第四周领教直播 · Hive

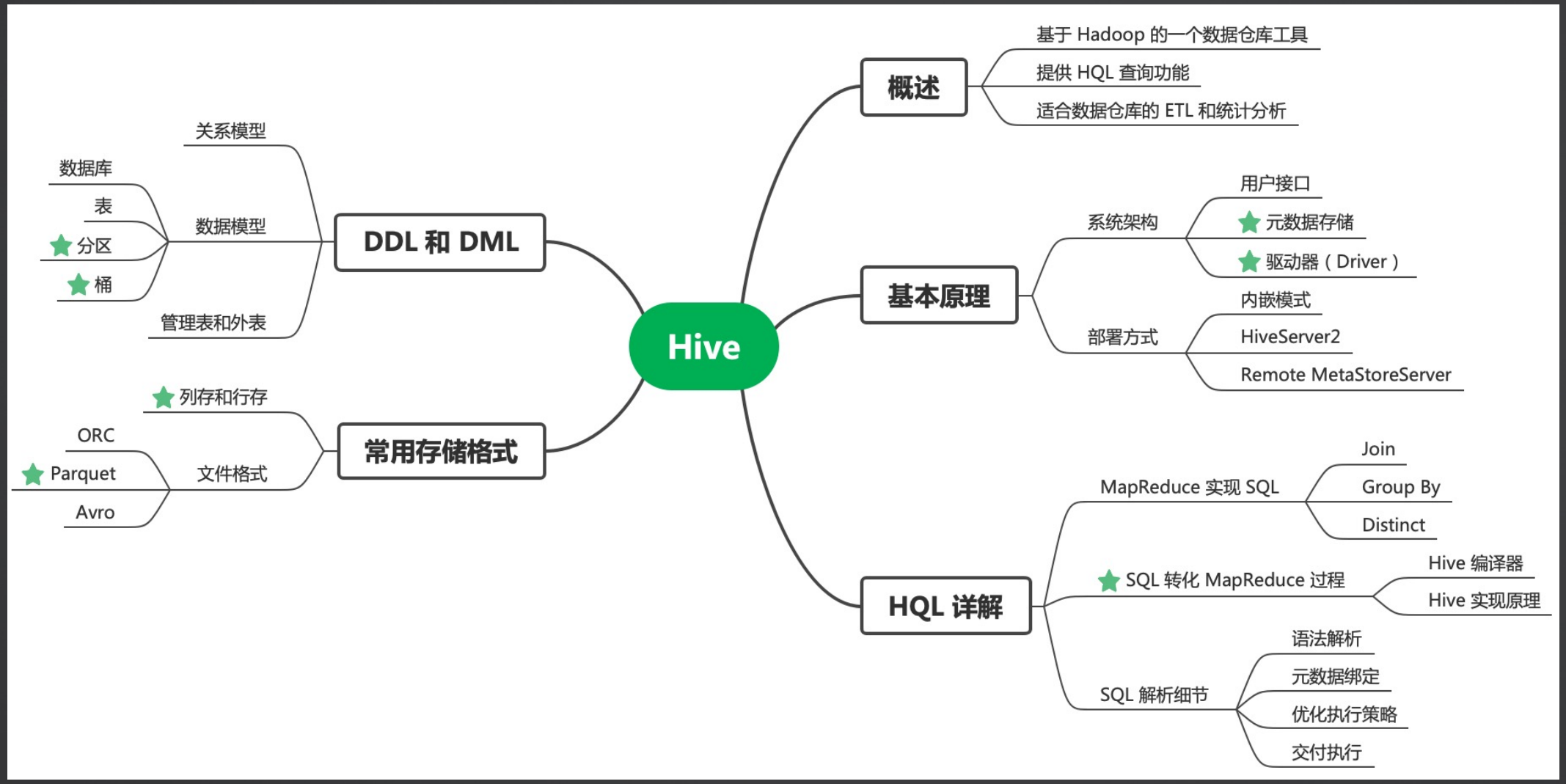
张语

目录

极客时间 训练营

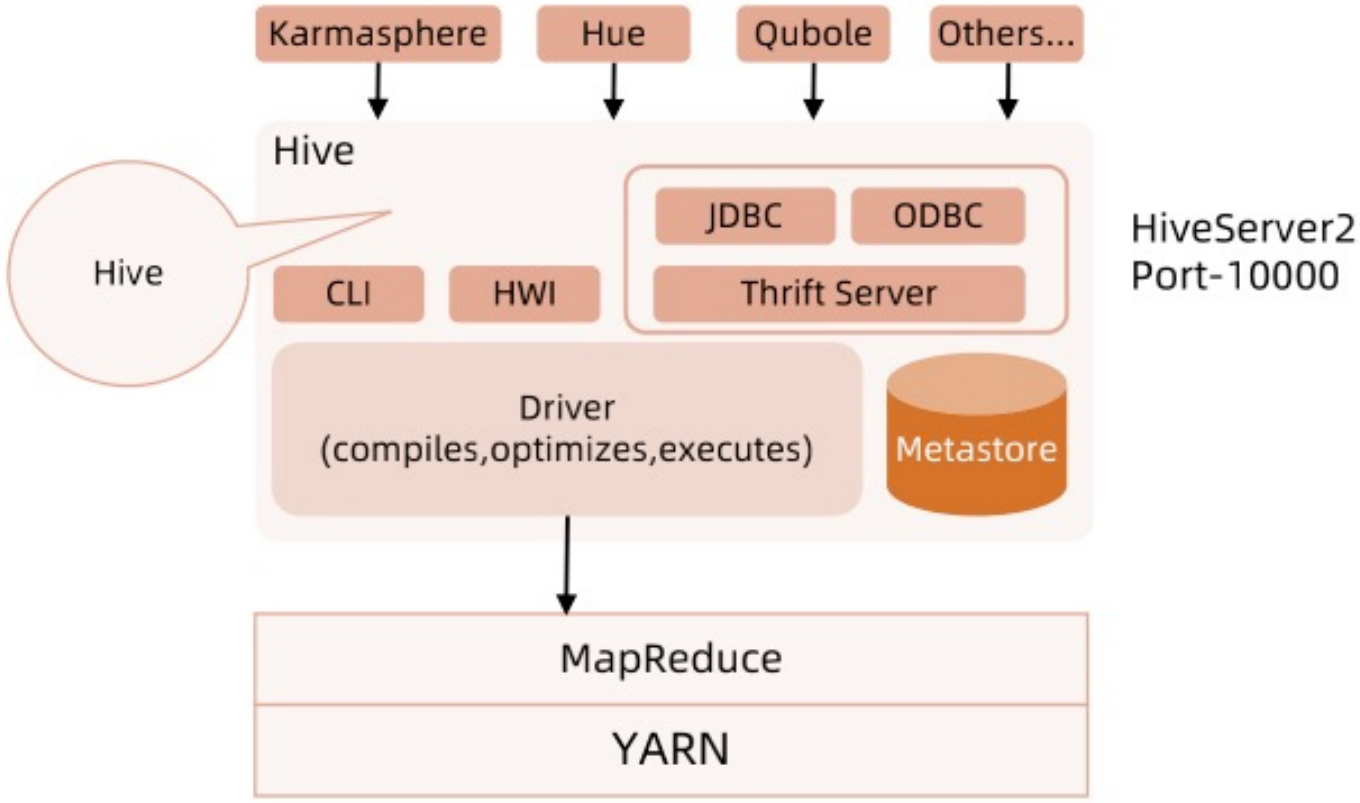
- 重点内容回顾
- 作业思路讲解
- 数据仓库基础
- Hive Metastore
- QA

