美团点评数据平台融合实战

谢语宸@ArchSummit 2017-07

CNUTCon 2017
全球运维技术大会

上海·光大会展中心大酒店 | 2017.9.10–11

智能时代的新运维

大数据运维
DevOps
安全
SRE
Kubernetes
Serverless
游戏运维
AIOps
智能化运维
基础架构
监控
互联网金融

主办方 Geekbang. 极客邦科技 InfoQ

StuQueue
斯达克学院

实践驱动的IT教育

斯达克学院(StuQ)， 极客邦旗下实践驱动的IT教育平台。通过线下和线上多种形式的综合学习解决方案 ， 帮助IT从业者和研发团队提升技能水平。

人工智能 | 大数据 | 前端开发 | 后端开发 | 架构设计

移动开发 | 运维设计 | 产品测试 | 产品经理 | 技术管理

10大职业技术领域课程

http://www.stuq.org

内容提要

分享

- 美团点评数据平台融合思路及实战经验

讨论

- Hadoop多机房架构的一种实现方案
- 大面积业务SQL任务重构的平滑方法
- 复杂平台系统平滑融合思路

Eat better, Live better

美团

点评

美团点评合并

两家公司 15年10月 合并, 业务层面上:

- 美团/点评 两个平台APP入口不变
- 业务跨团队划分, 1 + 1 > 2
  - 整合 综合业务 - 丽人/亲子/结婚/休闲娱乐, 评价UGC, 广告 等业务到上海原点评团队
  - 整合 餐饮业务 - 到店餐饮, 酒店旅游, 电影票 等业务到北京原美团对应团队

业务整合之后, 数据上:

- 数据生产地与数据使用地不同
- 数据关联分析越来越多
- 常态化的公司级别的流量, 交易数据表

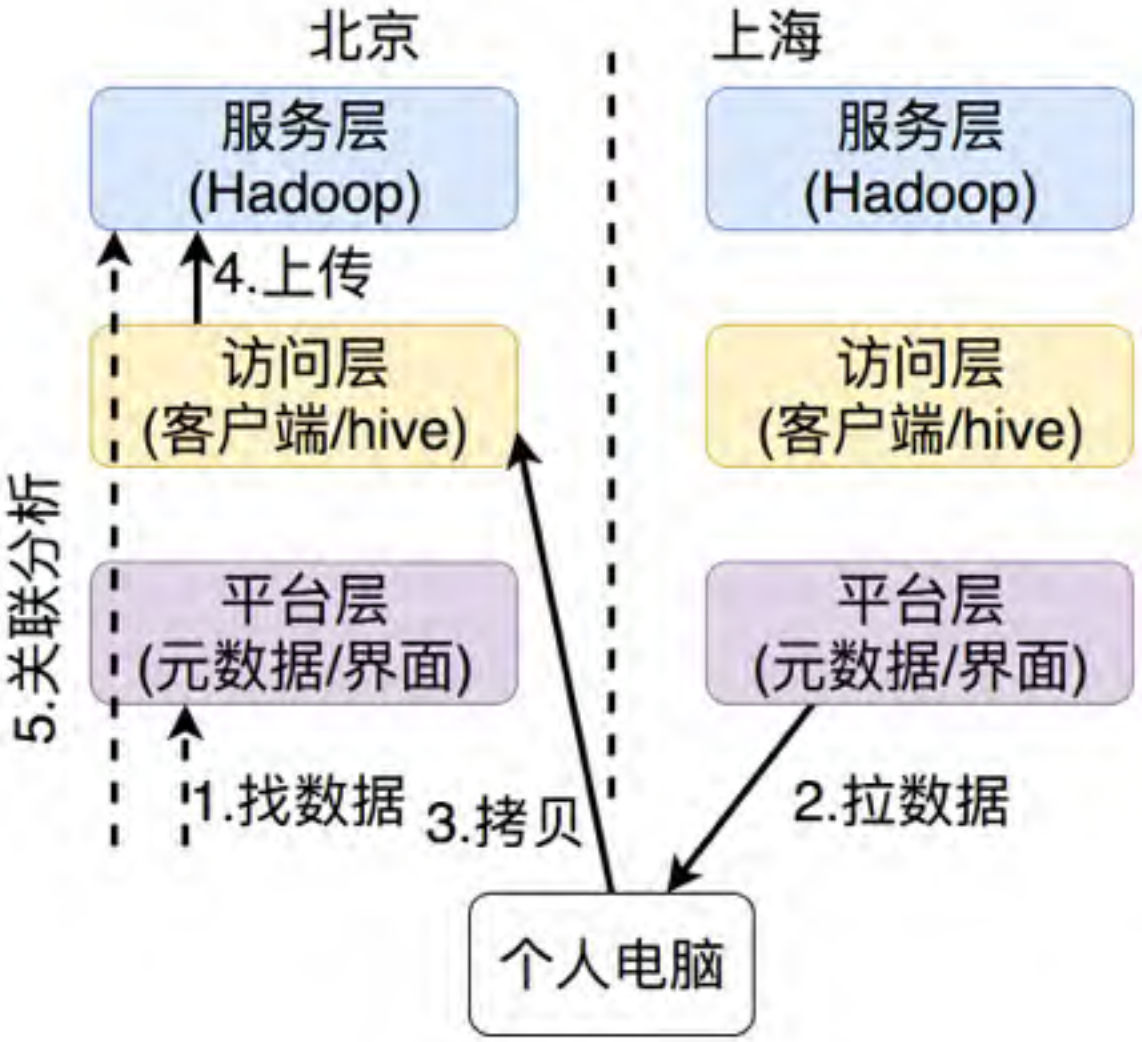
分析师的痛苦

分析: 最近一年访问过美团和点评两个App的重合度

用户: 原点评侧分析师

- 找数据 - 两套元数据系统, 两套规范学习成本高
- 做分析
  - 拉数据到本地 — 超出明细数据下载限制
  - 上传到原美团侧集群 — 没有账号, 超出上传量限制, 断点续传
  - 关联数据 — sql/udf不兼容, 分析工具差别太大, 申请计算资源
  - 做报表 — 每天人工走一遍流程
  - 改分析条件 — 中间结果白做了

结果: 分析师: “算了, 要不我们做个抽样分析好了…”



整体目标

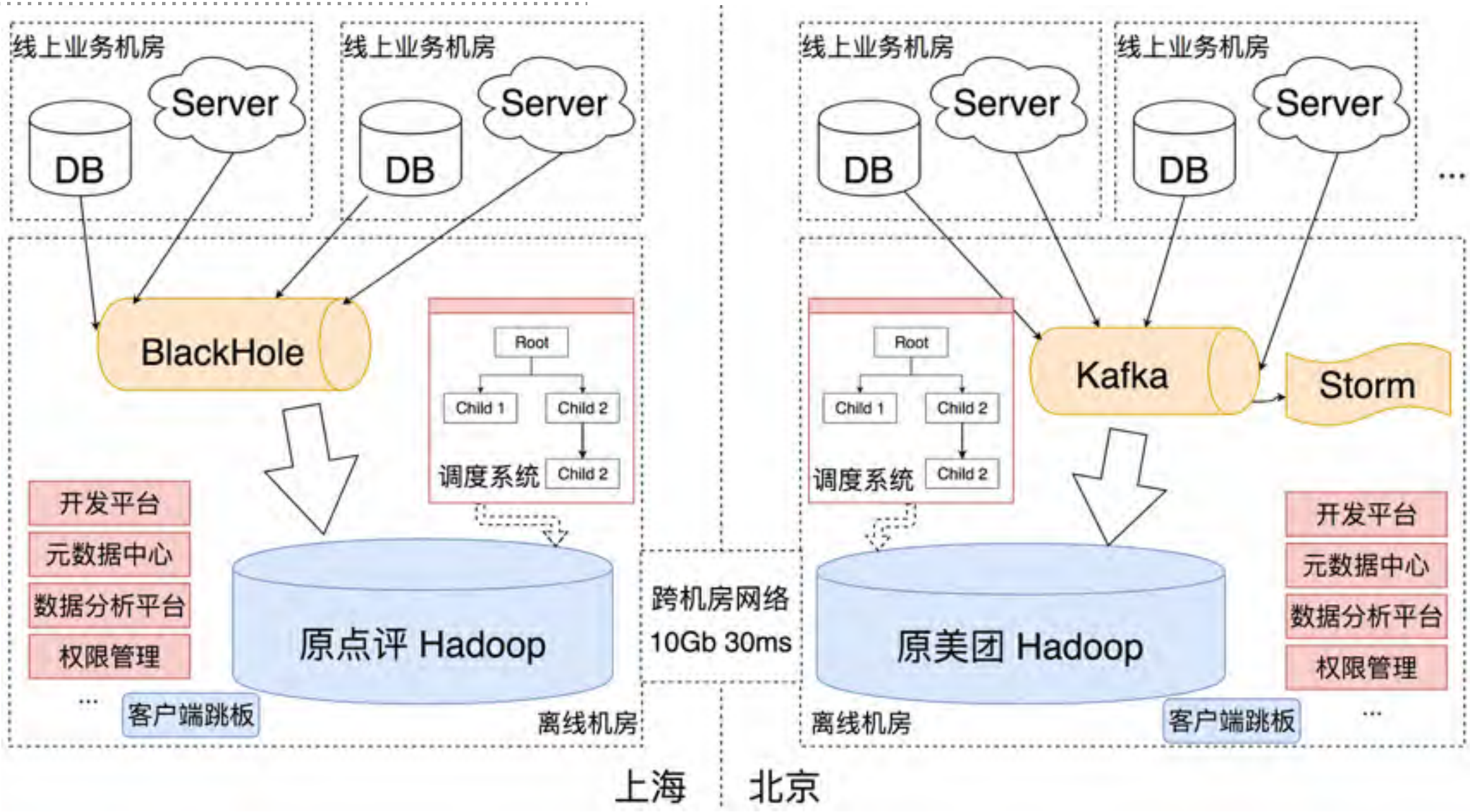
一个集群，一套工具，一套规范

整体目标

一个集群，一套工具，一套规范

至少看起来是　　至少明确留什么　至少认同有规范

难点 – 架构复杂, 基础设施限制



难点 – 可靠性要求

- 每日0点开始按天数据生产
- 工作日9点业务线总监以上高层会议过数据
- 工作日10点到22点分析师查数据, 数据科学家调模型

- 用户数千级, 我们有个所有数据应用用户群, 一旦波动...
  - “一句话证明你是美团人: 集群挂了, 又查不了数了”

