

# 小米基于 Paimon 的 Doris 优化与实践

王龙 高级开发工程师



# 目录

1.Doris 在小米的落地过程

2.Doris on Paimon 优化

3.未来规划

# 目录

1.Doris 在小米的落地过程

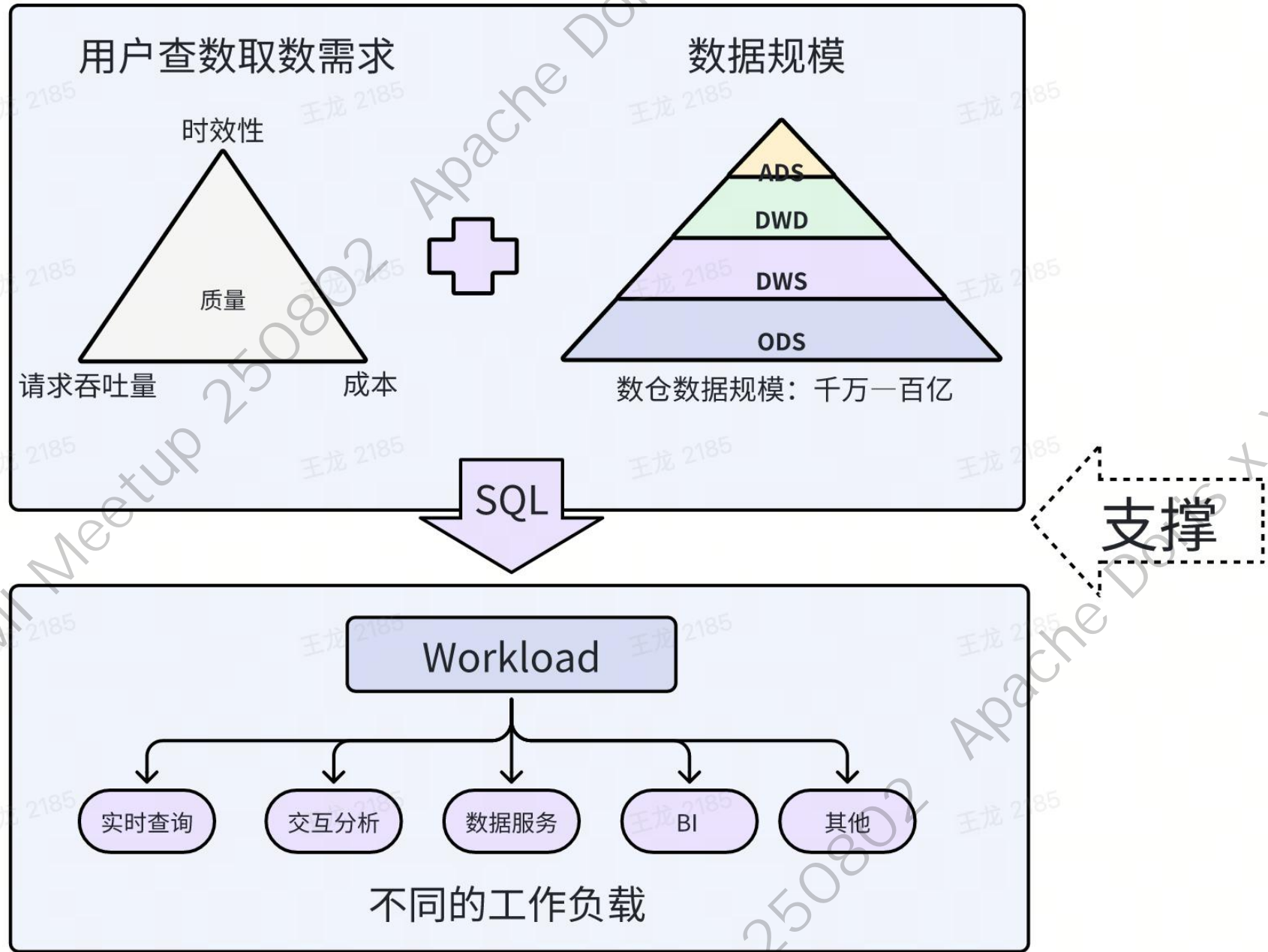
2.Doris on Paimon 优化

3.未来规划

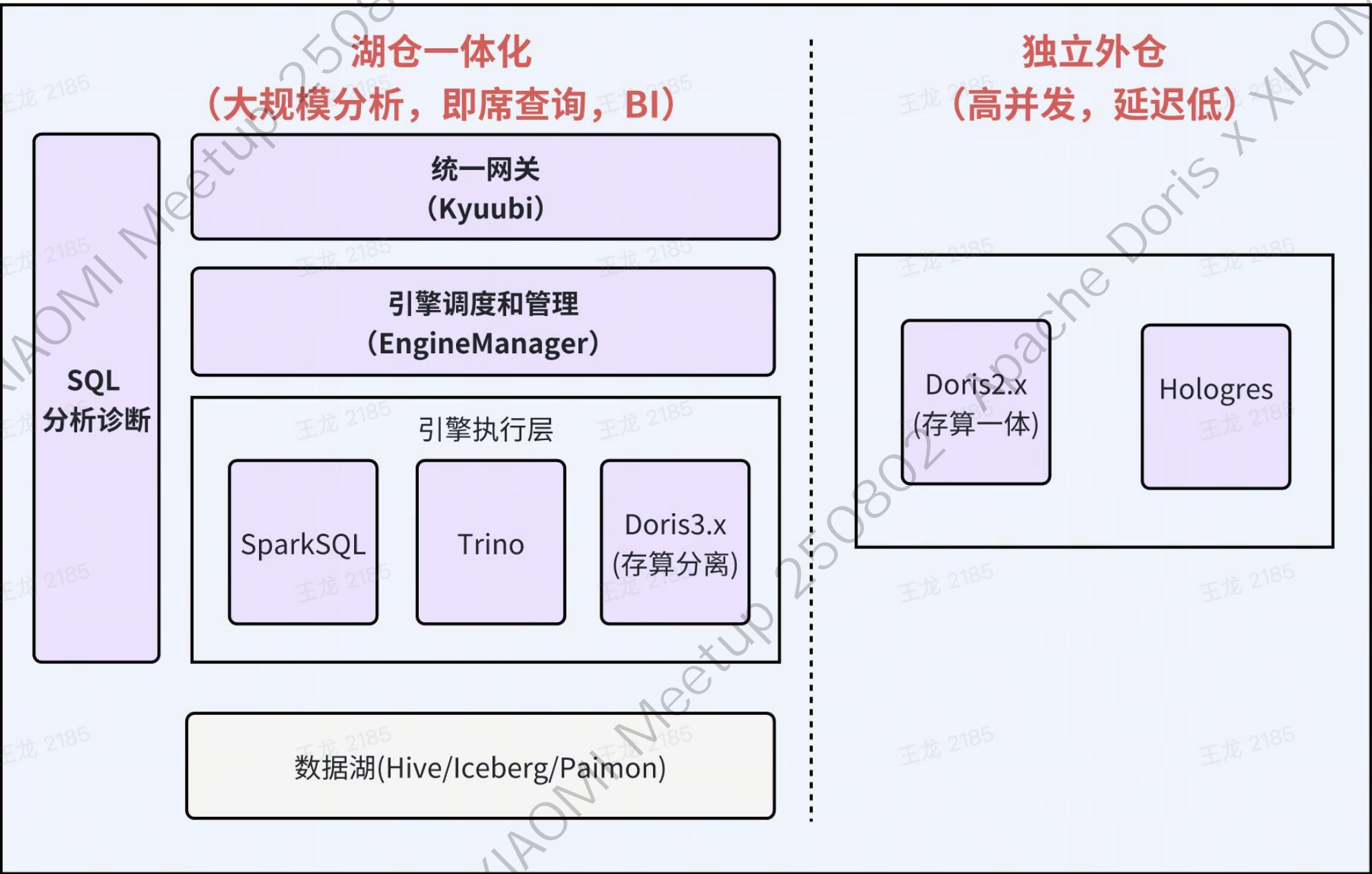


# OLAP 总体架构

## 多样性的业务需求

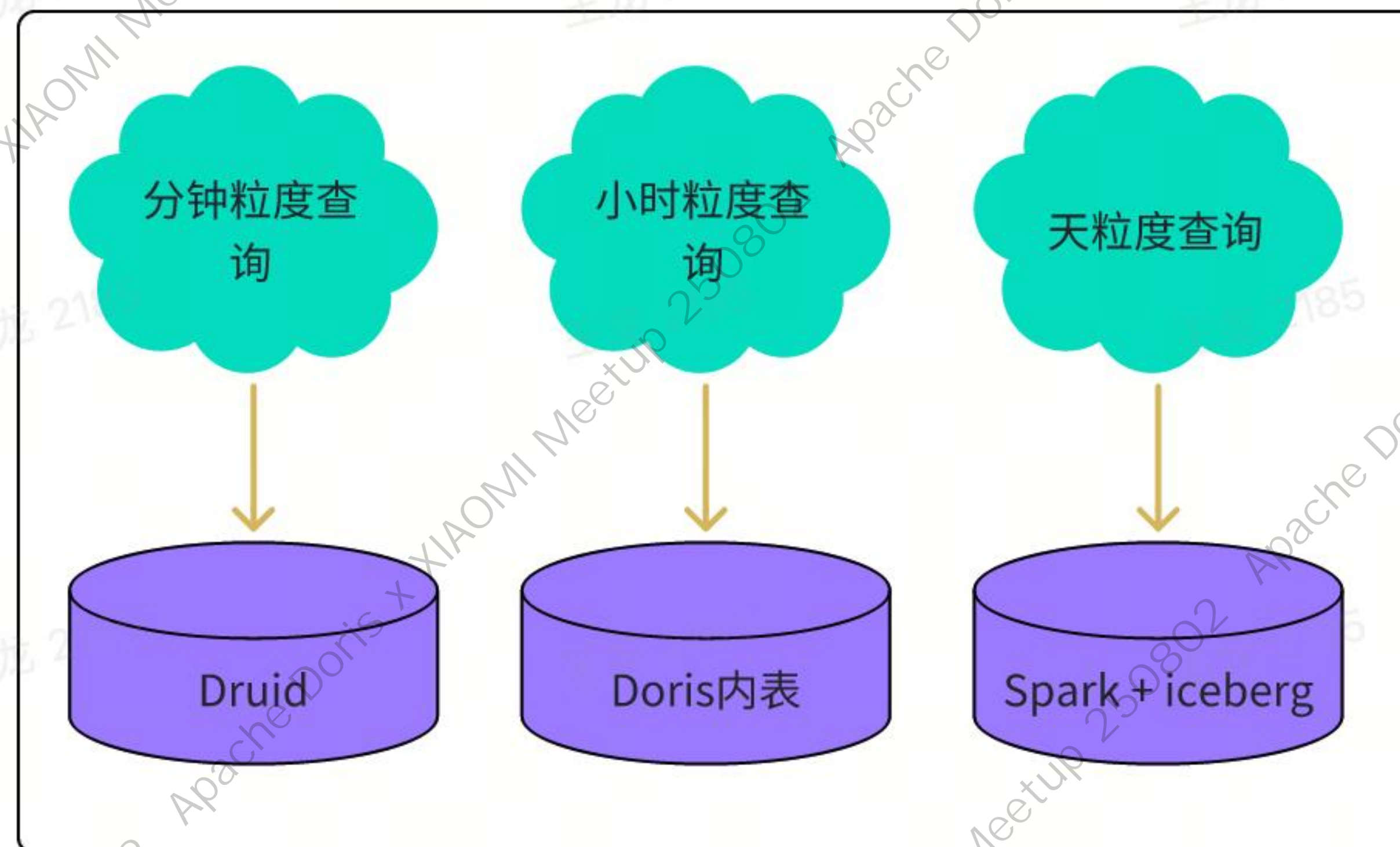


