网易大数据分析平台建设与实践

冯宇 @ 网易

网易大数据
Netease Big Data

Mammut

自我介绍

- 冯宇
- From 网易杭州研究院
- Apache Kylin Committer
- 目前主要关注大数据架构，OLAP查询引擎优化以及平台化服务等领域

大纲

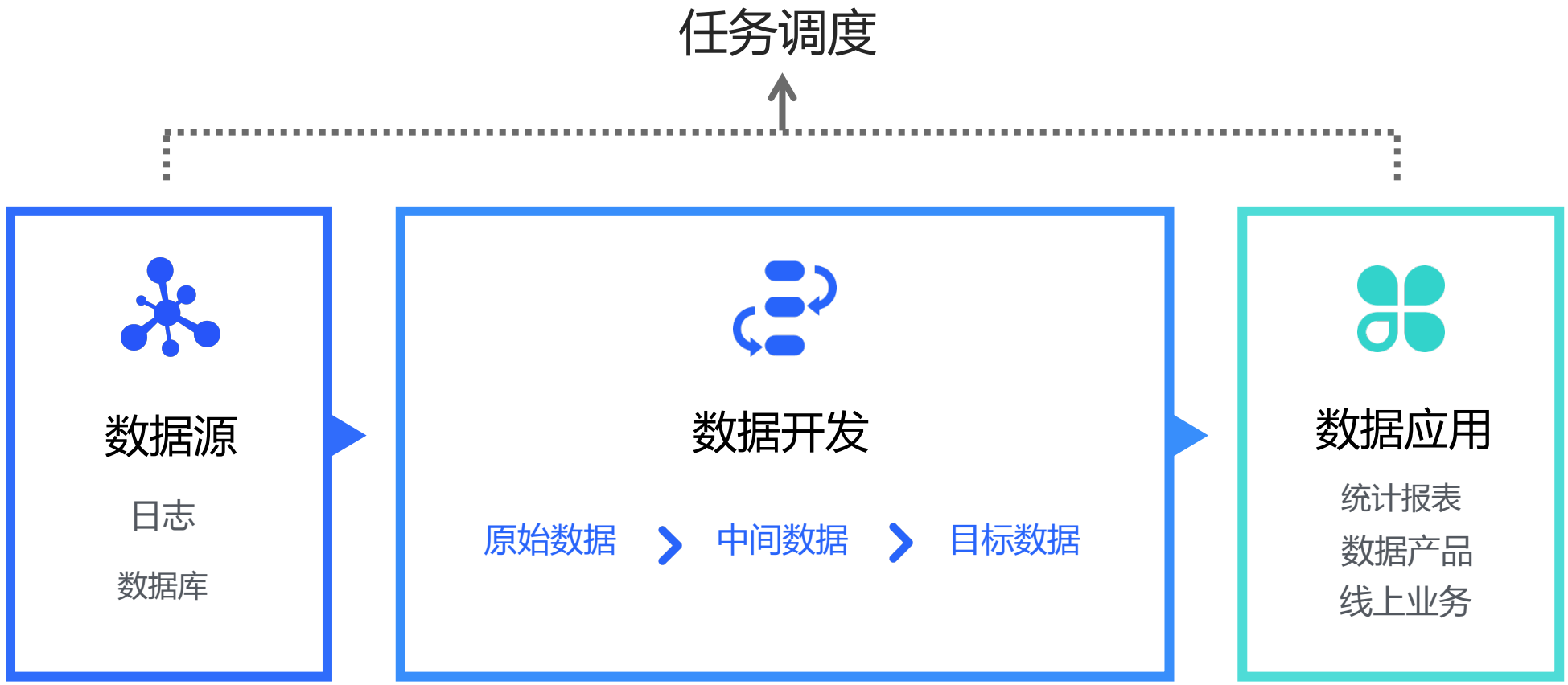
01 网易大数据平台猛犸介绍

02 Apache Kylin在网易的实践

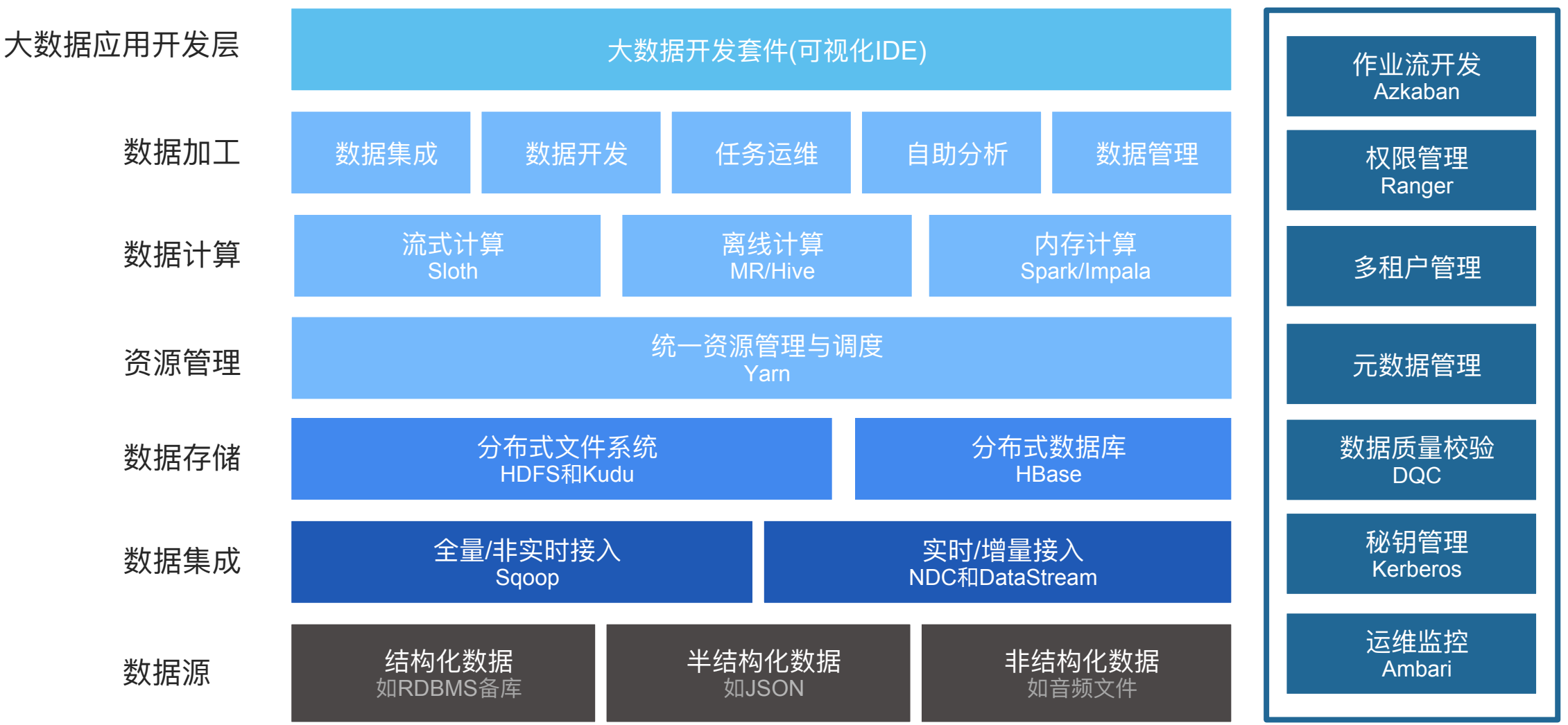
03 Cloudera Impala在网易的实践