- 考虑回滚, 运维时间窗口: 平日 — 1小时内, 周末 — 4小时内

难点 – 体量

美团点评数据仓库为主要特点:

- 业务快速增长, 每年节点数翻翻
- 数据重刷量大, 每日生成数据, 50%替换, 50%纯新

融合前	节点数	数据量	日生成数据量
上海 (原点评)	500+	11P	120T
北京 (原美团)	3000+	75P	800T

难点 – 平台化与复杂度

- 数据平台主要概念：数据表，计算任务
- 平台化后，开发，部分运维职责在业务线手里
- 两侧平台融合难免功能性，流程有所取舍，无法全面兼容

融合前	仓库任务数	业务团队数	开发平台日活	查询平台日活
上海 (原点评)	7000+	28	100+	400+
北京 (原美团)	14000+	50+	240+	900+

项目拆解

- 数据互访打通
  - 可以配置定期或提交数据同步任务, 进行两地数据的交互分析
- 集群融合
  - 接口不变, 集群搬迁, 缓解扩容压力
  - 逻辑上融合成一个集群, 各个入口提供原两地表同时访问能力
- 开发工具融合
  - 用户基于一套平台管理数据任务
- 原点评侧拆库
  - 统一数仓划分标准, 拆分业务

项目拆解

- **数据互访打通**
  - **可以配置定期或提交数据同步任务，进行两地数据的交互分析**
- 集群融合
  - 接口不变，集群搬迁，缓解扩容压力
  - 逻辑上融合成一个集群，各个入口提供原两地表同时访问能力
- 开发工具融合
  - 用户基于一套平台管理数据任务
- 原点评侧拆库
  - 统一数仓划分标准，拆分业务

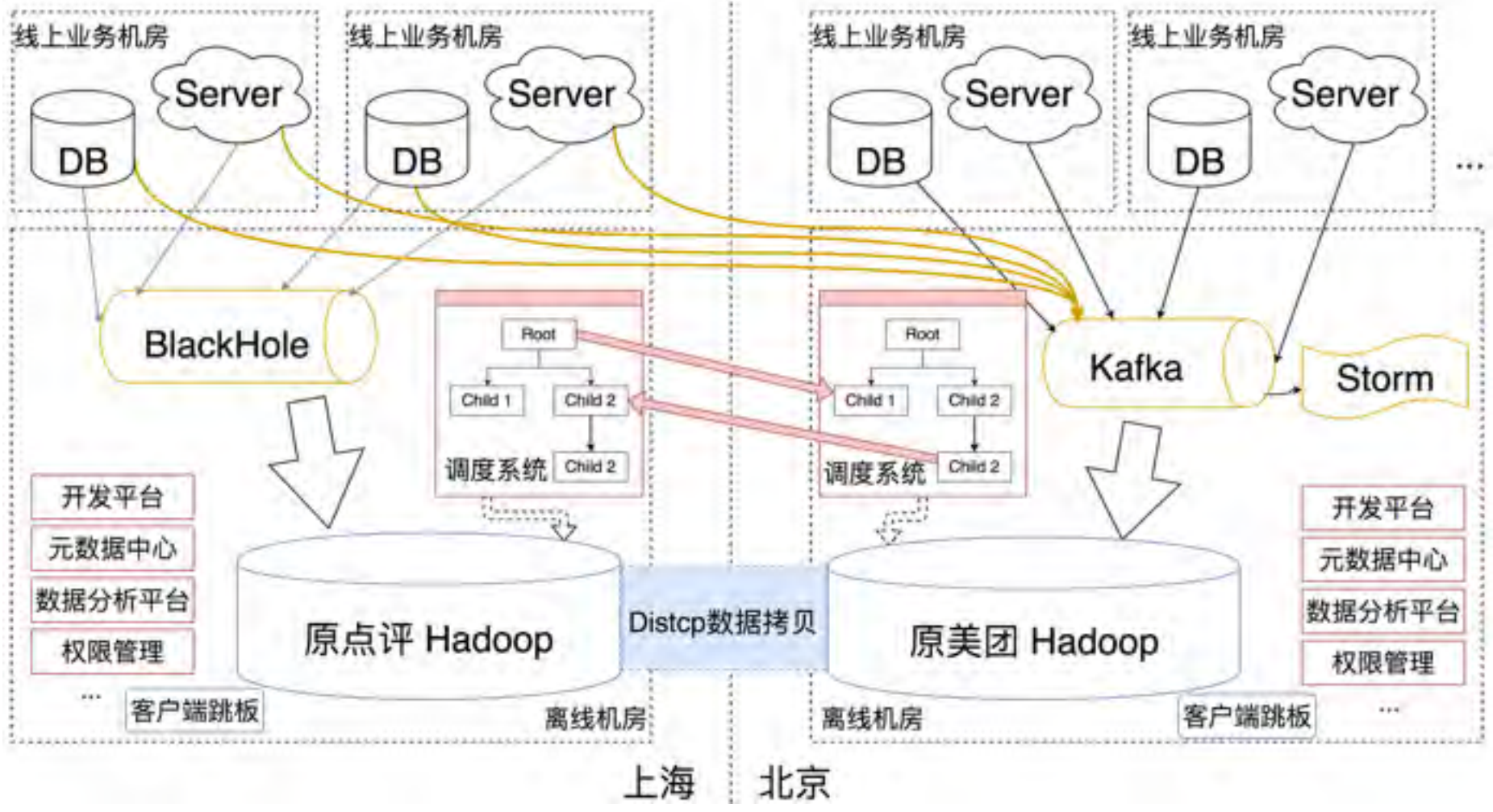
数据互访打通

目标: 用户可以配置定期或提交数据同步任务, 进行两地数据的交互分析

原始层数据收集

集群数据互拷

跨调度器依赖关系

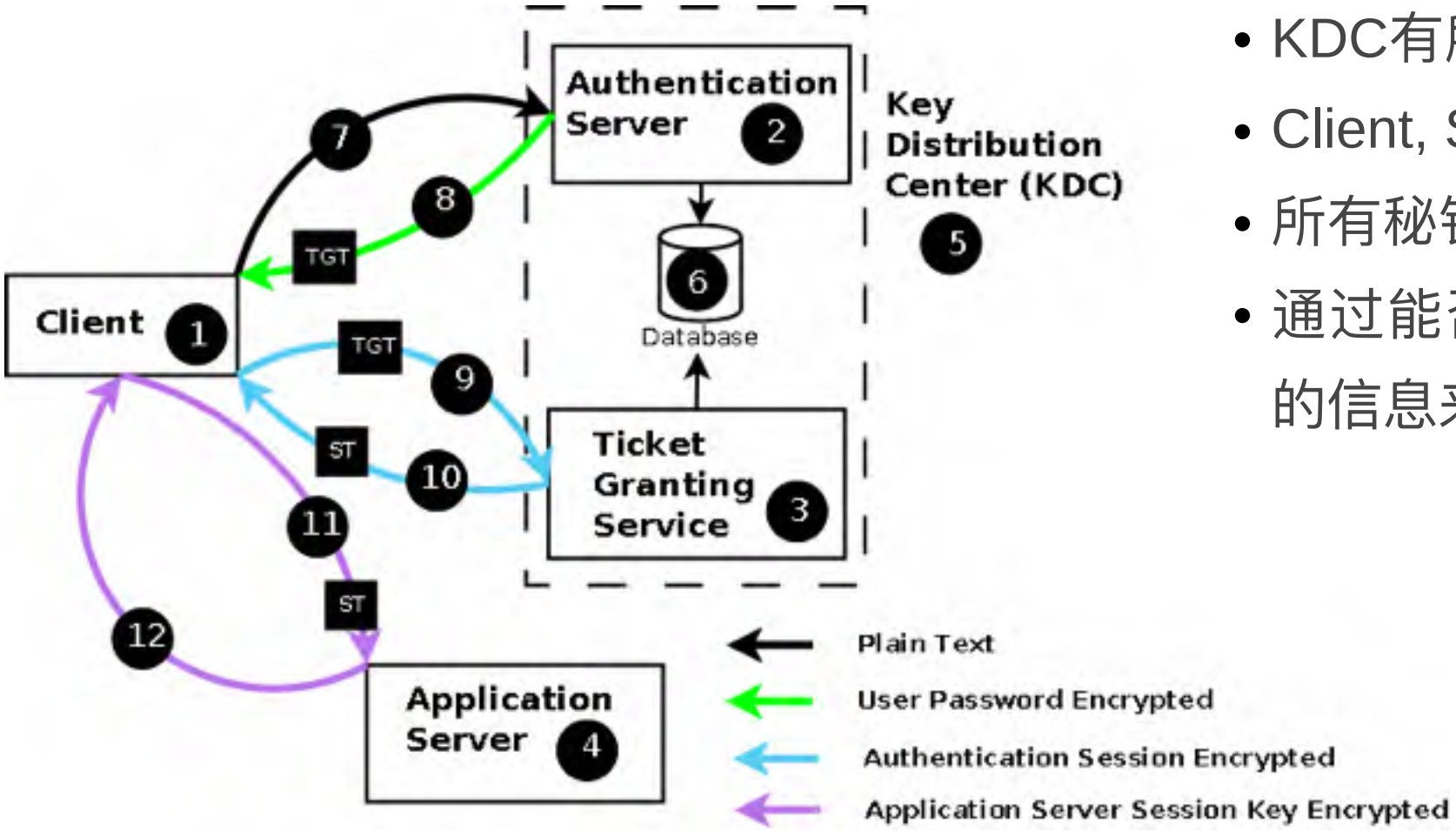


集群数据互访与拷贝任务

- Hadoop原生支持DistCP, 只要:
  - 网络通
  - 能认证账号
  - 有读取权限
  - 执行端有资源

- **权衡**
  - **平台统一管理同步任务**
  - **两侧集群分别建立一个用于同步的账号**
  - **在“读端” 集群上启动任务, 写入对应账号空间**
  - **写出数据默认开放本地各账号读取**