## OLAP 技术架构



## 业务痛点

- **数据冗余**  
数据同步多套系统
- **维护成本高**  
管理多个数据链路

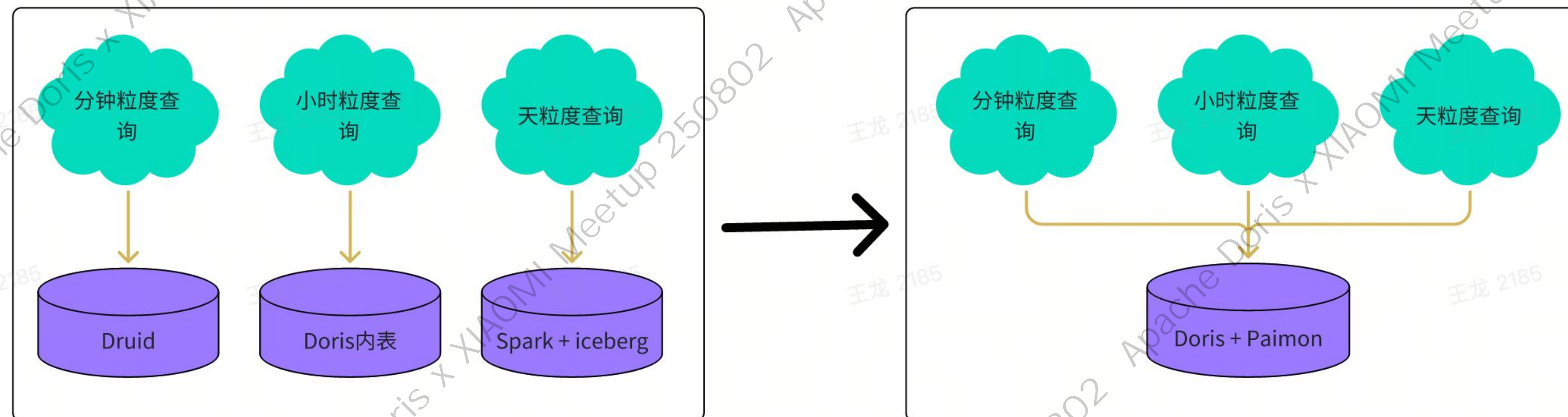




# 架构收敛

•引擎收敛  
统一使用 Doris

•存储收敛  
统一采用 Paimon



## 为什么选择 Doris?

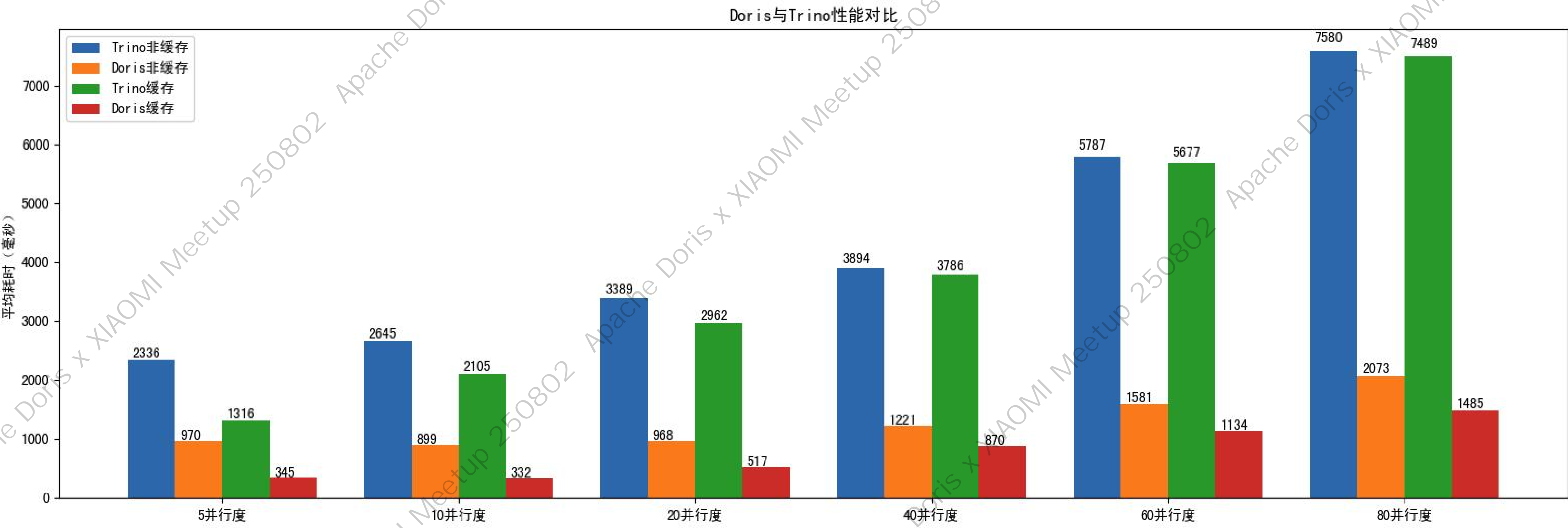
- **Native + 向量化**
- **支持本地缓存**
- **支持物化视图**
- **支持索引加速**