04 Q&A

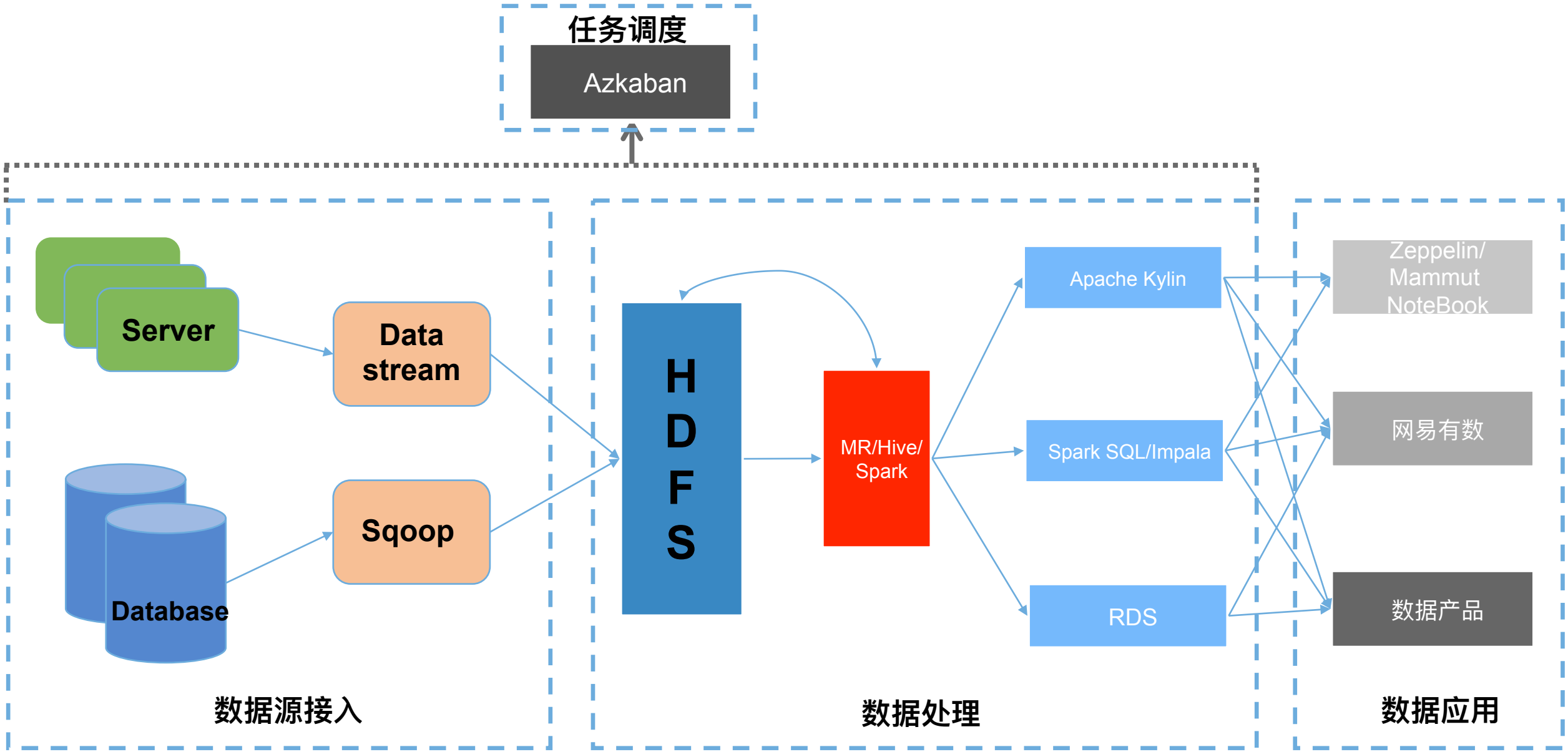
需求



猛犸：一站式大数据开发平台

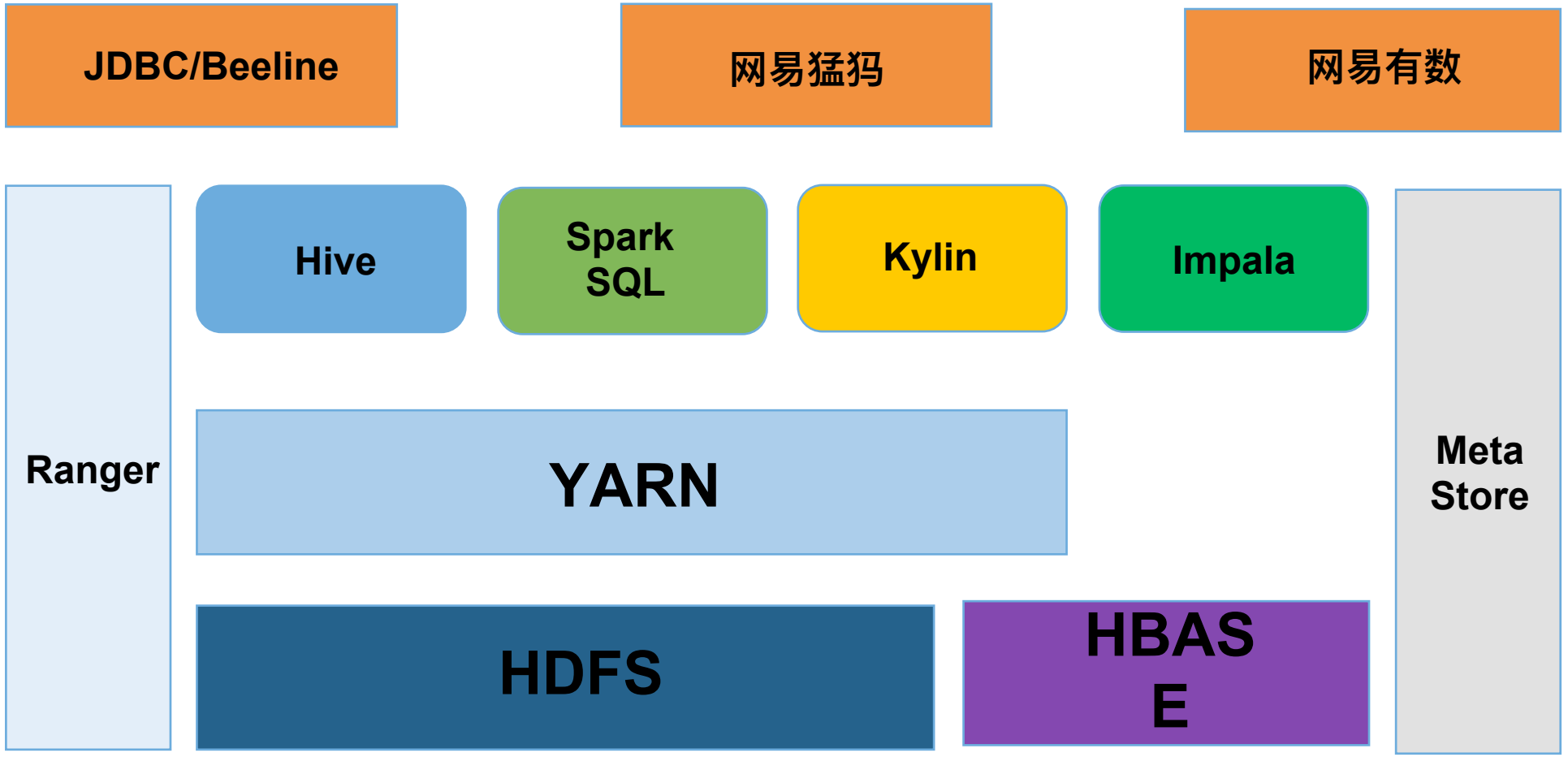


猛犸：离线数据处理


任务调度
Azkaban
Server
Data stream
Database
Sqoop
HDFS
MR/Hive/ Spark
Apache Kylin
Spark SQL/Impala
RDS
Zeppelin/ Mammut NoteBook
网易有数
数据产品
数据源接入
数据处理
数据应用

