Apache Doris 实时查询分析

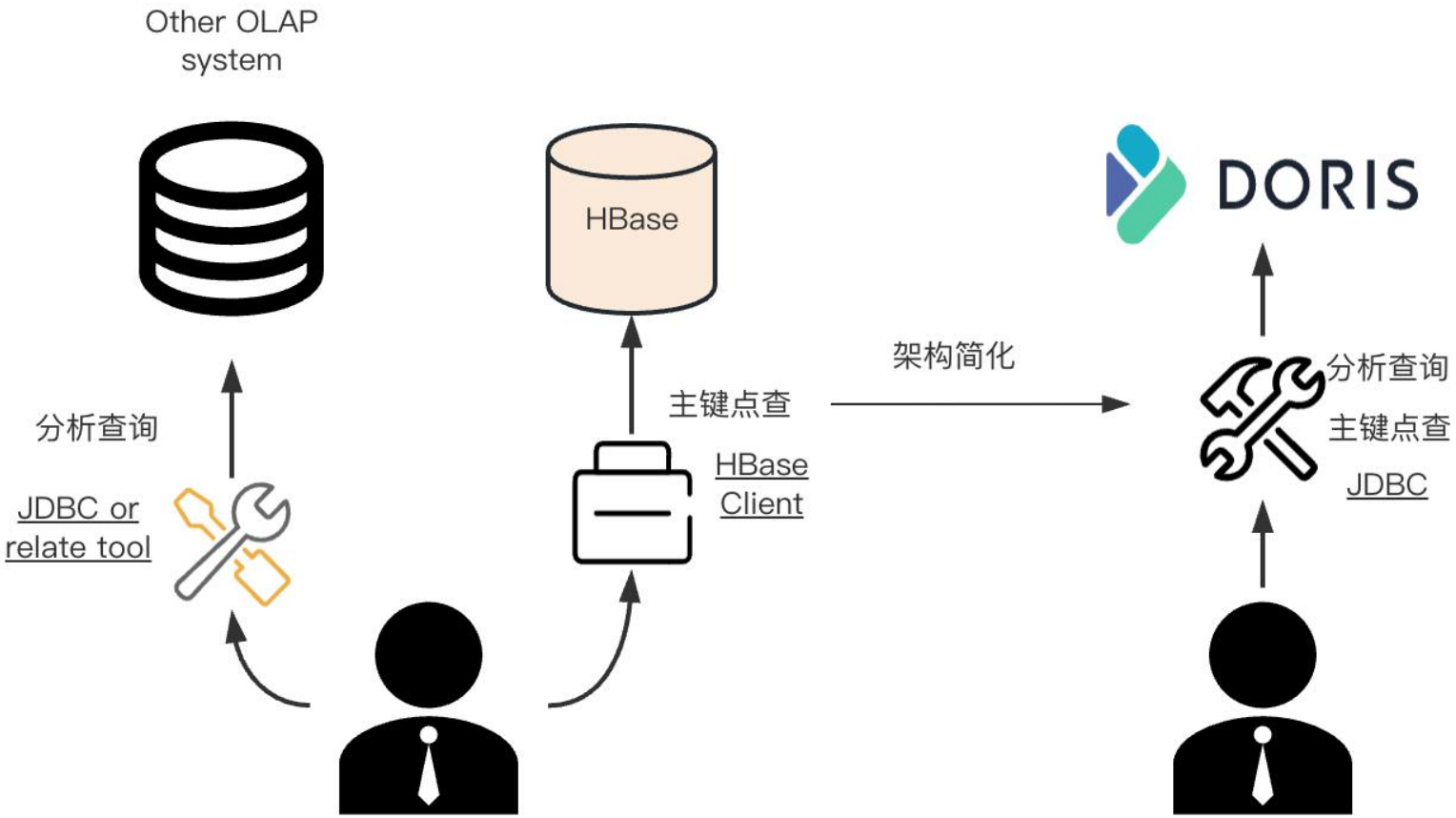
李航宇

SelectDB 数据库内核研发、Apache Doris Commiter

DataFunSummit # 2023

SELECTDB DataFun.

架构演变


Other OLAP
system
HBase
DORIS
分析查询
JDBC or
relate tool
主键点查
HBase
Client
架构简化
分析查询
主键点查
JDBC

