SELECTDB | 阿里云

Apache Doris 城市行 2024

Apache Doris × 阿里云联合 Meetup

10月26日（周六） 13:30-17:15

LNZI 霖梓控股

浙江霖梓控股基于 Apache Doris 的数仓实践

崔麒升

BI中心大数据架构师

LNZI 霖梓控股

崔 麒 升 | BI 中心大数据架构师

浙江霖梓控股有限公司

个人简介

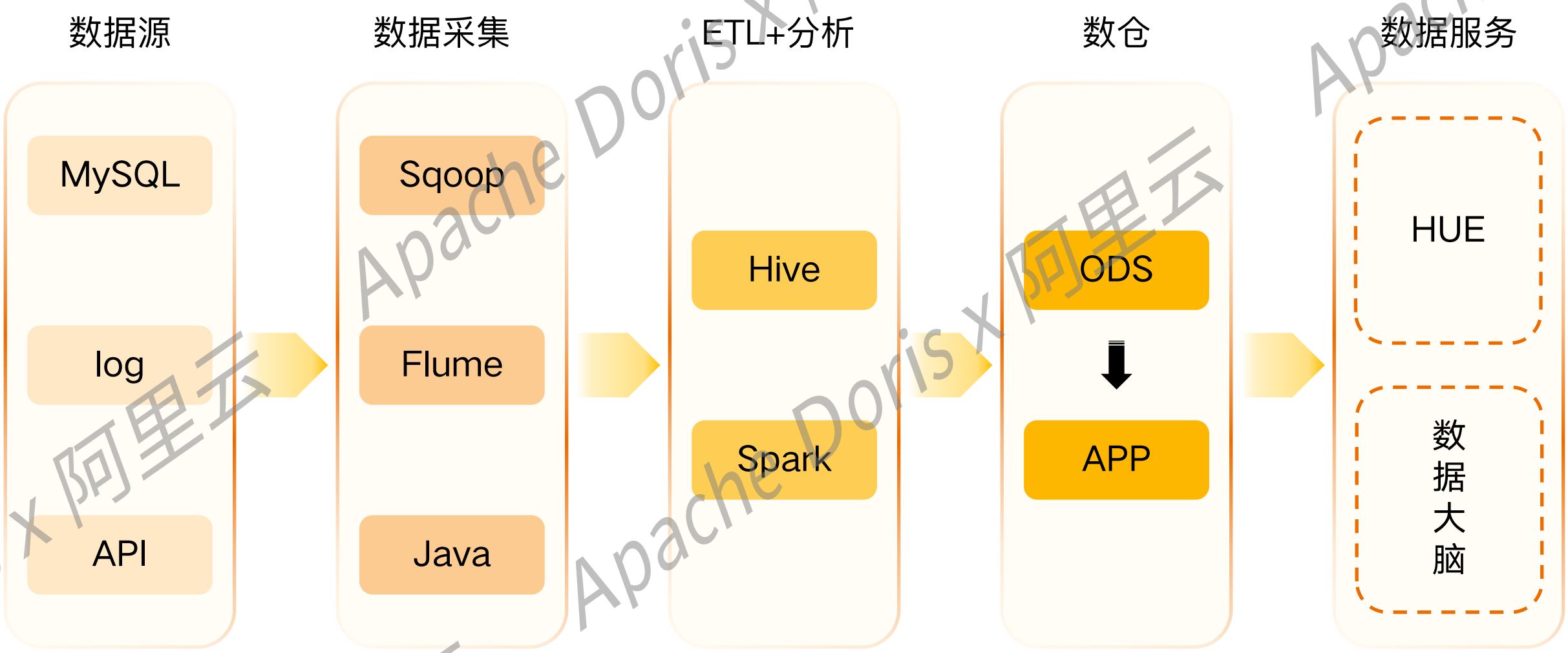
- 从事于 Java、大数据离线以及实时，负责公司大数据方向。熟悉数据湖，以及 AI 领域 Transformer、LOLO、UNet、CLIP、DALL-E 等算法，了解多模态、对抗生成网络架构思想。