大数据离线分析平台

网易云音乐　网易云课堂 study.163.com　网易理财 8.163.com　网易考拉海购　易信



网易猛犸数据查询界面

一份数据，多套引擎！

大纲

01 网易大数据平台猛犸介绍

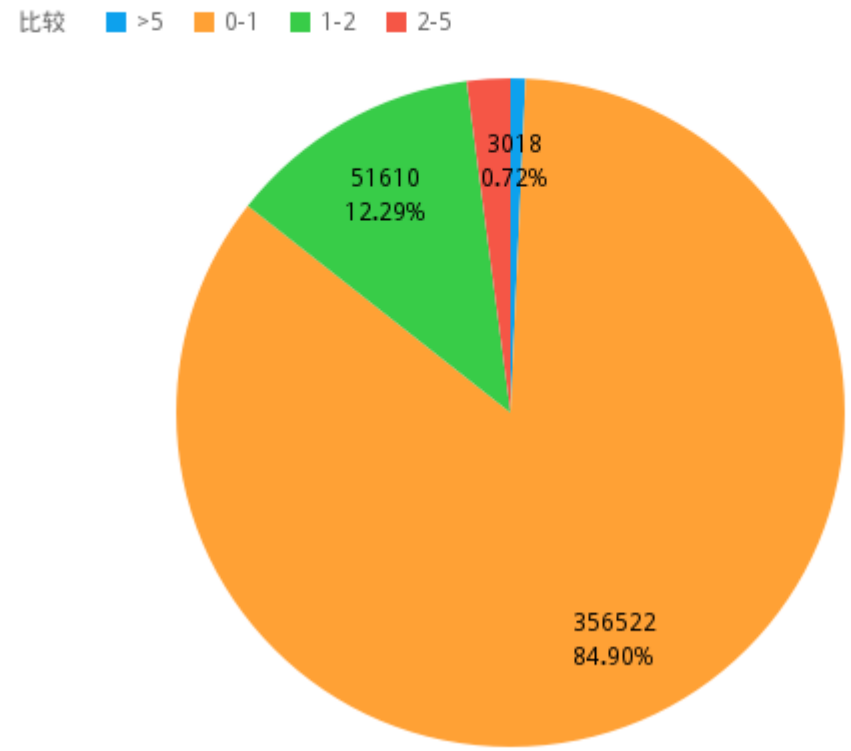
02 Apache Kylin在网易的实践

03 Cloudera Impala在网易的实践

04 Q&A

Apache Kylin在网易

查询时间分布图



累计Cube数

57

Cube总大小

7,448.9KM

输入总记录数

18,776亿

为什么选择Kylin

性能非常好，比SQL on Hadoop引擎支持更高的并发

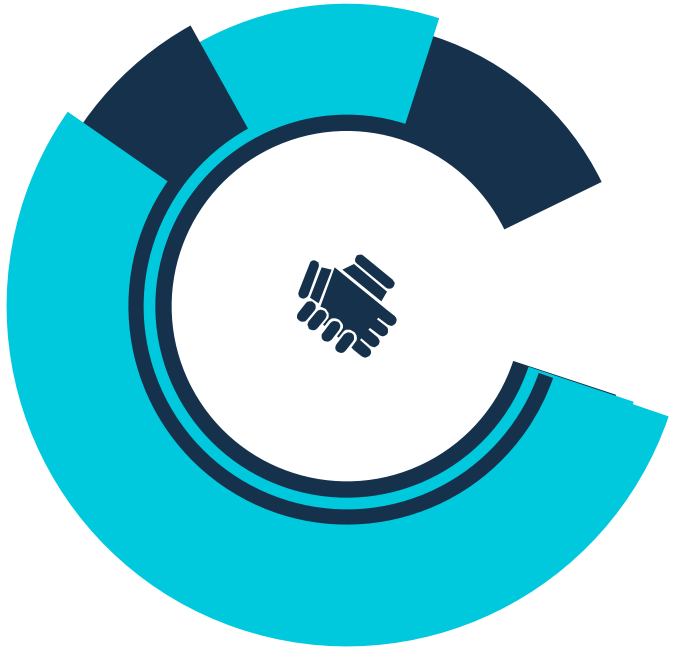
容易使用，开发人员不需要任何编写任何作业和代码

源码结构好，能够被开发人员掌握

支持SQL语法，兼容现有的BI工具，社区活跃，发展迅速

哪些需求适合Kylin

- 数据量，Kylin是面向大数据量的SQL引擎，数据量大，维度少的查询需求是最理想的
- 需求明确，不宜变动，修改Cube定义代价比较大
- 数据延迟，主要处理T+1的数据
- 数据修改，有些数据可能需要修改，需要考虑重新Build的代价