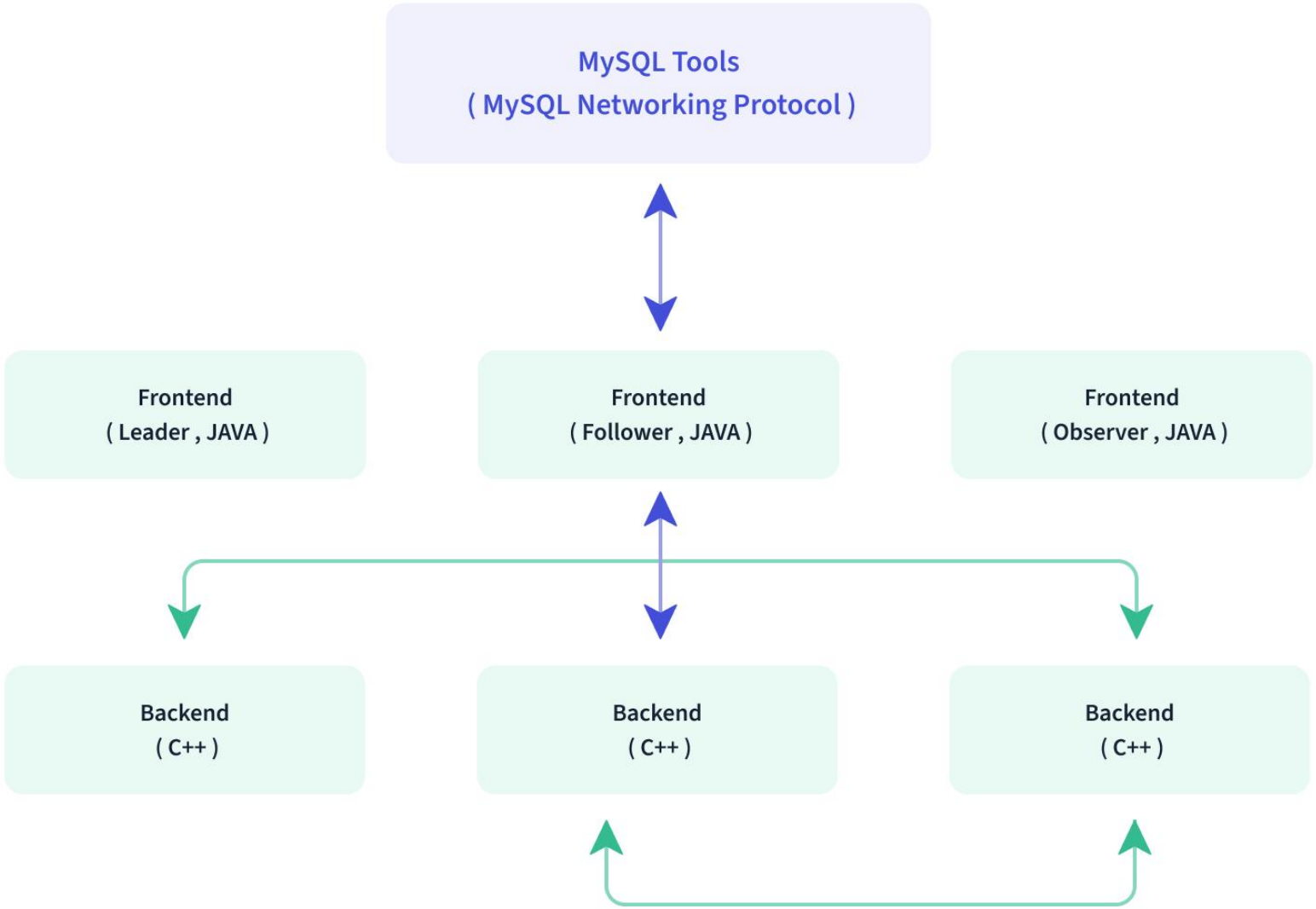
目录 CONTENT

Apache Doris 存储架构介绍

DataFunSummit # 2023

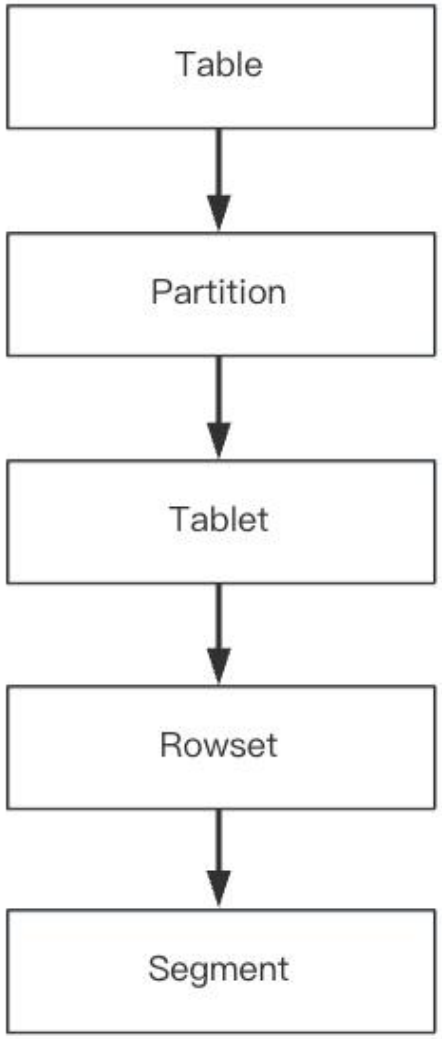
SELECTDB DataFun.

架构图

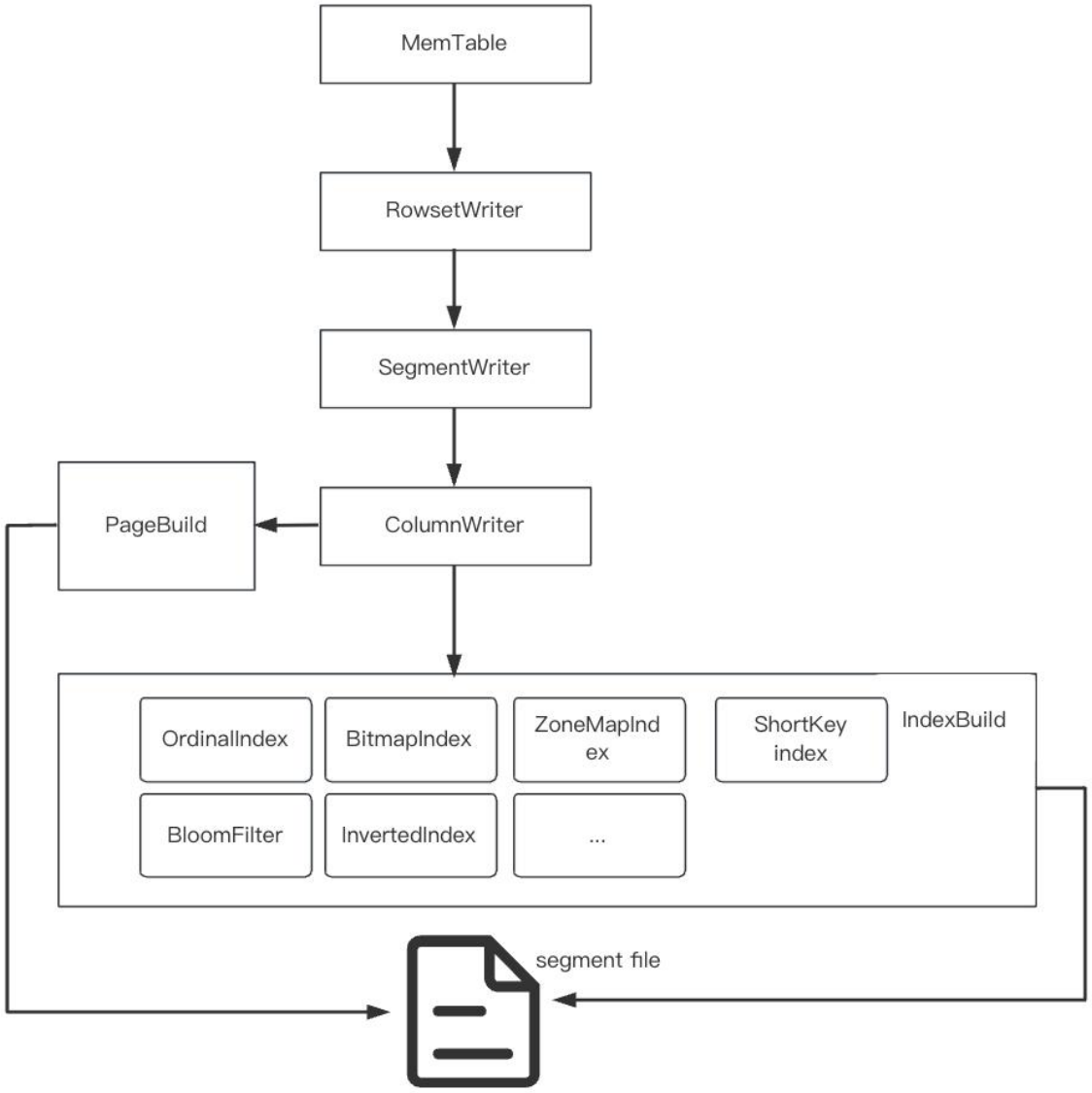

MySQL Tools
(MySQL Networking Protocol)
Frontend
(Leader , JAVA)
Frontend
(Follower , JAVA)
Frontend
(Observer , JAVA)
Backend
(C++)
Backend
(C++)
Backend
(C++)

层级关系

```
CREATE TABLE IF NOT EXISTS demo_table (
k1 date NOT NULL,
k2 varchar(20)
NOT NULL,
k3 int sum NOT NULL )

AGGREGATE KEY(k1,k2)
PARTITION BY RANGE(k1) ( )
DISTRIBUTED BY HASH(k1) BUCKETS 3
PROPERTIES (
  "dynamic_partition.enable"="true",
  "dynamic_partition.end"="3",
  "dynamic_partition.buckets"="10",
  "dynamic_partition.start"="-3",
  "dynamic_partition.prefix"="p",
...)
```