LNZI 霖梓控股

CONTENTS

目录

重构前大数据架构



改造前存在问题

01 数据采集性能差

MySQL->ODS 使用 Sqoop 每日所需时长 6H

02 数据变更能力差

无分区表，不支持数据的灵活变更
实现过程对性能消耗较大

03 数据分析性能差

烟囱式开发，使用不规范
计算冗余度大，没有复用性

04 离线任务跑批时间长

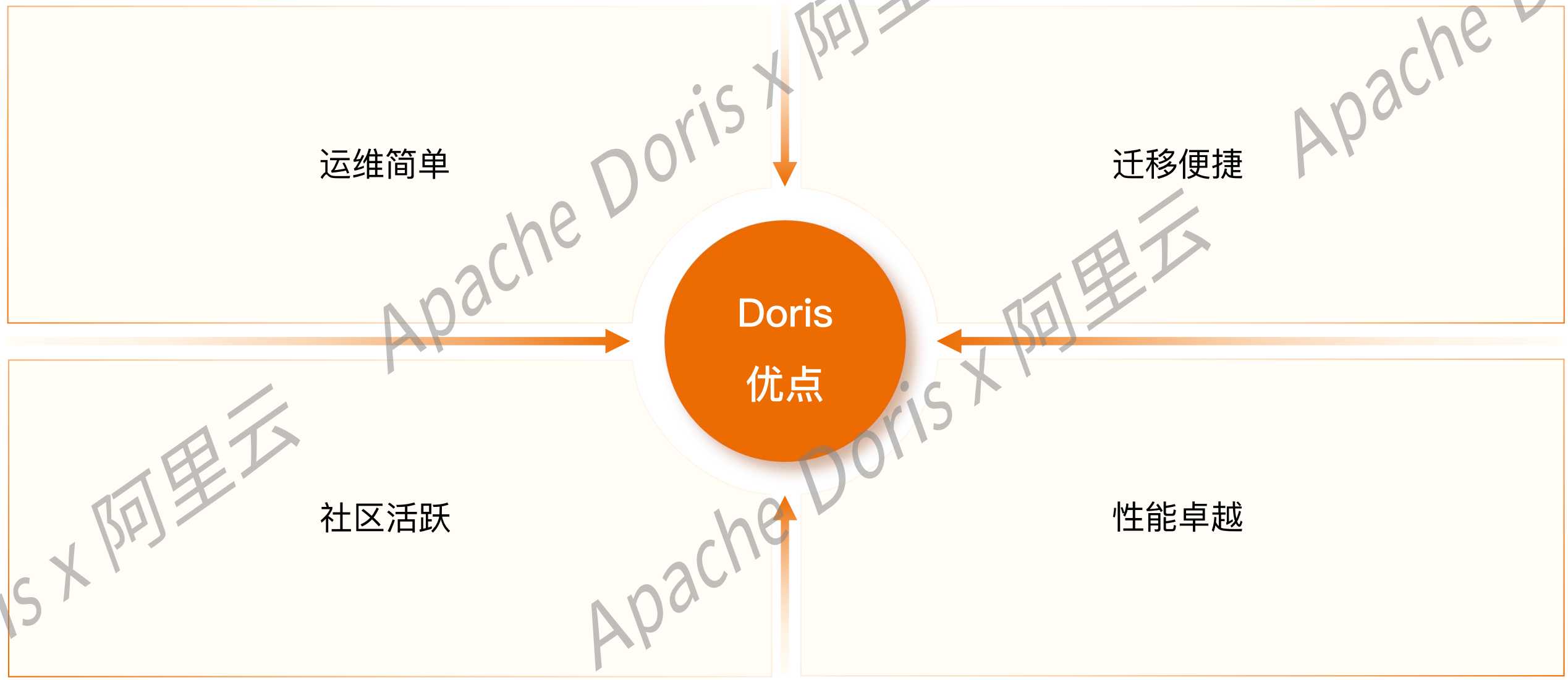
无数据模型
真正意义上的“仓库”