Kylin改造——多Hive数据源支持

问题
PROBLEM

公司内部没有统一元数据服务，每一个产品都有自己的Hive Metastore，使用不同的Hadoop集群

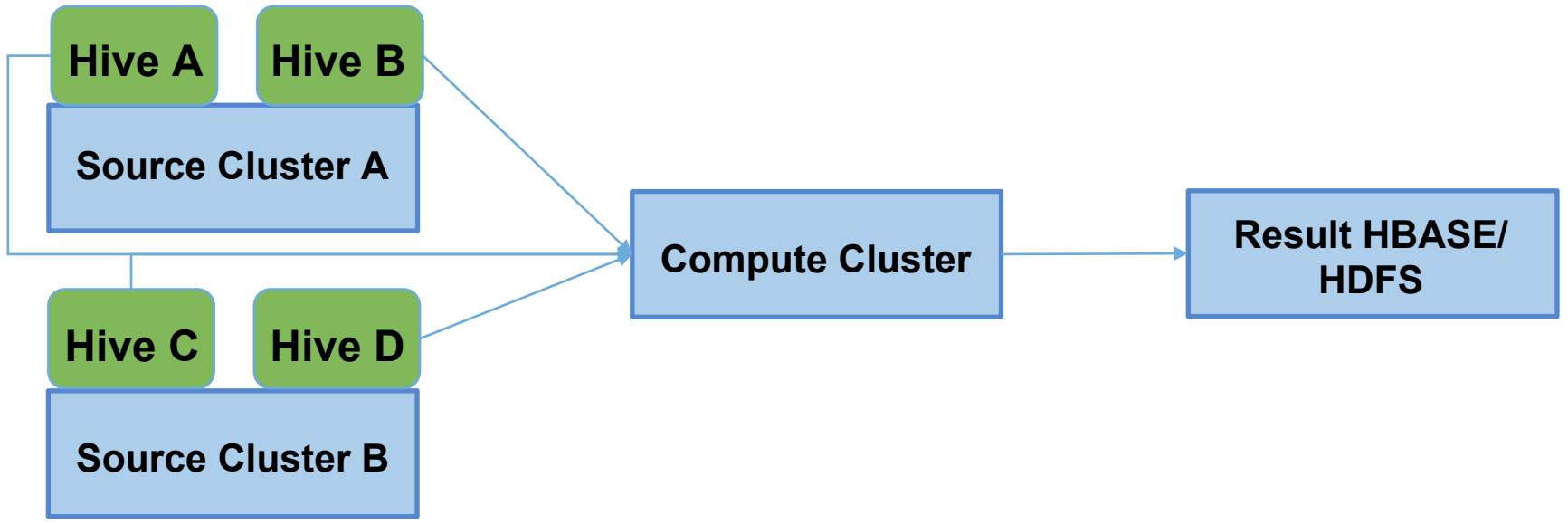
Kylin只支持一个Hive Metastore和一套Hadoop集群

为如果每一个Hive Metastore部署一套集群，维护代价太高

每一个Kylin服务需要一个Hbase作为存储，资源浪费，无法集中管理

Kylin改造——多Hive数据源支持

- 受益于Kylin插件式架构，将不同的Hive作为一种独立数据源，方便维护
- 增加数据拷贝任务



Kylin改造——结果缓存

Kylin中历史数据很少改变，对查询结果的缓存意义较大

Kylin基于ehcache的内部缓存，但是大小有限制，无法跨节点共享

改造
REMAKE

查询结果缓存至Hbase，集群共享。

惰性失效，根据缓存写入时间和查询涉及的Segment最后build时间判断是否有效。

Kylin改造——全局维度表

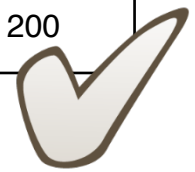
ID	NAME
1	A1
2	B1
3	C1

ID	NAME
1	A2
2	B2
3	C1

```
SELECT NAME ,
COUNT(1) FROM FACT
JOIN DIM1 GROUP BY
NAME ORDER BY
NAME;
```

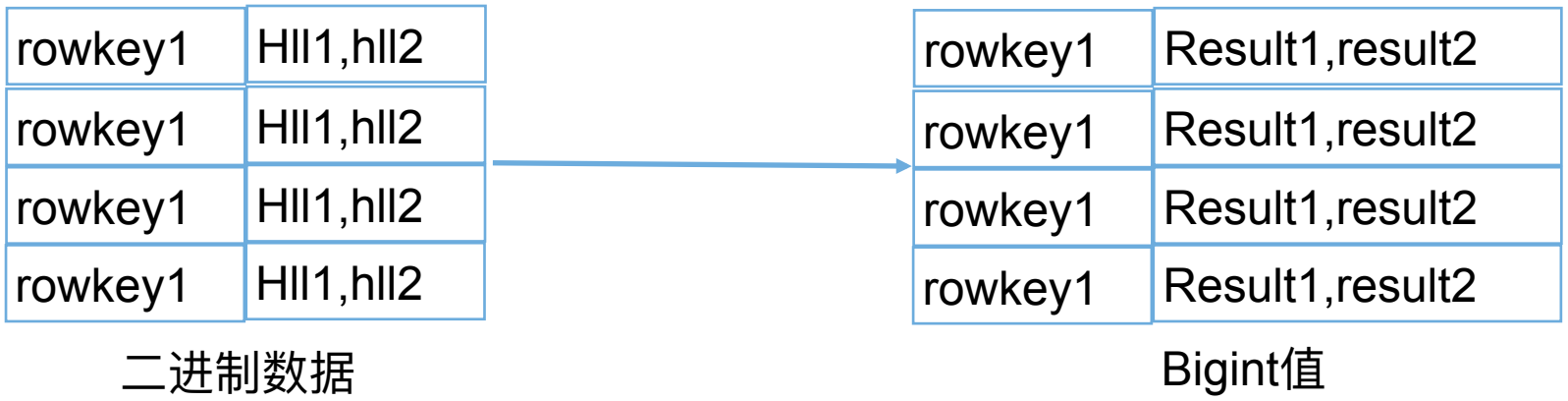
NAME	COUNT(1)
A1	100
A2	105
B1	90
B2	150
C1	200

NAME	COUNT(1)
A2	205
B2	240
C1	200



Kylin性能优化案例

- 做更多的计算，不进行在线聚合运算
- 查询时直接从Hbase中读取结果，不执行aggregate/merge
- 牺牲通用性，大大得提升查询性能和数据存储



大纲

01 网易大数据平台猛犸介绍

02 Apache Kylin在网易的实践

03 Cloudera Impala在网易的实践

04 Q&A

为什么选择Impala

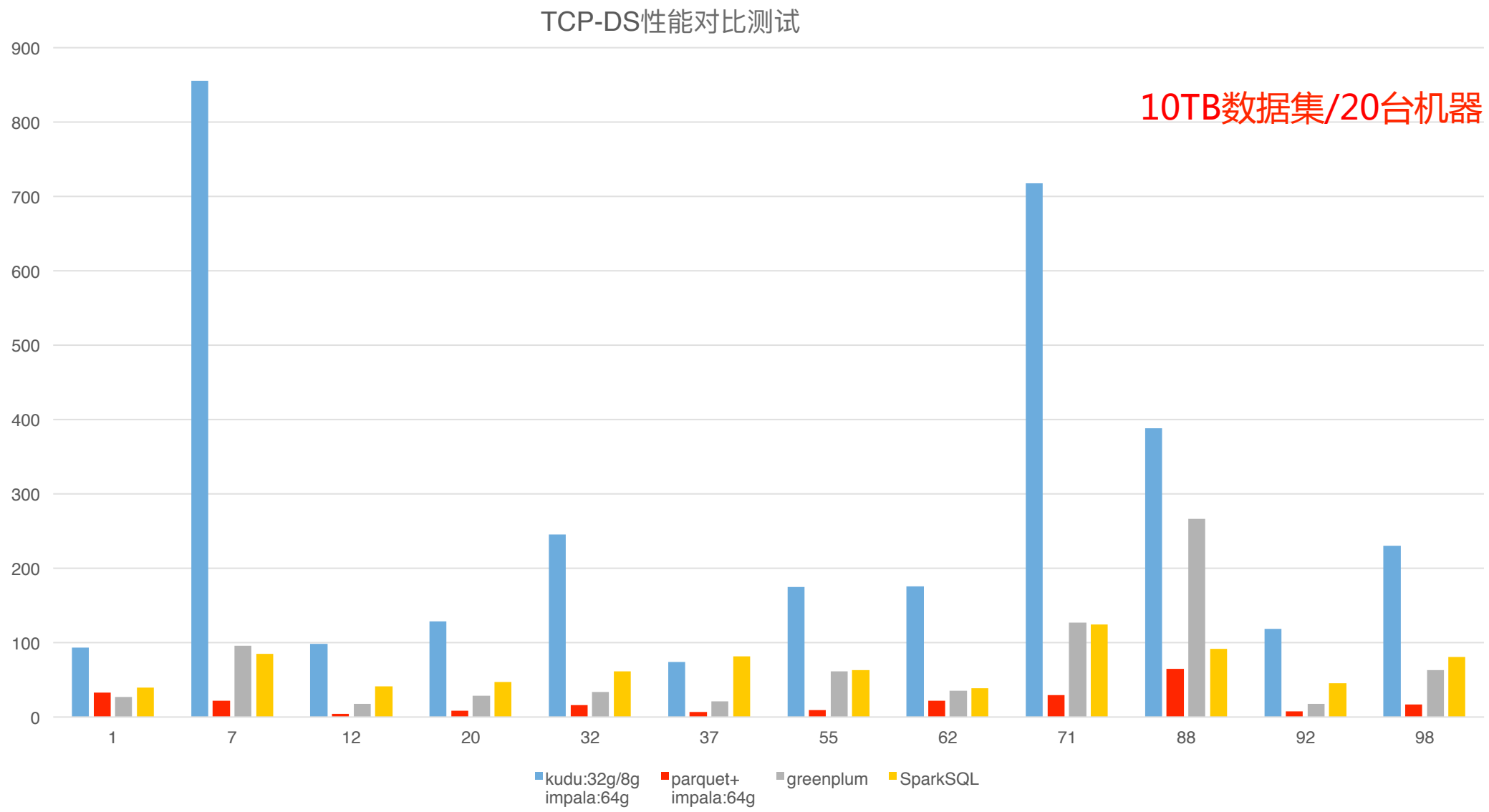
较之于Hive/Spark SQL等SQL on Hadoop引擎，性能有几倍的提升