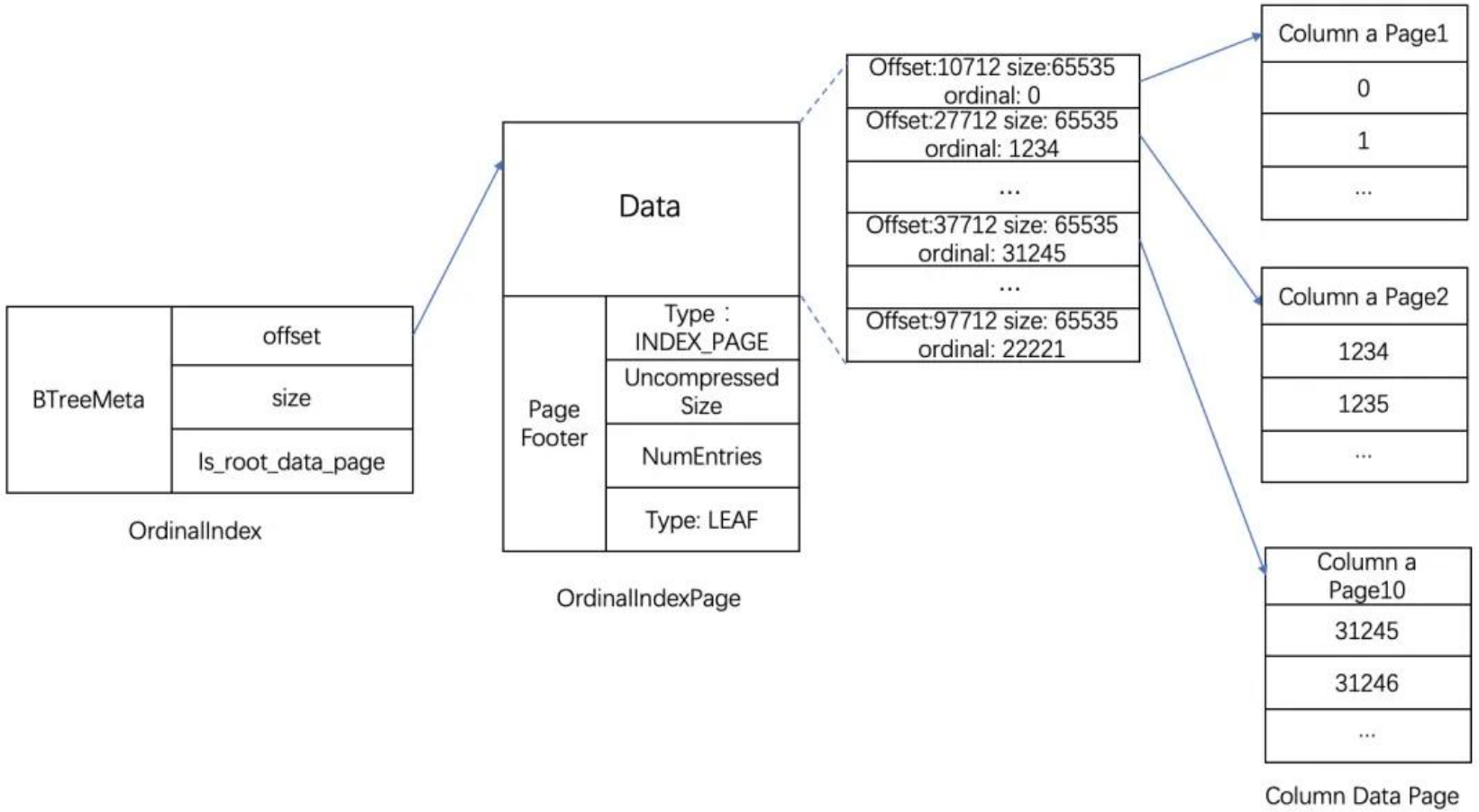


存储 and 索引


MemTable
RowsetWriter
SegmentWriter
PageBuild
ColumnWriter
OrdinalIndex
BitmapIndex
ZoneMapInd
ex
ShortKey
index
IndexBuild
BloomFilter
InvertedIndex
...
segment file

列式存储

Data Region	Column a	Page 0: Data Page
		...
		Page N: Data Page
	Column b	Page 0: Data Page
		...
		Page N: Data Page
	...	
	Column n	Page 0: Data Page
		...
		Page N: Data Page
Index Region	Column a Index	Bloom Filter Pages
		Bitmap Pages
		Ordinal Index Pages
	...	
	Short Key Index	
Footer	FileFooterPB	
	PB Checksum	PB Length
	MAGIC CODE	



Apache Doris 高并发核心技术

DataFunSummit # 2023

SELECTDB DataFun.

分区分桶

- Apache Doris 采用两级分区，第一级是 Partition，通常可以将时间作为分区键。第二级为 Bucket，通过 Hash 将数据打散至各个节点中，以此提升读取并行度并进一步提高读取吞吐。通过合理地划分区分桶，可以提高查询性能，以下列查询语句为例：

- select * from user_table where id = 5122 and create_time = '2022-01-01'

- 用户以 create_time 作为分区键、ID 作为分桶键，并设置了 10 个 Bucket， 经过分区分桶裁剪后可快速过滤非必要的分区数据，最终只需读取极少数据，比如 1 个分区的 1 个 Bucket 即可快速定位到查询结果，最大限度减少了数据的扫描量、降低了单个查询的延时。

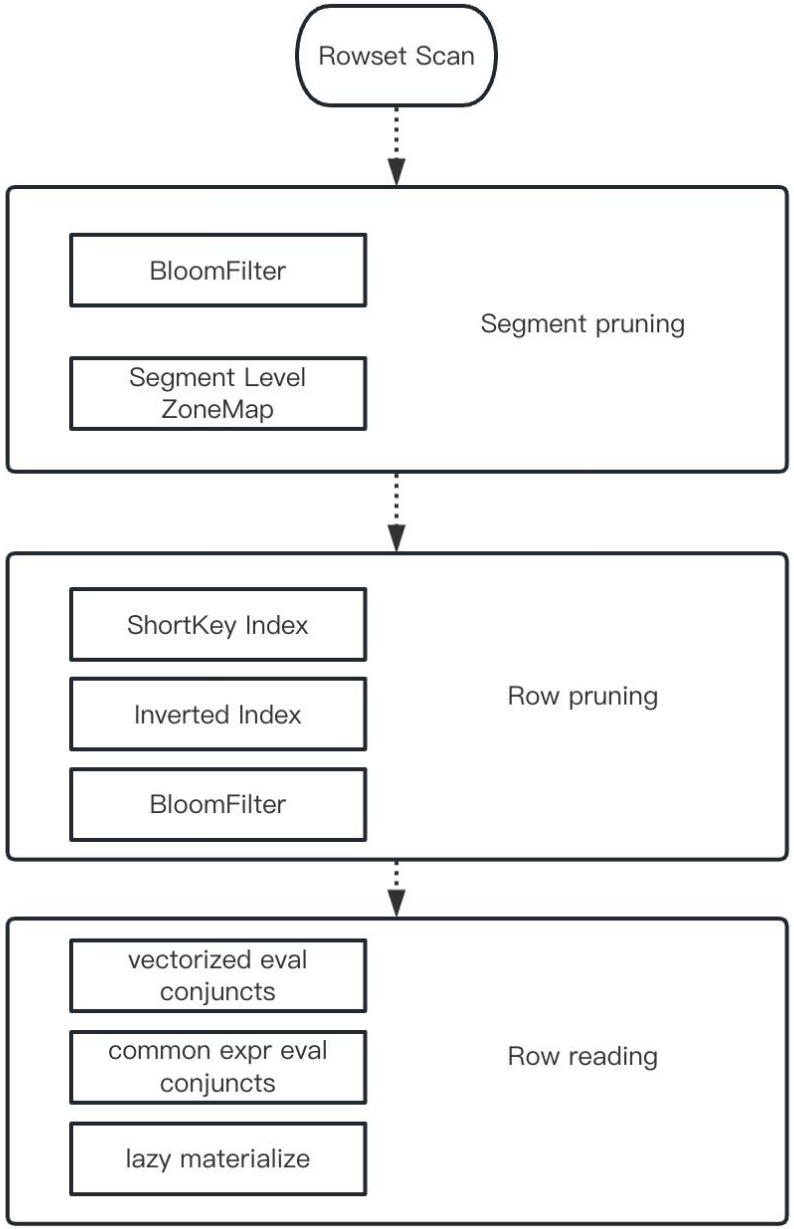
索引过滤机制

SELECT * from table where key = 1 and index_column = 'abc' and xyx = 10

- 按条件裁剪segment

- 前缀稀疏索引加速前缀匹配
- 倒排索引加速字符串匹配
- bloom过滤等值查询

- 表达式过滤(向量化)
- 延迟物化



案例

```
CREATE TABLE sales_order
(
    orderid     BIGINT,
    status      TINYINT,
    username    VARCHAR(32),
    amount      BIGINT DEFAULT '0',
    INDEX idx_user_name(username) USING BITMAP
)
UNIQUE KEY(orderid)
DISTRIBUTED BY HASH(orderid) BUCKETS 10
PROPERTIES (
"bloom_filter_columns"="amount"
)
```