05 运维成本高

架构依赖链长，运维复杂度高，
架构笨重，扩展性差

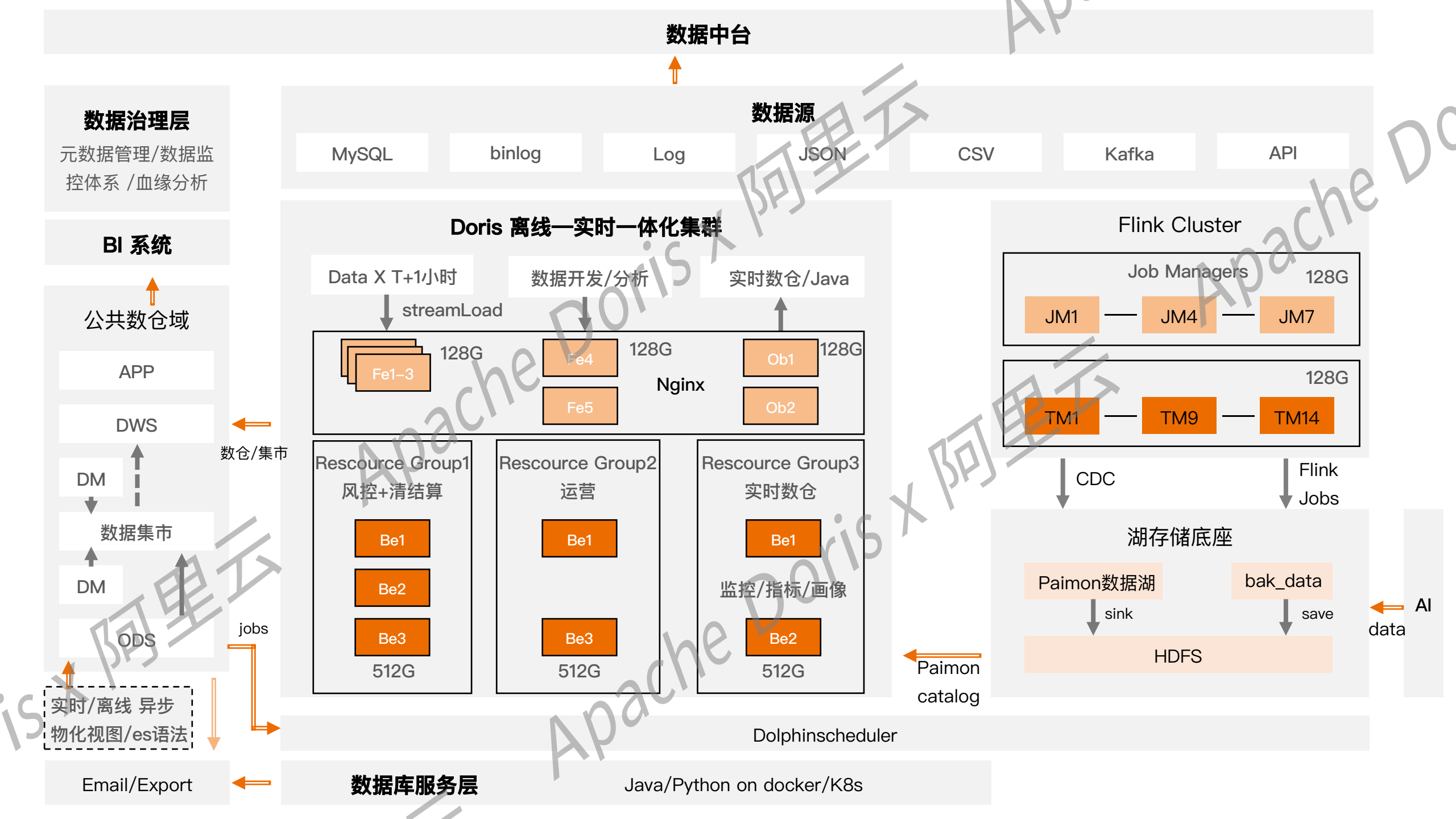
06 集群稳定性差

逻辑冗余度高，
基础参数配置不合理



LNZI 霖梓控股

改造后架构


数据中台
数据治理层
元数据管理/数据监控体系 /血缘分析
数据源
MySQL
binlog
Log
JSON
CSV
Kafka
API
BI 系统
公共数仓域
APP
DWS
DM
数据集市
DM
ODS
数仓/集市
jobs
实时/离线 异步
物化视图/es语法
Doris 离线一实时一体化集群
Data X T+1小时
数据开发/分析
实时数仓/Java
streamLoad
Fe1-3
128G
Fe4
Fe5
128G
Nginx
Ob1
Ob2
128G
Rescource Group1
风控+清结算
Be1
Be2
Be3
512G
Rescource Group2
运营
Be1
Be3
512G
Rescource Group3
实时数仓
Be1
监控/指标/画像
Be2
512G
Flink Cluster
Job Managers
128G
JM1
JM4
JM7
128G
TM1
TM9
TM14
CDC
Flink Jobs
湖存储底座
Paimon数据湖
bak_data
sink
save
HDFS
AI data
Paimon catalog
Dolphinscheduler
Email/Export
数据库服务层
Java/Python on docker/K8s

性能优化

01 分区分桶

- 建表时，已经设置分区分桶字段；
- 根据业务查询的粒度以及本身数据量决定分区粒度，如按天/周/月等；
- 将关联键设置为分桶字段，原理与 Hive分区分桶类似，为数仓中基础性能优化项。

动态分区

```sql
PARTITION BY RANGE(k1) ()
DISTRIBUTED BY HASH(k1)
PROPERTIES
(
    "dynamic_partition.enable" = "true",
    "dynamic_partition.time_unit" = "DAY",
    "dynamic_partition.start" = "-7",
    "dynamic_partition.end" = "3",
    "dynamic_partition.prefix" = "p",
    "dynamic_partition.buckets" = "32"
);

#修改动态分区属性
ALTER TABLE tbl1 SET
(
    "dynamic_partition.prop1" = "value1",
    ...
);
```

手动分区

```sql
PARTITION BY RANGE(col1[, col2, ...])
(
    PARTITION partition_name1 VALUES [("k1-lower1", "k2-lower1", "k3-lower1",...), ("k1-upper1", "k2-upper1", "k3-upper1", ...)),
    PARTITION partition_name2 VALUES [("k1-lower1-2", "k2-lower1-2", ...), ("k1-upper1-2", MAXVALUE, ))
)
#查看某表分区情况
show partitions from your_table
```