兼容Hive元数据库，接入简单

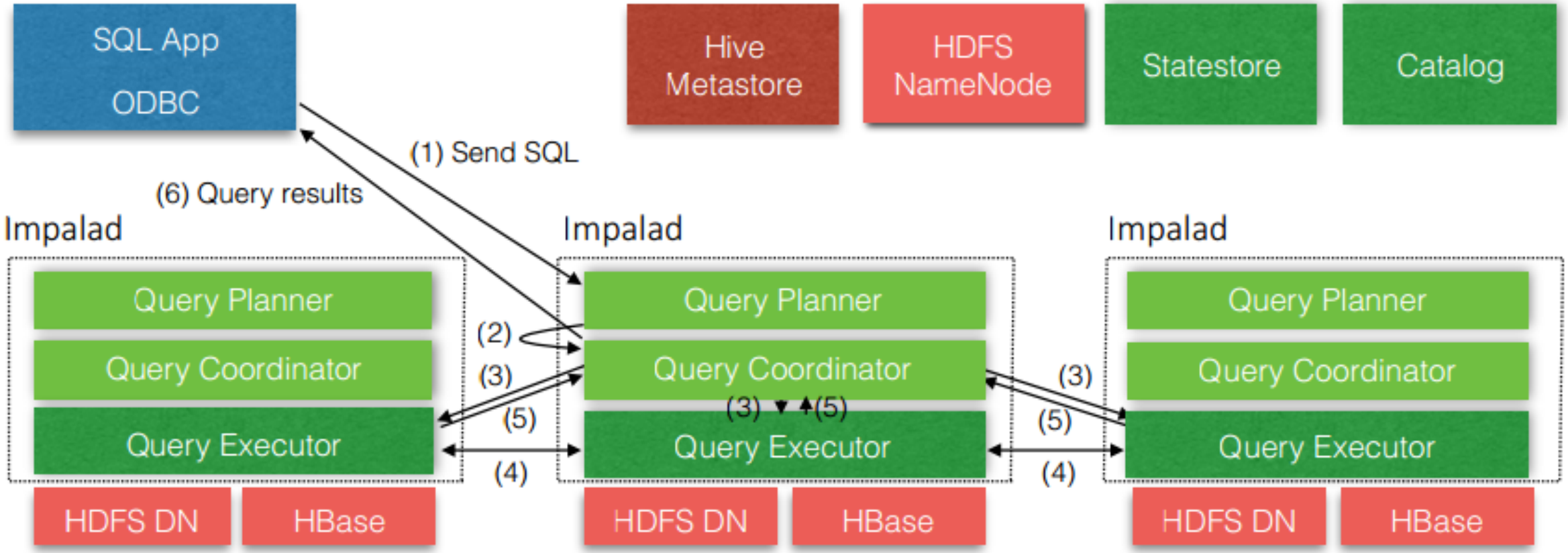
较之于GreenPlum，更好的融合Hadoop生态圈

经过测试，无论是系统稳定性和扩展能力impala表现的都比较好

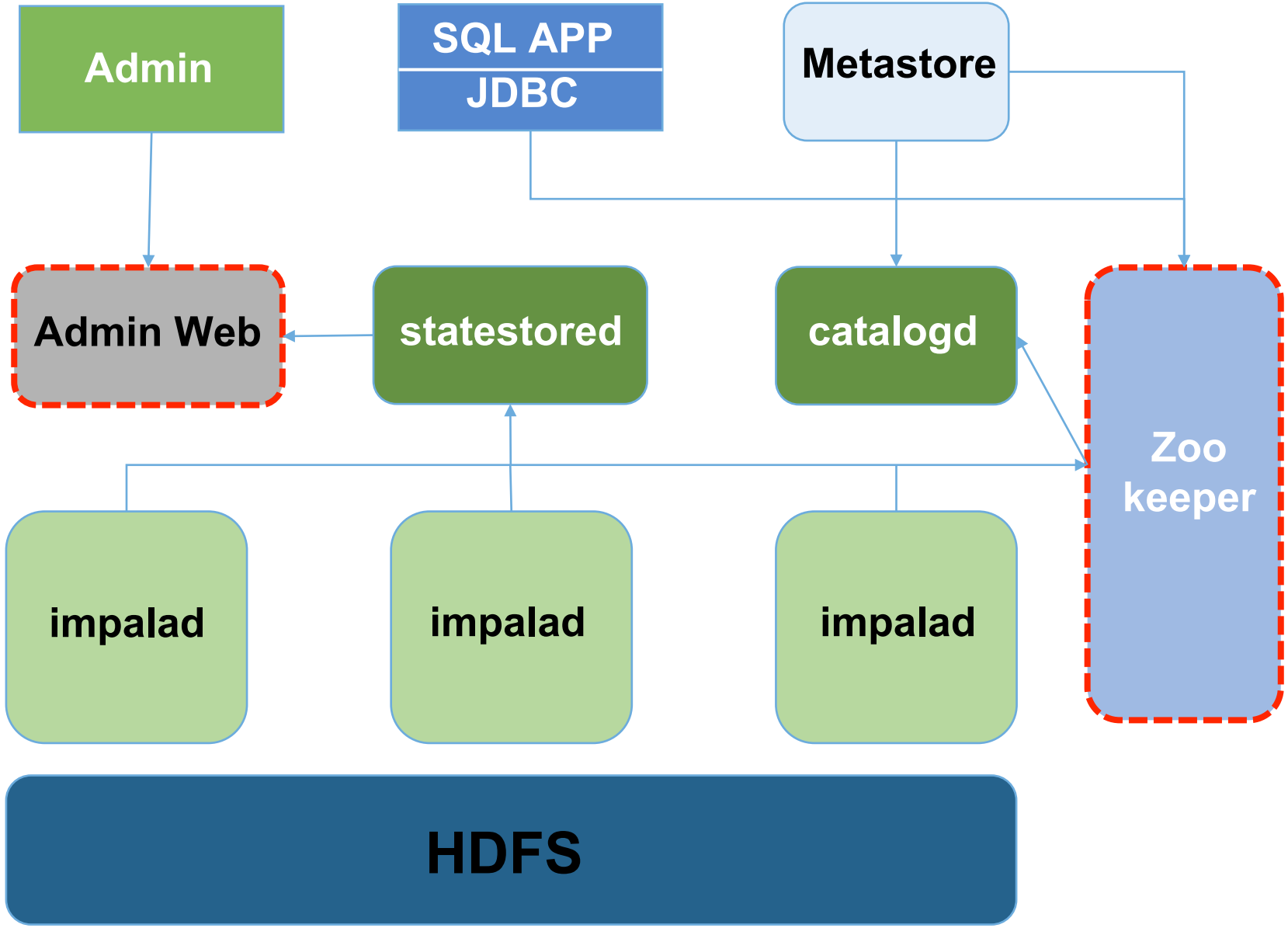
Impala性能对比测试



Impala架构


SQL App
ODBC
Hive Metastore
HDFS NameNode
Statestore
Catalog
(1) Send SQL
(6) Query results
Impalad
Query Planner
Query Coordinator
Query Executor
HDFS DN
HBase
(2)
(3)
(5)
(4)
Impalad
Query Planner
Query Coordinator
(3) (5)
Query Executor
HDFS DN
HBase
(3)
(5)
(4)
Impalad
Query Planner
Query Coordinator
Query Executor
HDFS DN
HBase

Impala架构改造


Admin
SQL APP
JDBC
Metastore
Admin Web
statestored
catalogd
Zoo
keeper
impalad
impalad
impalad
HDFS

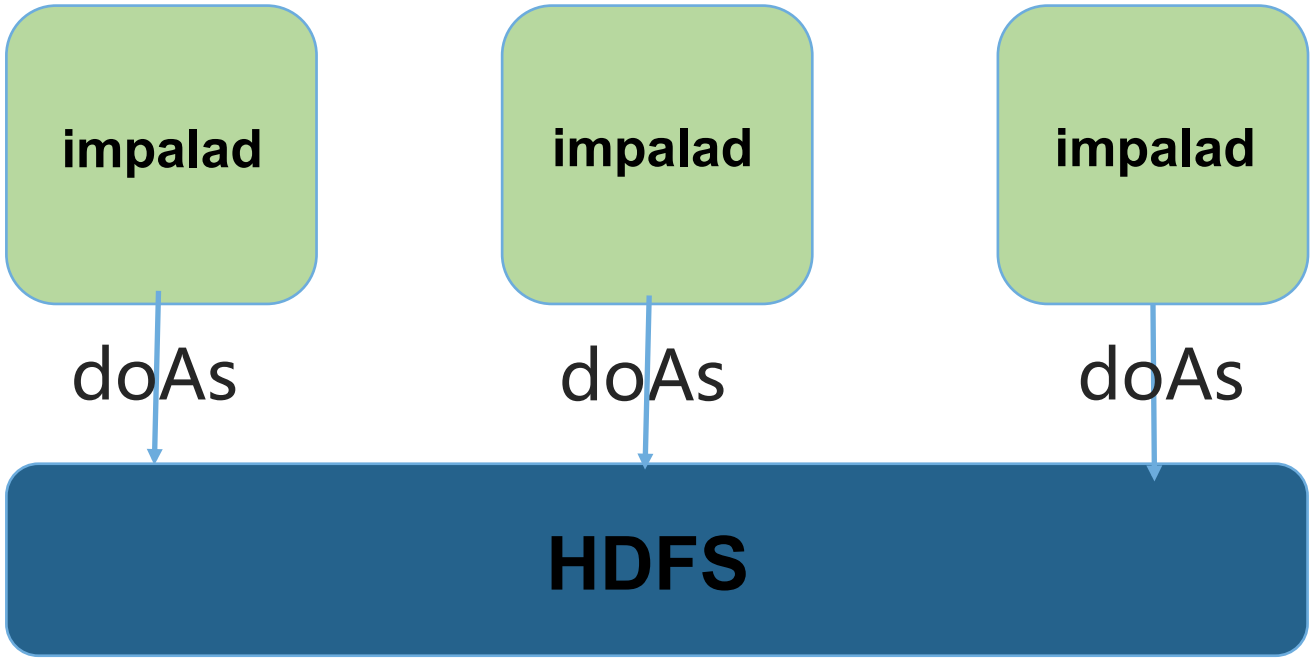
Impala平台化改造——用户权限隔离

问题
PROBLEM

- Kerberos环境下Impala使用配置的用户执行所有数据和元数据操作
- Hadoop管理员出于安全考虑不会授予Impala超级账号
- 即使有超级账号，所有写入的数据用户都没有权限查看
- 无法和Hive/Spark SQL共享同一套数据

