

# HBase在滴滴出行的应用场景及最佳实践

李扬 2017-05-19



# 背景介绍

---

## 滴滴出行:

提供一站式的出行服务，包括专车，快车，出租车，巴士，试驾，代驾，租车，共享单车（OFO）等出行服务。

## HBase:

Hadoop Database，是一个高可靠性、高性能、面向列、可伸缩的分布式存储系统

# HBase在滴滴的主要使用业务

---

## 在线业务：

- 服务于最终用户，需要实时快速地响应用户的操作
- 对数据访问的延时非常敏感，访问趋向随机
- 业务如派单，动调，计费，支付，客服等

## 离线业务：

- 通常是定时的大批量处理任务，对一段时间内的数据进行处理并产出结果
- 对任务完成的时间要求一般，处理逻辑复杂
- 业务如天报表，安全分析，用户行为分析，模型训练等

# 访问HBase的方式

---

HBase Native API

Thrift server (C++, PHP, Go, python)

Phoenix

Phoenix Queryserver

MapReduce job

Spark Job

Streaming

# 存放在HBase中的主要数据

---

数据类型：

## 1. 统计结果，报表数据

- 运营情况，运力情况，收入等结果
- 通常配合Phoenix进行SQL查询
- 数据量小，查询灵活性高，延时要求一般

## 2. 原始事实类数据

- 订单，司机，乘客等，GPS和日志等
- 主要用作在线和离线数据供给
- 数据量大，一致性和可用性要求高，延时要求高，实时写入，单点或者批量查询

## 3. 生产中间数据和结果数据

- 模型训练所需数据等
- 数据量大，可用性和一致性要求一般，批量查询对吞吐要求高

## 4. 线上系统的备份数据

- 历史数据，查询频率不高，延时要求高

## 场景一：订单事件

---

需要满足三个需求：

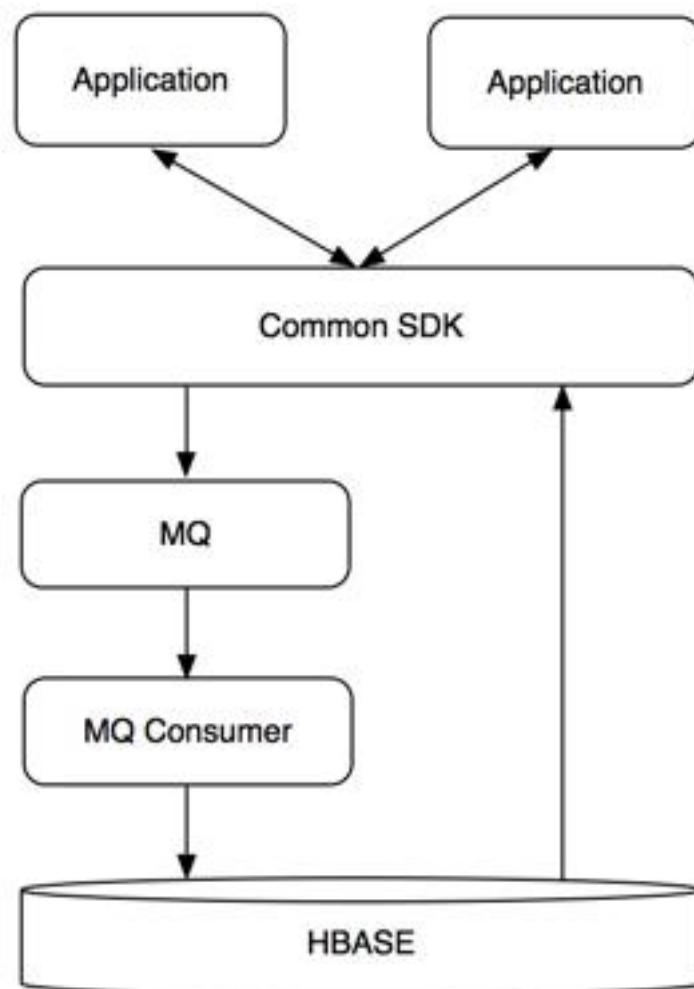
1. 在线查询订单的生命周期的各个状态
  1. 包括status, event\_type, order\_detail等信息
  2. 主要查询来自客服系统
2. 在线历史订单详情查询
  1. 同时由redis来存储近期的订单，当redis不可用，查询会直接落到HBase
3. 离线对订单的状态进行分析

## 场景一：订单事件

写入，满足每秒10K事件

读取，满足每秒1K事件

时效性，5s以内数据可用



# 场景一：订单事件

---

## 1. 订单状态表

- Rowkey:  $reverse(order\_id) + (MAX\_LONG - timestamp)$
- Columns: 该订单各种状态

## 2. 历史订单表

- Rowkey:  $reversed(passenger\_id \mid driver\_id) + (MAX\_LONG - timestamp)$
- Columns: 用户在时间范围内的所有订单