02 前缀索引

- 在 Doris 中，前缀索引是一种稀疏索引，即表中按照相应的行数的数据构成一个逻辑数据块 (Data Block)。
- 每个逻辑数据块在前缀索引表中存储一个索引项，长度不超过 36 字节，内容为数据块中第一行数据的排序列组成的前缀，在查找前缀索引表时，可以帮助确定该行数据所在逻辑数据块的起始行号。
- 由于前缀索引比较小，可以通过**全量缓存**的形式快速定位数据块，大大提升查询效率。比如在指定日期或者区域范围内的查询，可加速查询性能。

性能优化

03 倒排索引

- 在 Doris 的倒排索引实现中，Table 的一行对应一个文档、一列对应文档中的一个字段，因此倒排索引可以根据关键词快速定位包含它的行，达到 WHERE 子句加速的目的。经常用于一些大明细表以及日志分析中加速查询或者联表前提前过滤数据，减少资源占用率，比如 where 中包含全文检索、日期范围过滤、数值过滤以及字符串过滤等。

注意

A

存在精度问题的浮点数类型 FLOAT 和 DOUBLE 不支持倒排索引，原因是浮点数精度不准确。解决方案是使用精度准确的定点数类型 DECIMAL，DECIMAL 支持倒排索引。

B

部分复杂数据类型还不支持倒排索引，包括：MAP、STRUCT、JSON、HLL、BITMAP、QUANTILE_STATE、AGG_STATE。其中 MAP、STRUCT 会逐步支持，JSON 类型可以换成 VARIANT 类型获得支持。其他几个类型因为其特殊用途暂不需要支持倒排索引。

C

DUPLICATE 和 开启 Merge-on-Write 的 UNIQUE 表模型支持任意列建倒排索引。但是AGGREGATE 和 未开启 Merge-on-Write 的 UNIQUE 模型仅支持 Key 列建倒排索引，非Key 列不能建倒排索引，这是因为这两个模型需要读取所有数据后做合并，因此不能利用索引做提前过滤。

性能优化

04 BitMap 去重

Record No.	Name	Gender	Address	Income_Level
0	John	M	Perryridge	L1
1	Diana	F	Brooklyn	L2
2	Mary	F	Jonestown	L1
3	Peter	M	Brooklyn	L4
4	Kathy	F	Perryridge	L3

Bitmaps for Gender	
M	10010
F	01101

Bitmaps for Income_Level	
L1	10100
L2	01000
L3	00001
L4	00010
L5	00000

示例 SQL

```sql
#建表
create table metric_table (
  datekey int,
  hour int,
  device_id bitmap BITMAP_UNION
)
aggregate key (datekey, hour)
distributed by hash(datekey, hour) buckets 1
properties(
  "replication_num" = "1"
);
#查询
select hour, BITMAP_UNION_COUNT(pv) over(order by hour) uv from(
   select hour, BITMAP_UNION(device_id) as pv
   from metric_table -- 查询每小时的累计UV
   where datekey=20200622
group by hour order by 1
) final;
```

性能优化

05 高并发点查优化

- 上述的行存用于在 Unique 模型下开启 Merge-On-Write 策略是减少点查时的 IO 开销。当 enable_unique_key_merge_on_write 与 store_row_column 在创建 Unique 表开启时，对于主键的点查会走短路径来对 SQL 执行进行优化，仅需要执行一次 RPC 即可执行完成查询

示例 SQL

```
#开启行存
"store_row_column" = "true"
```

- 也可以通过增加 Oberver 的数量来提升处理查询的能力

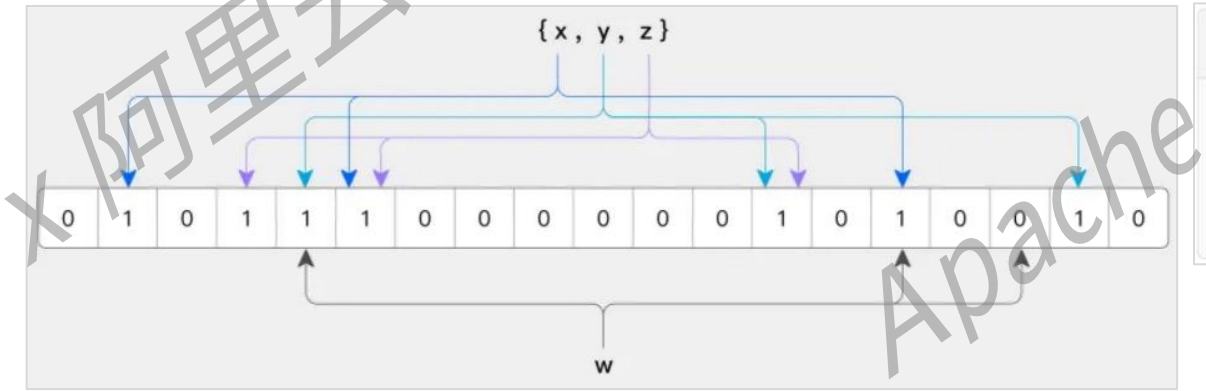
06 日志全文检索性能优化

- 在检索上，Doris 支持简单的等值与范围查询加速，也支持文本字段（中英文）的全文检索、多关键词检索（MATCH_ANY、MATCH_ALL）、短语检索、slop、多字段检索，相较 LIKE 模糊匹配，性能有很大改善。