Impala平台化改造——用户权限隔离

- Impalad和Catalogd端使用doAs的方式执行数据和元数据操作
- 修改hadoop-common代码，需要维护特定版本



Impala平台化改造——元数据同步

Impala启动时缓存全部元数据，其它SQL引擎的写操作无法同步

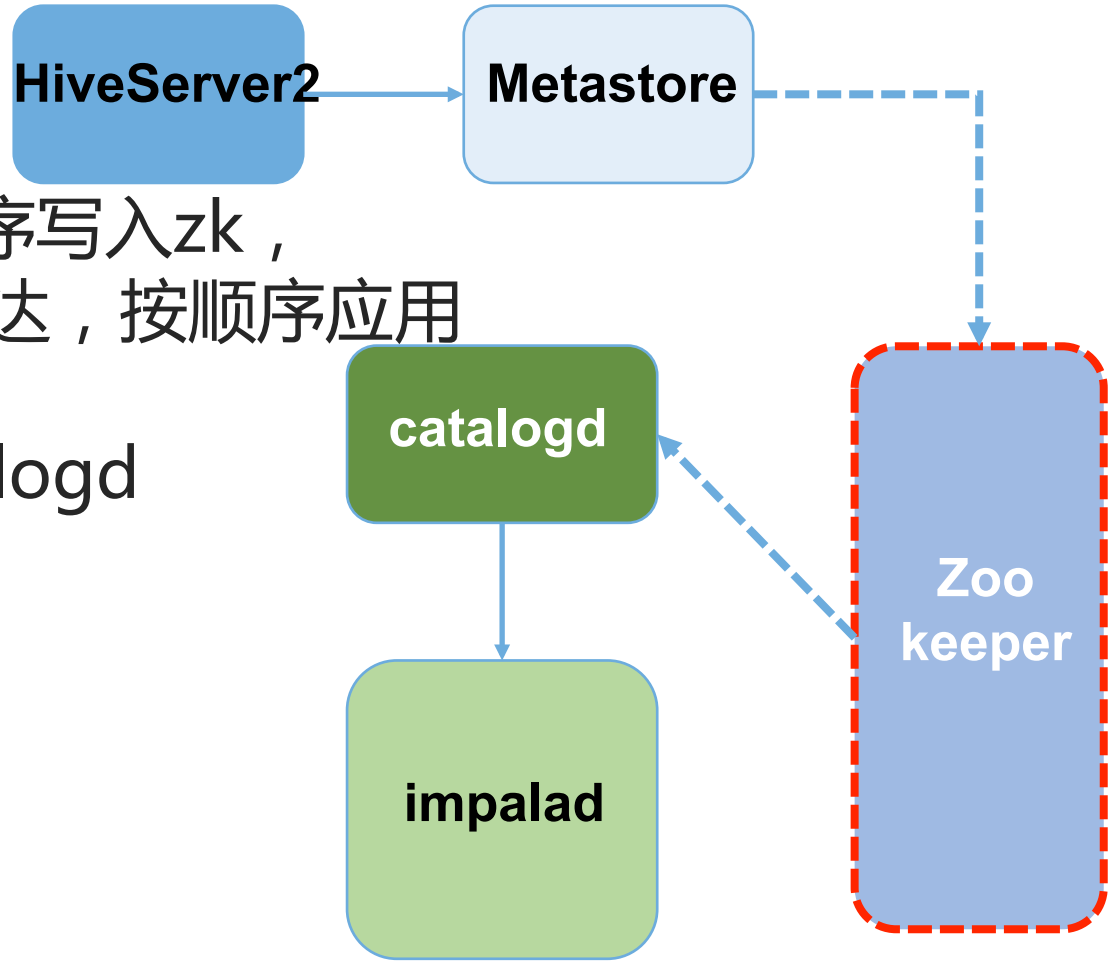
问题
PROBLEM

官方建议在执行所有写操作之后执行Invalidate metadata同步，低效并且不友好

> In Impala 1.2 and higher, a dedicated daemon (`catalogd`) broadcasts DDL changes made through Impala to all Impala nodes. Formerly, after you created a database or table while connected to one Impala node, you needed to issue an `INVALIDATE METADATA` statement on another Impala node before accessing the new database or table from the other node. Now, newly created or altered objects are picked up automatically by all Impala nodes. You must still use the `INVALIDATE METADATA` technique after creating or altering objects through Hive. See The Impala Catalog Service for more information on the catalog service.

Impala平台化改造——元数据同步

- 通过Zookeeper实现消息队列，
  Metastore修改时按照版本递增的顺序写入zk，
  Catalogd注册watcher监听新数据到达，按顺序应用
- 出现错误时重试，甚至报警重启Catalogd



Impala平台化改造——高可用

问题
PROBLEM

任何一个impalad节点都可以接收查询请求并执行

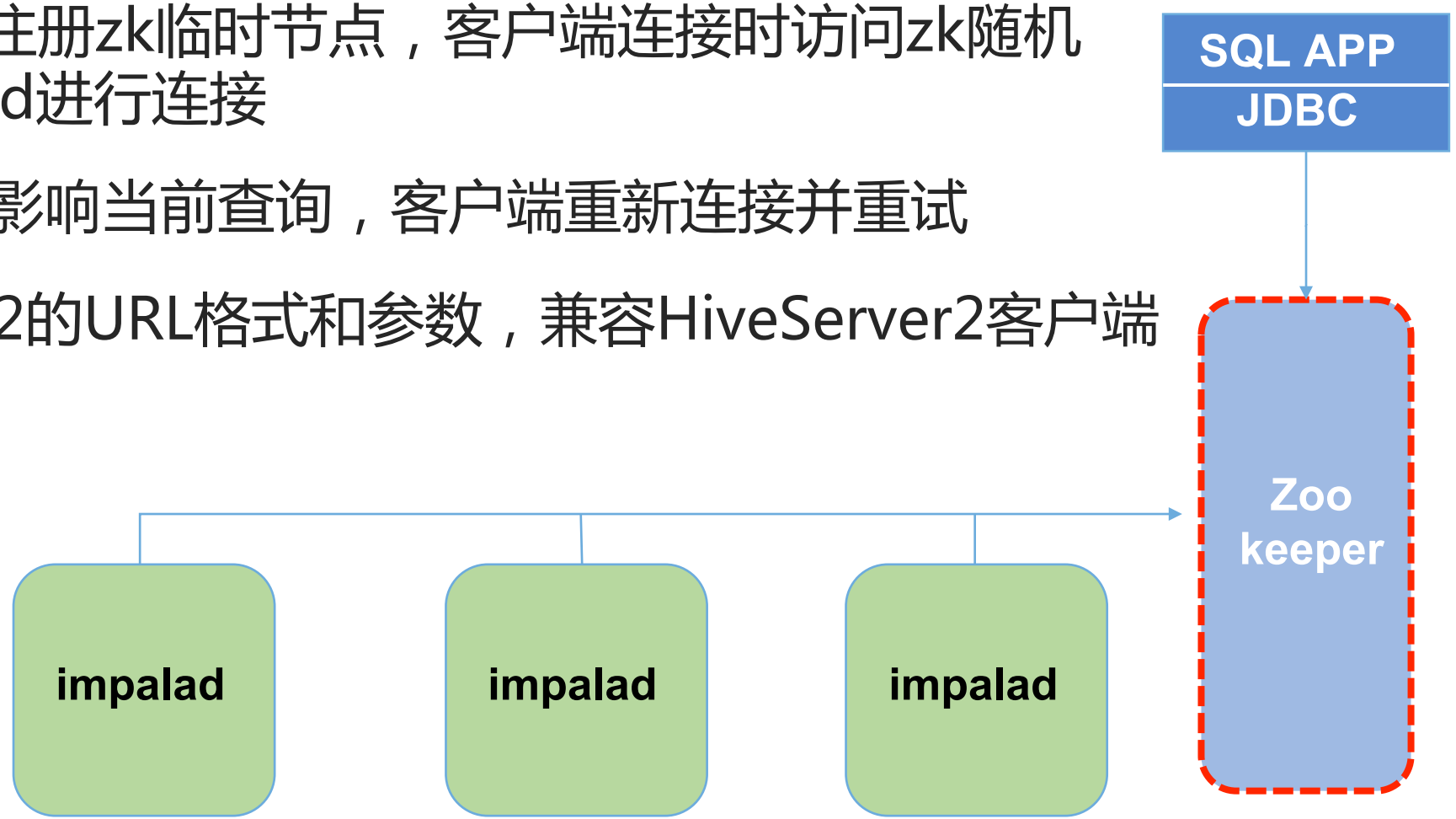
单个impalad节点出现故障只会影响当前执行的查询

为了避免单点和负载均衡，需要一种负载均衡方案提交任务到任意节点

官方推荐Haproxy+Keeplived方案比较复杂，增加组件依赖

Impala平台化改造——高可用

- Impalad启动时注册zk临时节点，客户端连接时访问zk随机选择一个impalad进行连接
- 单个节点挂掉只影响当前查询，客户端重新连接并重试
- 兼容HiveServer2的URL格式和参数，兼容HiveServer2客户端