## 场景二：司机乘客轨迹

---

需求：

- 1. 满足实时或者准实时司机乘客的轨迹坐标查询
- 2. 满足离线大规模的轨迹分析

场景：

- 1. 给定ID，查询其历史移动轨迹
- 2. 给定时间和空间范围，查询符合条件的所有轨迹

## 场景二：司机乘客轨迹

---

使用坐标的业务：

客服系统

查询某客户的某个订单的轨迹

可视化系统

查询指定地理范围的轨迹情况

坐标+半径距离，坐标矩形

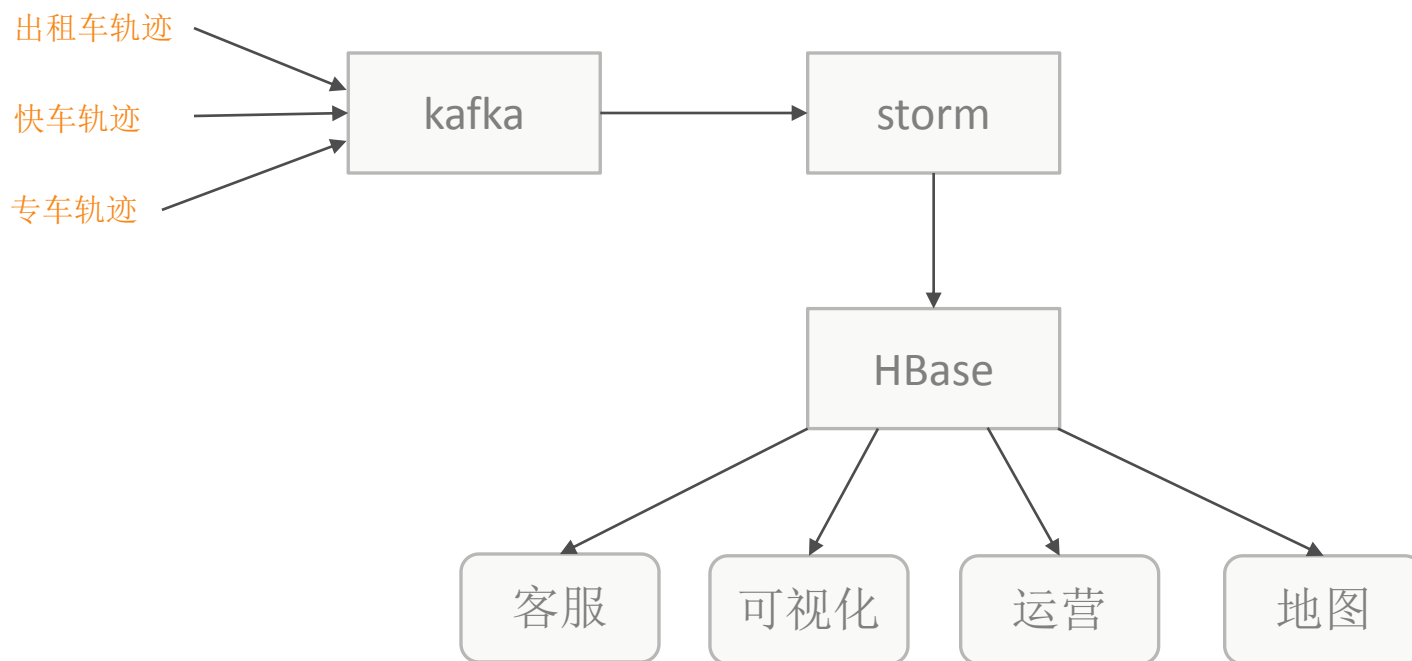
运营系统

地图交通分析

质量控制

## 场景二：司机乘客轨迹

坐标数据流水线：



## 场景二：司机乘客轨迹

---

### 通过ID查询轨迹

- Rowkey: ID+Timestamp
- Column: 轨迹详细信息
- 提供java API给用户使用

### 通过地理范围查找全部出现的轨迹

需要建立空间索引表

GeoHash分区

Rowkey: Reversed\_geohash + Timestamp + ID

提供3种方式访问

1. 小范围或短时间数据：API一次性查询， 延时小，成本低
2. 中等范围或中等时间数据： 提供iterator/scanner批量查询结果，延时较高，成本低
3. 大范围或者长时间数据：提供Base mapper等离线查询方法，延时高，成本高