性能优化

07 BloomFilter

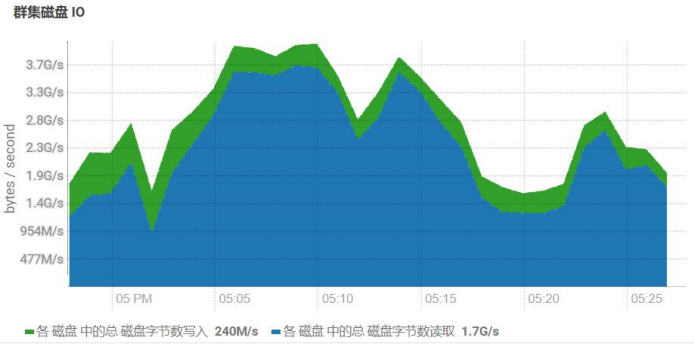
BloomFilter 索引是基于 BloomFilter 的一种跳数索引。原理是利用 BloomFilter 跳过等值查询指定条件不满足的数据块，达到减少 I/O 查询加速的目的。大致原理就是根据哈希冲突来判别某位上是否有元素，但 hashcode 不能唯一确定一个元素，所以布隆过滤器判别一定不存在的元素是准确的，但不全面。因此基于 BloomFilter 的索引只能跳过不满足条件的数据，不能精确定位满足条件的数据。使用布隆过滤器索引，写入时，对于数据块中的每个值，经过 Hash 存入数据块对应的 BloomFilter。查询时，根据等值条件的值，判断每个数据块对应的 BloomFilter 是否包含这个值，不包含则跳过对应的数据块不读取，达到减少 I/O 查询加速的目的。

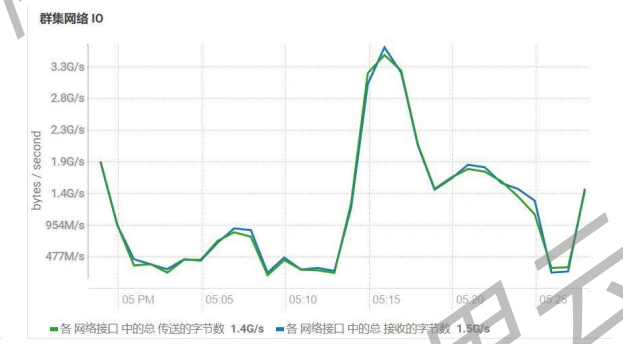


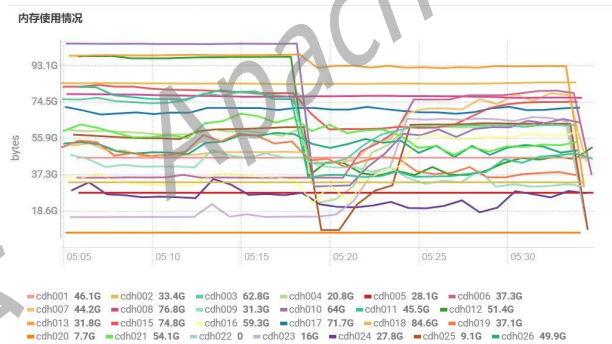
创建BloomFilter索引 SQL

```
PROPERTIES (
.....
"bloom_filter_columns"="saler_id,category_id"
);
```