Impala平台化改造——查询管理系统

问题
PROBLEM

每一个Impalad提供它所执行的SQL的执行状态，资源消耗等查询

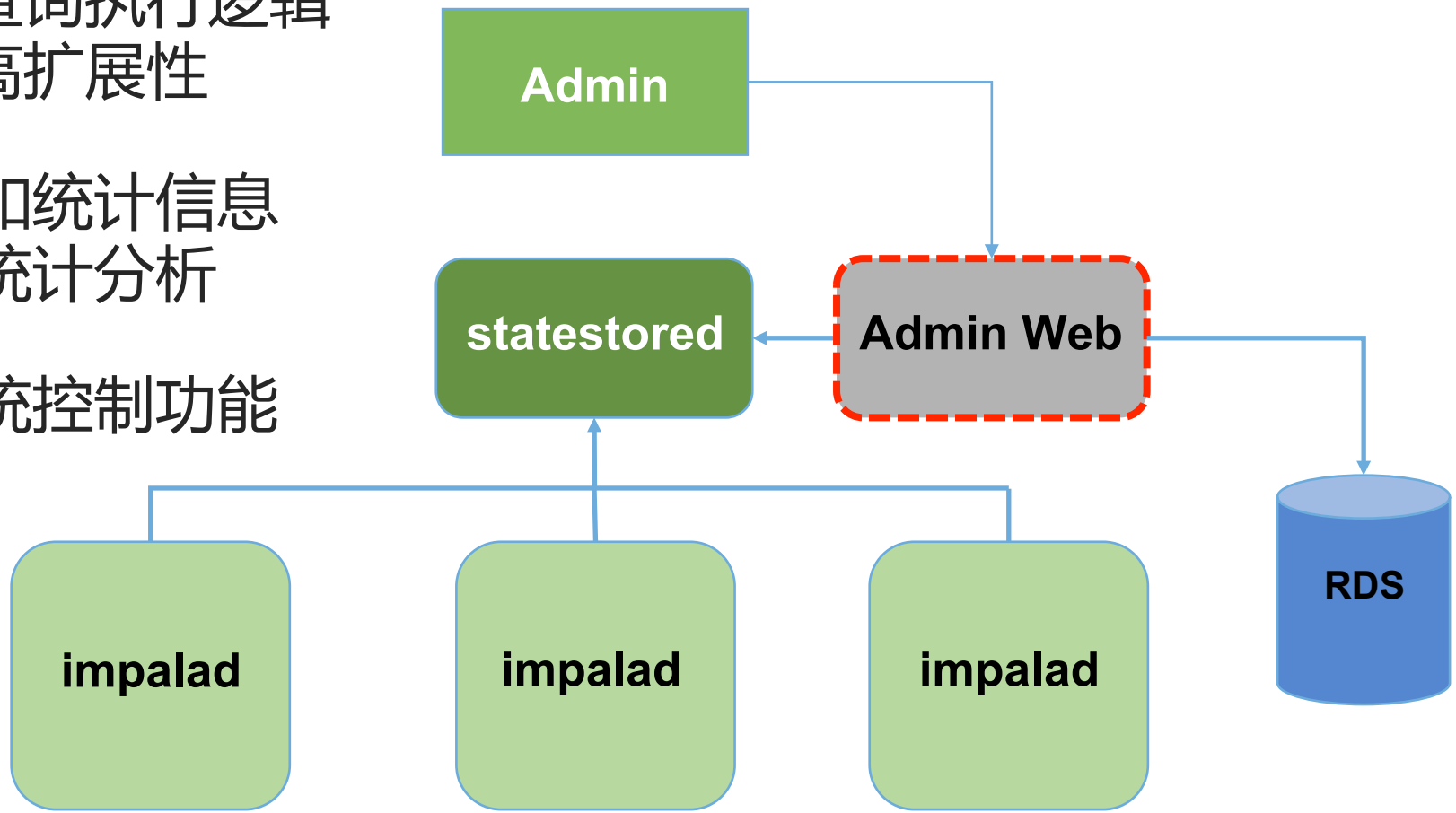
该信息非常重要，但是impalad没有持久化，并且只保留最近N条（可配置）

Impalad高可用之后任何一个impalad都散布着查询，无法集中管理

由于impalad的分散，一些管理操作的实现都需要侵入impalad逻辑

Impala平台化改造——查询管理系统

- 基于现有impala架构，通过statestored服务实现，有延迟（1-4s）
- 几乎不侵入impalad查询执行逻辑
  与impalad无耦合，高扩展性
- 持久化存储全部查询和统计信息
  方便后期排查问题和统计分析
- 未来方便实现一些系统控制功能



Impala的坑——count distinct

```
select province,
count(distinct accountid)
from tbl where day =
'2017-03-01' group by
province
```

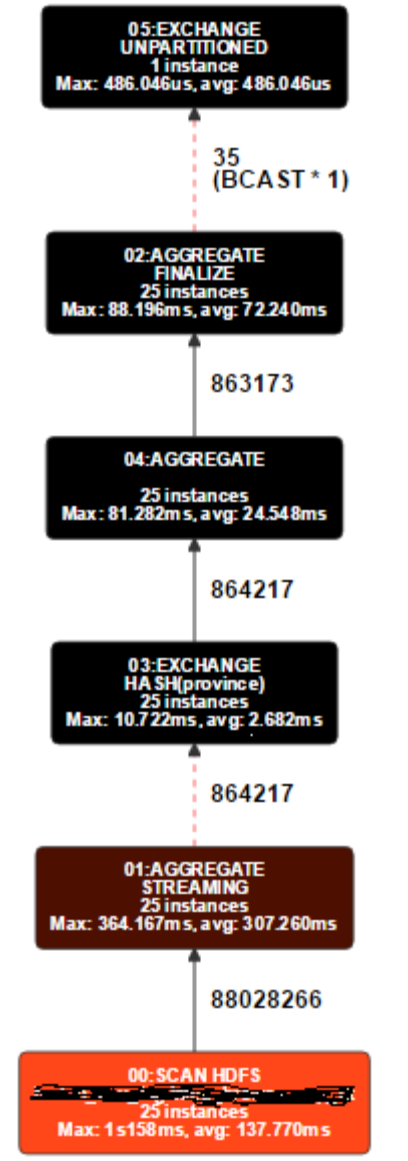
=

```
select province,
count(accountid) from (select
province, accountid from tbl
where day = '2017-03-01'
group by province, accountid)
as a group by province
```

- 查询性能非常好，并发度高
- 不支持一条SQL多个count distinct，通过union或者join方式改变
- 通过UDAF实现多个版本的count distinct

To produce the same result as multiple COUNT(DISTINCT) expressions, you can use the following technique for queries involving a single table:

```
select v1.c1 result1, v2.c1 result2 from
  (select count(distinct col1) as c1 from t1) v1
    cross join
  (select count(distinct col2) as c1 from t1) v2;
```