# DORIS

# 为什么选择 Doris?

高并发情况下，Doris 性能是 Presto 的 5 倍





# 目录

1.Doris 在小米的落地过程

2.Doris on Paimon 优化

3.未来规划

## Doris on Paimon 面临的问题

- **Paimon 读取速度慢：需要合并文件**
- **基于 Paimon 的物化视图只支持全量/分区更新，速度慢，代价高**
- **读取 HDFS 上的 Paimon 数据频繁长尾**
- **Paimon 元数据缓存更新延时，时效性差，命中率低**

## 优化方案

- **Paimon merge on Doris 优化**
- **基于 Paimon 的增量物化视图**
- **基于重试机制的 HDFS 长尾 Task 优化**
- **定时刷新 Paimon 元数据缓存**



# Paimon merge on Doris 优化

## 主要性能瓶颈：合并文件

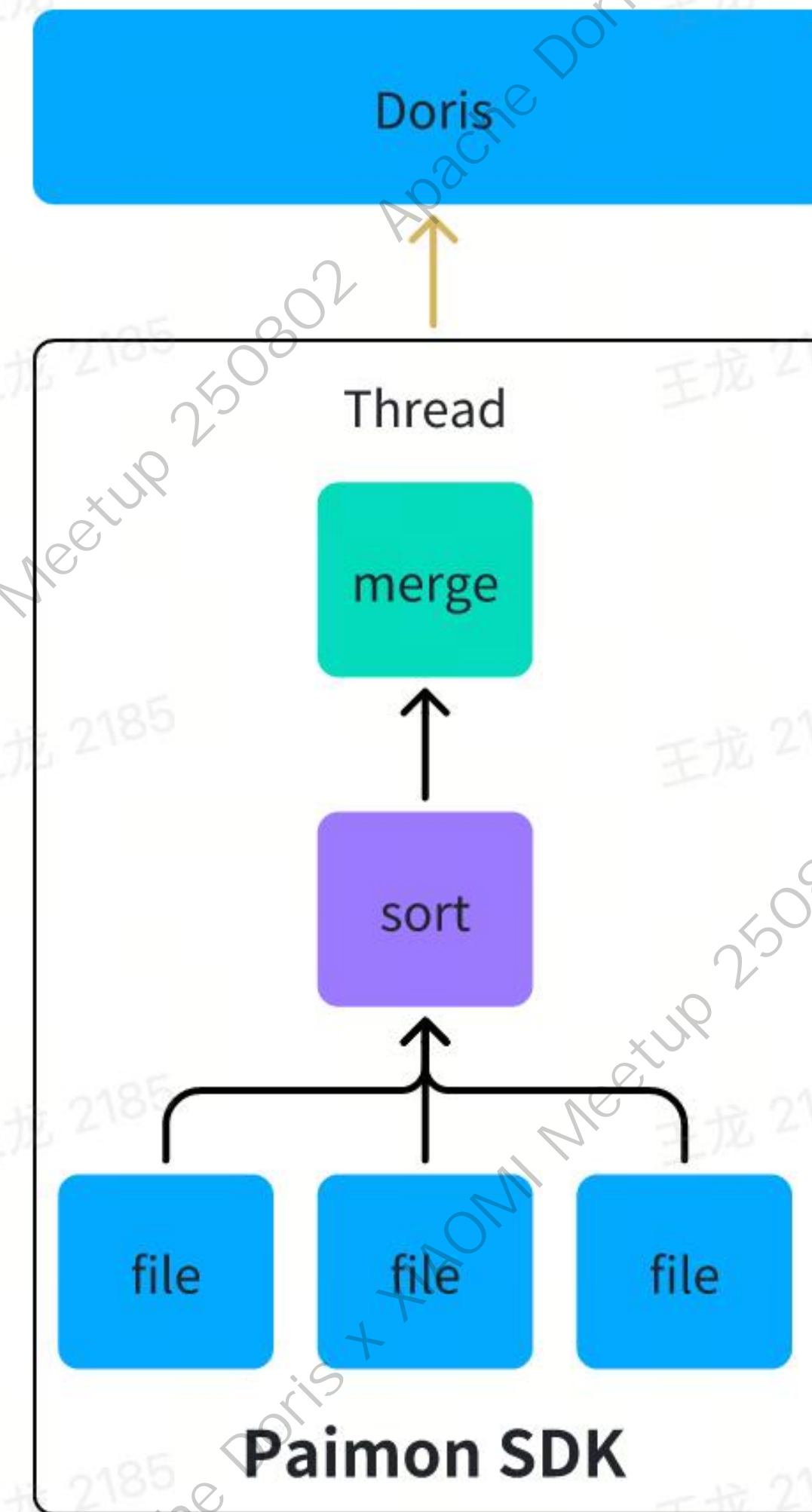




# Paimon merge on Doris 优化

## 为什么慢?

- 单线程合并多文件
- 数据排序
- JAVA 执行

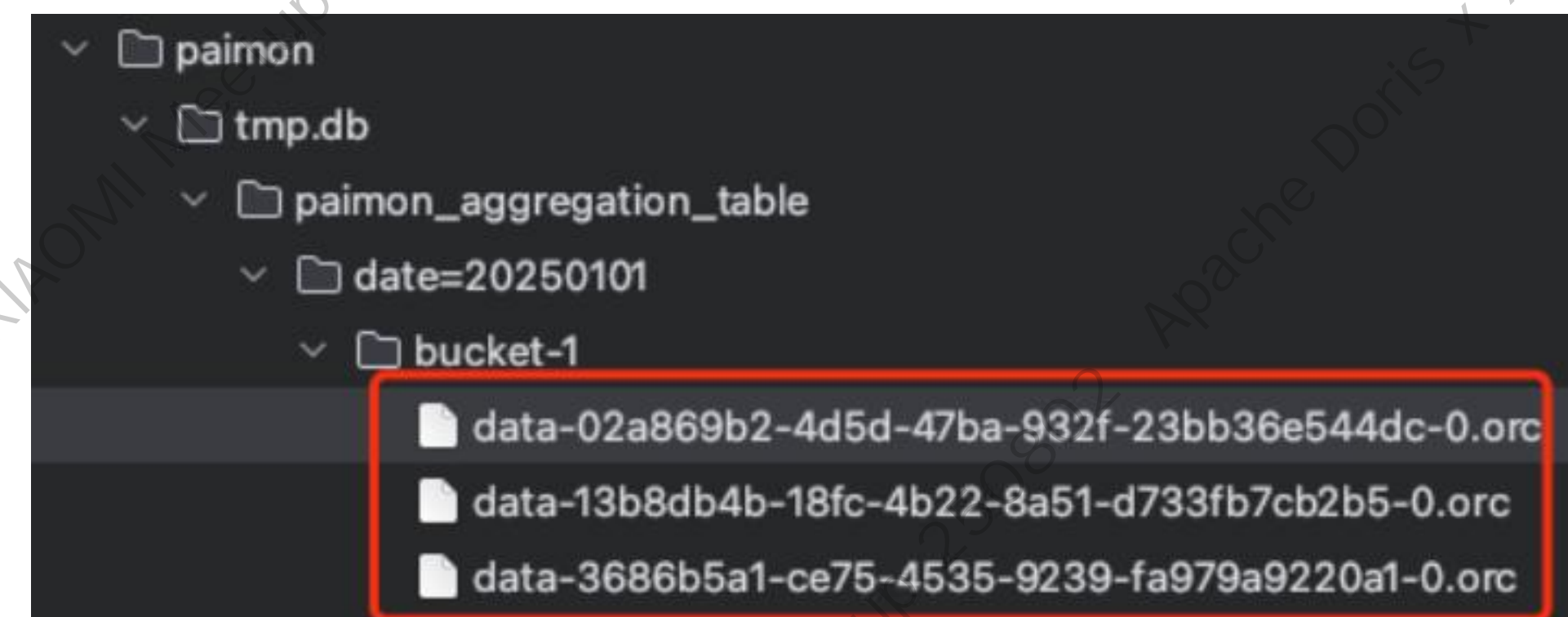


# Paimon merge on Doris 优化

## Paimon 文件结构

```
CREATE TABLE paimon_aggregation_table (  
  date bigint,  
  k bigint,  
  s bigint)  
USING paimon  
PARTITIONED BY (date)  
TBLPROPERTIES (  
  'merge-engine' = 'aggregation',  
  'fields.s.aggregate-function' = 'sum',  
  'primary-key' = 'date,k');
```

```
INSERT INTO paimon_aggregation_table VALUES (20250101, 1, 1);  
INSERT INTO paimon_aggregation_table VALUES (20250101, 1, 2);  
INSERT INTO paimon_aggregation_table VALUES (20250101, 1, 3);
```