重点内容回顾


Hive
概述
基于 Hadoop 的一个数据仓库工具
提供 HQL 查询功能
适合数据仓库的 ETL 和统计分析
基本原理
系统架构
用户接口
元数据存储
驱动器（Driver）
部署方式
内嵌模式
HiveServer2
Remote MetaStoreServer
HQL 详解
MapReduce 实现 SQL
Join
Group By
Distinct
SQL 转化 MapReduce 过程
Hive 编译器
Hive 实现原理
SQL 解析细节
语法解析
元数据绑定
优化执行策略
交付执行
DDL 和 DML
关系模型
数据模型
数据库
表
分区
桶
管理表和外表
常用存储格式
列存和行存
文件格式
ORC
Parquet
Avro

Hive 系统架构

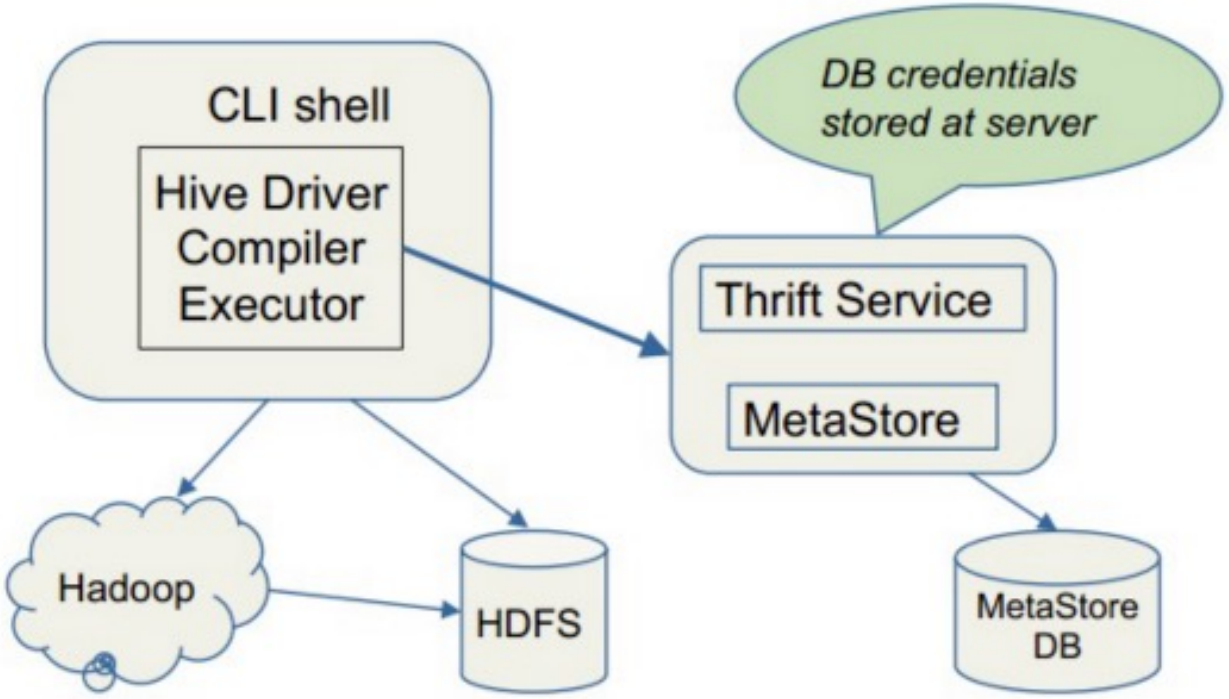
- 接口
  - CLI
  - HWI
  - ThriftServer
  - JDBC/ODBC
- 元数据存储
- 驱动器（Driver）
  - 编译器
  - 优化器
  - 执行器
- Hadoop
  - MapReduce 计算
  - HDFS 数据存储



部署方式

极客时间 训练营

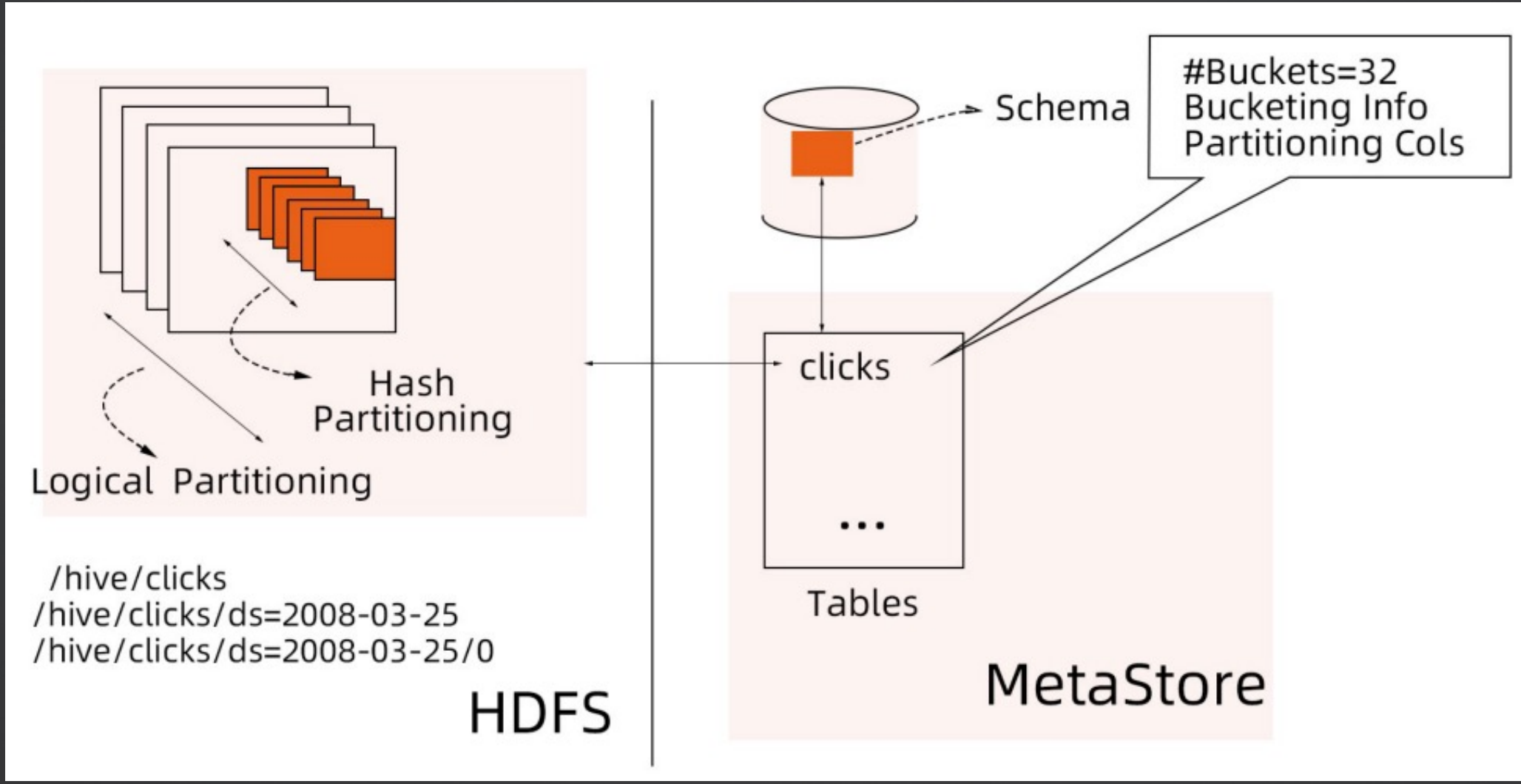
- Remote MetaStoreServer
  - MetaStore 作为一个独立的 Thrift 服务，而不是每个 Driver 自己维护一个瘦 MetaStore 访问 DB。