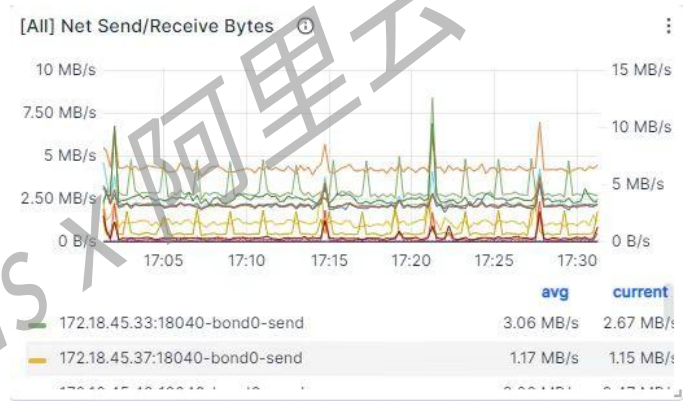
老架构

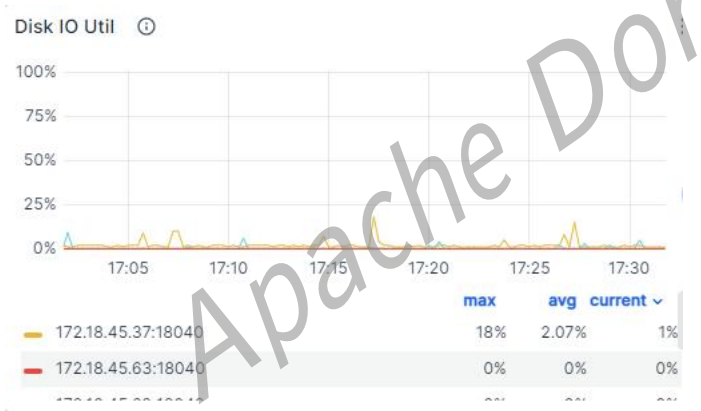






新架构

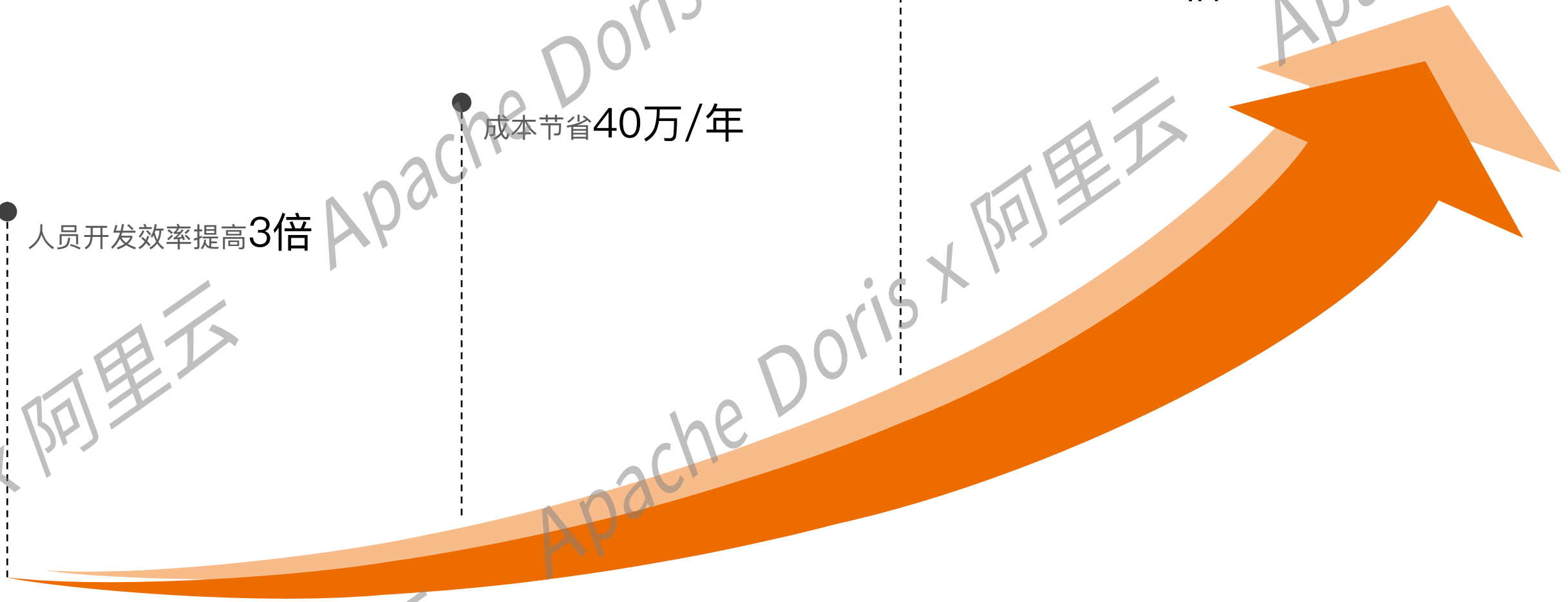


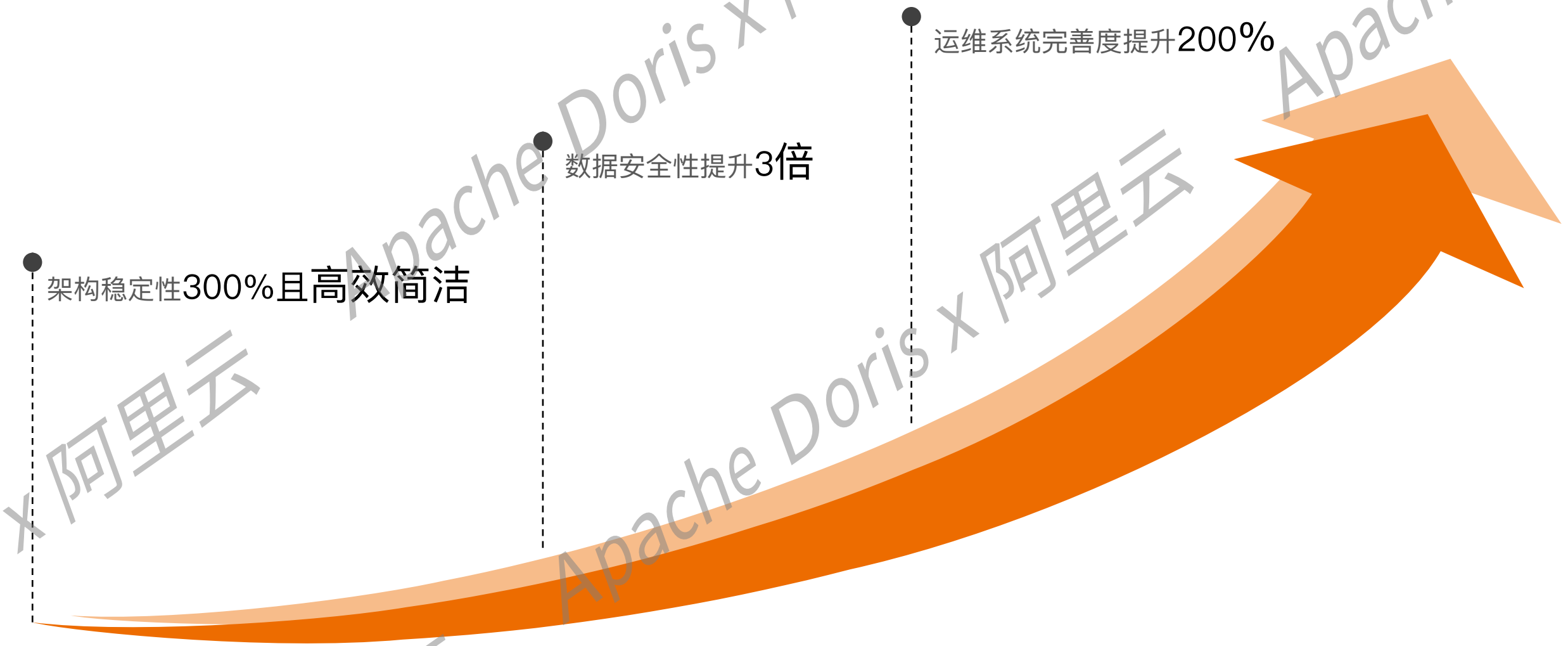


人员开发效率提高3倍

成本节省40万/年

计算效率提升7–30倍





问题解决

问题回顾1：基础配置不当引起集群健康状况异常

解决方案

在Doris FE 部署时，如果未显式指定 meta 目录的存储路径，则默认在 FE 部署目录下，而一般服务都会集中部署到 /opt/xxx 下，该目录的空间分配很可能导致元数据存储目录空间不足，出现元数据无法写入进而集群异常的情况。

- 经排查，重新配置了 FE meta 的存储路径。

配置项 SQL 复制代码

```
meta_dir =/xxxx
```

问题回顾2：通过 DECOMMISSION 下线 BE 节点时，部分 tablet 残留

解决方案

在下线过程中，通过 show backends 查看下线节点的 tabletNum，观察到 tabletNum 数量在减少，说明数据分片正在从这个节点迁移走。当数量减到 0 时，系统会自动删除这个节点。但某些情况下，tabletNum 下降到一定数值后不再变化。