前缀索引　　bitmap索引　　bloom filter

```
select * from sales_order where orderid = 81991 and username = 'lucy' and amount = 10;
```

物化视图

1. 空间换时间

2. 预聚合

3. 前缀匹配

物化视图

/ 对于聚合操作， 直接读物化视图预聚合的列

create materialized view store_amt as select store_id, sum(sale_amt) from sales_records group by store_id;

SELECT store_id, sum(sale_amt) FROM sales_records GROUP BY store_id;

// 对于查询， k3满足物化视图前缀列条件， 走物化视图加速查询

CREATE MATERIALIZED VIEW mv_1 as SELECT k3, k2, k1 FROM tableA ORDER BY k3;

select k1, k2, k3 from table A where k3=3;

其他

- **包括多种 Cache**
  - PageCache
  - SQL Cache
  - PartitionCache
- **RuntimeFilter**
  - 根据小表数据过滤大表
- **TopN optimization**
  - limit下推
  - key filter
  - 延迟物化
- **....**

Doris 高并发点查优化

DataFunSummit # 2023

SELECTDB DataFun.

优化背景以及思路

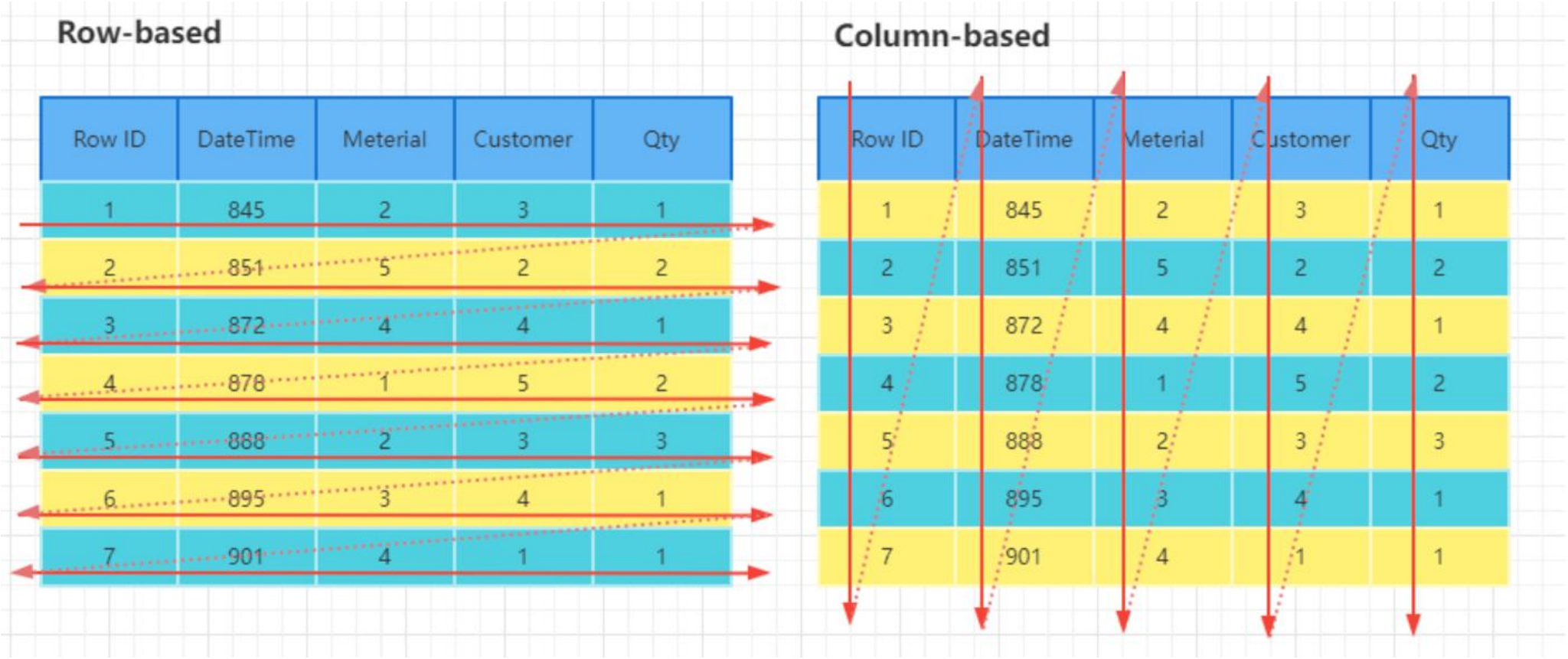
Apache Doris 已经在单节点上实现了支持数千次每秒的查询（Queries Per Second, QPS）。然而，在对超高并发性能要求的数据服务场景中（例如，数万次每秒的 QPS），仍然存在一些瓶颈。以下是所面临的挑战：

1. 列式存储引擎对于读取行级数据不友好，在宽表模型中会导致随机读 I/O 大幅增加。
2. OLAP 数据库的执行引擎和查询优化器对于某些简单查询（例如点查询）过于繁重，需要进行短路径规划来处理此类查询。
3. SQL 请求访问、查询计划解析和生成由 FE 模块使用 Java 处理，当在高并发场景中解析和生成大量查询执行计划时，会导致高 CPU 开销。
4. 利用高效的 PrimaryIndex 以及 bloomfilter 加速点查

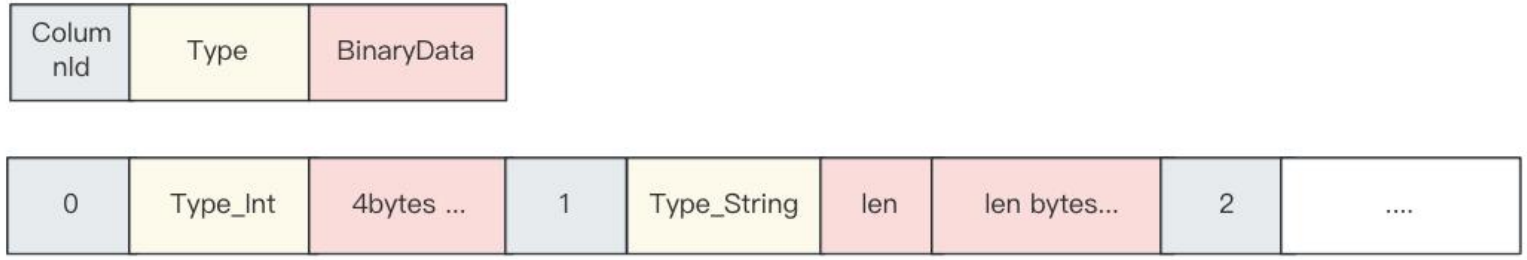
为了解决这些挑战，Apache Doris 从以下三个设计要点实施了一系列优化：