SQL 转化为 MapReduce 的过程

1. 词法语法解析：Antlr 定义 SQL 的语法规则，完成 SQL 词法，语法解析，将 SQL 转化为抽象语法树（AST Tree）。
2. 语义解析：遍历 AST Tree，抽象出查询的基本组成单元查询块（QueryBlock）。
3. 生成逻辑执行计划：遍历查询块，翻译为逻辑执行计划，Hive 用操作树（OperatorTree）表示。
4. 优化逻辑执行计划：对操作树进行变换，合并不必要的操作，减少 shuffle 数据量，得到优化过的逻辑执行计划。
5. 生成物理执行计划：遍历操作树，翻译为 MapReduce 任务，即物理执行计划。
6. 优化物理执行计划：继续对物理执行计划进行变换，生成最终的 MapReduce 任务。

Hive 数据模型


Schema
#Buckets=32
Bucketing Info
Partitioning Cols
Hash
Partitioning
Logical Partitioning
clicks
...
Tables
/hive/clicks
/hive/clicks/ds=2008-03-25
/hive/clicks/ds=2008-03-25/0
HDFS
MetaStore

行存和列存

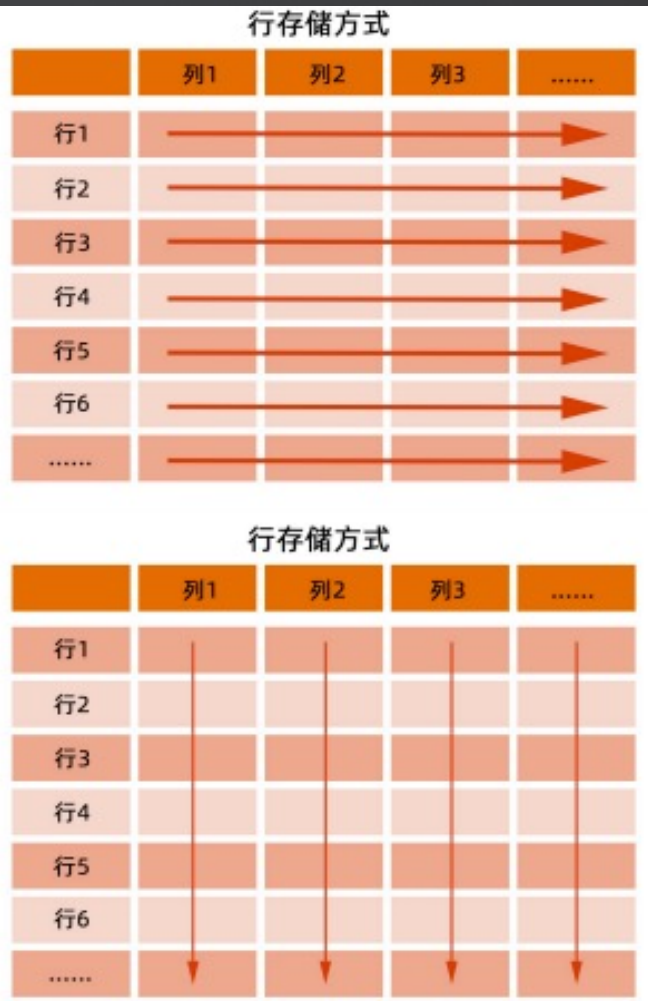
极客时间训练营

行存储

- 适合增加、插入、删除、修改的事务处理处理。
- 对列的统计分析却需要耗费大量的 I/O。对指定列进行统计分析时，需要把整张表读取到内存，然后再逐行对列进行读取分析操作。

列存储

- 对增加、插入、删除、修改的事务处理 I/O 高、效率低。
- 非常适合做统计查询类操作，统计分析一般是针对指定列进行，只需要把指定列读取到内存进行操作。



作业思路讲解

极客时间 训练营

题目一（简单）
展示电影 ID 为 2116 这部电影各年龄段的平均影评分。
提示：group by

题目二（中等）
找出男性评分最高且评分次数超过 50 次的 10 部电影，展示电影名，平均影评分和评分次数。
提示：group by、having、order by

题目三（选做）
找出影评次数最多的女士所给出最高分的 10 部电影的平均影评分，展示电影名和平均影评分（可使用多行 SQL）。
提示： group by、order by、子查询

推荐资料：
《 SQL必知必会》https://book.douban.com/subject/35167240/

OLTP 与 OLAP

极客时间 训练营

- OLTP 系统通常面向的主要数据操作是随机读写，主要采用实体关系模型存储数据，使用事务解决数据一致性问题。
- OLAP 系统面向的主要数据操作是批量读写，主要关注数据的整合以及在复杂大数据查询和处理性能。

分类	OLTP	OLAP
分析粒度	细节的	细节的，综合的或提炼的
时效性	在存取瞬间是准确的	代表过去的数据
数据更新需求	可更新	一般情况，无需更新
操作可预知性	操作需求事先可知道	操作需求事先可能不知道
实时性	对性能要求高，响应毫秒级、秒级	对性能要求相对宽松，响应分钟级、小时级
数据量	一个时刻操作一条或几条记录，数据量小	一个时刻操作一集合，数据量大
驱动方式	事务驱动	分析驱动
应用类型	面向应用	面向分析
应用场景	支持日常运营	支持管理需求
典型应用	银行核心系统、信用卡系统	ACRM、风险管理

数据建模

极客时间 训练营

- 数据模型就是数据组织和存储方法，它强调从业务、数据存取和使用角度合理存储数据。
- 大数据系统需要数据模型方法来帮助更好地组织和存储数据，以便在性能、成本、效率和质量之间取得最佳平衡。

维度模型

极客时间训练营

- 维度模型是数据仓库领域的 Ralph Kimball 大师所倡导的。
- 维度建模从分析决策的需求出发构建模型，为分析需求服务。
- 重点关注用户如何更快速地完成需求分析，同时具有较好的大规模复杂查询的响应性能。
- 典型的代表是星形模型，以及在一些特殊场景下使用的雪花模型。

维度模型

- 事实和维度是维度模型中的核心概念。
- 事实表示对业务数据的度量，而维度是观察数据的角度。
- 事实通常是数字类型的，可以进行聚合和计算，而维度通常是一组层次关系或描述信息，用来定义事实。
- 例如，销售金额是一个事实，而销售时间、销售的产品、购买的顾客、商店等都是销售事实的维度。
- 维度模型按照业务流程领域即主题域建立，例如进货、销售、库存、配送等。