- 这些 tablet 属于刚被删除的表、分区或物化视图，会保留在回收站中，而下线逻辑不会处理这些分片。可通过修改 FE 的配置参数 catalog_trash_expire_second 来修改对象在回收站中驻留时间。当对象从回收站中被删除后，这些 tablet 就会被处理了。

问题查看 SQL 复制代码

```
show proc "/cluster_health/tablet_health";
#注意：可以先通过以上命令查看集群是否还有 unhealthy 的分片，如果为 0，则可以直接通过
drop backend 语句删除这个 BE。否则，还需要具体查看不健康分片的副本情况
show proc "/cluster_balance"
```

问题解决

问题回顾3：节点新增加了新的磁盘，数据没有均衡到新的磁盘上

解决方案

- Doris 的均衡策略以节点为单位，也就是说，是按照节点整体的负载指标（分片数量和总磁盘利用率）来判断集群负载。
- 将数据分片从高负载节点迁移到低负载节点，如果每个节点都增加了一块磁盘，则从整体角度看，负载并没有改变，所以无法触发均衡逻辑。
- 此外，Doris 目前不支持单个节点内部，各磁盘间的均衡操作。所以新增磁盘后，不会将数据均衡到新的磁盘。
- 但数据在节点之间迁移时，Doris 会考虑磁盘的因素。比如一个分片从 A 节点迁移到 B 节点，会优先选择 B 节点中，磁盘空间利用率较低的磁盘。

- ① 重新建表
- ② 通过 Decommission 命令

前置执行 SQL 复制代码

```
admin set frontend config("drop_backend_after_decommission" = "false");
```

