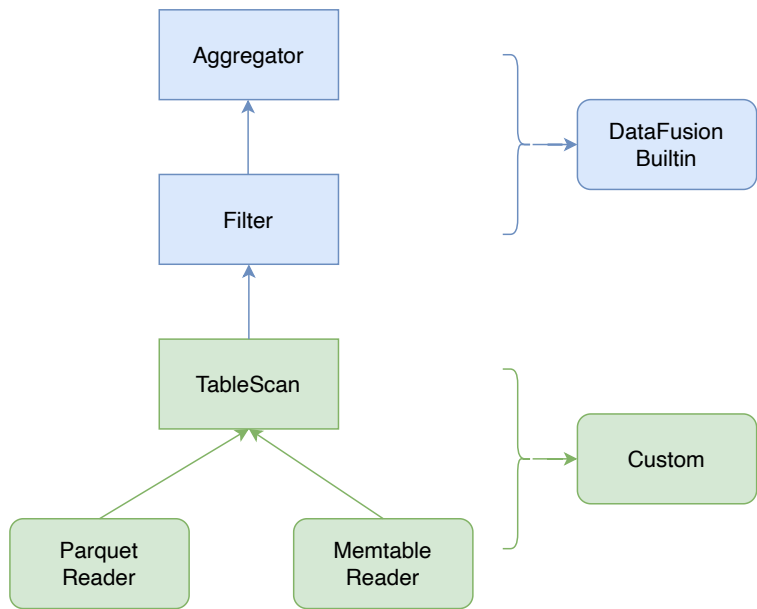
Open as a "normal" table (single point hotspot)



Open as a "virtual" table

```
SELECT
    time_bucket(`timestamp`, '5 min') AS `timestamp`,
    SUM(`value`) AS `value_sum`
FROM
    `table`
WHERE
    `timestamp` >= '2023-12-15 07:17:00'
    AND `timestamp` < '2023-12-14 08:17:00'
    AND ((`col2` IN ('T'))
GROUP BY
    time_bucket (`timestamp`, '5 min')
```

Query pattern in HoraeDB



Simplified physical plan

```
pub enum AggregateMode {  
    /// Partial aggregate that can be applied in parallel across input part  
    Partial,  
    /// Final aggregate that produces a single partition of output  
    Final,  
    ....  
}
```

AggregateMode

ProjectionExec:

AggregateExec: mode=FinalPartitioned, gby=[..], aggr=[SUM(value)],

CoalesceBatchesExec:

RepartitionExec:

AggregateExec: mode=Partial, gby=[..], aggr=[SUM(value)]

ProjectionExec:

ScanTable: table=my_table, filters=[..]

Pushdown

5. Looking Forward

- Inverted index for low cardinality tags
- Teach query engine aware of data distribution patterns(TSID)
- Release more ASF-compliant versions, growing community

Thanks

<https://horaedb.apache.org/>