Kerberos

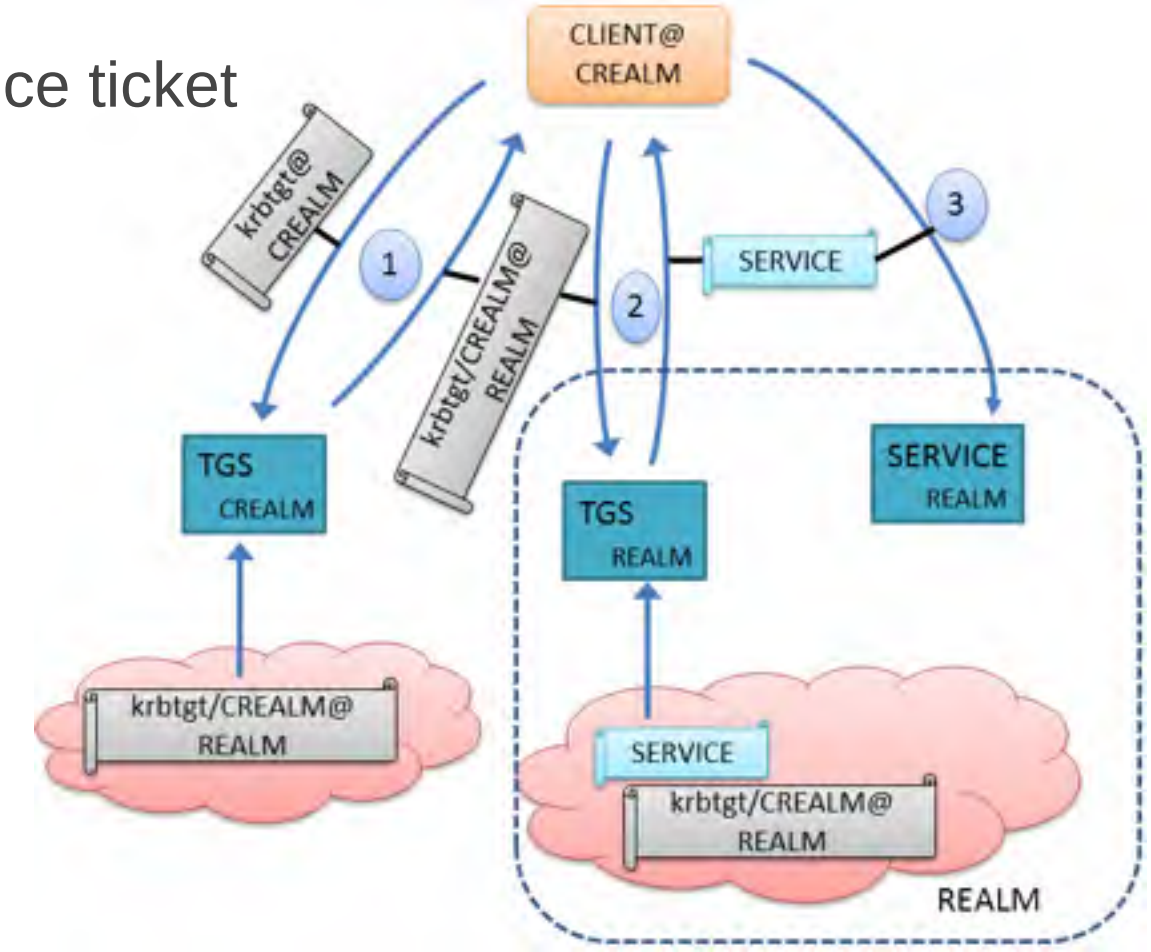


简述

- KDC有所有的Client, Server的秘钥
- Client, Server各自有各自的秘钥
- 所有秘钥都不进行传输
- 通过能否解密出用自有秘钥加密过的信息来认证

Kerberos跨域认证架构

- **对于某个方向的认证, 两个KDC共享一个秘钥**
- client请求本域KDC, 拿跨域ticket
- 使用跨域ticket认证, 请求他域KDC, 拿Service ticket
- client拿Service ticket访问Service



Kerberos跨域认证架构

- **两个KDC同时建立两个principal，krbtgt/A@B, krbtgt/B@A**
  - 要求秘钥编码一致, 具体查手册
- **krb5.conf 配置对应realm的KDC位置, 并能通过domain_realm策略找到**
  - 使用A realm的principal进行认证(kinit), 使用kvno命令尝试获取B realm的server principal来确定是否跨域认证成功
- **hadoop client端 dfs.namenode.kerberos.principal.pattern 以及yarn.resourcemanager.principal.pattern 设置为***
  - 绕过hadoop对于service principal 判断是否合法
- **在所有RPC Server端配置 hadoop.security.auth_to_local 规则**
  - 以上两条开-Dsun.security.krb5.debug 看log, hadoop代码中有存在对于realm隐改的行为

项目拆解

- 数据互访打通
  - 可以配置定期或提交数据同步任务，进行两地数据的交互分析
- **集群融合**
  - **接口不变，集群搬迁，缓解扩容压力**
  - **逻辑上融合成一个集群，各个入口提供原两地表同时访问能力**
- 开发工具融合
  - 用户基于一套平台管理数据任务
- 原点评侧拆库
  - 统一数仓划分标准，拆分业务

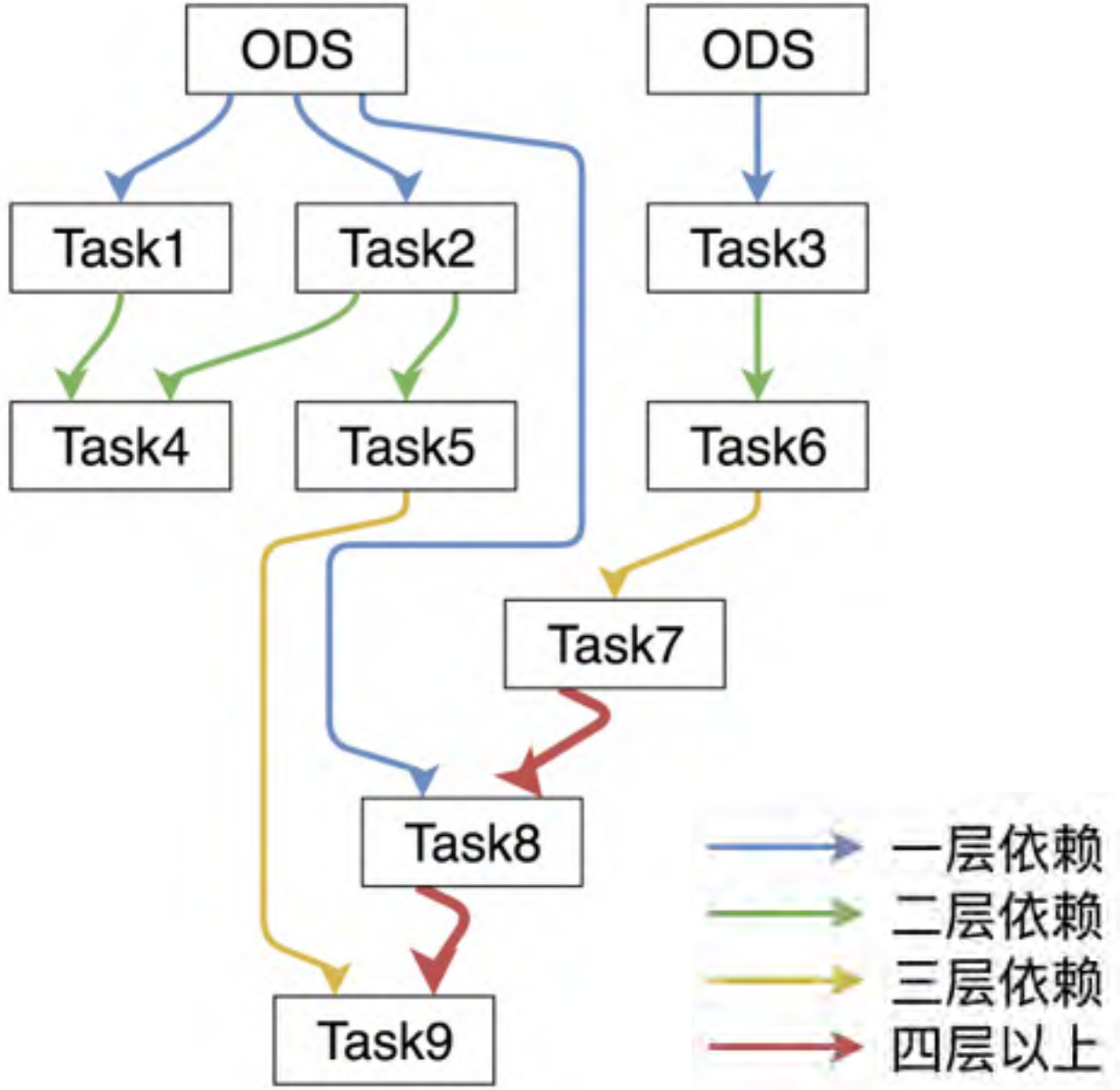
回顾目标

一个集群，一套工具，一套规范

至少看起来是　　至少明确留什么　至少认同有规范

平台必须融合，如果我们什么都不做…

- 数据RD需要在目标平台重建数据
  - 平台工具大量 概念，功能，流程，规范不一致
- 从源头开始逐层迁移，整体阻塞
  - 7000+任务，平均深度10层
- 数据表拷贝，上线任务双跑，数据校验
  - 拷贝1-4天，改代码开发测试1-2天，数据可信3-5天
- 长期两倍数据，两份任务
  - 承载能力问题，任务逻辑持续修改需求



问题本质

- 不能双跑，一旦双跑，势必会常态化的生成两份数据，引发一系列问题
- 所以，必须一份数据，一套任务执行机制，后续任务改写，相当于系统内部上线
- 因此，需要自底向上融合，**先合集群，后推动切换平台**