# Paimon merge on Doris 优化

## Paimon 文件结构



```
spark.read().orc("data-02a869b2-4d5d-47ba-932f-23bb36e544dc-0.orc").show();  
spark.read().orc("data-13b8db4b-18fc-4b22-8a51-d733fb7cb2b5-0.orc").show();  
spark.read().orc("data-3686b5a1-ce75-4535-9239-fa979a9220a1-0.orc").show();
```

_KEY_k	_SEQUENCE_NUMBER	_VALUE_KIND	date	k	s
1	0	0	20250101	1	1

_KEY_k	_SEQUENCE_NUMBER	_VALUE_KIND	date	k	s
1	0	0	20250101	1	2

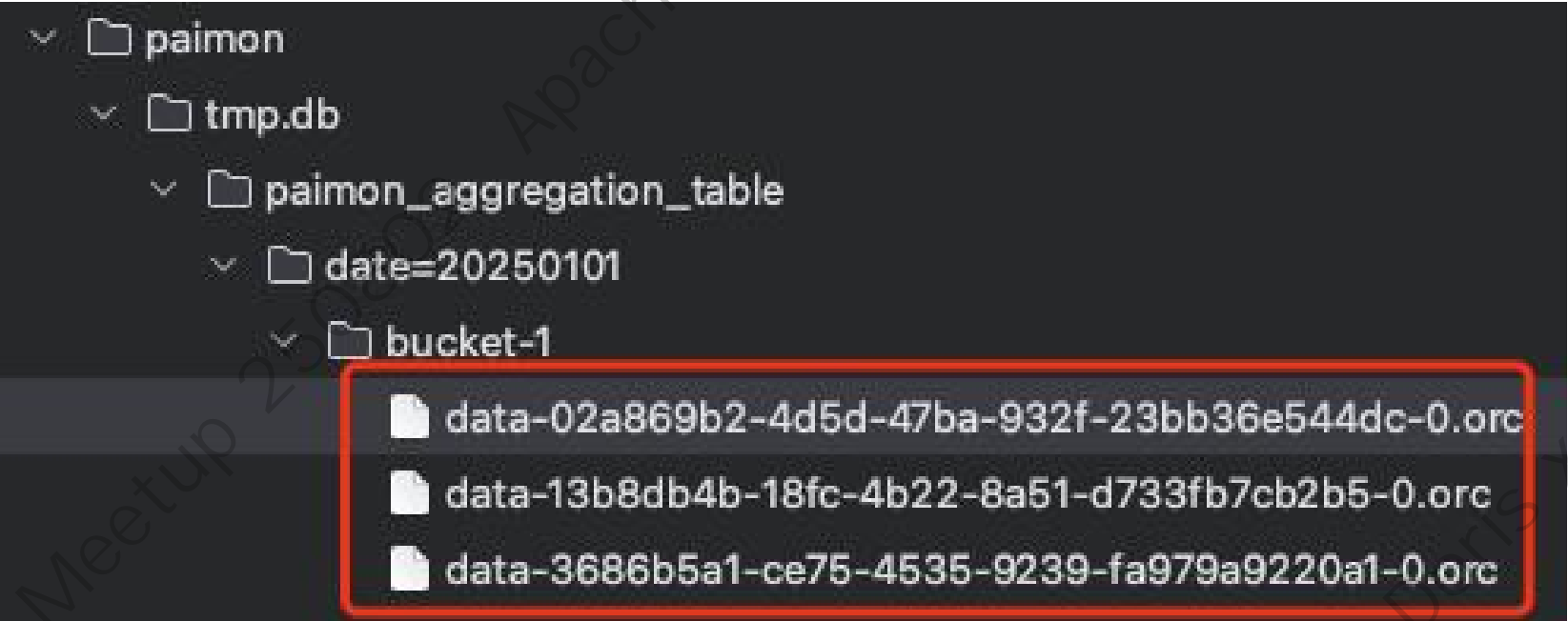
_KEY_k	_SEQUENCE_NUMBER	_VALUE_KIND	date	k	s
1	0	0	20250101	1	3



# Paimon merge on Doris 优化

## Paimon 文件读合并

select \* from paimon\_aggregation\_table



```
xxxxxx> select * from paimon_aggregation_table;
```

date	k	s
20250101	1	6

1 row in set (0.28 sec)