维度模型-设计

极客时间训练营

1. 选择需要进行分析决策的业务过程。业务过程可以是单个业务事件，也可以是某个事件的状态，还可以是一系列相关业务事件组成的业务流程。

2. 选择粒度：在事件分析中，要预判所有分析需要细分的程度， 从而决定选择的粒度。粒度是维度的一个组合。

3. 识别维表：基于粒度设计维表，包括维度属性，用于分析时进行分组和筛选。

4. 选择事实：确定分析需要衡量的指标。

数据仓库分层

数仓建模是从分析决策的需求出发构建模型，它主要是为分析需求服务，因此它重点关注用户如何更快速地完成需求分析，同时具有较好的大规模复杂查询的响应性能。

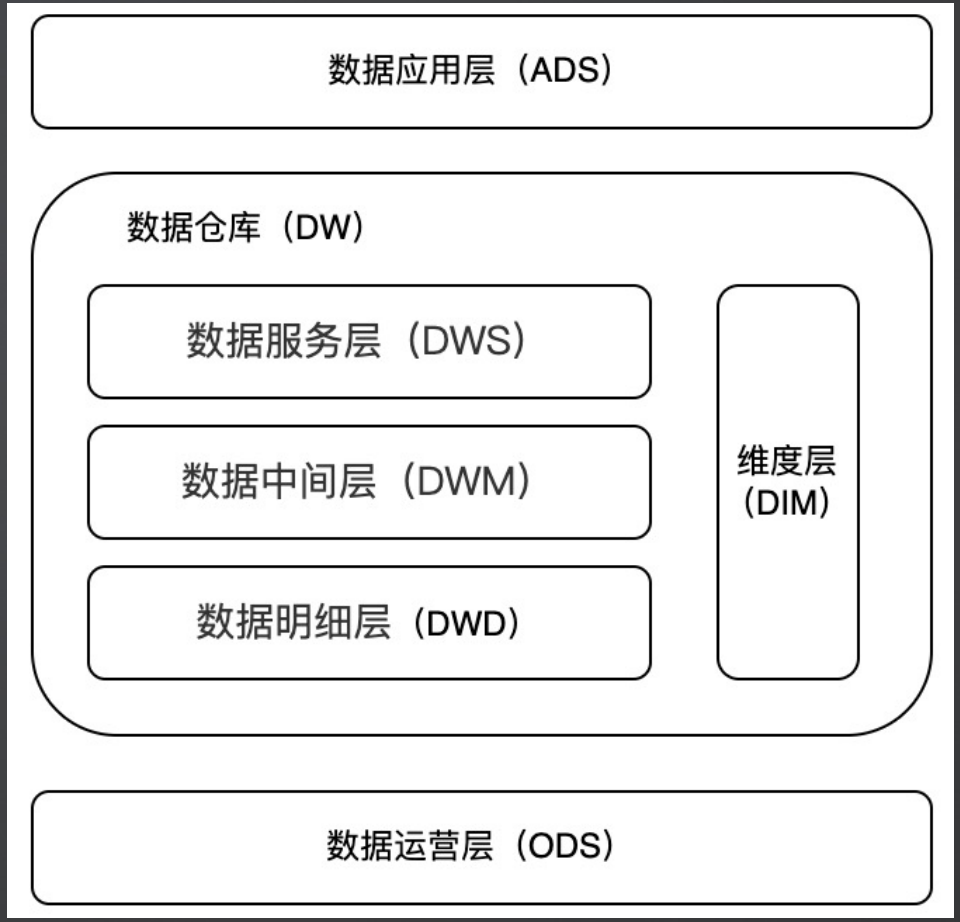
数据分层的好处：

- 清晰数据结构：每一个数据分层都有它的作用域和职责，在使用表的时候能更方便地定位和理解。
- 减少重复开发：规范数据分层，开发一些通用的中间层数据，能够减少极大的重复计算。
- 统一数据口径：通过数据分层，提供统一的数据出口，统一对外输出的数据口径。
- 复杂问题简单化：将一个复杂的任务分解成多个步骤来完成，每一层解决特定的问题。

数据仓库分层

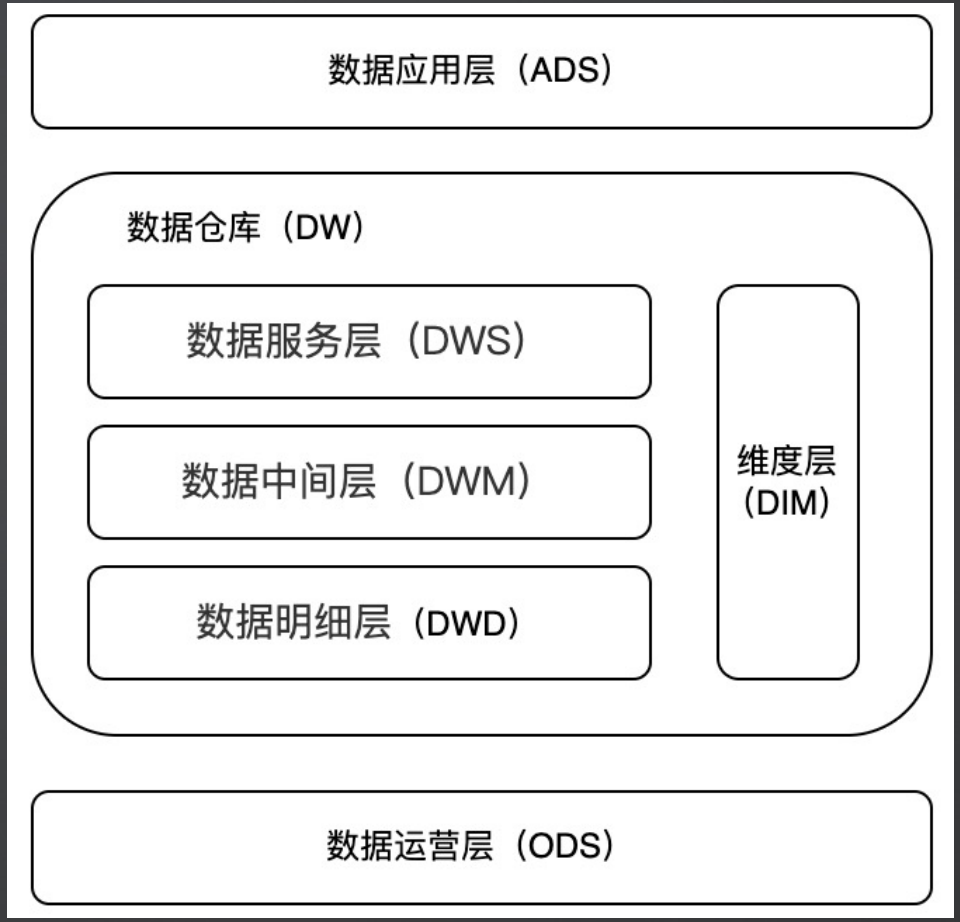
极客时间 训练营

- 数据运营层（ODS，Operational Data Store）
  - 最接近数据源中数据的一层，数据源中的数据，经过抽取、加载，保存到本层。
- 数据明细层（DWD，Data Warehouse Detail）
  - 基于维度表建模，明细宽表，复用关联计算，减少数据扫描。
- 数据中间层（DWM，Data Warehouse Middle）
  - 在DWD层的数据基础上，对数据做轻度的聚合操作，生成一系列的中间表，提升公共指标的复用性，减少重复加工。



数据仓库分层

- 数据服务层（DWS，Data Warehouse Service）
  - 建立命名规范、口径一致的统计指标，为上层提供公共指标，建立汇总宽表。
- 维度层（DIM，Dimension）
  - 建立一致数据分析维表、降低数据计算口径和算法不统一风险。
- 数据应用层（ADS，Application Data Store）
  - 个性化指标加工：定制化、复杂性指标。基于应用的数据组装：宽表集市、趋势指标。