我们的思路

集群融合先行

- 统一 — Hadoop服务架构与代码版本统一
- 拷贝 — 原点评集群Block同步北京机房2个副本
- 切换 — 存储计算服务切换到北京机房
- 融合 — 统一客户端同时访问原来两集群

基于统一的数据/集群元数据/访问入口后, 工具链融合即可:

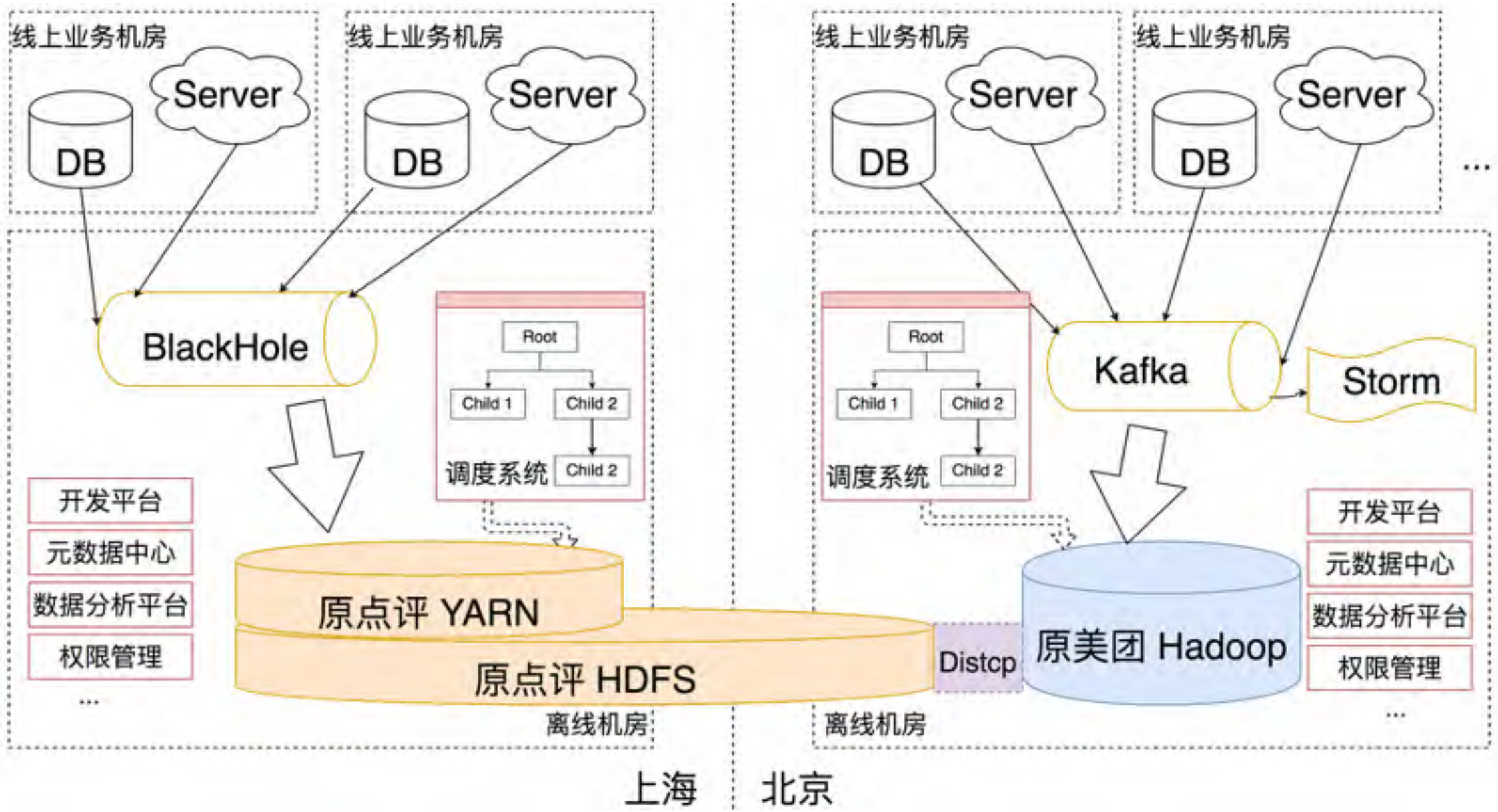
- 任务表示兼容
- 灰度模式逐个任务替换
- 各个管理工具, 数据访问工具接管新数据

统一 — Hadoop服务架构与代码版本

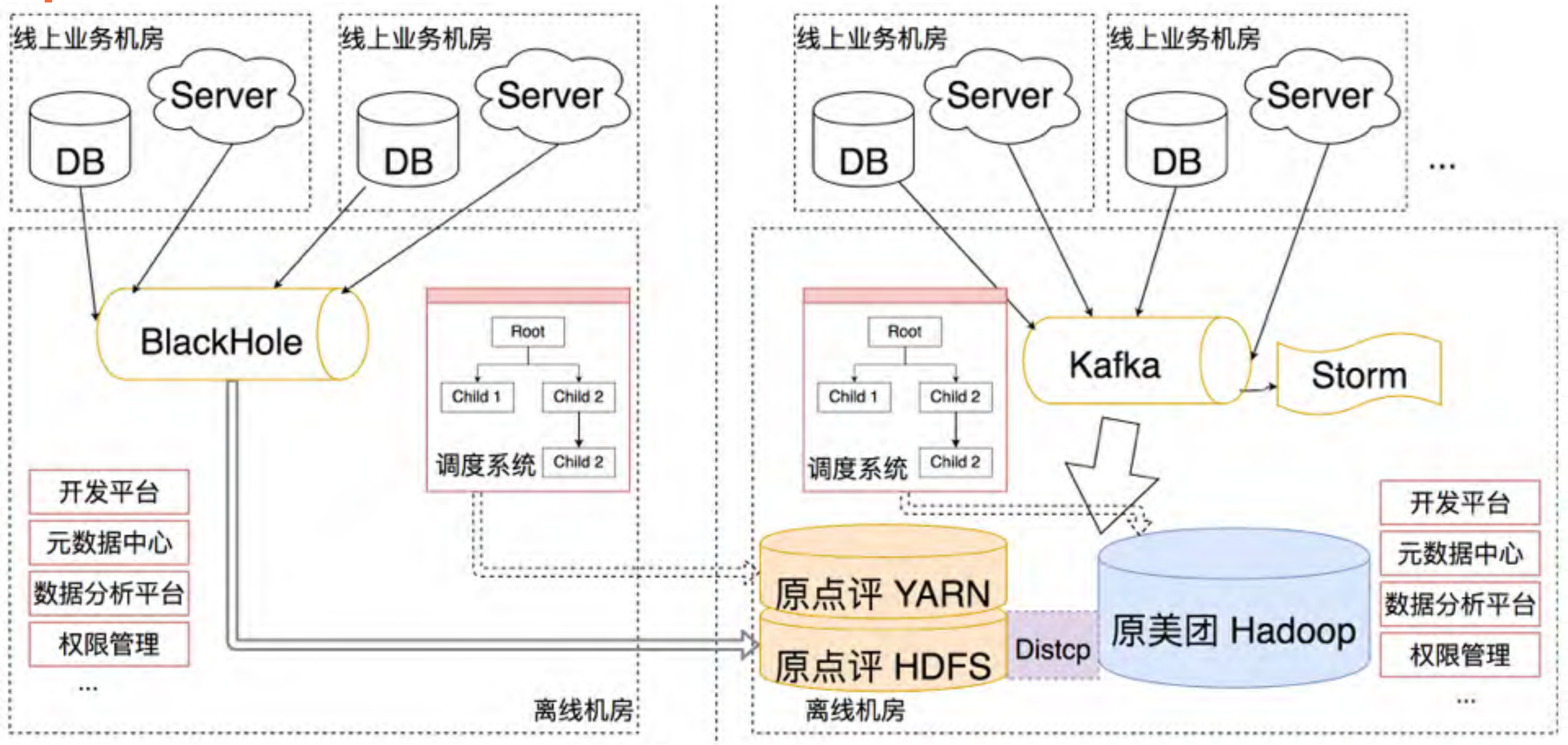
环境对比	北京侧	上海侧
JDK	1.7.0_76	1.6.0_43
Hadoop	2.7.1	2.4.1
HDFS Arch.	HA using QJM, Federation	无
Hive	0.13, 1.2	0.11
Kerberos	keytab	keytab, token, password
日志收集	自研Agent, Kafka, camus	自研收集服务 BlackHole
任务调度	自建	自建
……	……	…….