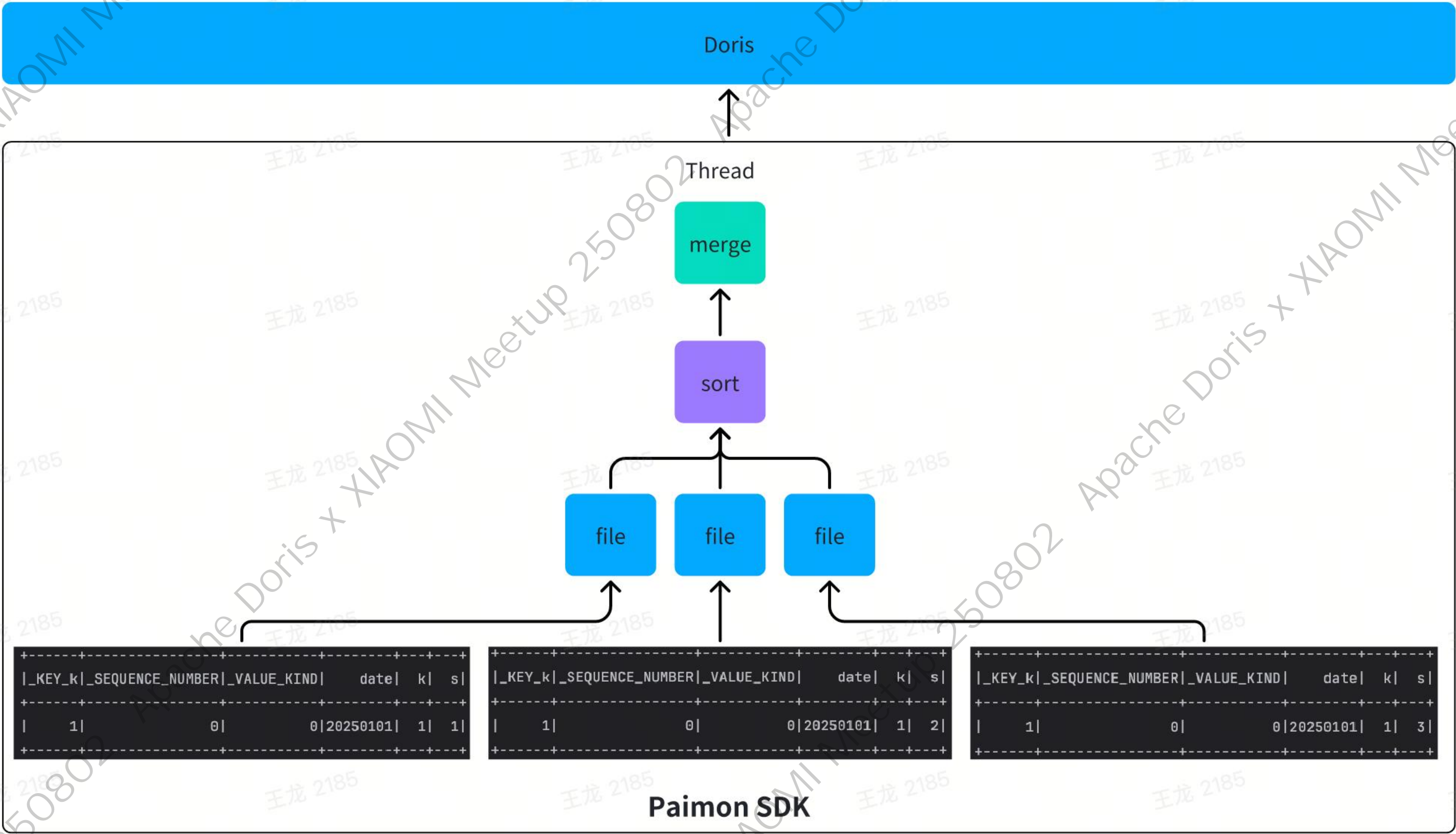


# Paimon merge on Doris 优化

## Merge on Paimon

select date, k, sum(s)  
from paimon\_table  
group by date, k

- 单线程
- 排序
- JAVA

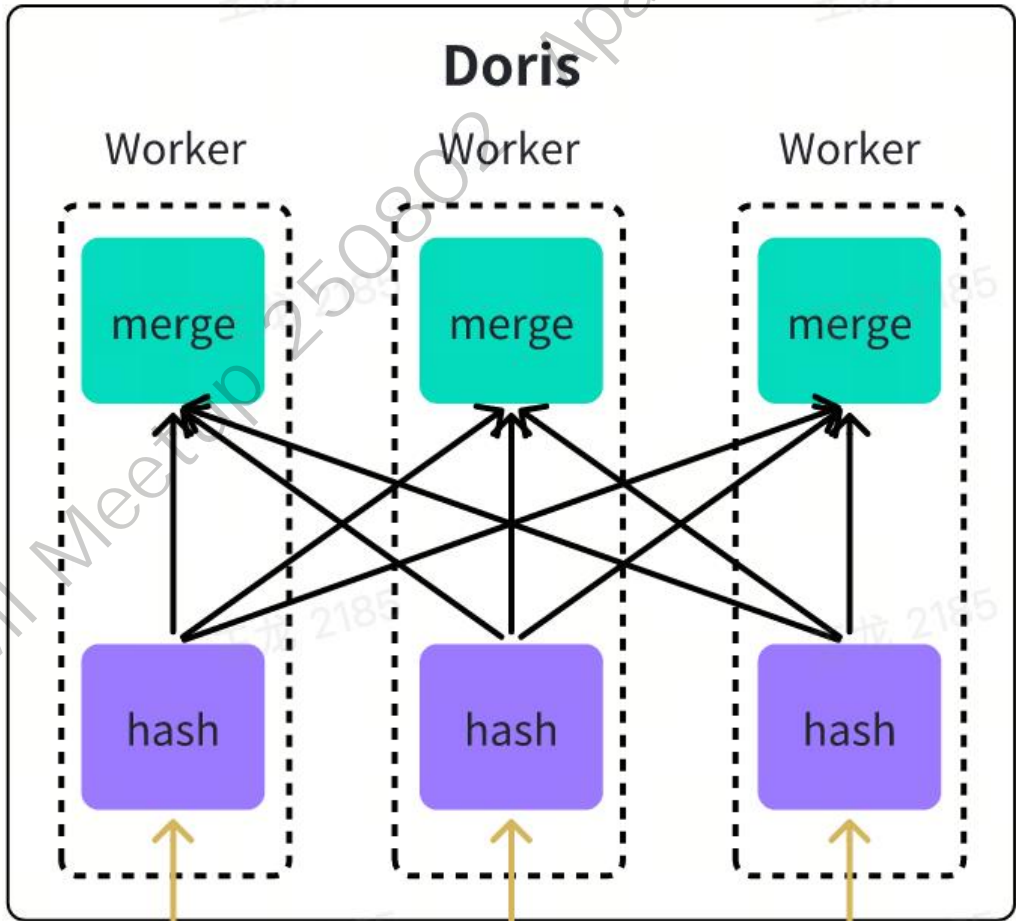


# Paimon merge on Doris 优化

## Merge on Paimon

```
select date, k, sum(s)
from paimon_table
group by date, k
```

- 分布式
- 无排序
- C++



_KEY_k	_SEQUENCE_NUMBER	_VALUE_KIND	date	k	s
1	0	0	20250101	1	1

_KEY_k	_SEQUENCE_NUMBER	_VALUE_KIND	date	k	s
1	0	0	20250101	1	2

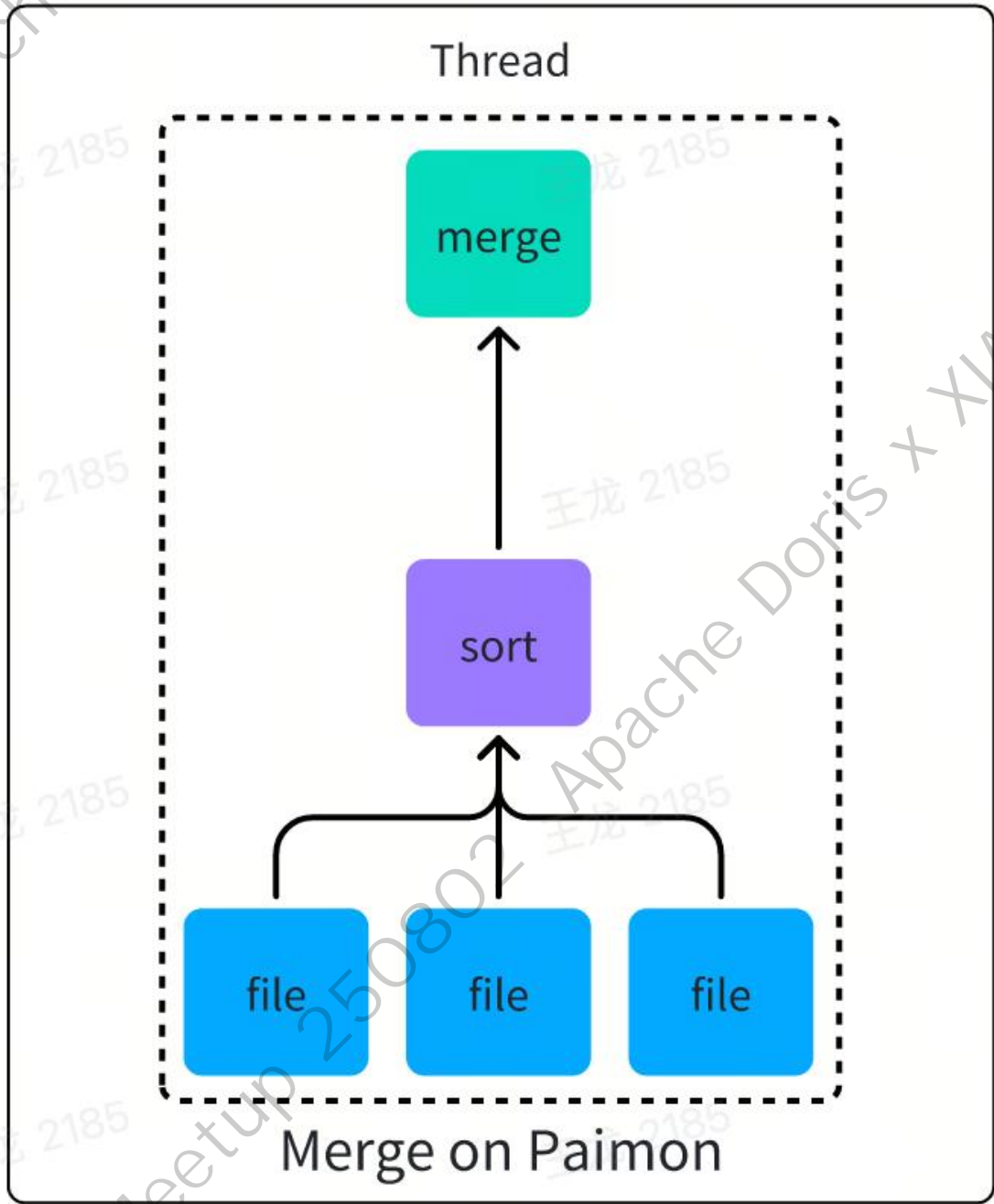
_KEY_k	_SEQUENCE_NUMBER	_VALUE_KIND	date	k	s
1	0	0	20250101	1	3

