基于Hadoop快速构建离线elasticsearch索引

钟华

滴滴
滴滴一下 美好出行

目录

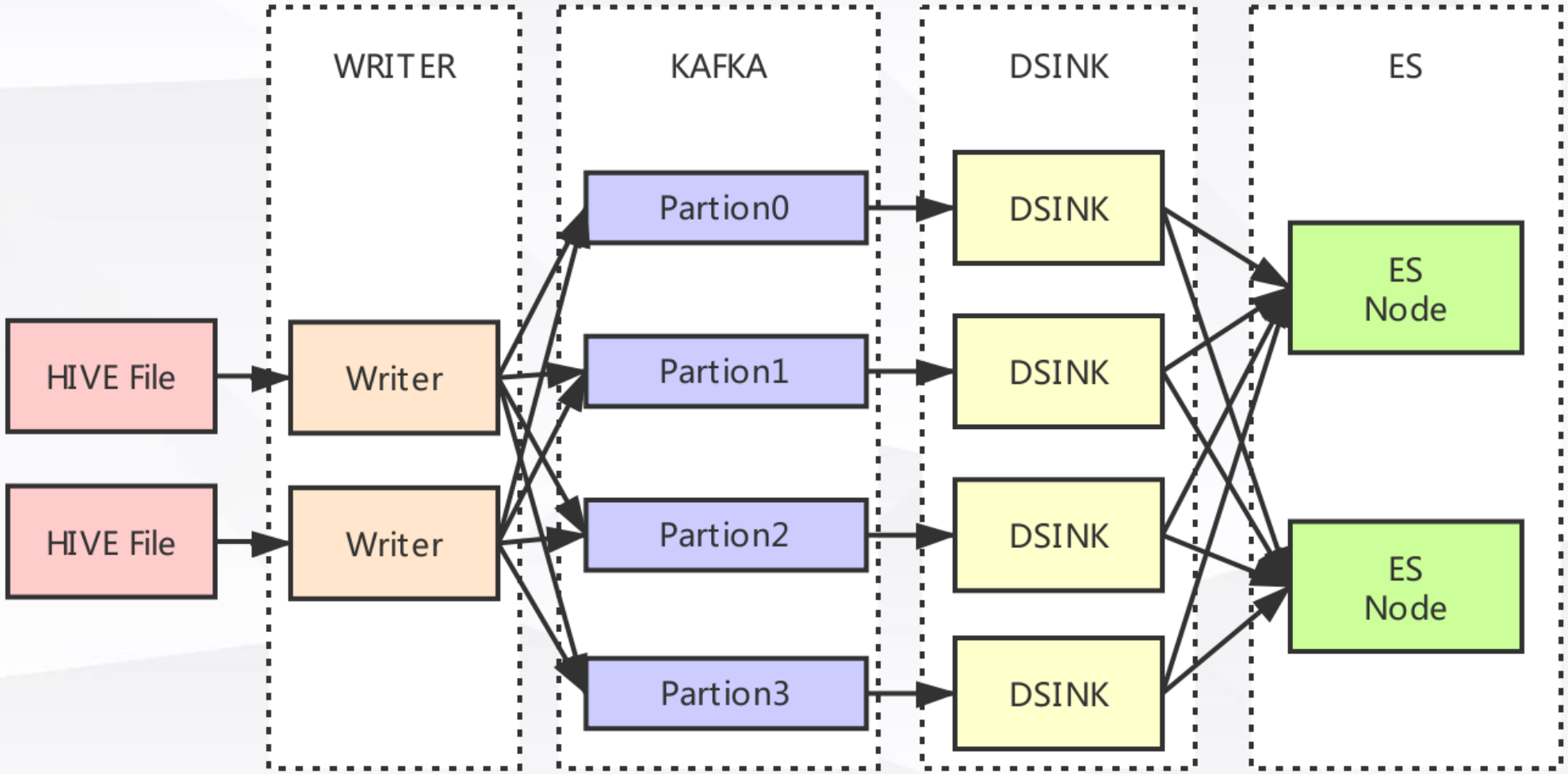
背景

背景 - 标签系统


服务
对象
运营人员
分析师
用户分群
人群提取
人群判定服务
服务
内容
目标
运营活动
新手保护
客服支持
人群AB实验
司乘定向教育
广告投放