自研Patch要能区分, 合版本时互相Review了上百个Patch

拷贝 — 原点评集群Block同步北京机房2个副本



切换 — 存储计算服务切换到北京机房



拷贝 & 切换

拷贝 — 原点评集群Block同步北京机房2个副本

切换 — 存储计算服务切换到北京机房

如何做到的: Hadoop多机房架构

设计目标

- Hadoop单集群多机房
- 块粒度控制数据副本分布与迁移
- 业务无感知的集群平滑切换

Hadoop多机房架构

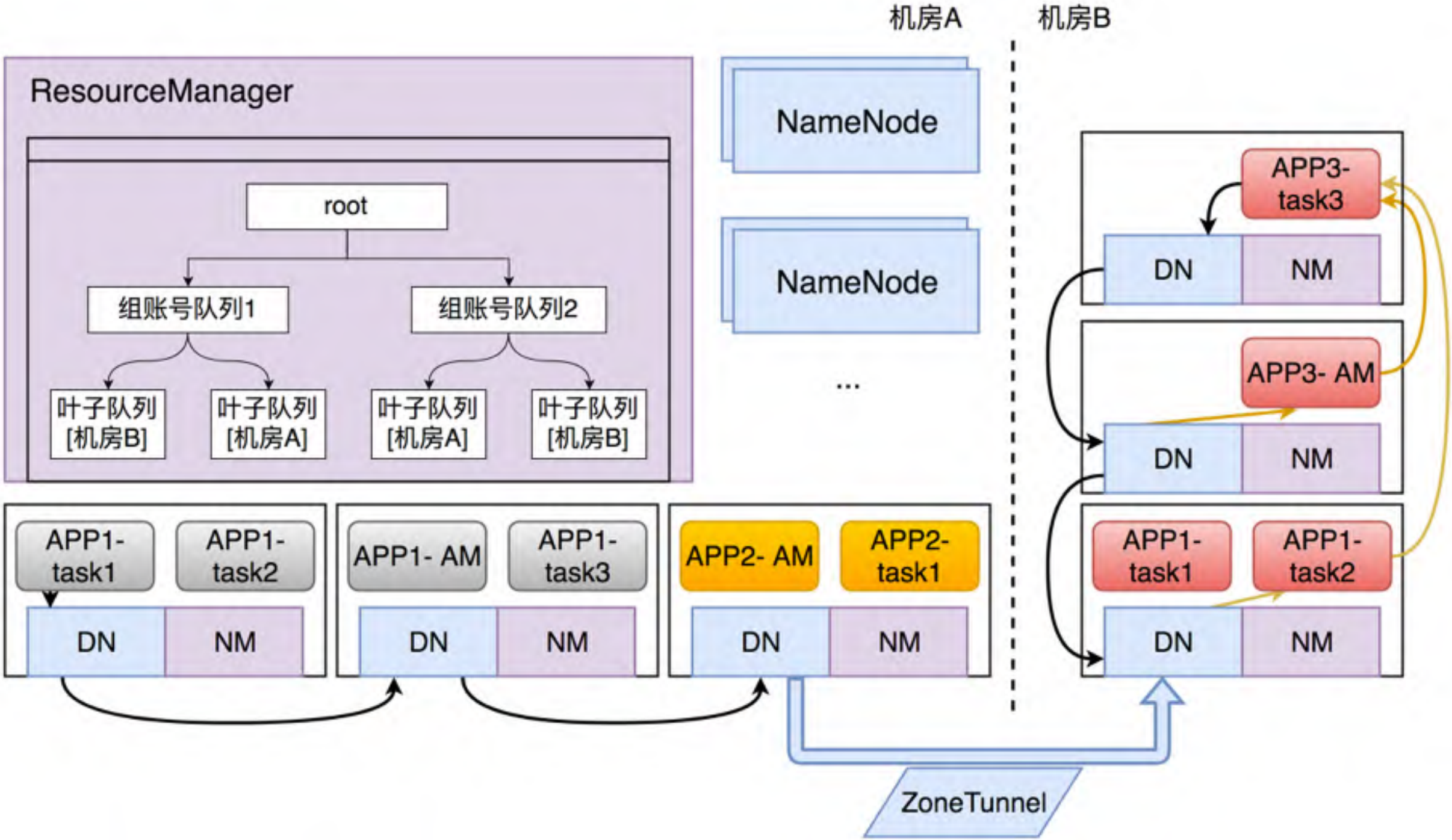
- Hadoop原生架构没有机房的概念,所有服务器都被认为在“同一个机房”
  - 不可避免的衍生出大量的、不受控制的跨机房读写数据的操作
- 机房间网络带宽资源受限
  - 原点评集群日新增120T, 跨机房带宽10G
- 机房间通信有延时,机房内 << 1ms,机房间 30ms
  - 举例:作业要读取30万个文件,split阶段要和NN串行交互30万+次

核心在于跨机房带宽与流量管控

Hadoop多机房架构

核心思路: 梳理跨机房场景, 规避, 减量, 管控

- 写数据DN间 pipeline 跨机房
  - NN增加 机房 “zone” 概念, 修改NN逻辑, 建立pipeline默认与client同机房
- Application内数据同步
  - YARN队列增加 “zone” 概念, 修改调度逻辑, 当前调度NM如果与队列不同机房, 则跳过
- 读取数据跨机房
  - 读取块选择本地优先
  - 任务临时文件上传任务所在目标机房
- 拷贝数据: 基于Balancer使用的接口, 工具驱动DN之间直接进行跨机房的Block粒度拷贝


机房A
机房B
ResourceManager
root
组账号队列1
组账号队列2
叶子队列 [机房B]
叶子队列 [机房A]
叶子队列 [机房A]
叶子队列 [机房B]
NameNode
NameNode
...
APP1- task1
APP1- task2
APP1- AM
APP1- task3
APP2- AM
APP2- task1
DN
NM
APP3- task3
APP3- AM
APP1- task1
APP1- task2
ZoneTunnel

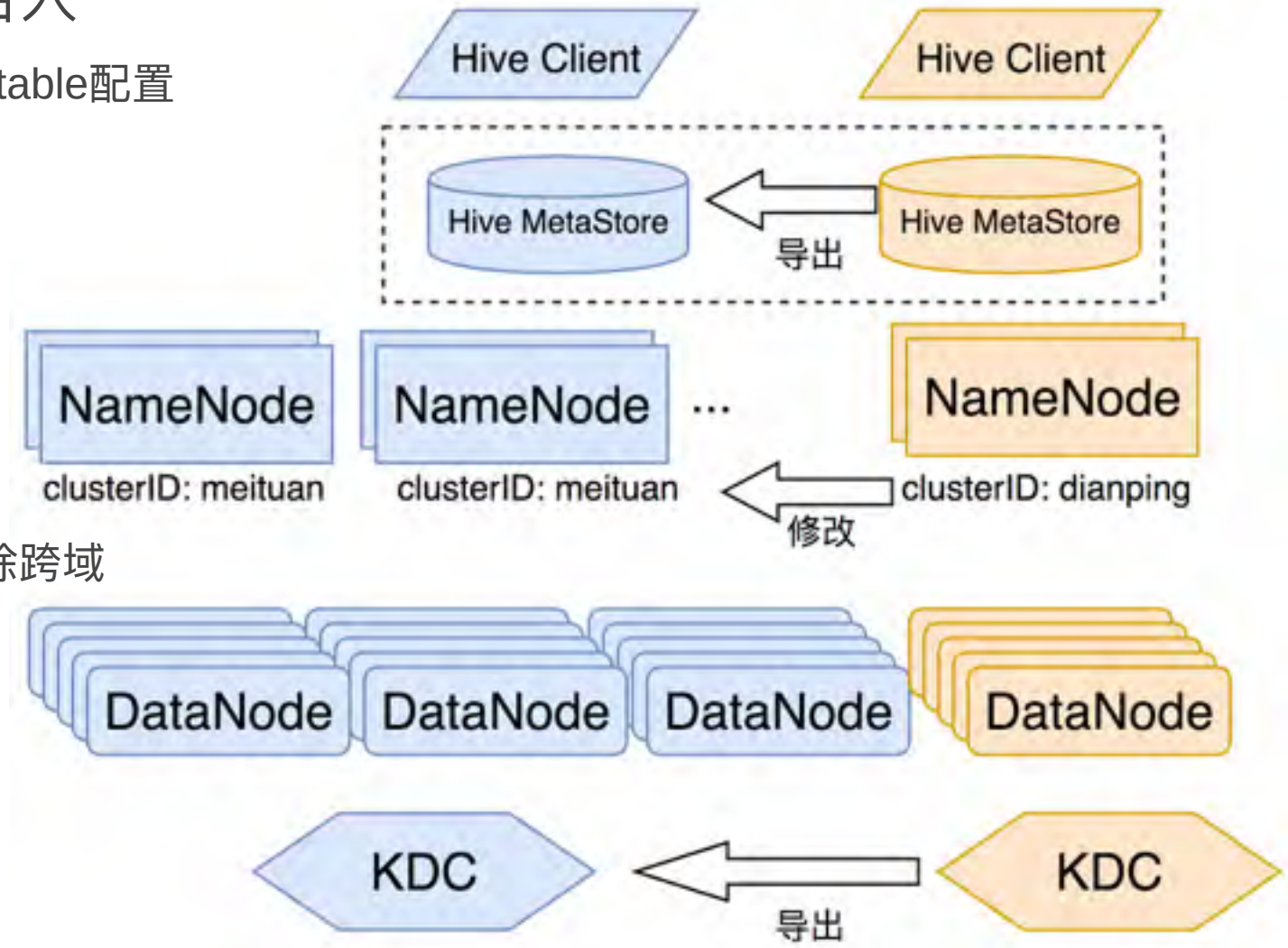
Hadoop多机房架构

应用经验

- 带宽到底要多少
  - 新增量, 替换量, 拷贝策略/留存率, 我们拷贝了近4个月
- 元数据分析能力
  - Rpc抓NN 到 基于FSImage + StandbyNN请求
- 任务提交与任务元数据上传管理
  - 切计算集群同时切任务提交机器
- 切换当天新增任务拷贝
  - 分析EditLog找变更
- 跨机房带宽与线上业务隔离