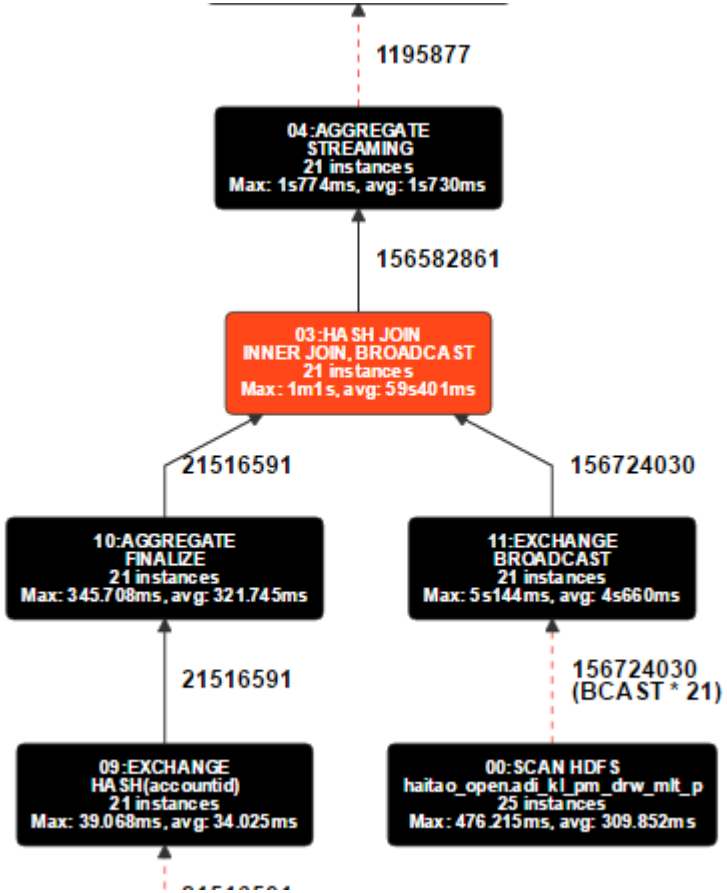
Impala的坑——资源隔离

Exec Summary

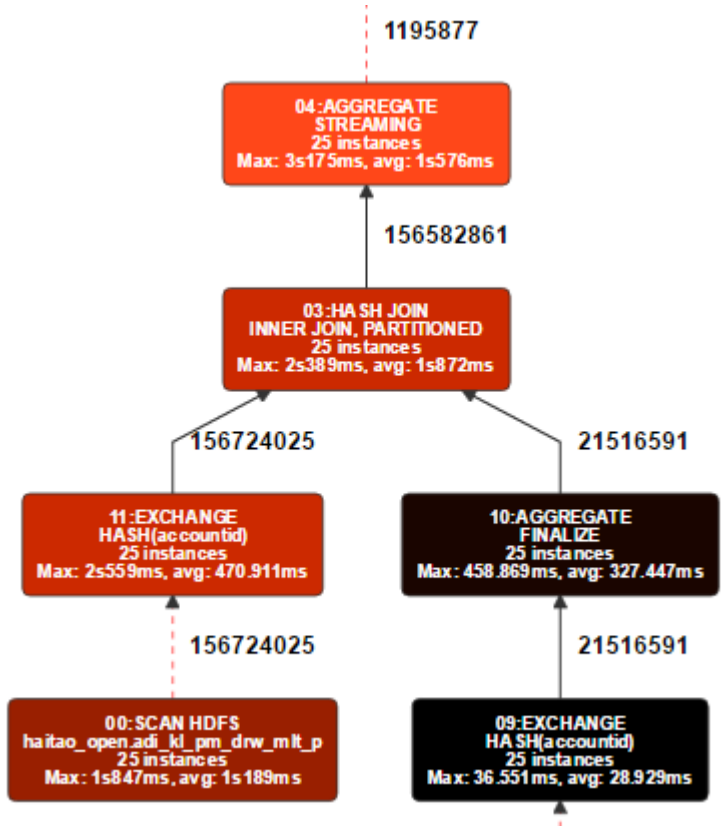
Operator	#Hosts	Avg Time	Max Time	#Rows	Est. #Rows	Peak Mem	Est. Peak Mem	Detail
31:EXCHANGE	0	0.000ns	0.000ns	0	77.66K	0	-1.00 B	UNPARTITIONED
19:HASH JOIN	25	151.760ms	232.796ms	0	77.66K	69.12 KB	20.66 MB	INNER JOIN, BROADCAST
|--30:EXCHANGE	25	0.000ns	0.000ns	0	77.66K	0	0	BROADCAST
| 18:HASH JOIN	25	0.000ns	0.000ns	0	77.66K	78.05 GB	35.25 MB	INNER JOIN, PARTITIONED
| |--17:NESTED LOOP JOIN	25	9m5s	9m34s	337.29B	4.83M	240.70 MB	464.65 MB	LEFT OUTER JOIN, BROADCAST
| | |--28:EXCHANGE	25	441.070ms	532.839ms	3.85M	9.56M	0	0	BROADCAST
| | | 27:AGGREGATE	25	811.498ms	862.117ms	3.85M	9.56M	14.23 MB	20.44 MB	FINALIZE
| | | 26:EXCHANGE	25	137.637ms	146.831ms	79.04M	9.56M	0	0	HASH(account_id)
| | | 04:AGGREGATE	25	13s464ms	13s936ms	79.04M	9.56M	410.04 MB	511.12 MB	STREAMING
| | | 03:SCAN HDFS	25	938.625ms	11s992ms	425.92M	1.59B	1.12 GB	352.00 MB	haitao.tb_coupon
| | 16:HASH JOIN	25	41.926ms	73.588ms	96.87K	4.83M	155.04 MB	14.59 MB	RIGHT OUTER JOIN, PARTITIONED
| | |--25:EXCHANGE	25	19.839ms	24.923ms	7.36M	4.83M	0	0	HASH(haitao.tb_coupon.accou...
| | | 00:SCAN HDFS	25	3s762ms	18s174ms	7.36M	4.83M	1.46 GB	1.29 GB	haitao.tb_coupon
| | 24:AGGREGATE	25	2s308ms	2s444ms	229.38K	9.56M	154.60 MB	20.44 MB	FINALIZE
| | 23:EXCHANGE	25	401.169ms	423.593ms	241.71M	9.56M	0	0	HASH(account_id)
| | 02:AGGREGATE	25	42s277ms	44s710ms	241.71M	9.56M	920.74 MB	511.12 MB	STREAMING
| | 01:SCAN HDFS	25	1s671ms	1s826ms	5.02B	15.93B	465.41 MB	176.00 MB	haitao.tb_coupon
| 29:EXCHANGE	25	303.468us	479.700us	0	17.53M	0	0	HASH(account_id)
| 22:AGGREGATE	25	975.842ms	1s088ms	5.50M	17.53M	146.38 MB	41.65 MB	FINALIZE
| 21:EXCHANGE	25	93.160ms	113.997ms	32.40M	17.53M	0	0	HASH(account_id,cat1_id)
| 13:AGGREGATE	25	1s843ms	3s517ms	32.40M	17.53M	340.00 MB	1.43 GB	STREAMING
| 12:HASH JOIN	25	1s349ms	2s526ms	79.77M	17.53M	160.00 MB	9.73 MB	INNER JOIN, BROADCAST
| |--20:EXCHANGE	25	7.431ms	21.861ms	221.45K	221.45K	0	0	BROADCAST
| | 11:SCAN HDFS	1	327.956ms	327.956ms	221.45K	221.45K	15.06 MB	32.00 MB	haitao_db_dump.curr_dimen_c...
| 10:SCAN HDFS	25	184.865ms	304.541ms	80.29M	80.31M	178.92 MB	176.00 MB	haitao_open.adi_kl_usr_beha...
06:SUBPLAN	25	0.000ns	0.000ns	0	235.16M	8.00 KB	0	
|--09:NESTED LOOP JOIN	25	102.324us	180.146us	0	10	16.00 KB	122.00 B	CROSS JOIN
| |--07:SINGULAR ROW SRC	25	0.000ns	0.000ns	0	1	0	0	
| 08:UNNEST	25	17.920us	36.254us	0	10	0	0	kaola_dmp.crm_user_metrics_...
05:SCAN HDFS	25	166.013ms	323.815ms	0	23.52M	0	440.00 MB	kaola_dmp.crm_user_metrics_...

设置mem_limit参数

Impala的坑——表统计信息



COMPUTE STATS
xxx



Cost : 80s

Cost : 8s