1. 减少 SQL 内存 I/O 开销
2. 提高点查询执行效率
3. 降低 SQL 解析开销

行存 vs 列存

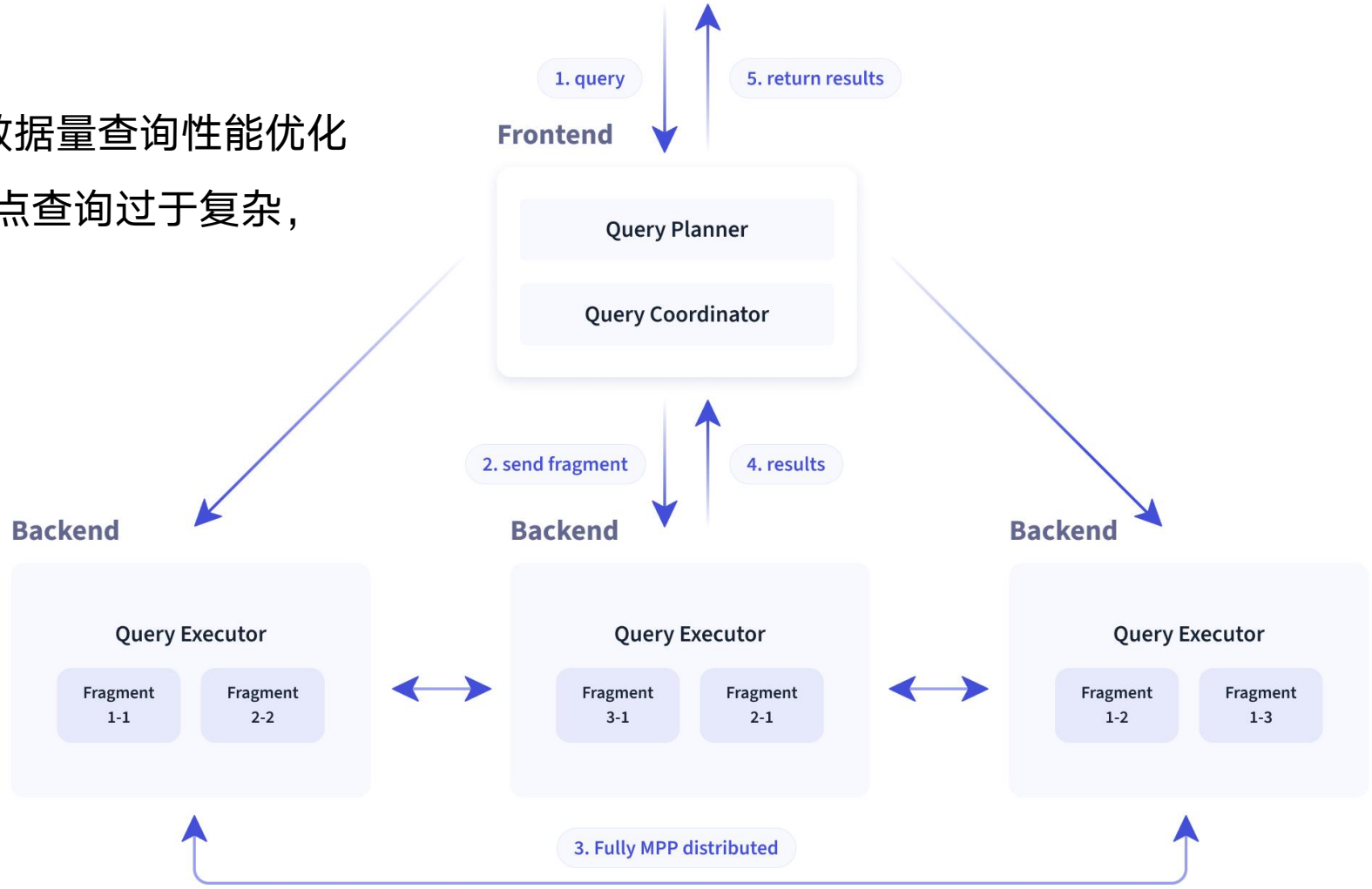


引入新的编码方式， 将一行紧凑的编码到一块空间，较少读取整行数据的IO放大



短路径规划

MPP 查询为跑大数据量查询性能优化极致、但是对于单点查询过于复杂，路径冗余

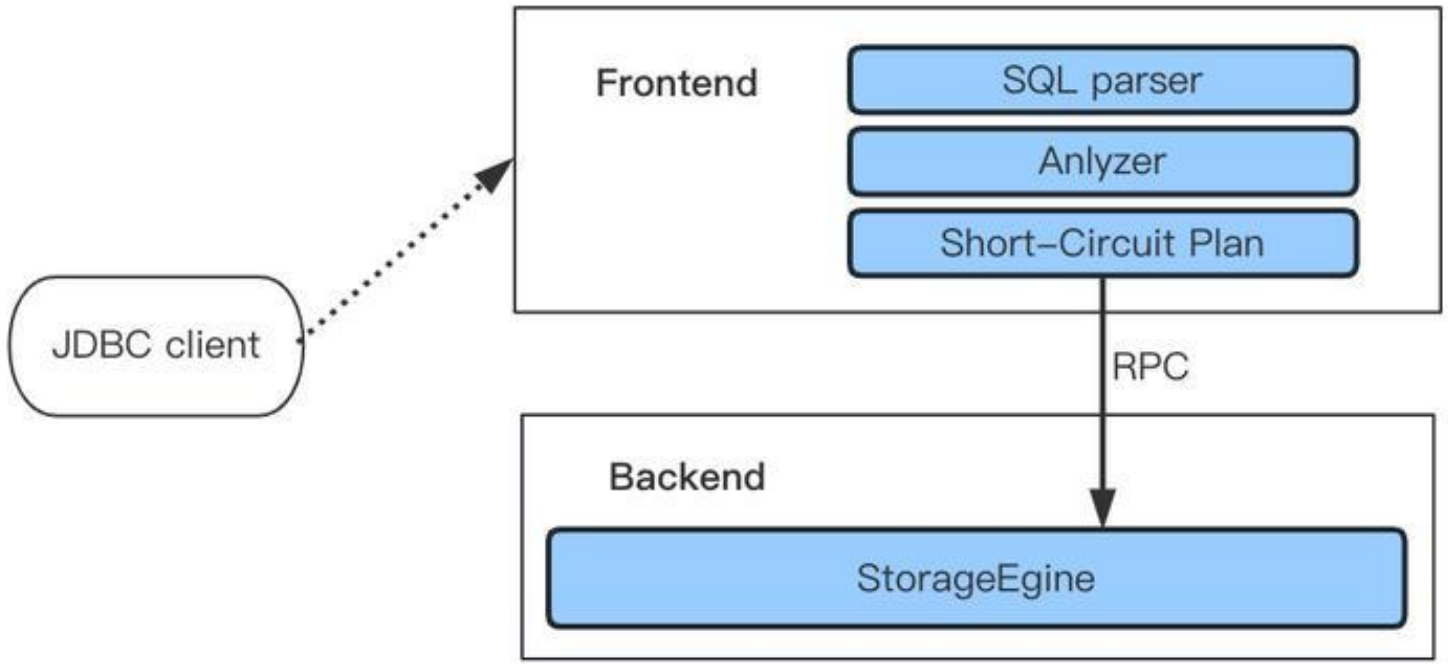


短路径规划

Short-circuit query 直接访问存储引擎

select * from table id = 10

id 是主键

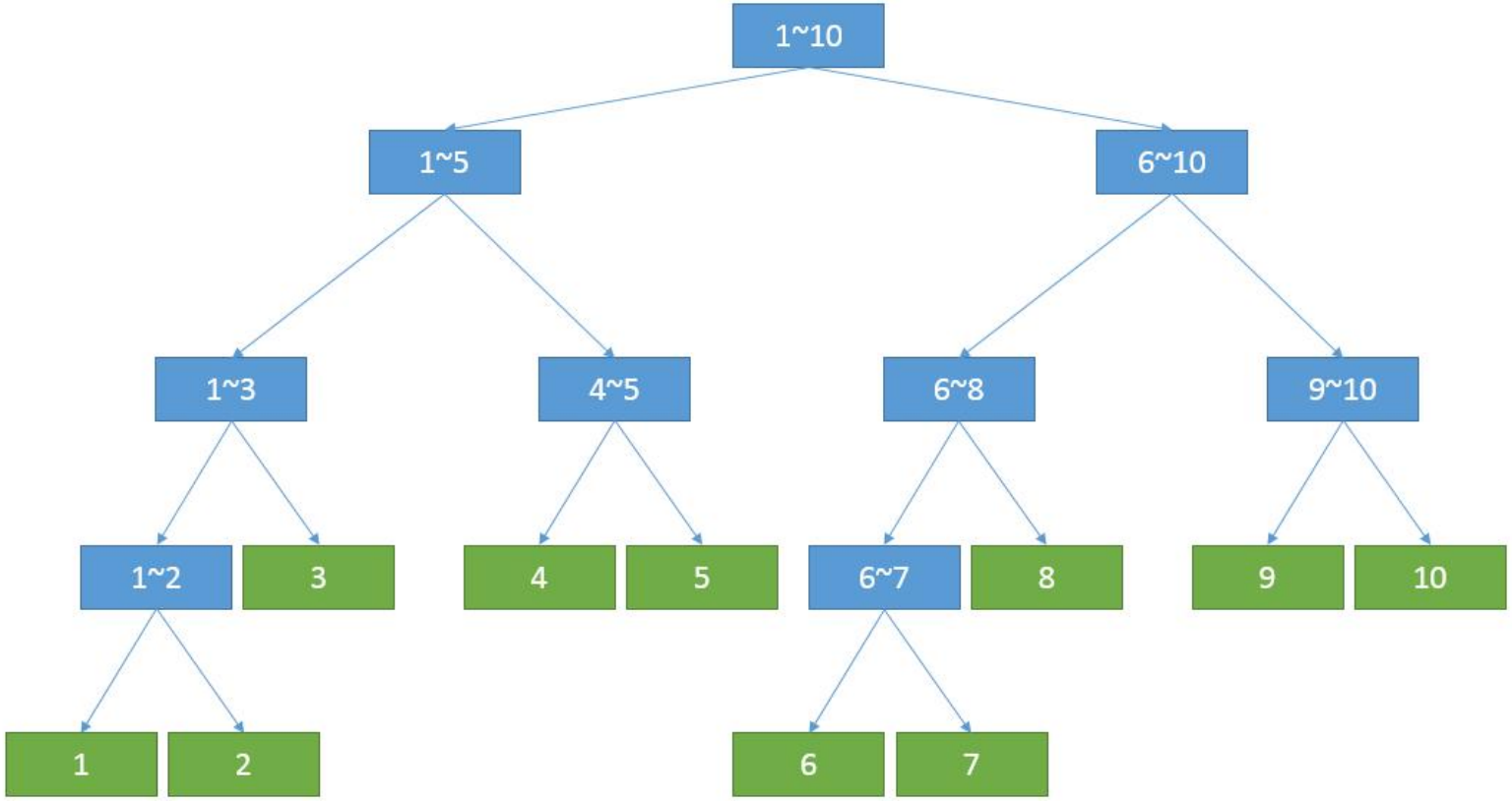