融合 — 统一客户端同时访问原来两集群

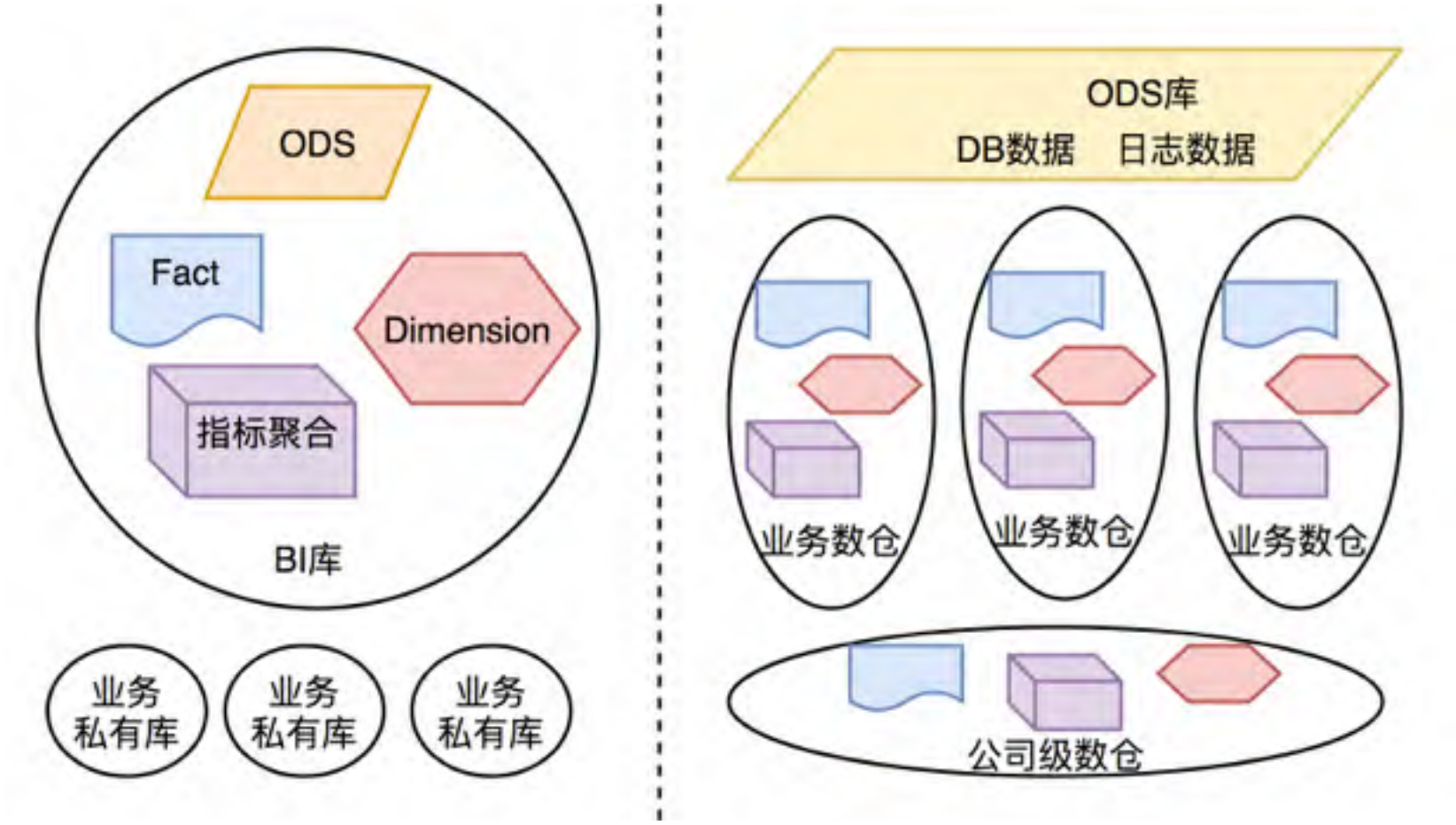
- NameNode 作为一个Federation合入
  - 修改元数据clusterID, 修改客户端mount table配置
- Hive 元数据融合
  - mysqldb存储合并, 切换配置
- 统一认证服务& 统一Master账号
  - KDC账号导出与导入, 切换KDC
  - 修改NN, DN, RM, NM服务账号配置
  - 由于有跨域, 灰度进行账号切换, 最终拆除跨域



项目拆解

- 数据互访打通
  - 可以配置定期或提交数据同步任务，进行两地数据的交互分析
- 集群融合
  - 接口不变，集群搬迁，缓解扩容压力
  - 逻辑上融合成一个集群，各个入口提供原两地表同时访问能力
- 开发工具融合
  - 用户基于一套平台管理数据任务
- **原点评侧拆库**
  - **统一数仓划分标准，拆分业务**

原点评侧拆库 – 背景



点评侧 - 重整体性, 所有业务一套账号, 共享一个Hive库

特殊需求使用更加自由的私有库

美团侧 - 平台化, 业务已经进行分拆

有公司数仓团队进行数据整合

原点评侧拆库

原任务内容:

```
insert into bi.table_a
select x, y, z
from bi.table_b join bi.table_c on ***
where *** group by ***
```

希望改写成

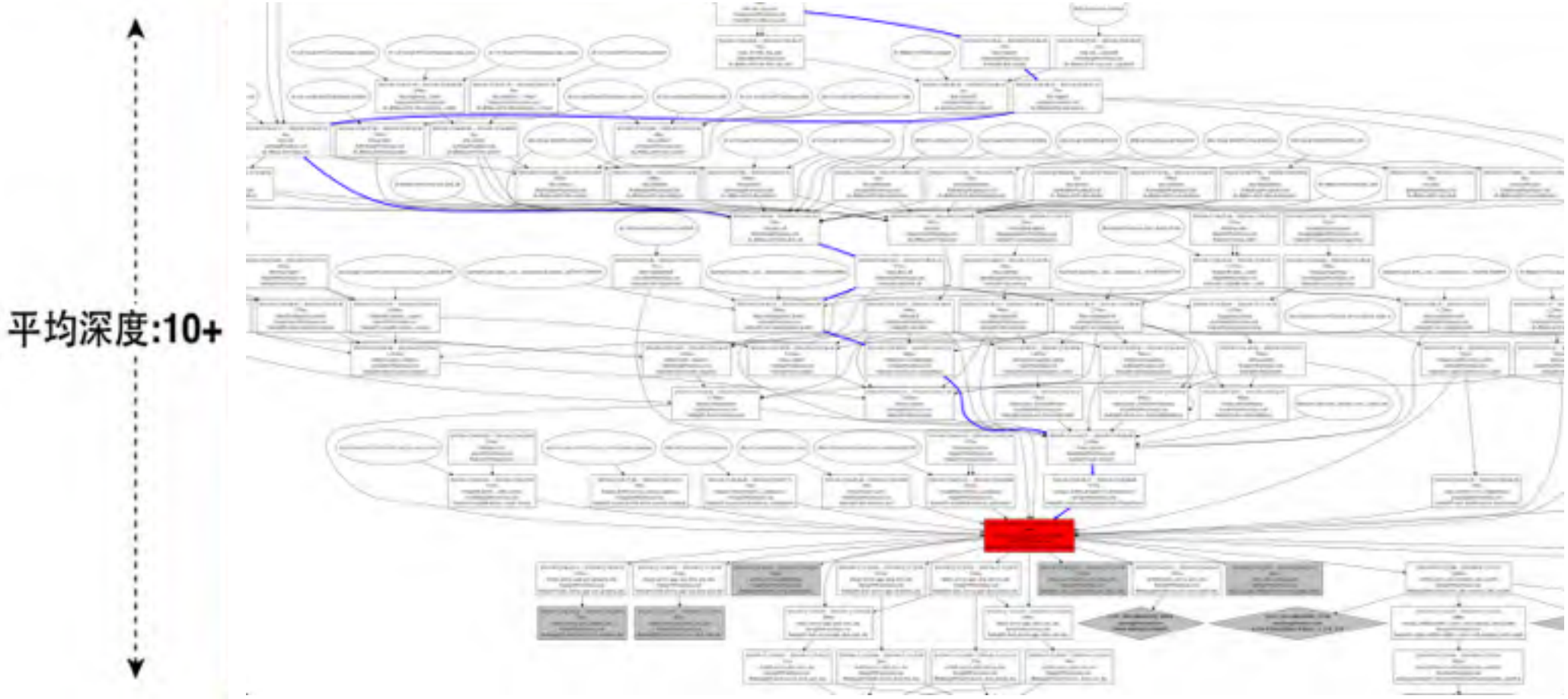
```
insert into mart_xxx.table_a
select x, y, z
from mart_yyy.table_b join mart_zzz.table_c on ***
where *** group by ***
```

原点评侧拆库

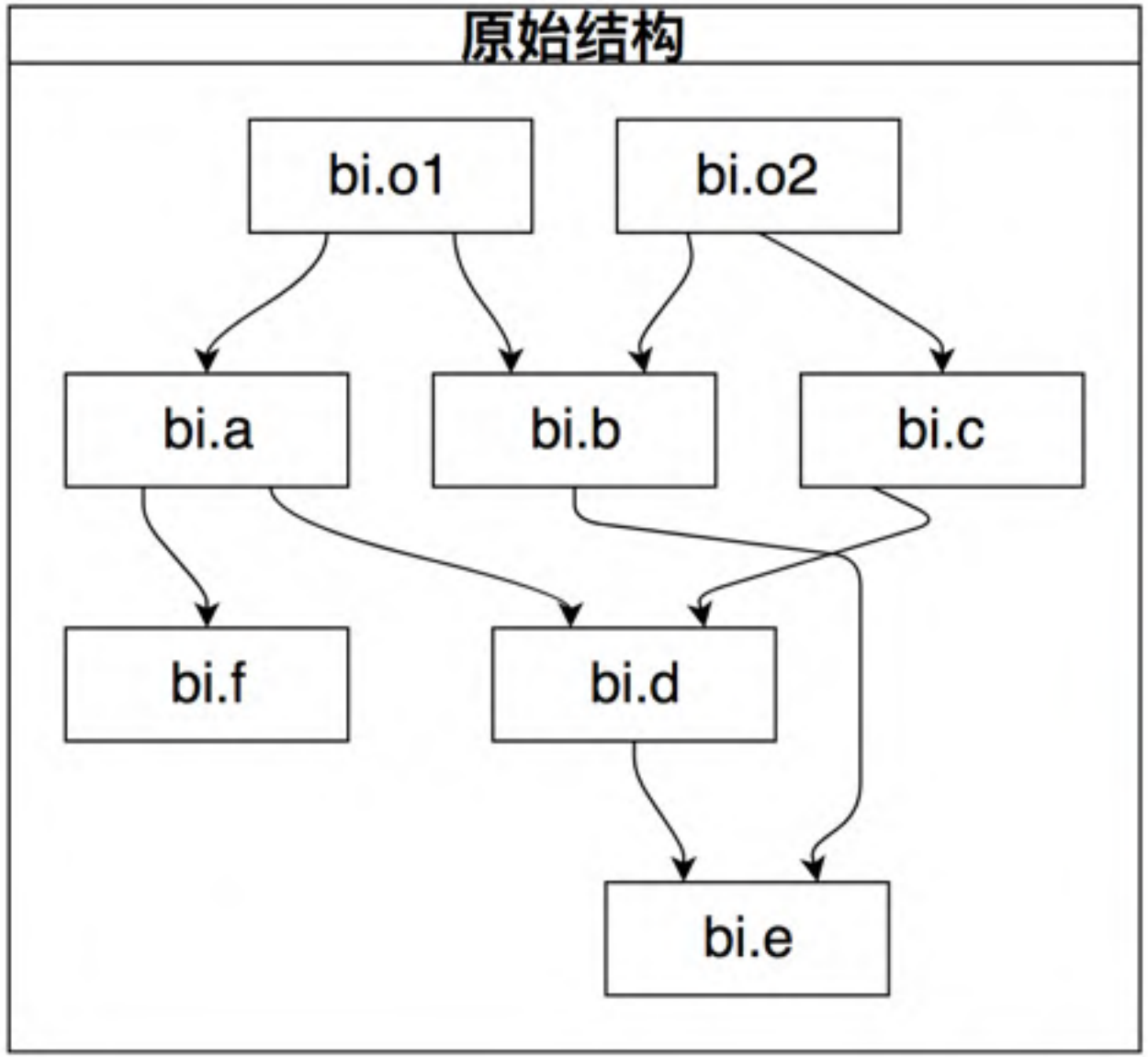
- 我们有 8000 个这样的任务要改写, 同时...