## 场景三：ETA

---

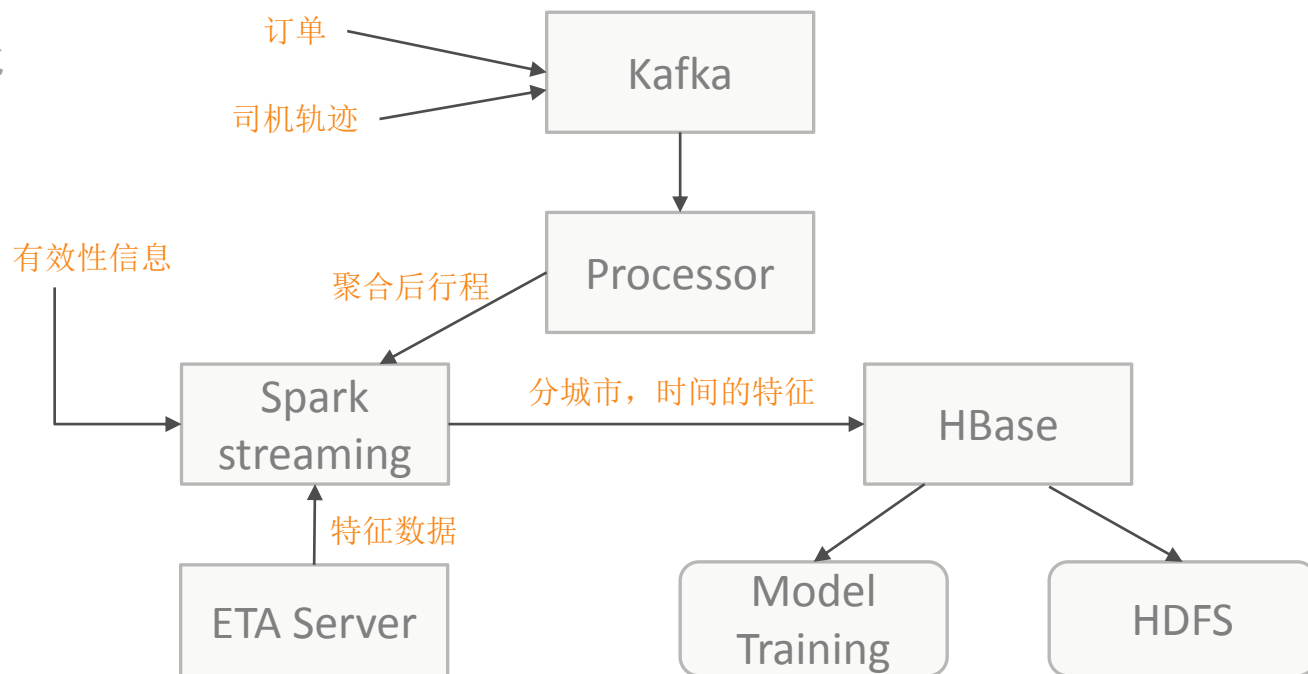
### ETA（预计到达时间）模型实时训练

- 减少训练时间，准实时生产
- 多城市并行训练
- 增加灵活性
- 减少人工干预造成的问题

## 场景三：ETA

ETA流程：

- 1. 原始数据汇集
- 2. 清洗过滤
- 3. 特征提取
- 4. 存储和持久化
- 5. 模型训练



## 场景三：ETA

---

模型训练通过spark任务，每30分钟对各个城市训练一次

模型训练第一个阶段，在5分钟内，按照设定条件从HBase读取所有城市数据

模型训练第二阶段在25分钟之内完成ETA的计算

Rowkey: Salting+CityId+Type0+Type1+Type2+Timestamp

Columns: Order, Feature

HBase中的数据会每隔一段时间持久化至HDFS中，供新模型测试和新特征提取

## 场景四：监控工具

---

Hadoop集群资源监控和查询

将hdfs文件的信息和job history定期导入HBase

通过phoenix来做复杂交互查询

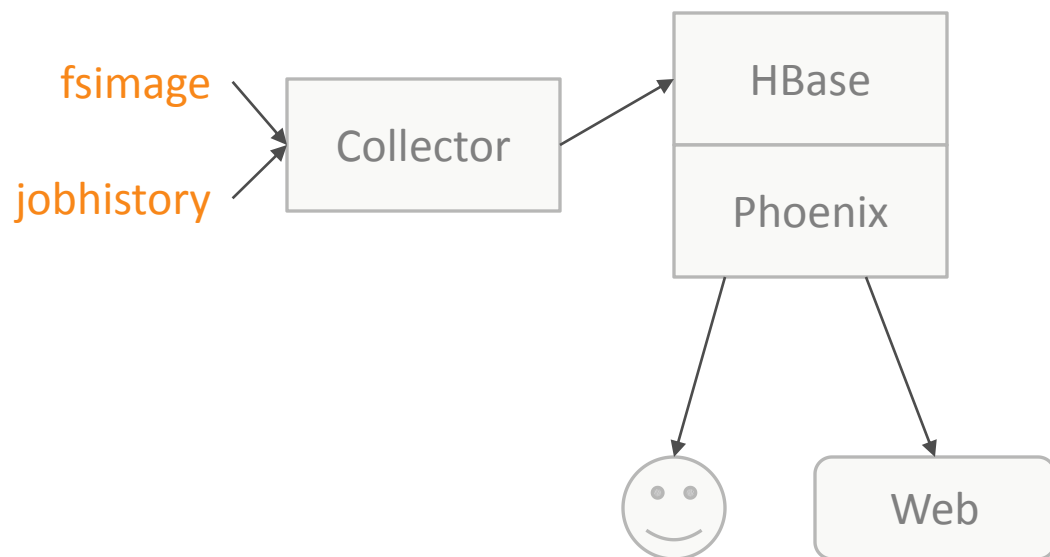
生产各种报表在前端展示



## 场景四：监控工具

---

大量数据每日汇聚到HBase，用户通过phoenix进行查询



## 场景四：监控工具

---

该监控工具每天将几亿的路径信息和当天执行的任务历史信息写入HBase

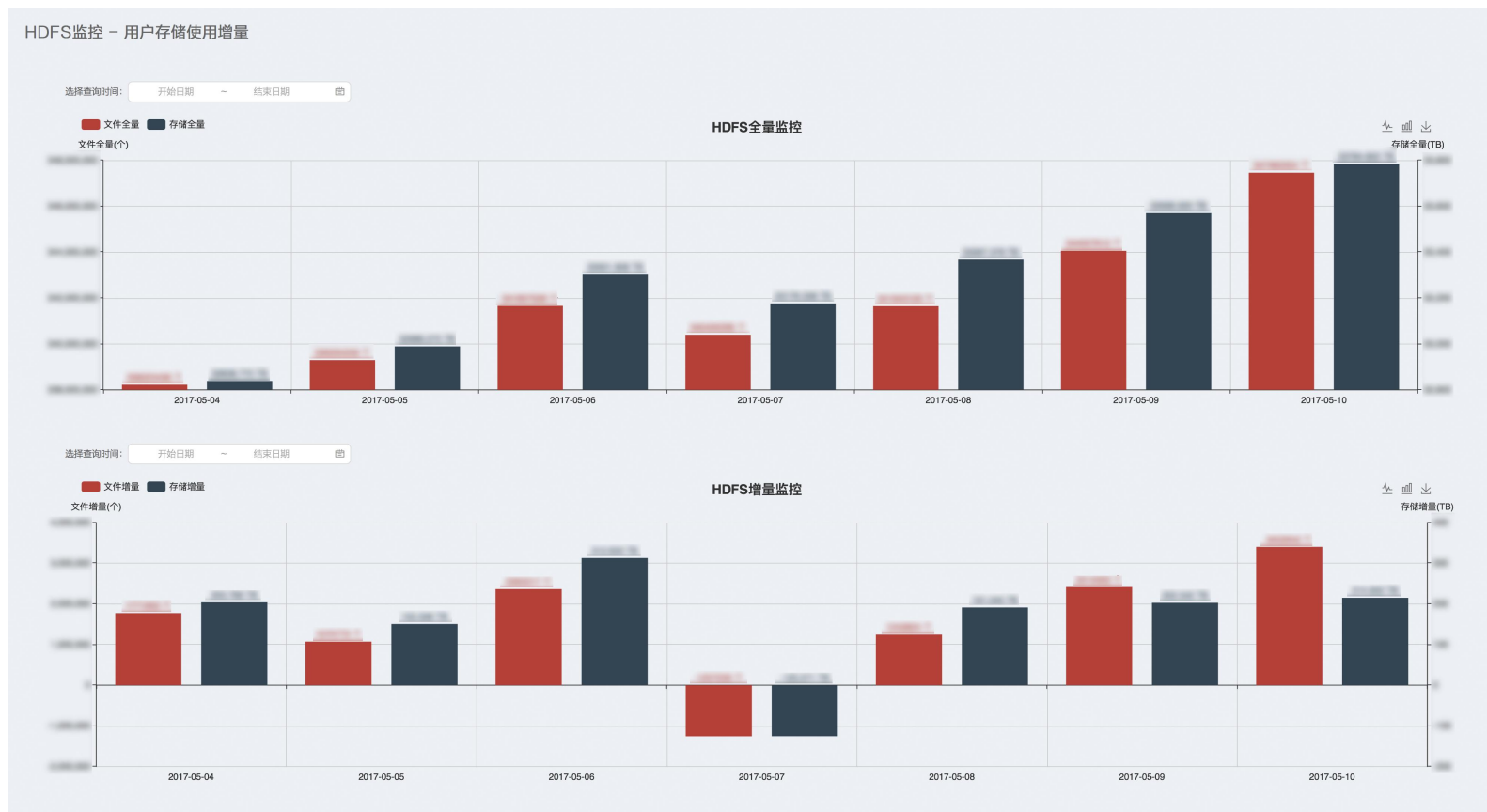
Rowkey: path / jobId

Columns: 多列的相关信息

用户通过phoenix用SQL对数据统计

结果在秒级别返回

## 场景四：监控工具



## 场景四：监控工具

DCM

HDFS监控 ▾Hive监控 ▾冷热数据 ▾计算监控 ▾用户配额Yarn监控 ▾成本监控 ▾liyanglee

HDFS监控 – 用户存储使用增量

用户日增量数据：

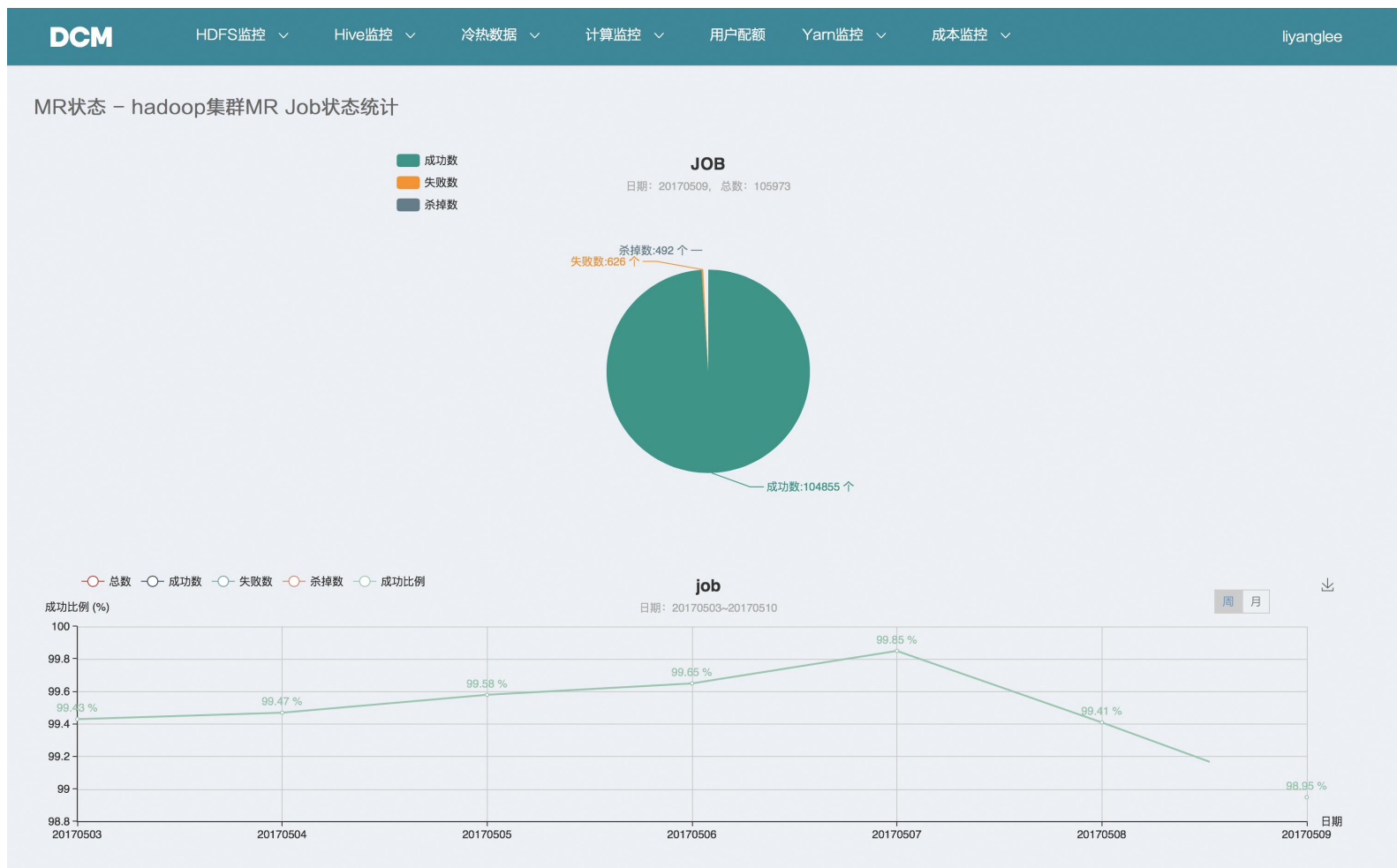
用户名 ○：

查询日期：

搜索

用户名	当天存储量 ▾	存储量日增量 ▾	当天文件数 ▾	文件数日增量 ▾
root	10.000 TB	10.000 TB	100000000	1000000
hadoop-hdfs	1.000 TB	1.000 TB	10000000	1000000
hadoop-yarn	1.000 TB	1.000 TB	10000000	1000000
hadoop-mapreduce	1.000 TB	1.000 TB	10000000	1000000
hadoop-streaming	1.000 TB	1.000 TB	10000000	1000000
hadoop-common	1.000 TB	1.000 TB	10000000	1000000
hadoop-ant	1.000 TB	1.000 TB	10000000	1000000
hadoop-ivy	1.000 TB	1.000 TB	10000000	1000000
hadoop-jetty	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-core	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-api	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-impl	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-layout	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-encoder	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-core	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-api	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-impl	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-layout	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-encoder	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-core	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-api	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-impl	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-layout	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-encoder	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-core	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-api	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-impl	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-layout	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-encoder	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-core	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-api	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-impl	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-layout	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-encoder	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-core	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-api	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-impl	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-layout	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-encoder	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-core	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-api	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-impl	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-layout	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-encoder	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-core	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-api	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-impl	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-layout	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-encoder	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-core	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-api	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-impl	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-layout	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-encoder	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-core	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-api	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-impl	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-layout	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-encoder	1.000 TB	1.000 TB	10000000	1000000
hadoop-log4j2-core	1.000 TB	1.000 TB	10000000	10000