利用 RowsetTree 快速定位 segment

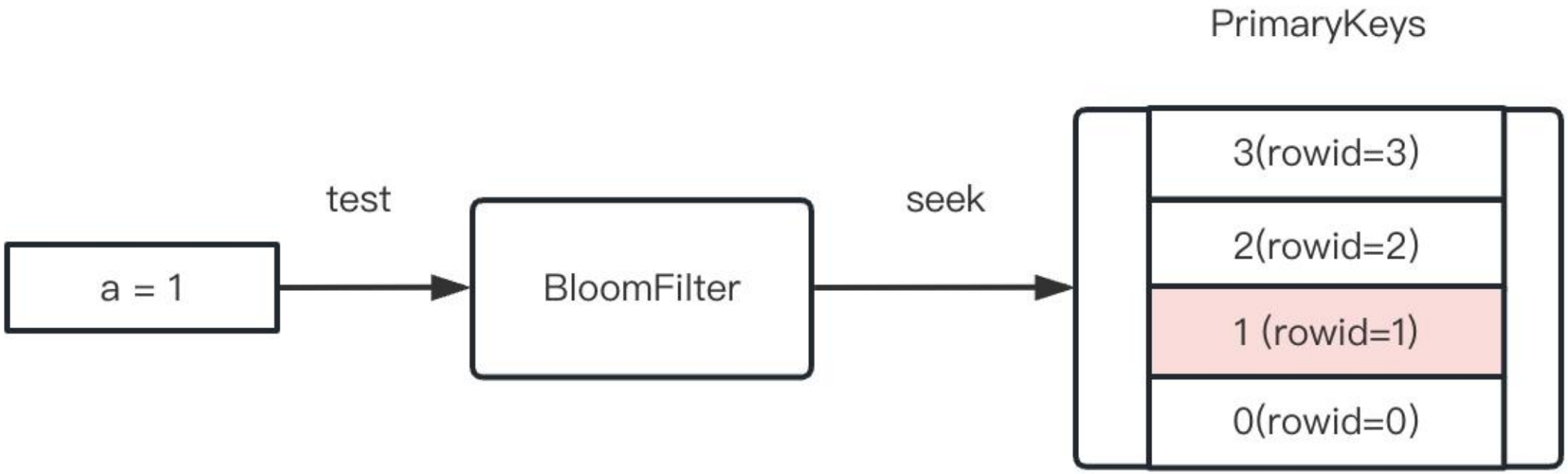
线段树二分查找快速定位 rowset & segment

区间为主键编码后的 [lower_bound, upper_bound),

叶子节点是对应 rowset/segment-id



主键索引 & bloom filter



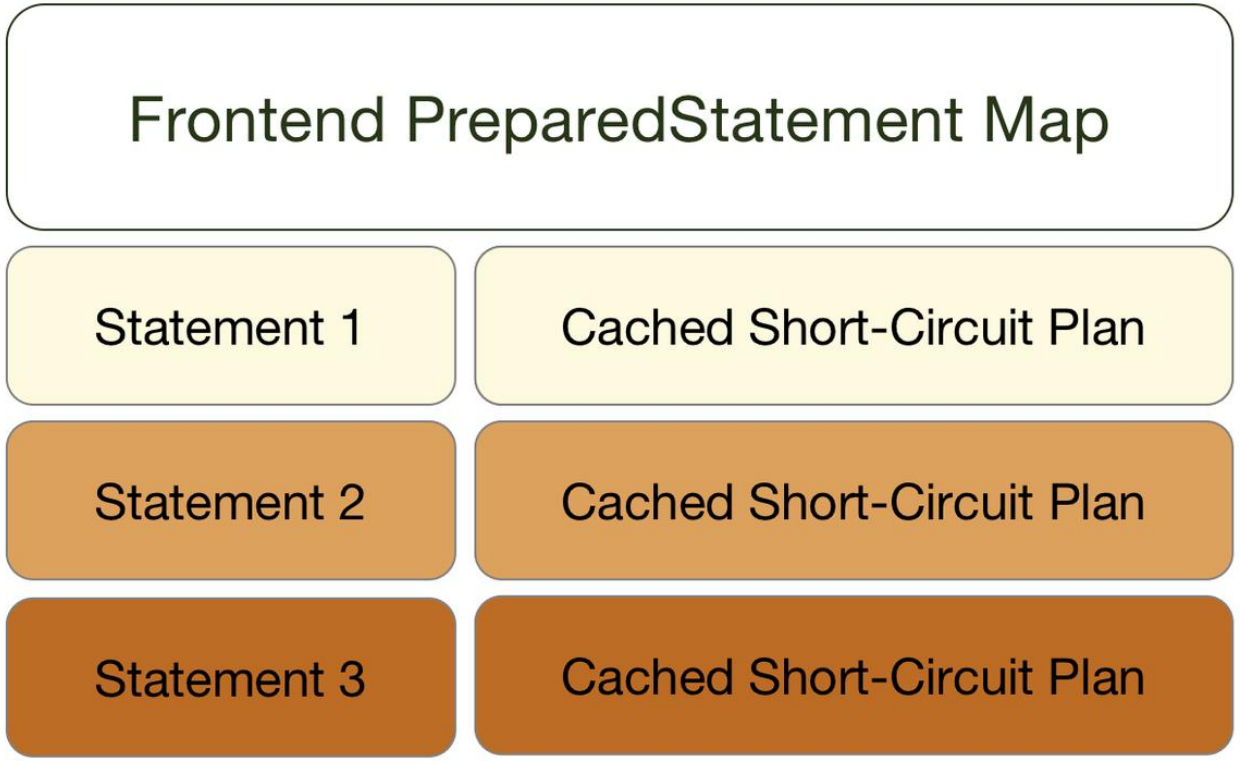
FE 性能瓶颈

一条 SQL 语句的执行需要经过三个步骤：首先通过 SQL Parser 解析语句，生成抽象语法树 (AST)，随后通过 Query Optimizer 生成可执行计划（Plan），最终通过执行该计划得到计算结果