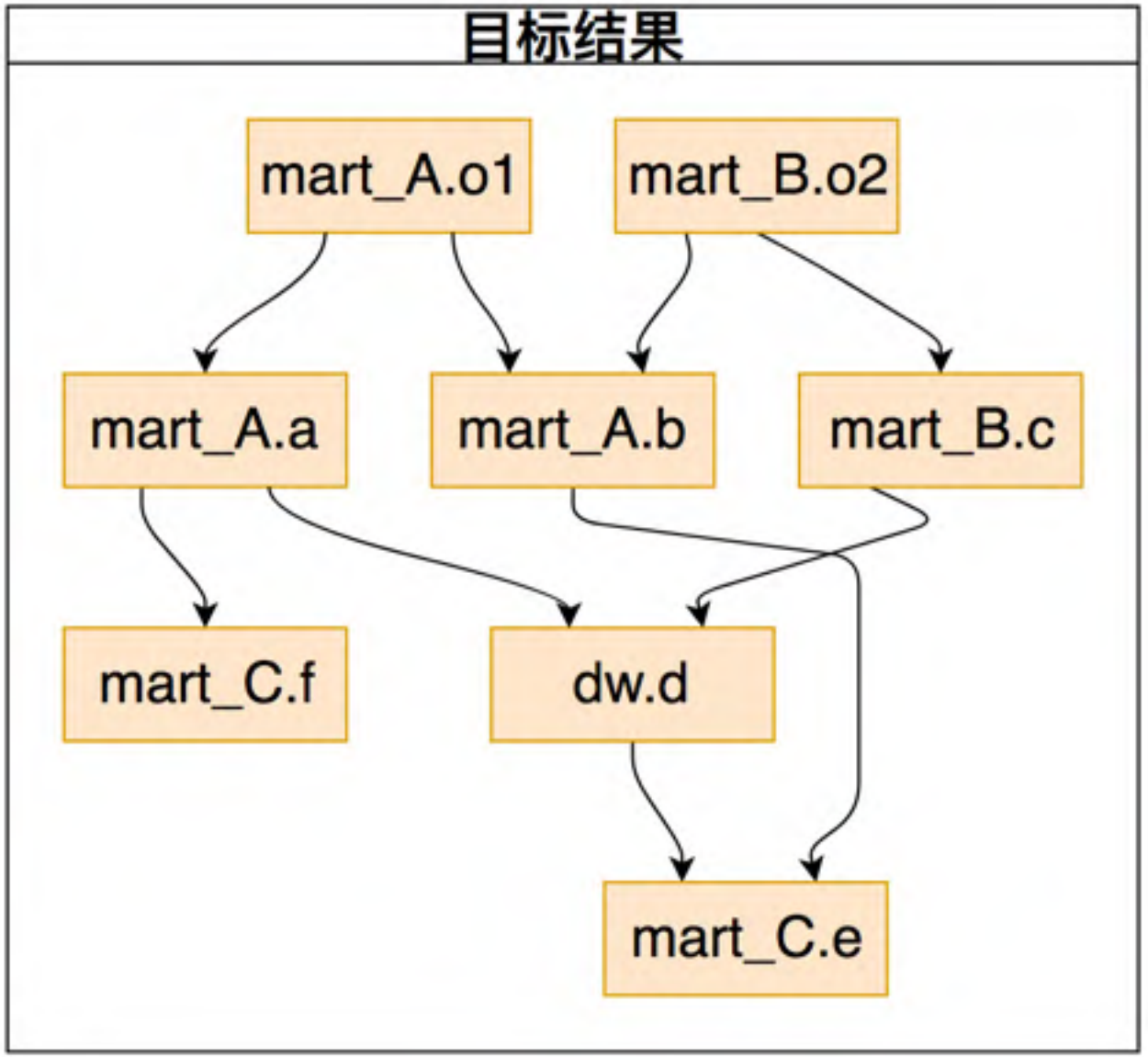
原点评侧拆库

- 我们有 8000 个这样的任务要改写，同时这些任务与表互相依赖，改一个表，依赖它的任务都得改一次，只能分层推动，项目周期不可控

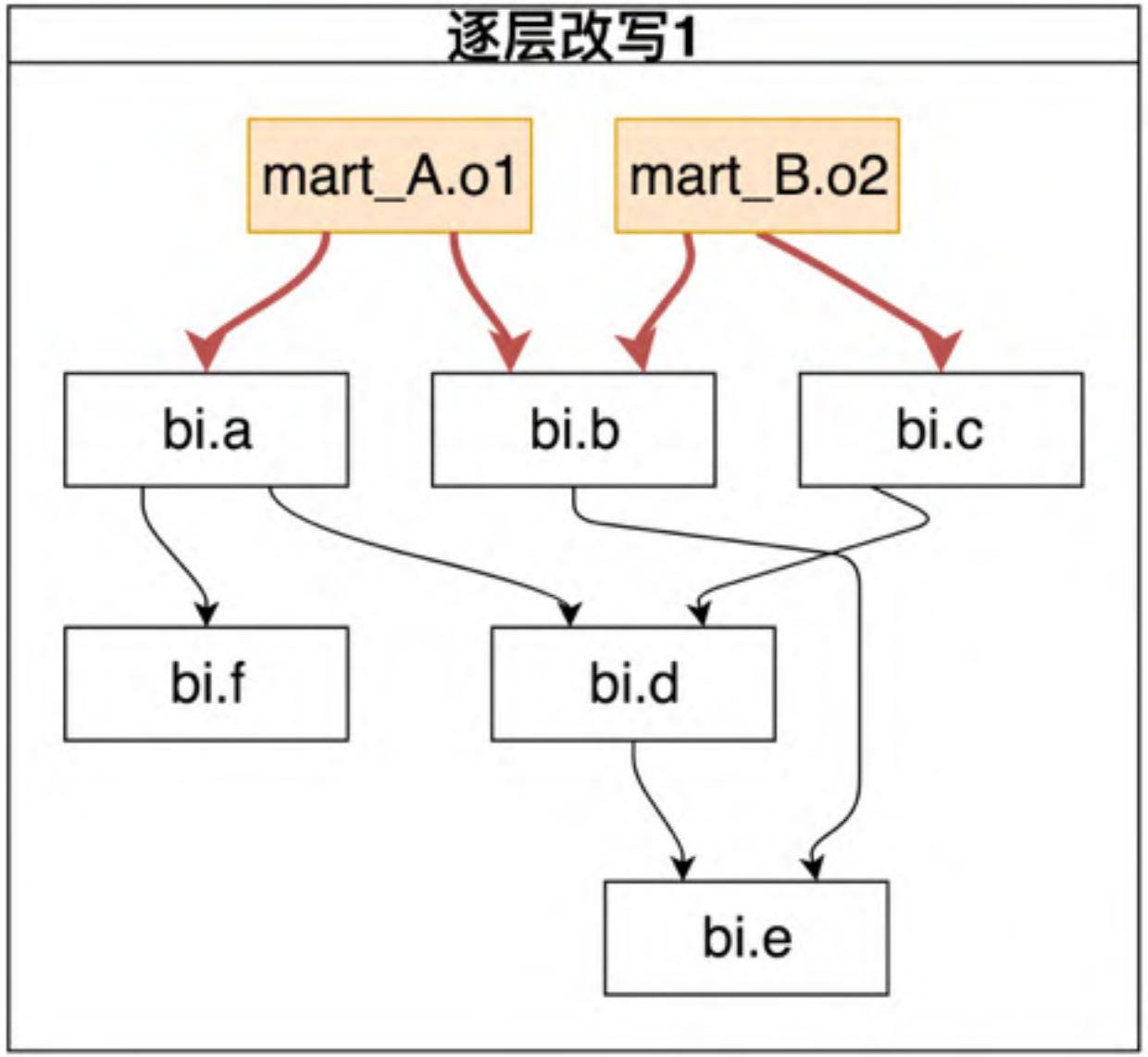


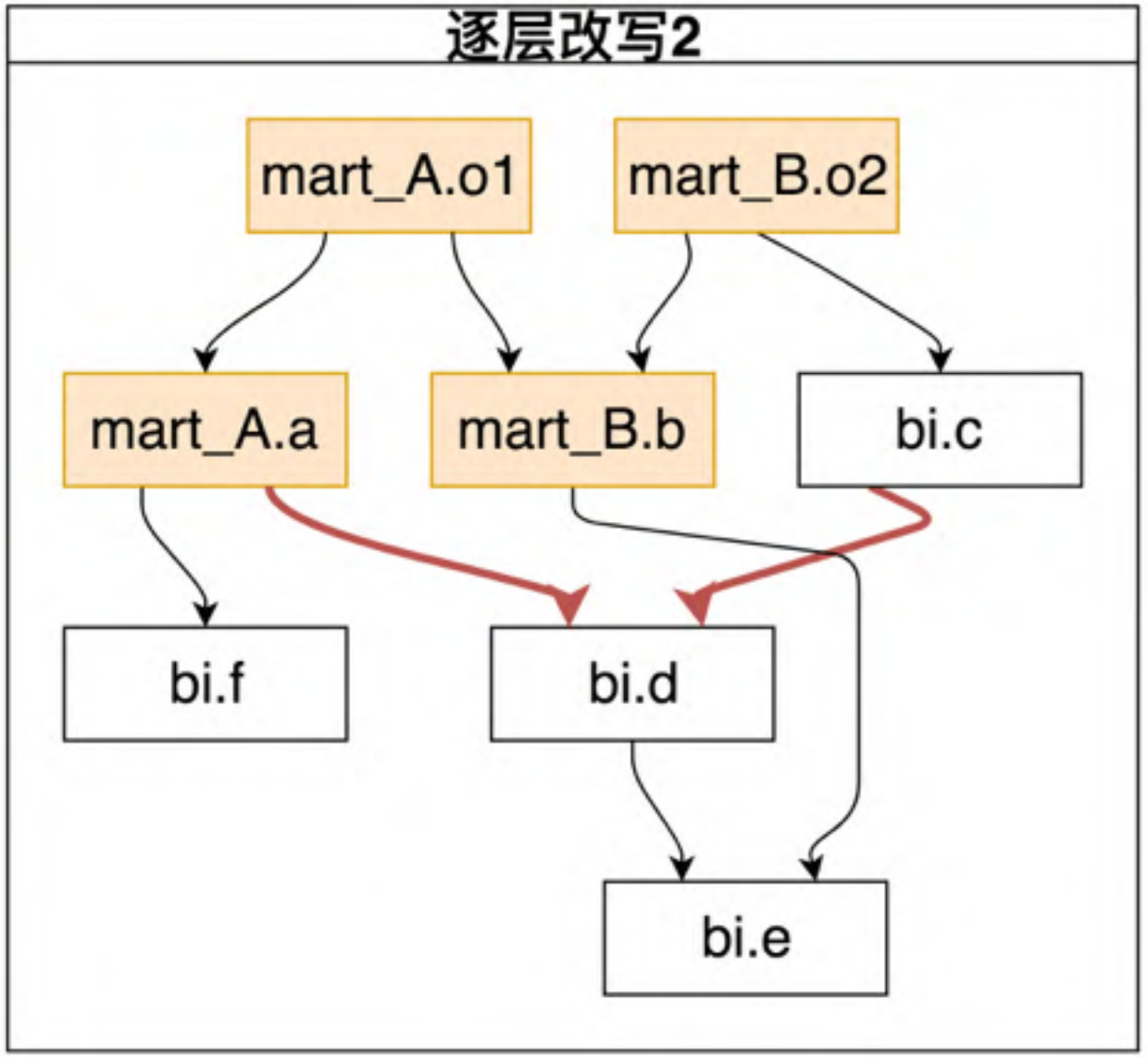
原点评侧拆库 – 目标


原始结构
bi.o1
bi.o2
bi.a
bi.b
bi.c
bi.f
bi.d
bi.e


目标结果
mart_A.o1
mart_B.o2
mart_A.a
mart_A.b
mart_B.c
mart_C.f
dw.d
mart_C.e

逐层改写方案


逐层改写1
mart_A.o1
mart_B.o2
bi.a
bi.b
bi.c
bi.f
bi.d
bi.e


逐层改写2
mart_A.o1
mart_B.o2
mart_A.a
mart_B.b
bi.c
bi.f
bi.d
bi.e

我们的方案

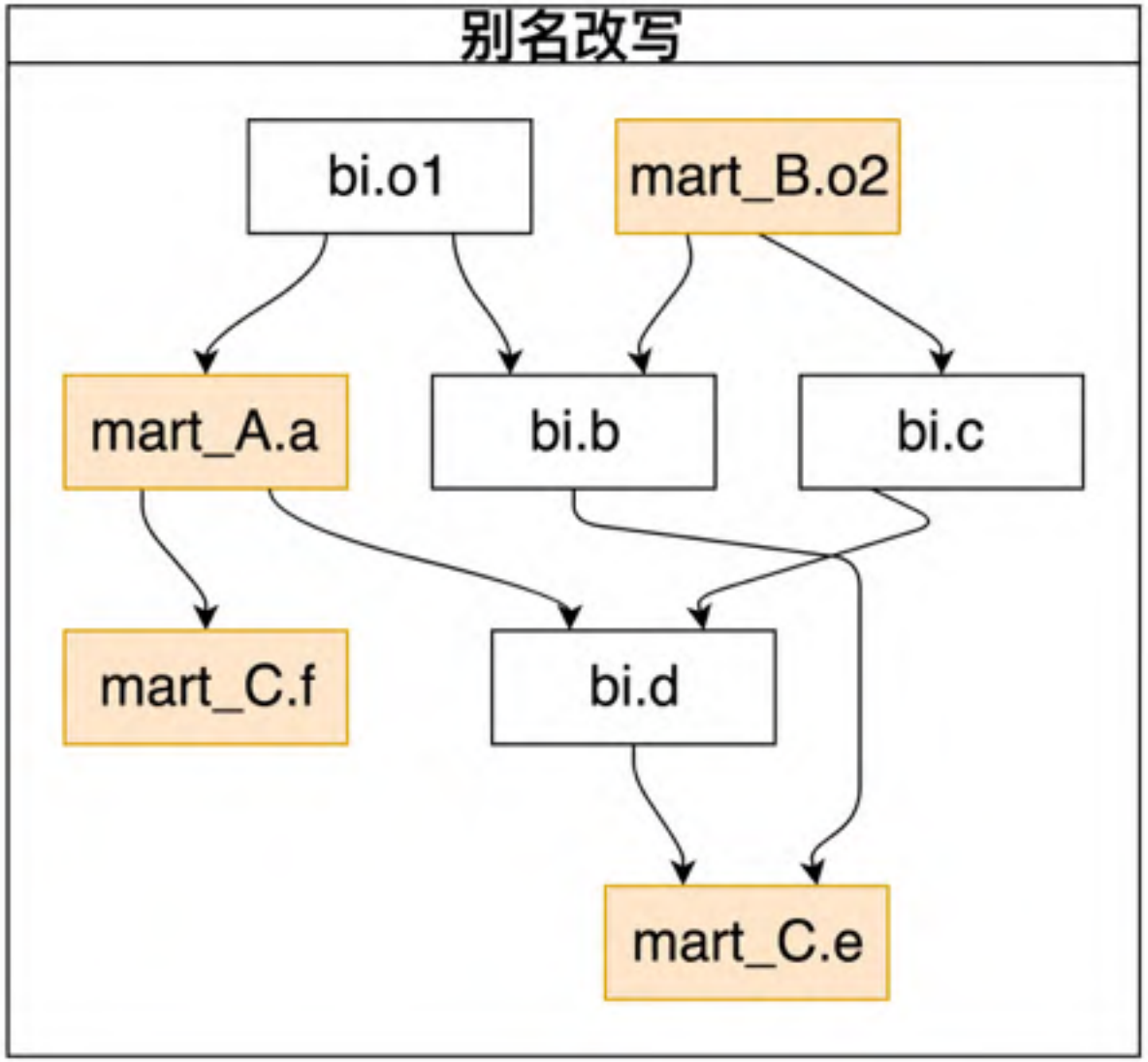
通过Hive元数据访问层做新名称别名映射

迁移任意表名不需要双跑

不需要同步修改下游任务内容

改写顺序任意, 高度并行

改写进度可打点分析



Hive元数据库表映射

具体实施流程

- 梳理业务, 确定映射关系
- Hive元数据入口上线别名能力, 形成目标库表名对原有路径映射
- 分批改写, 单任务内以及任务链条内均不做强制要求, 新旧表名”双跑”
- 基于Hive元数据以及HDFS访问审计, 确认依赖清理完成
- HDFS&Hive元数据层面同时进行 “物理切分” 拆除映射关系

效果

两个季度完成了7000任务中90%的批量改写上线

总结与展望

未来 – 常态化多机房方案

- 之前的方案以整体搬迁为主, 新的目标
  - 支持常态异地单集群多机房Hadoop服务 - 入口, 元数据统一
  - 为用户提供多机房计算资源和存储副本管理能力 - 跨机房带宽资源意识