问题解决

问题回顾4：夜间大批量离线跑批时经常有任务超时

解决方案

优化完基于 Doris 的新架构后，我们将所有运营侧任务在 dolphinscheduler 中上线并分配定时时间，第二天发现大量任务执行失败告警信息：

- ① 合理调整各任务以及工作流之间的并行度
- ② 适当调大参数max_user_connections

问题回顾5：单 Query 异常导致某 BE 服务下线

解决方案

在 2.0 版本压测过程中，我们发现 where 条件数量超过 80时，会导致查询报错并且其中一台 BE 下线。经排查，原因在于边界场景下造成的 OOM。

- 我们在调整线程栈空间大小的基础上，限制了 where 条件的数量，也对全表扫描、笛卡尔积做了强限制。

问题解决

问题回顾6：偶发查询性能不稳定

解决方案

在压测过程中，有同事发现某条 SQL 在 Doris 不执行任何任务的情况下，多次执行速度相差较大：

- 查看SQL中是否存在笛卡尔积或者数据倾斜，对SQL中的某些低基字段做了额外的索引优化，并从业务的角度上优化了SQL；
- 同时，我们发现该账号的资源可用率被强制限制为集群总资源的 20%，优化后，我们使用该账号执行发现执行时间缩减到了 14–20s；
- 对执行参数进行会话设置。

```
        union all

        select gmt_create verify_time,unique_code consumer_no,'s' sx_status,case when channel regexp 'CJTG|WYD' and channel <> 'CJTG_XCX_LZ' then 'API' else 'APP' end channel,'中原API'
        ,row_number()over(partition by to_date(gmt_create),unique_code order by gmt_create desc) rn
        from ods.lt_lt_stage_borrow_new
        where gmt_create >= '2024-01-01 00:00:00' and is_delete = 0 and supplier_code = '100000286' and risk_order_no regexp 'wrlt'
    )a
left join ods.lt_lt_stage_borrow_new b on a.consumer_no = b.unique_code and b.success_num = 1
where a.rn = 1
group by to_date(a.verify_time)
    ,case when b.gmt_arrival < a.verify_time then '老客' else '新客' end
    ,a.channel
    ,a.zf
```

Execution Time: 14467 ms

days	kq	channel	zf	etsq	ettg
2024-04-10	老客	APP	银		
2024-04-20	老客	APP			
2024-08-14	老客	APP			
2024-08-11	老客	APP			
2024-08-03	老客	APP	中		507

参数设置 SQL 复制代码

```
set enable_pipeline_engine = true;
set parallel_pipeline_task_num = 6 #根据实际任务所需进行设置
```

01

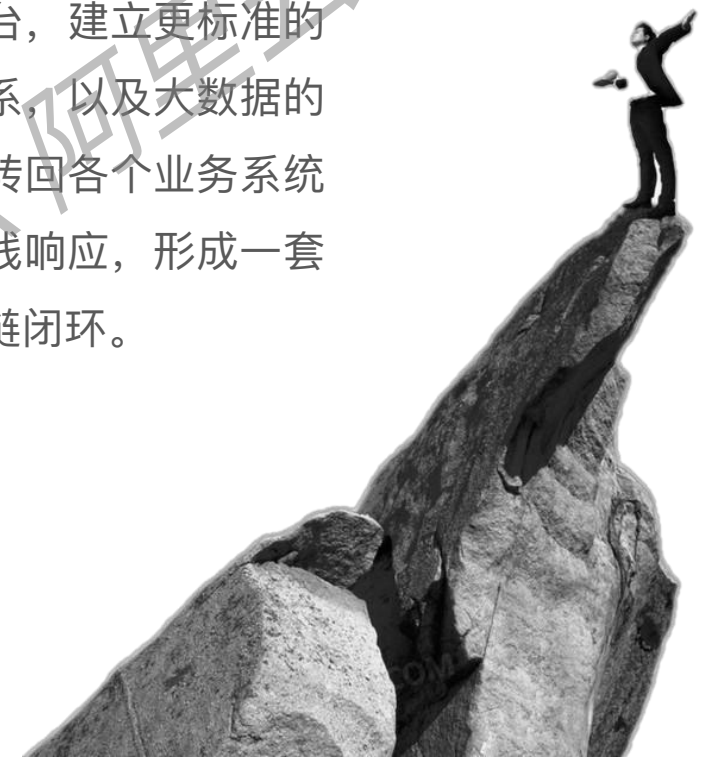
后期我们将进行全部架构重整，将离线与实时结合为业务提供看板、决策以及问题定位能力；

02

为了使这套架构能兼容 PB 级数据分析，我们将引入 Doris + Paimon 湖仓一体架构

03

研发数据中台，建立更标准的行业数据体系，以及大数据的成型数据流转回各个业务系统为业务做在线响应，形成一套完整的数据链闭环。



LNZI 霖梓控股

THANK YOU

谢谢观看

崔麒升

BI 中心大数据架构师