## 场景四：监控工具



# HBase多租户的挑战

---

用户管理

项目管理

资源隔离

性能优化

成本控制

# 基础平台管理者和用户的战斗

---

## 用户方面常见的问题：

- 对使用资源情况不做分析
- 数据量变化后不做调整
- 项目上下线无计划
- 永远想要最多的权限
- 永远想要最多的资源

## 平台管理者常见的问题：

- 难以理解所有的用户的业务
- 对项目目前的状态不清楚
- 不能判断用户的需求是否合理
- 出现问题定位排查时间长
- 一个用户的问题会影响其它用户

# 滴滴HBase用户和项目的管理方案

---

1. 通过DHS (Didi HBase Service) 来管理项目
2. 通过namespace, RS group等技术来分隔用户数据和权限
3. 通过计算开销并计费的方法来管控资源分配



# Didi HBase Service (DHS) 系统

---

## 项目生命周期管理

- 立项
- 资源预估和申请
- 项目需求调整
- 需求讨论

## 用户管理

- 权限管理
- 项目审批

## 资源管理

- 集群管理

# DHS系统

## 创建项目：

Didi Hbase Service

功能菜单

首页

项目列表

hbase简介

常用场景

帮助

平台介绍使用帮助李扬

首页 / 项目列表 / 项目创建

\* 项目名称

\* 英文名称

\* 所在部门

产品

\* 项目owner

请输入owner的邮箱前缀, 只允许填写一个owner

(权限: 读写)

\* 项目成员

请输入成员的邮箱前缀, 多个项目成员请用逗号分隔

(权限: 只读)

\* 组账号

(权限: 读写)

\* 预约onboarding时间

选择日期

(若非紧急任务, 请选择周二、周四onboarding)

项目标签

☐ 星辰花 ☐ 蓝莲花 ☐ 风铃花 ☐ 别乱花

\* 项目背景

HBASE使用的业务背景以及上下游系统描述。如: gps定位数据, 上游为Hadoop, 下游为应用: 每天存储离线数据进入hbase, 应用在线访问 (必填大于50个字符)

重置

提交

# DHS系统

## 新建表以及性能需求预估

Didi Hbase Service

平台介绍使用帮助李扬

功能菜单

返回列表

项目动态

项目概览

表管理

新建表

项目监控

帮助

项目列表 / 表管理 / 表创建

表类型: ☒ 普通表 ☐ phoenix表

namespace

表名

rowkey设计

列族

列族名Cversion1TTL7776000秒

COMPRESSIONSNAPPYDATA\_BLOCK\_ENCODINGFAST\_DIFF

split方式 ☒ 常用 ☐ 范围 ☐ 枚举

region 设计

通用类型请选择region数量

访问量

峰值scan均值scan存储容量GB

峰值get均值get每条记录大小

峰值put均值put每次scan条数

峰值delete均值deletecache命中率

重置

提交

# DHS系统

## 项目流程审批

Didi Hbase Service

功能菜单

返回列表

项目动态

项目概览

表管理

新建表

项目监控

帮助

项目列表 / 项目概况

1

2

3

4

5

6

7

项目创建/更改

表创建/更改

开发审核

运维审核

测试建表

线上建表

已上线

项目 kuaiche\_history

项目介绍:  
项目分为两个部分, 第一部分是订单事件流, 第二部分是历史交易订单。上游系统从消息队列中获取数据(java程序)写入hbase;  
下游系统通过php访问thrift来访问hbase

用户信息

部门: 平台技术

产品: 历史交易订单

owner:

用户:

组账户:

连接信息

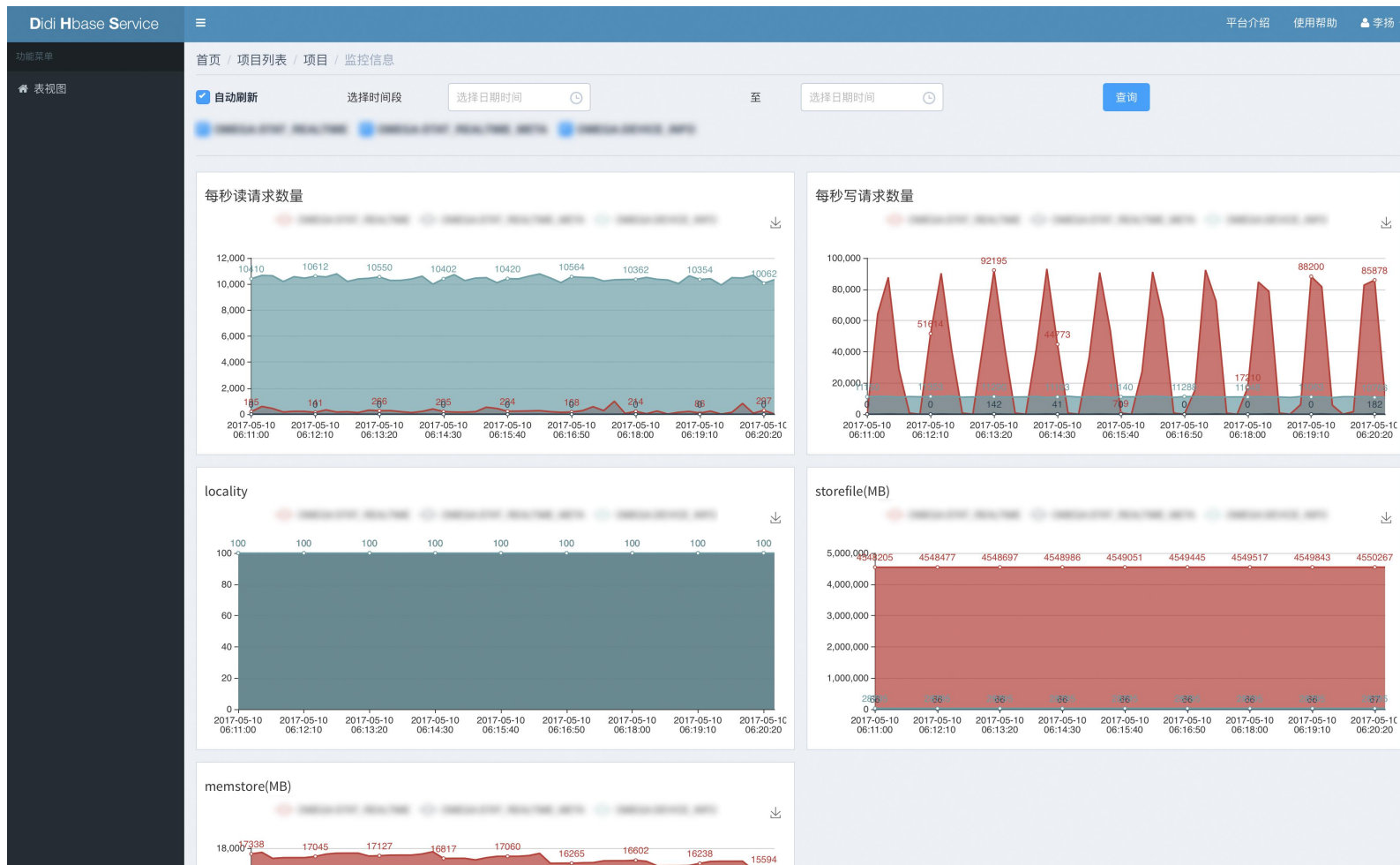
测试环境: 办公网测试

预发环境: 星光预发集群

生产环境: 广州存储集群

# DHS系统

## 项目表监控



# DHS系统

资源管理，集群整体的情况，方便分配资源

Didi Hbase Service

平台介绍使用帮助李扬

DHS管理端

资源管理

权限管理

流程审批

首页 / 资源管理

+ 添加集群

集群名称	集群类型	zk地址	group数量	rs使用量/总量	已使用量/总存储量(TB)	rs使用率	存储使用率	hbase版本	维修中rs数	监控表	操作
星光预发集群	预发							0.98.21-hadoop2-ddh-0.1.0	0		<a href="#">编辑</a> <a href="#">删除</a>
办公网测试	测试							0.98.21-hadoop2-ddh-0.1.0	0		<a href="#">编辑</a> <a href="#">删除</a>
星光存储集群	线上							0.98.20-hadoop2	0		<a href="#">编辑</a> <a href="#">删除</a>
广州存储集群	线上							0.98.21-hadoop2-ddh-0.1.0	0		<a href="#">编辑</a> <a href="#">删除</a>
广州计算集群	线上	...						0.98.20-hadoop2	1		<a href="#">编辑</a> <a href="#">删除</a>