# Paimon merge on Doris 优化

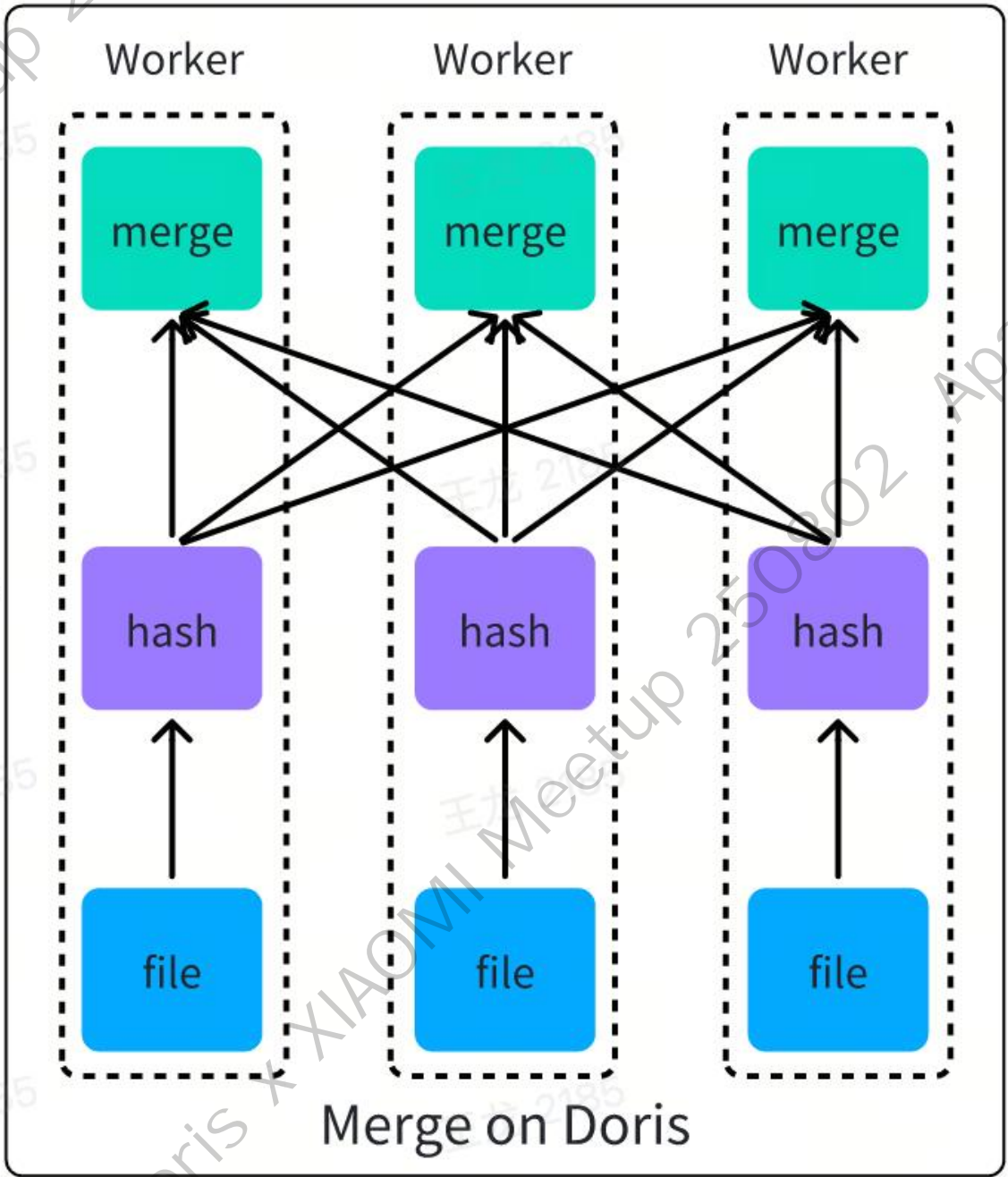
## 优化效果

平均执行时间：40s -> 8s

单线程，需排序，Java执行



分布式，无需排序，C++执行

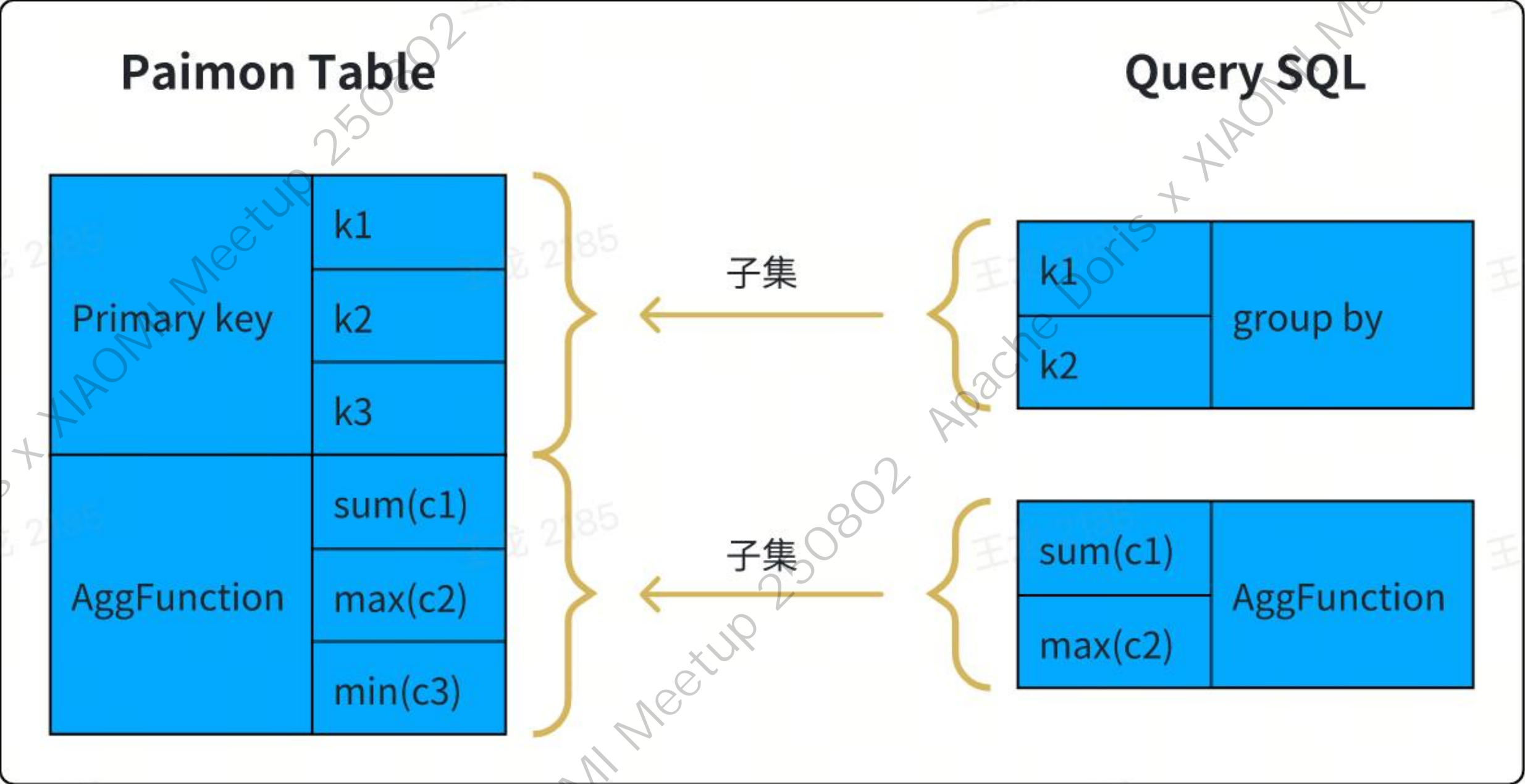




# Paimon merge on Doris 优化

## 应用场景

- 执行计划是聚合，如：Agg -> Project -> filter -> Scan
- Group by 列是 Paimon 主键子集
- SQL 聚合函数是 Paimon 聚合函数子集
- SQL 聚合函数只包括 MIN、MAX、SUM