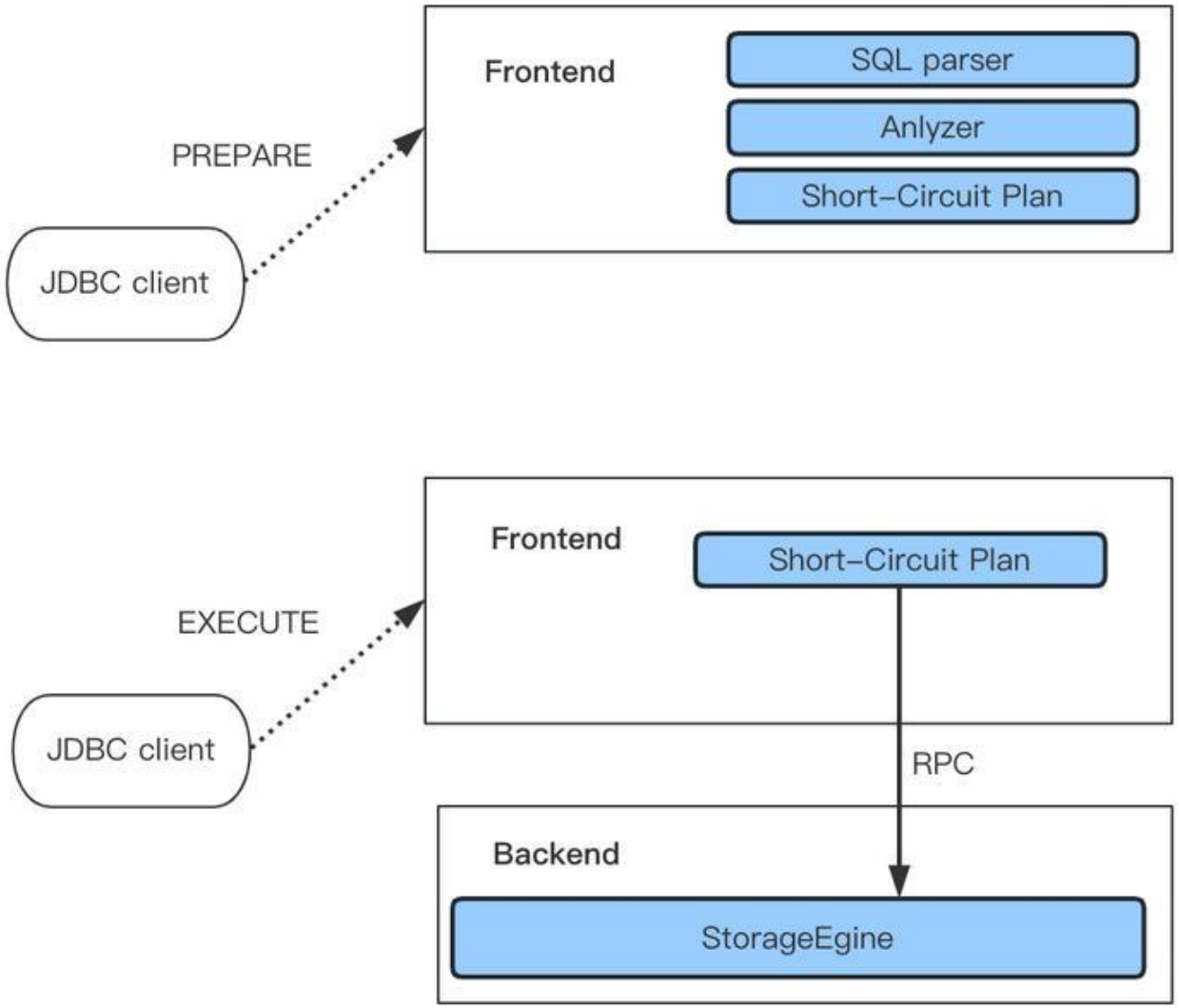
由于 FE 是 Java 编写，性能难以优化到极致，考虑到类似的查询 pattern 足够简单，很多数据结构是可以复用的。这里我们用 Mysql 的 ServerSide PreparedStatement(SSP) 来解决该问题。

Prepared Statement

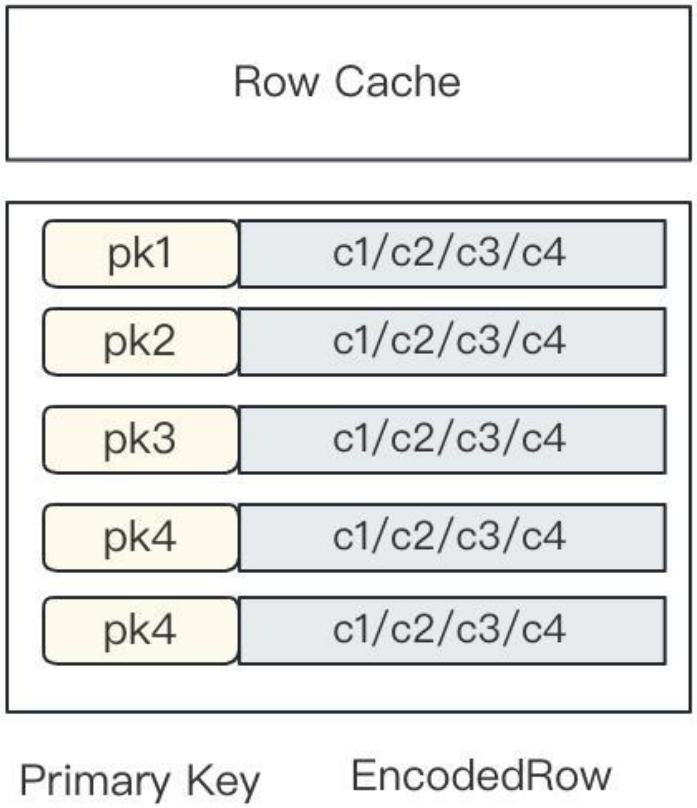
Prepared Statement 的工作原理是通过在 Session 内存 HashMap 中缓存预先计算好的 SQL 和表达式，在后续查询时直接复用缓存对象即可