# DHS系统

---

正在开发中的功能：

与自动化运维系统连接，自动化on-boarding的整个流程，包括建表，权限，资源分配，配置设置等

与监控系统连接，用户可以查看项目下的集群健康状况，访问情况

自动报警，按照用户指定的条件（如延时，存储等）报警

管理员配置项目health checker，按照设置条件和频率检测项目健康状况并做出反应

加入其它必要工具，如bulk load工具

# 资源隔离与分配

---

资源共享还是独占？

资源利用率和服务质量的矛盾

多租户共享资源

- 好处：资源利用率高，维护简单
- 坏处：用户竞争资源，难以发现问题

多租户独占资源

- 好处：资源冲突减少，可用性高，细粒度维护
- 坏处：业务低峰时段资源浪费，维护成本高



# 资源隔离与分配

---

共享与独占共存

按照业务的特性来选择不同方案

共享资源：

- 对访问延时要求低
- 访问量小
- 可用性要求低
- 备份或者测试阶段的数据

独占资源：

- 延时，吞吐要求高
- 高峰时段访问量大
- 可用性要求高
- 在线业务

# 资源分配的方法

---

## 需求分析

- 用户需要给出预估的表大小，访问方式和吞吐，表的属性等
- 需要给出均值和最大值，如果可能，给出未来几个季度的预计增长情况

## 上线流程

- 开发集群→测试集群→线上集群

## HBase Regionserver Group分配

- 按照需求和测试集群的状况，计算出所需的regionserver个数
- 通常会在额外给20%-30%的资源

## 定期报告和账单

- 每个月自动检测资源使用情况
- 计算开销发送给用户

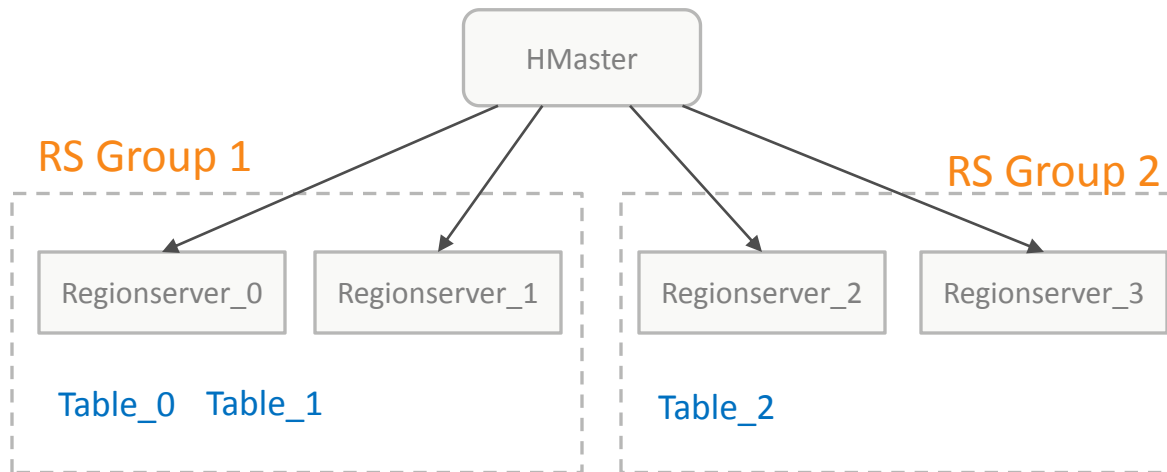
# HBase Regionserver Group

HBASE-6721: RegionServer Group based Assignment

通过namespace和RS group两个功能对资源和权限进行隔离

用户的一个或者多个table可以分配在指定的regionserver列表中，这个列表称为一个RS group。

一个HBase集群的计算资源被逻辑上分成了多个groups。对每个用户按需分配group。



# HBase Regionserver Group

---

使用RS group可以帮助我们：

1. 更容易的权限分配（通过绑定group的namespace权限）
2. 避免多用户资源争抢造成的不公平和性能问题
3. 可以对一个group进行单独的优化，hbase conf, gc等
4. 异构集群更好管理
5. 成本计算更加容易
6. 日常维护滚动升级可以增量进行，而且可以并行化

# 成本控制

---

资源分配考虑的因素：

- 表的总大小
- 读写吞吐
- 访问方式
- 存活时长
- 延时要求

资源计算的方式：

- 用户预估
- 测试环境评估
- 线上定期监控

保证灵活合理地分配资源

# 付费服务

---

服务不是免费的！

公司内部的一二级部门都会定期收到账单，按照使用的资源付费。

付费的原因：

- 降低公司的总成本
- 减少平台维护者的不必要工作
- 鼓励用户优化业务，用更少的资源做更多的事
- 各个部门资源使用情况透明化

# 付费服务

---

计费标准：

1. 存储使用
2. 计算使用

计费方法：

$$\text{Cost} = \text{TableSize} * x + \text{RSCount} * y$$

x: 每GB的费用

y: 每个regionserver的费用

x, y的值按照不同的配置有不同的标准

# 未来的工作

---

自动化运维工具

更完善的监控报警

异地双活

更好的周边服务管理，thrift/restful/queryserver

数据安全

更细粒度的共享资源计费方式



# THANK YOU


李扬 [albertlee166@gmail.com](mailto:albertlee166@gmail.com)



北京滴滴无限科技发展有限公司

北京市海淀区东北旺路8号院尚东·数字山谷B1号楼



期待、现在 

Haidian, Beijing



Scan the QR code to add me on WeChat