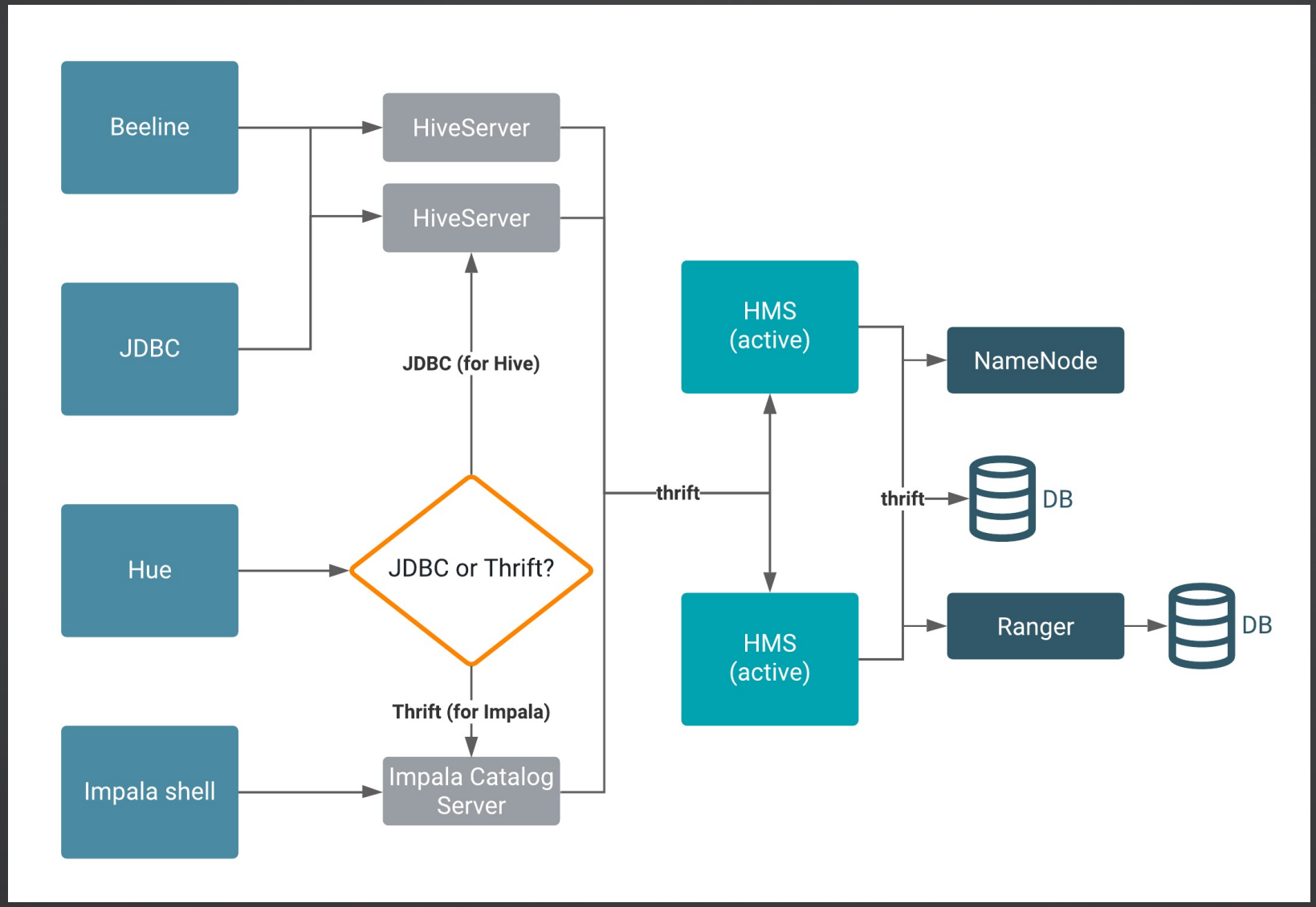
Metastore

- Metastore 功能
  - 数据抽象：数据信息在创建表时给出，并在每次引用表时重复使用
  - 数据发现：使用户能够发现和探索仓库中的相关和具体数据
- Metastore 对象
  - 数据库（Database）
  - 表（Table）
  - 分区（Partition）
  - 桶（Bucket）

Metastore 架构

- Metastore 是一个带有数据库的对象存储。
- 数据库支持的存储是通过一个名为 DataNucleus 的对象关系映射（ORM）解决方案实现的。
- Metastore 服务是无状态的。

Metastore 架构



Metastore 架构-支持数据库

极客时间 训练营

RDBMS	Minimum Version	javax.jdo.option.ConnectionURL	javax.jdo.option.ConnectionDriverName
MS SQL Server	2008 R2	jdbc:sqlserver://<HOST>:<PORT>;DatabaseName=<SCHEMA>	com.microsoft.sqlserver.jdbc.SQLServerDriver
MySQL	5.6.17	jdbc:mysql://<HOST>:<PORT>/<SCHEMA>	com.mysql.jdbc.Driver
MariaDB	5.5	jdbc:mysql://<HOST>:<PORT>/<SCHEMA>	org.mariadb.jdbc.Driver
Oracle*	11g	jdbc:oracle:thin:@//<HOST>:<PORT>/xe	oracle.jdbc.OracleDriver
Postgres	9.1.13	jdbc:postgresql://<HOST>:<PORT>/<SCHEMA>	org.postgresql.Driver

Hive Metastore 实践-知乎

知乎将元信息存储在 MySQL 内的，随着业务数据的不断增长，MySQL 内已经出现单表数据量两千多万的情况，当用户的任务出现 Metastore 密集操作的情况时，往往会出现缓慢甚至超时的现象，极大影响了任务的稳定性。

已有方案

业内目前有两种方案可供借鉴：

1. 对 MySQL 进行分库分表处理，将一台 MySQL 的压力分摊到 MySQL 集群。

2. 对 Hive Metastore 进行 Federation，采用多套 Hive Metastore + MySQL 的架构，在 Metastore 前方设置代理，按照一定的规则，对请求进行分发。

方案对比

- 对 MySQL 进行分库分表，首先面临的直接问题就是需要修改 Metastore 操作 MySQL 的接口，涉及到大量高风险的改动，后续对 Hive 的升级也会更加复杂。

- 对 Hive Metastore 进行 Federation，尽管不需要对 Metastore 进行任何改动，但是需要额外维护一套路由组件，并且对路由规则的设置需要仔细考虑，切分现有的 MySQL 存储到不同的 MySQL 上，并且可能存在切分不均匀，导致各个子集群的负载不均衡的情况。