  - 保持原有生产查询模式 - 用户不介入的切分/迁移
- 设计思路
  - 基于访问数据依赖分析的业务拆分方案
  - 基于yarn标签调度和多级队列的资源划分与调度方案
  - NameNode元数据中新增文件粒度 block副本分布策略标记
  - 写出后增加同步数据阶段, 读取数据时, 独立服务接管跨机房请求, 统一块粒度预拷贝
  - 网络层带宽使用监控

- 本质: 跨机房数据分布是个 Cache 管理的事

反思 – 技术换运营

产生原因:

- 数据平台对上层不只是RPC接口 (还有数据表, 计算任务)
- 从开源系统拼接到平台化, 是一个规范(限制)增多的过程
- 平台的变更即使做到 “兼容”, 历史包袱也需要尽快 “扫清”

- “可运营性”:
  - 可灰度
  - 可关门
  - 进度可知
  - 分工可知
  - 变更兼容/替代方案

- 使用技术降低 “运营” 成本:
  - 找核心问题, 是否能规避 “运营”
  - “运营” 并行化, 批量化, 自动化
  - 小范围试用找问题
  - 引入的架构复杂度后期可回收

复杂系统重构与融合

- 不可能不出问题, 关键在于影响范围可控

- 清晰明确定义 “目标”, 并拆分到团队, 保证团队间并行推动 (“流水线”)
- 团队内目标再拆分为变更
- 变更 -> 分析影响 -> 改动/测试 -> 制定上线/回滚方案
- 变更间依赖分析 -> 上线排期 -> 通告 -> 上线 -> 验证

- 监控: 知道发生了什么
- 抓手: 能现场调整什么

复杂系统重构与融合


美团
meituan.com

让创新技术推动社会进步

HELP TO BUILD A BETTER SOCIETY WITH
INNOVATIVE TECHNOLOGIES

Geekbang.
极客邦科技

InfoQ

专注中高端技术人员的技术媒体

EGO EXTRA GEEKS' ORGANIZATION NETWORKS

高端技术人员学习型社交平台

StuQ 斯达克学院

实践驱动的 IT 教育平台

地址：北京市朝阳区洛娃大厦C座8层1801室　　网址：www.geekbang.org