Prepared Statement



Cache

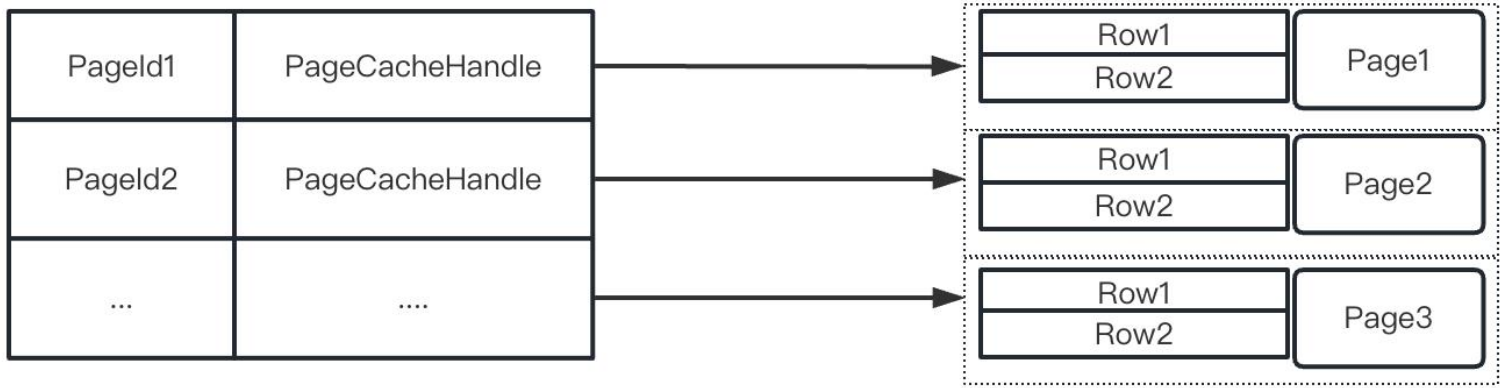


多级 cache：

Page 级别 Cache， 类似 Rocksdb Block Cache，极大减少磁盘 IO

Row 级别 Cache，缓存原始行存，减少 codec

都是 LRU 缓存



Cache

- Meta Cache
- 缓存 Schema、Expression、thrift、proto 等结构

YCSB Benchmark

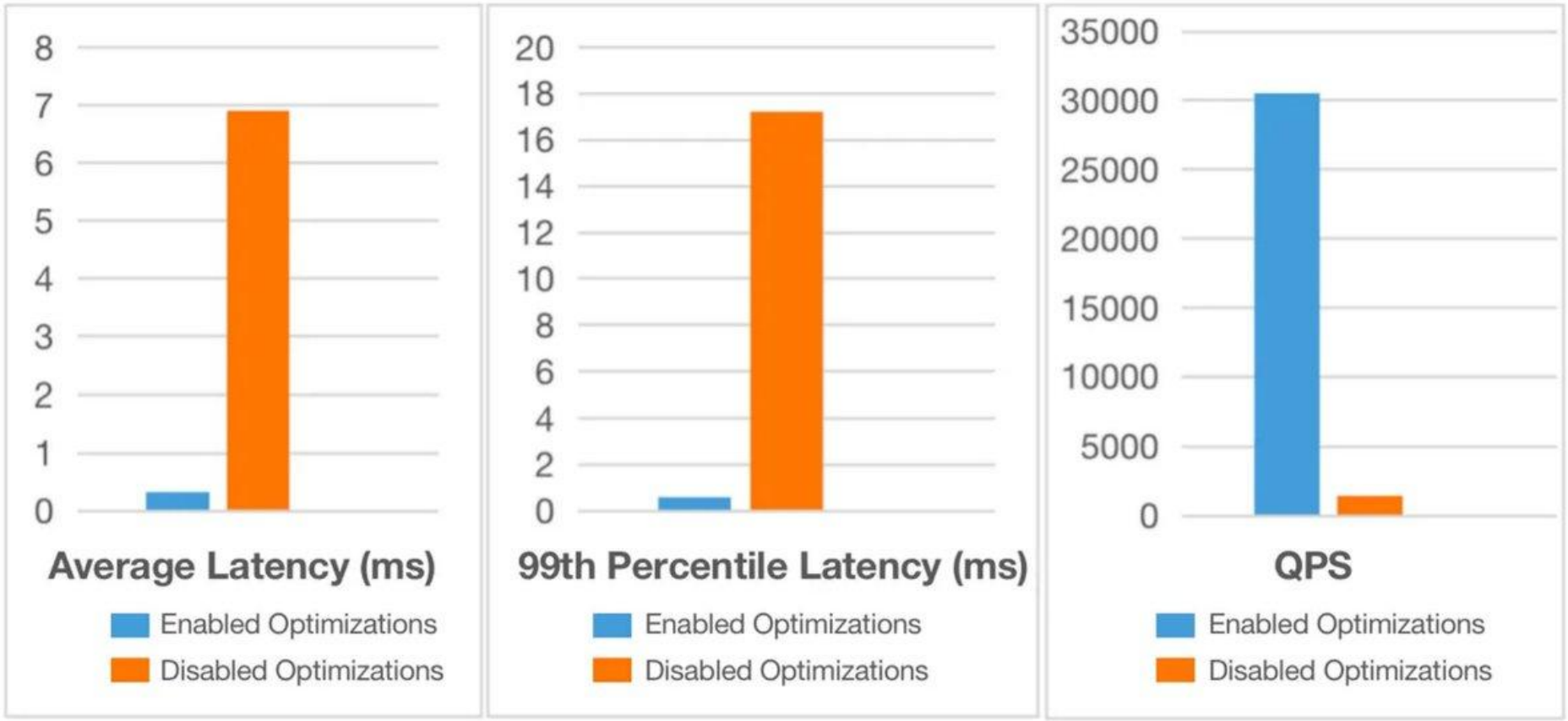
Yahoo! Cloud Serving Benchmark （YCSB）用于专门评测NoSQL系统

- 数据规模：一共 1 亿条数据，平均每行在 1K 左右，测试前进行了预热。
- 表结构为 YCSB 提供， 10 个字段，都是varchar(100)，第一个字段做主键，随机生成

```
// Table creation statement:

CREATE TABLE `usertable` (
 `YCSB_KEY` varchar(255) NULL,
 `FIELD0` text NULL,
 `FIELD1` text NULL,
 `FIELD2` text NULL,
 `FIELD3` text NULL,
 `FIELD4` text NULL,
 `FIELD5` text NULL,
 `FIELD6` text NULL,
 `FIELD7` text NULL,
 `FIELD8` text NULL,
 `FIELD9` text NULL
) ENGINE=OLAP
UNIQUE KEY(`YCSB_KEY`)
COMMENT 'OLAP'
DISTRIBUTED BY HASH(`YCSB_KEY`) BUCKETS 16
PROPERTIES (
"replication_allocation" = "tag.location.default: 1",
'enable_unique_key_merge_on_write" = "true",
'light_schema_change" = "true",
'store_row_column" = "true",
);

// Query statement:

SELECT * from usertable WHERE YCSB_KEY = ?
```

YCSB Benchmark

提升了超过 20 倍，整体性能表现和并发承载实现数据量级的飞跃！



0.3 ms vs 6.9 ms　　0.6 ms vs 17 ms　　30301 QPS vs 1404 QPS

展望

展望

1. 更加智能的 cache
2. 优化存储空间
3. 更智能的调优
4. 支持更丰富的查询语法
5. 支持物化视图，提供二级索引

如何加入 Apache Doris 社区

- **订阅开发者邮件组**

  订阅社区开发者邮件 dev@doris.apache.org 并参与社区的邮件讨论中

- **双周开发者会议(线上会议)**

  通过会议链接参与到社区开发者双周会： https://meeting.tencent.com/dm/6iQBgklhn5LO

- Doris Improvement Proposals(DSIP)

  通过DSIP查阅社区核心功能设计方案及相关进展：

  https://cwiki.apache.org/confluence/display/DORIS/Doris+Improvement+Proposals

欢迎关注：

SELECTDB DataFun.

欢迎关注 SelectDB 微信公众号

获取最新活动资讯、技术解析、社区动态

公司邮箱：support@selectdb.com

SelectDB 官网：www.selectdb.com

欢迎关注 Apache Doris GitHub & 官网

Apache Doris GitHub：https://github.com/apache/doris

Apache Doris 官网：https://doris.apache.org/


感谢观看
SELECTDB
DataFun.