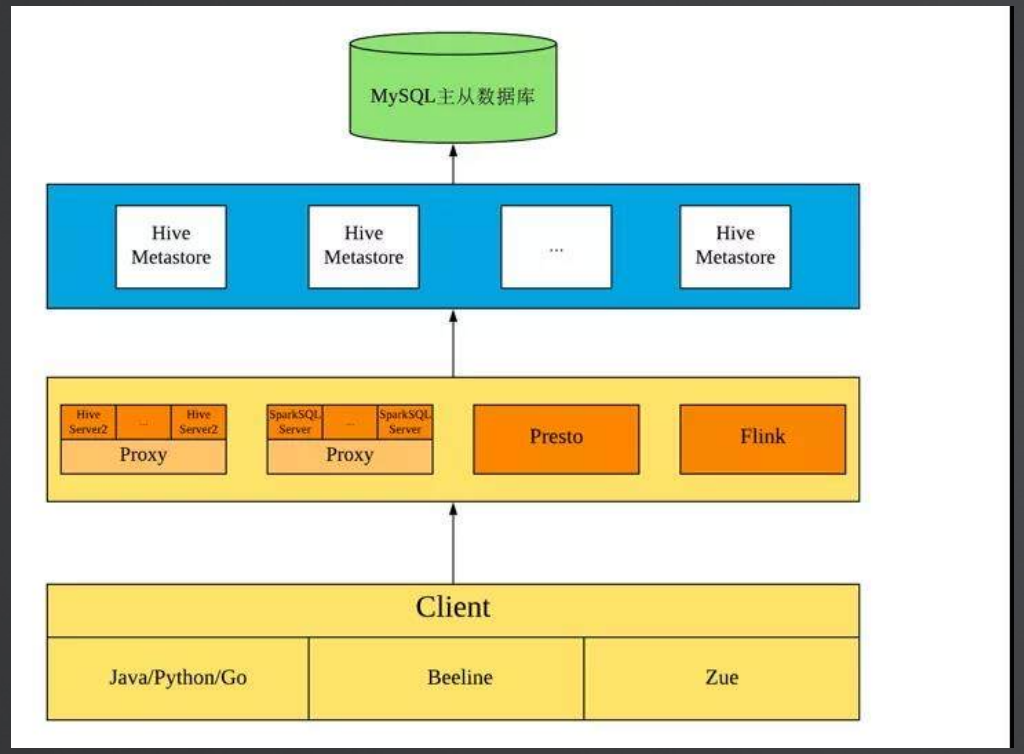
最终方案

- 核心问题：当数据量增加时，MySQL 受限于单机性能，很难有较好的表现，而将单台 MySQL 扩展为集群复杂度较高。如果能够找到一款兼容 MySQL 协议的分布式数据库，就能解决这个问题。

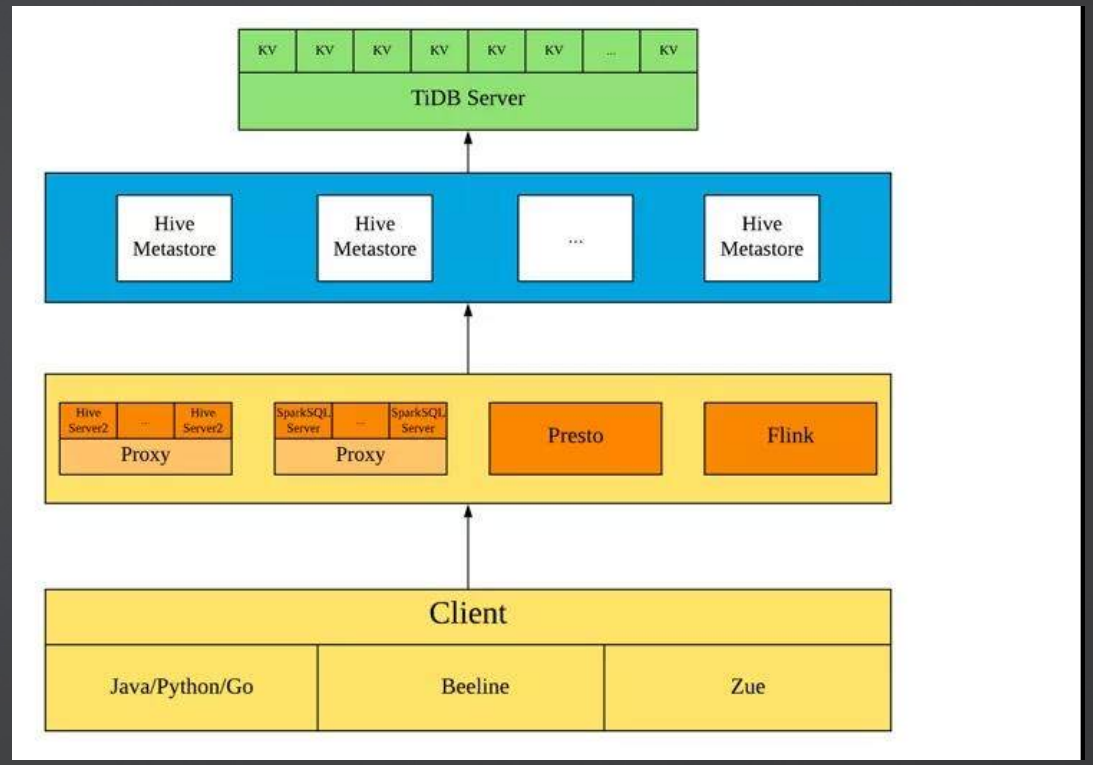
- 方案选型：使用 TiDB 代替 MySQL。

- TiDB 是 PingCAP 开源的分布式 NewSQL 数据库，它支持水平弹性扩展、ACID 事务、标准 SQL、MySQL 语法和 MySQL 协议，具有数据强一致的高可用特性，是一个不仅适合 OLTP 场景还适 OLAP 场景的混合数据库。

迁移对比


MySQL主从数据库
Hive Metastore
Hive Metastore
...
Hive Metastore
Hive Server2
...
Hive Server2
Proxy
SparkSQL Server
...
SparkSQL Server
Proxy
Presto
Flink
Client
Java/Python/Go
Beeline
Zue

迁移前


KV
KV
KV
KV
KV
KV
...
KV
TiDB Server
Hive Metastore
Hive Metastore
...
Hive Metastore
Hive Server2
...
Hive Server2
Proxy
SparkSQL Server
...
SparkSQL Server
Proxy
Presto
Flink
Client
Java/Python/Go
Beeline
Zue

迁移后

迁移流程

1. 将 TiDB 作为 MySQL 的从库，实时同步数据；
2. Metastore 缩容至 1 个，防止多个 Metastore 分别向 MySQL 及 TiDB 写入，导致元数据不一致；
3. 选取业务低峰期，主从切换，将主切为 TiDB，重启 Metastore ；
4. Metastore 扩容；
5. 此迁移过程对业务几乎无感，成功上线。