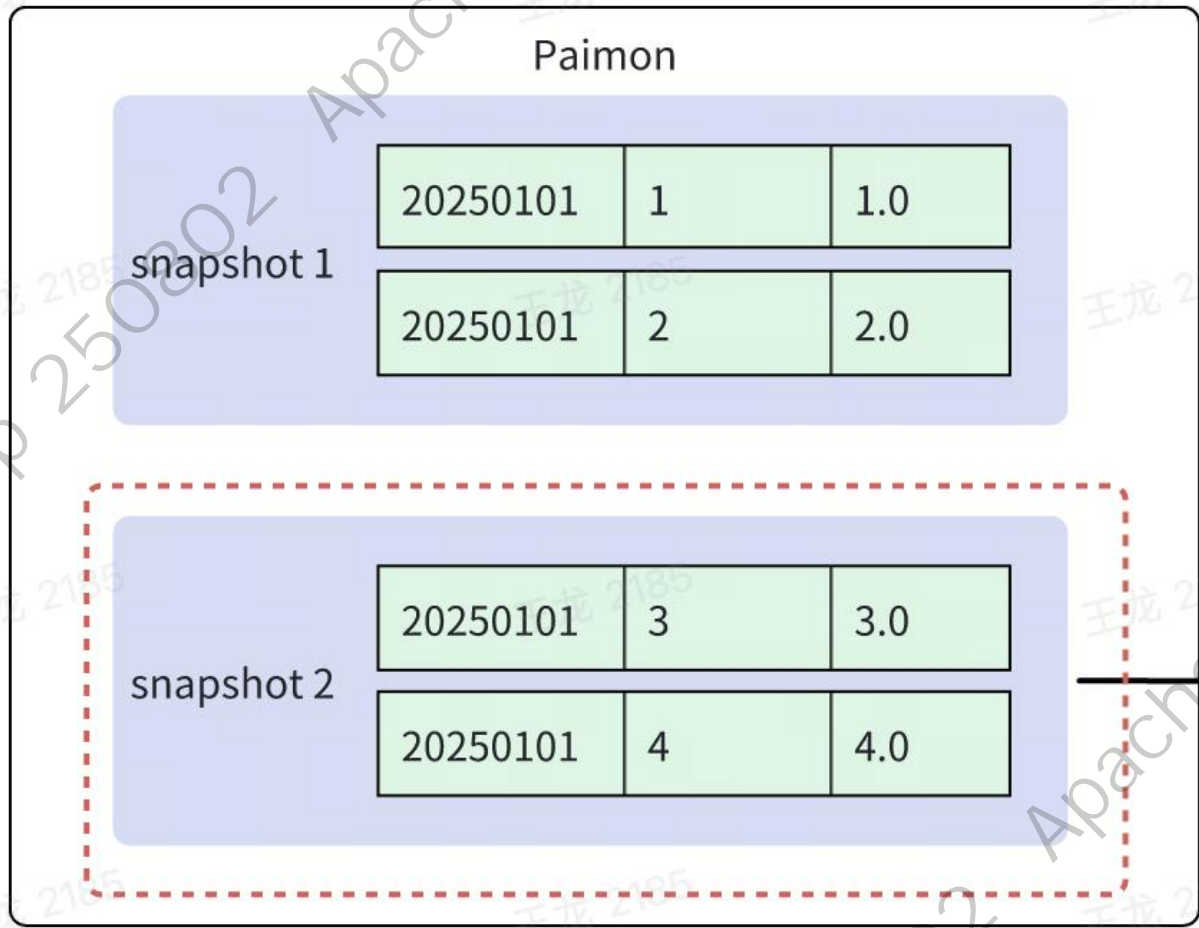
# 基于 Paimon 的增量物化

## 增量更新原理

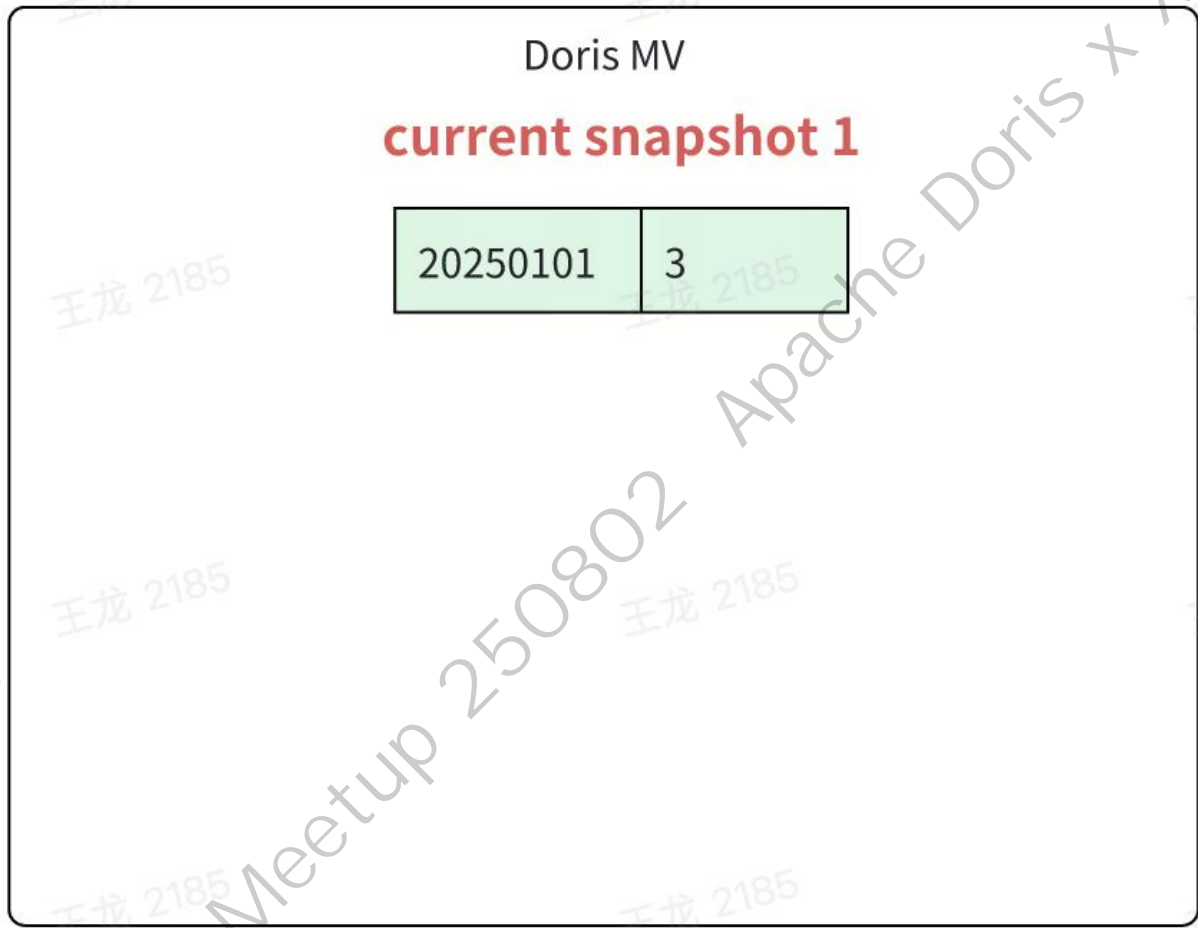
```
CREATE TABLE paimon_table (  
  c1 bigint,  
  c2 bigint,  
  c3 bigint)  
USING paimon;
```

```
create materialized view paimon_table_mv  
BUILD DEFERRED  
REFRESH INCREMENTAL  
ON SCHEDULE EVERY 3 second  
as  
select c1, sum(c2) as c2  
from paimon_table  
where c1 >= 0  
group by c1;
```

```
CREATE TABLE paimon_table_mv (  
  c1 bigint,  
  c2 SUM)  
AGGREGATE KEY(c1)
```



snapshot 1 和 snapshot 2  
之间的增量数据



```
insert into paimon_table_mv  
select c1, sum(c2) as c2  
from paimon_table@incr('startSnapshotId='1', 'endSnapshotId'='2')  
where c1 >=0 group by c1
```

# 基于 Paimon 的增量物化

## 增量更新原理

```
CREATE TABLE paimon_table (  
  c1 bigint,  
  c2 bigint,  
  c3 bigint)  
USING paimon;
```

Paimon			
snapshot 1	20250101	1	1.0
	20250101	2	2.0
snapshot 2	20250101	3	3.0
	20250101	4	4.0

```
create materialized view paimon_table_mv  
BUILD DEFERRED  
REFRESH INCREMENTAL  
ON SCHEDULE EVERY 3 second  
as  
select c1, sum(c2) as c2  
from paimon_table  
where c1 >= 0  
group by c1;
```

```
CREATE TABLE paimon_table_mv (  
  c1 bigint,  
  c2 SUM)  
AGGREGATE KEY(c1)
```

Doris MV		
current snapshot 2		
20250101	10	

insert into paimon\_table\_mv  
select c1, sum(c2) as c2  
from paimon\_table@incr('startSnapshotId='1', 'endSnapshotId'='2')  
where c1 >=0 group by c1

# 基于 Paimon 的增量物化

## 物化视图初始化

```
CREATE TABLE paimon_table (  
  c1 bigint,  
  c2 bigint,  
  c3 bigint)  
USING paimon;
```

Paimon			
snapshot 1	20250101	1	1.0
	20250101	2	2.0
snapshot 2	20250101	3	3.0
	20250101	4	4.0

```
create materialized view paimon_table_mv  
BUILD DEFERRED  
REFRESH INCREMENTAL  
ON SCHEDULE EVERY 3 second  
as  
select c1, sum(c2) as c2  
from paimon_table  
where c1 >= 0  
group by c1;
```

```
CREATE TABLE paimon_table_mv (  
  c1 bigint,  
  c2 SUM)  
AGGREGATE KEY(c1)
```

Doris MV		
current snapshot 2		
20250101	10	

insert overwrite paimon\_table\_mv  
select c1, sum(c2) as c2  
from paimon\_table /\*+ options('scan.snapshot-id'='xx') \*/  
where c1 >=0 group by c1



# 基于 Paimon 的增量物化

## 物化表模型设计

- **预过滤 SQL**  
create materialized view as  
Select \* from t where a = 'x'
- **预聚合 SQL**  
create materialized view as  
select k, sum(s)  
from t group by k

Paimon 表类型	物化 SQL 模型	物化表模型
Duplicate	预过滤	Duplicate
	预聚合	Aggregate
Aggregate	预过滤	Aggregate
	预聚合	Aggregate
Unique	预过滤	Unique
	预聚合	不支持