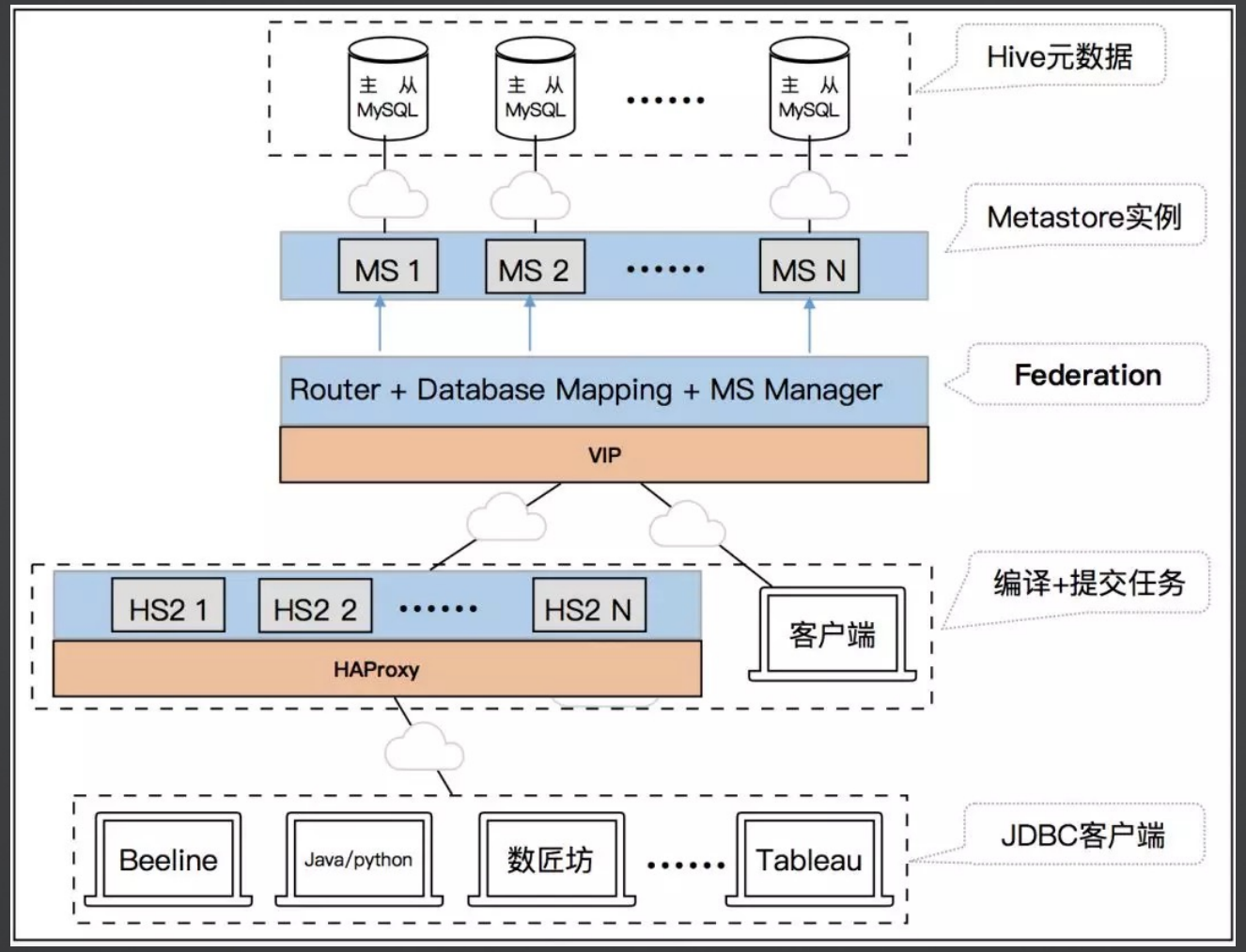
滴滴的方案

- 采用基于 Hive Metastore 层实现 Federation，实现思路主要是以 Hive DB 切分，在 Hive DB 层面将元数据分布多套 MySQL 环境存储，这样对 Hive Metastore 本身不需要做任何修改。

- 基于 waggle-dance 做二次开发。

- waggle-dance 项目主要是联合多个 Hive Metastore 数据查询的服务，实现了一个统一的路由接口解决多套 Hive Metastore 环境间的元数据共享问题。

滴滴的方案

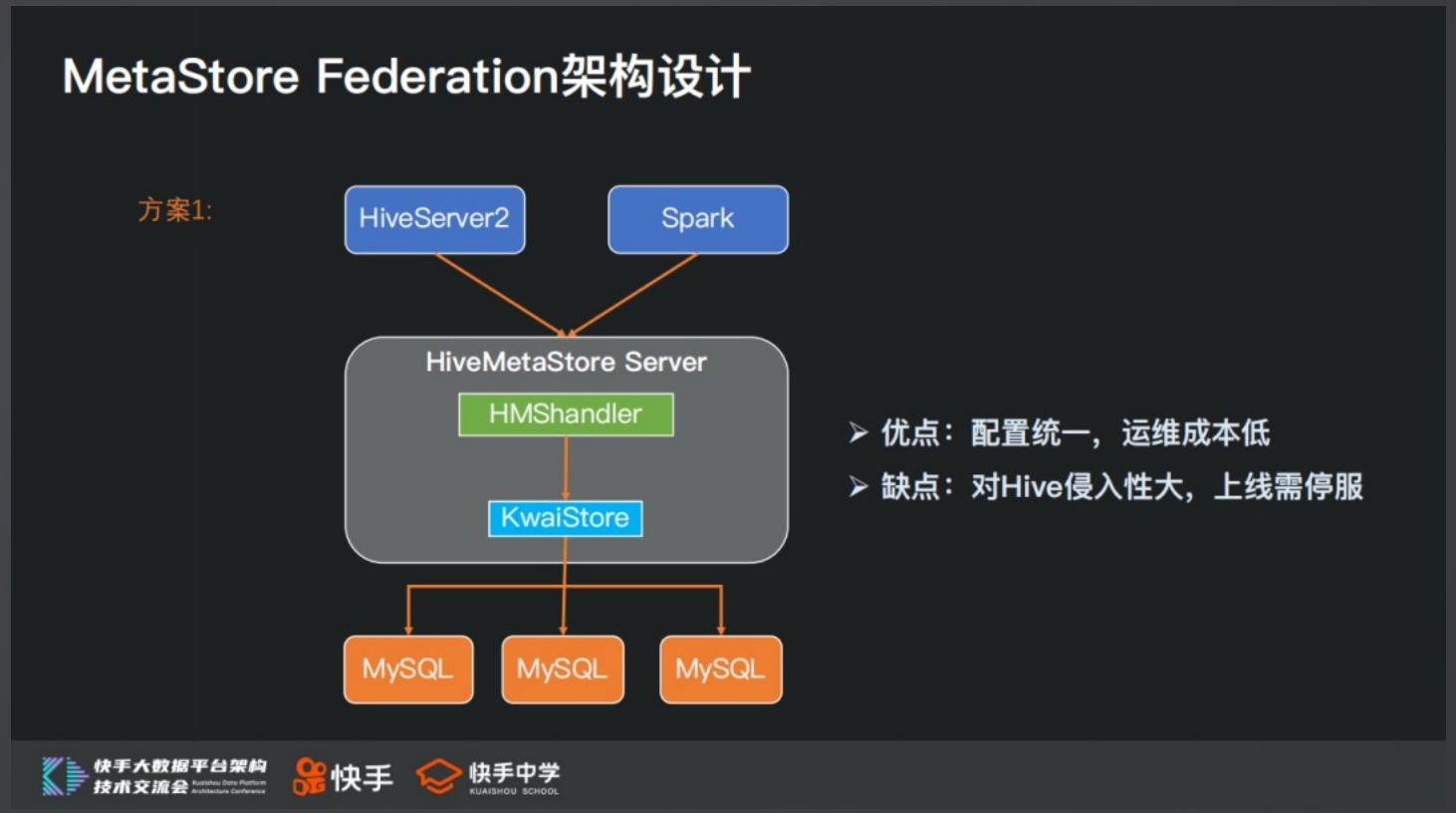

主 从
MySQL
主 从
MySQL
主 从
MySQL
Hive元数据
MS 1
MS 2
MS N
Metastore实例
Router + Database Mapping + MS Manager
VIP
Federation
HS2 1
HS2 2
HS2 N
HAProxy
客户端
编译+提交任务
Beeline
Java/python
数匠坊
Tableau
JDBC客户端

快手的方案

- 采用 Federation 方案。
- Hive MetaStore 内部实现逻辑，持久化元数据层被抽象成了 RawStore，比如 MySQL 对应的实现时 ObjectStore。

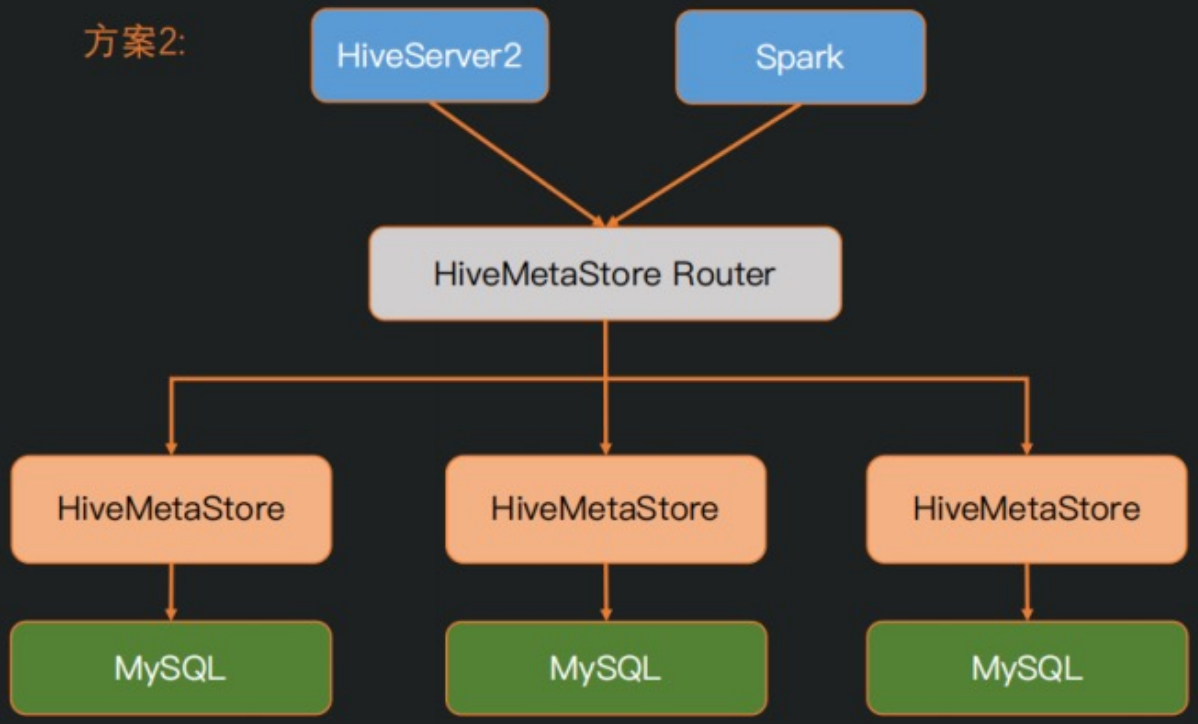
快手的方案

基于 ObjectStore 已有功能和代码实现 KwaiStore，在 KwaiStore 中实现 Hive DB 路由数据源的功能，配置不同 Hive DB 到对应 MySQL 数据源的映射关系。



快手的方案

MetaStore Federation架构设计

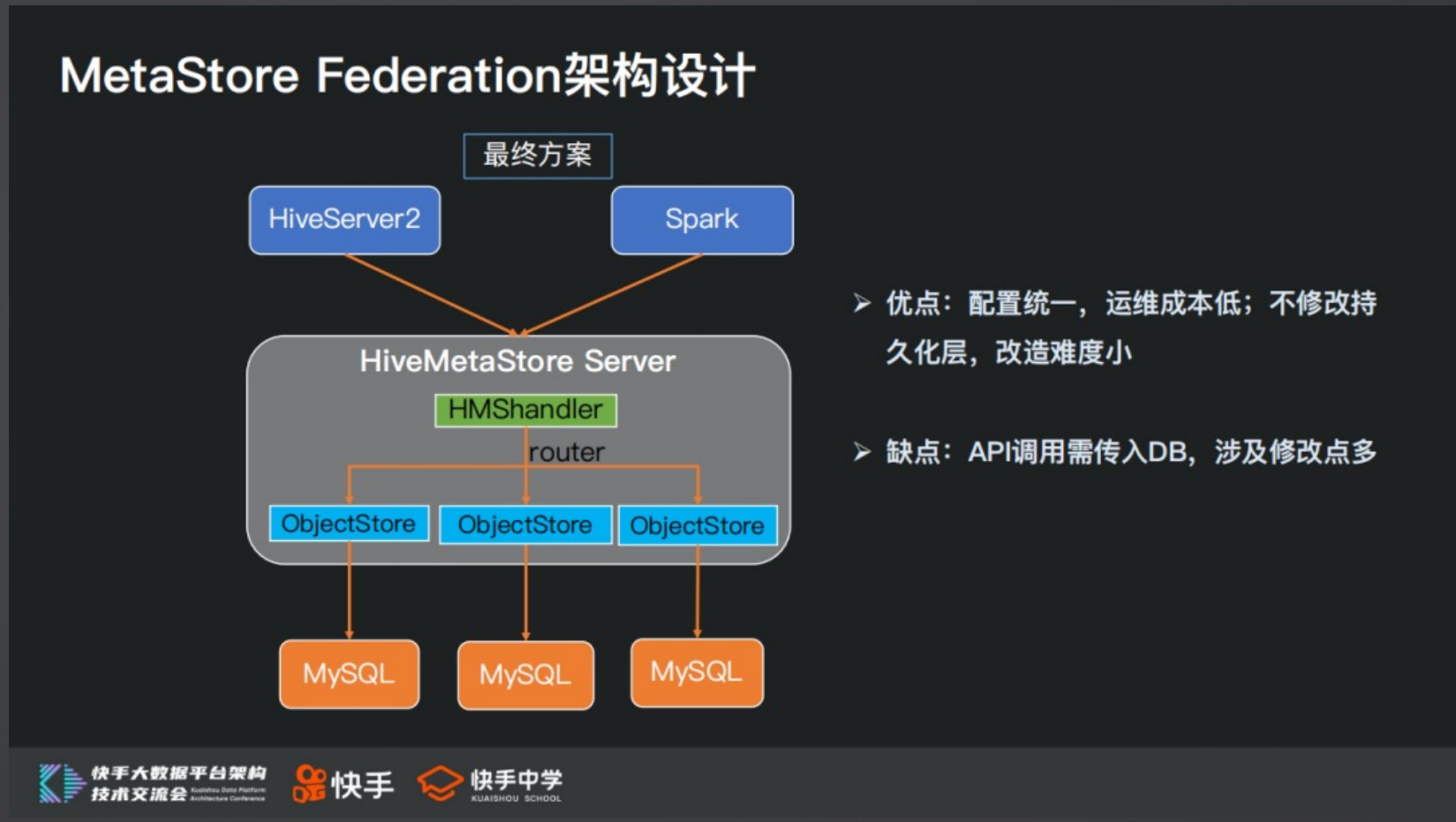


- 优点：MetaStore无需改动，router策略定制灵活，可水平扩容，上线风险低
- 缺点：新引入服务层，MetaStore配置不统一，运维成本高

滴滴应用经验：https://segmentfault.com/a/1190000019370850

快手的方案

在 HMSHandler 层来实现路由功能，在 HMSHandler 中维护一组 HiveDB 与 RawStore 的映射关系，在 getMS()时传入 Hive DB，路由根据 DB 判断应该使用那个 RawStore，不修改 RawStore 中 API 的实现，不涉及到持久化层的侵入改造。



QA

THANKS