# 基于 Paimon 的增量物化

## 适用场景和局限性

### 适用场景

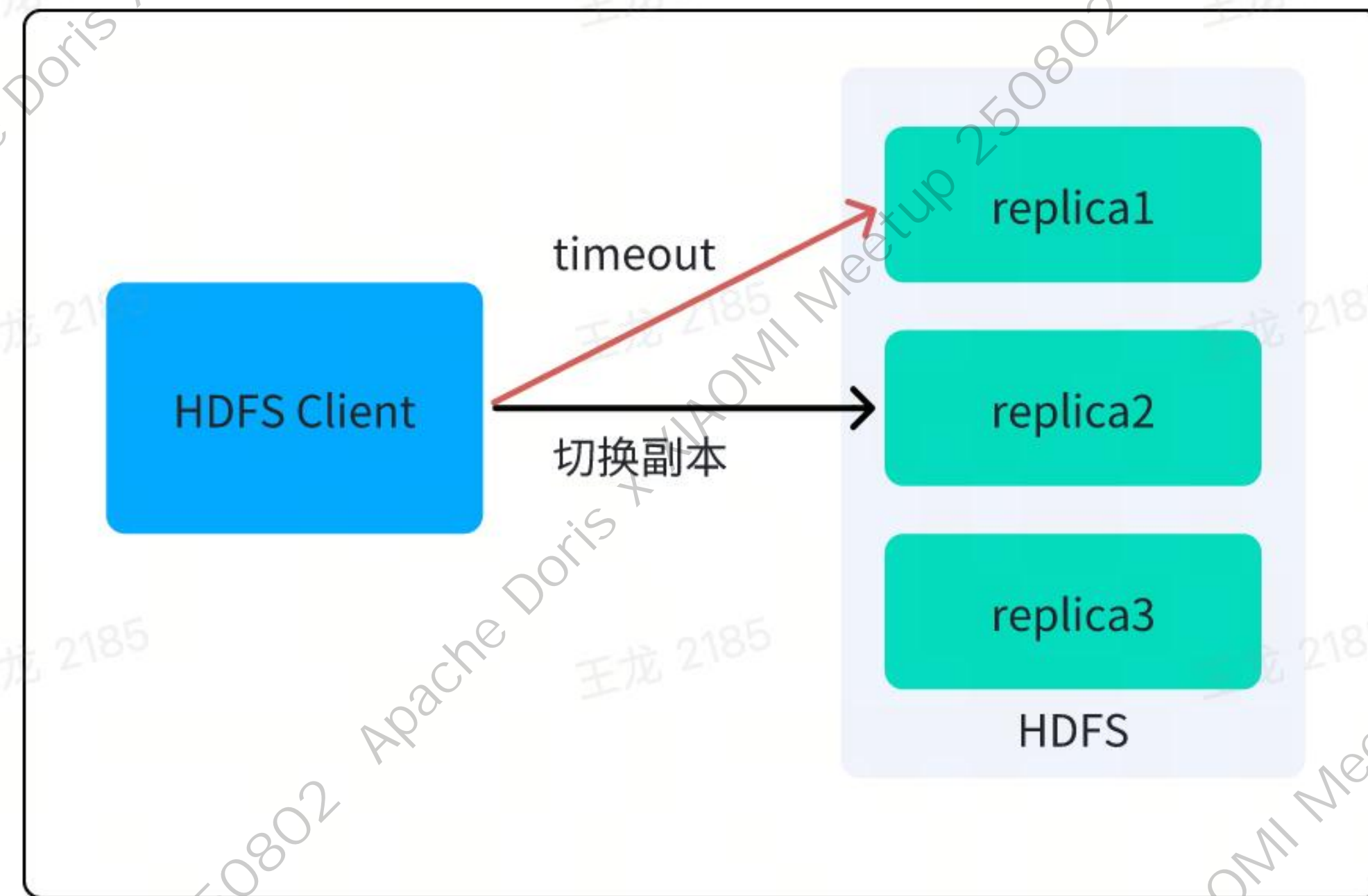
- 预聚合加速
- 预过滤加速
- 前缀索引加速

### 限制

- 只支持单表 SELECT 语句，支持 WHERE、GROUP BY 等子句，但不支持 JOIN、ORDER BY、HAVING、LIMIT、LATERAL VIEW、窗口函数
- 不支持源表数据删除、OVERWRITE
- 聚合函数只支持 sum、min、max

# 基于重试机制的 HDFS 长尾 Task 优化

## HDFS 副本切换原理



# 基于重试机制的 HDFS 长尾 Task 优化

## 副本超时时间设置

副本读取超时配置：`dfs.client.socket-timeout=60000`

- 超时时间太短：导致读取失败

```
org.apache.hadoop.hdfs.BlockMissingException: Could not obtain block: BP-119742716-10.38.161.51-1496386339466:blk_6606440313_5543075691 No live nodes contain current block Block locations
```

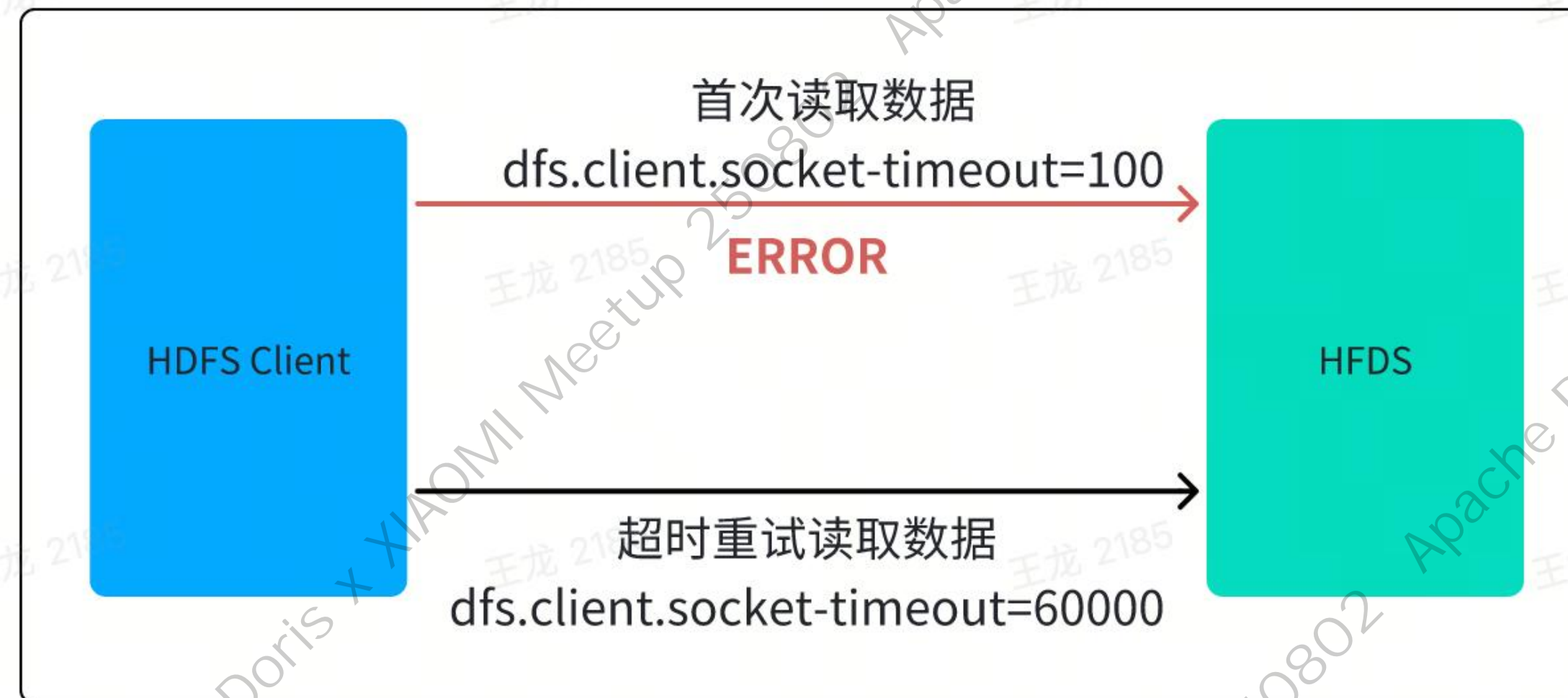
- 超时时间太长：导致读取长尾



# 基于重试机制的 HDFS 长尾 Task 优化

## 重试优化

- 快速试错，解决长尾问题
- 异常重试，解决超时问题



## 最终优化效果

业务查询性能提升 5 倍，平均查询时间变化：60s -> 10s

# 目录

1.Doris 在小米的落地过程

2.Doris on Paimon 优化

3.未来规划



## 未来规划

- **Doris 替换 Presto**
- **Doris 湖上增量物化能力增强**
- **Doris 湖仓容器化**
- **Doris 基于 compute group 的资源隔离**

# Thanks !



# 活动已结束

关注 SelectDB 公众号解锁更多技术资讯!



解锁更多技术资讯



加入社区