背景

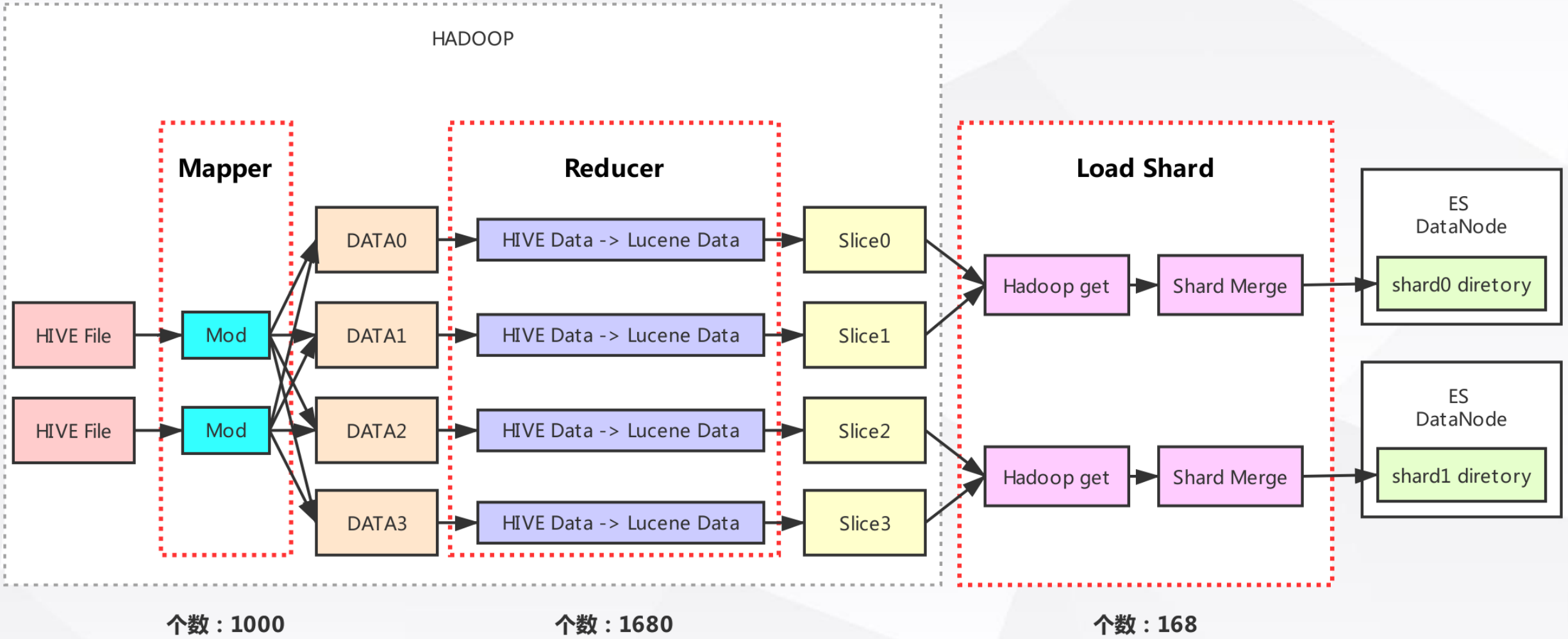
老架构


WRITER
KAFKA
DSINK
ES
HIVE File
HIVE File
Writer
Writer
Partion0
Partion1
Partion2
Partion3
DSINK
DSINK
DSINK
DSINK
ES
Node
ES
Node

背景

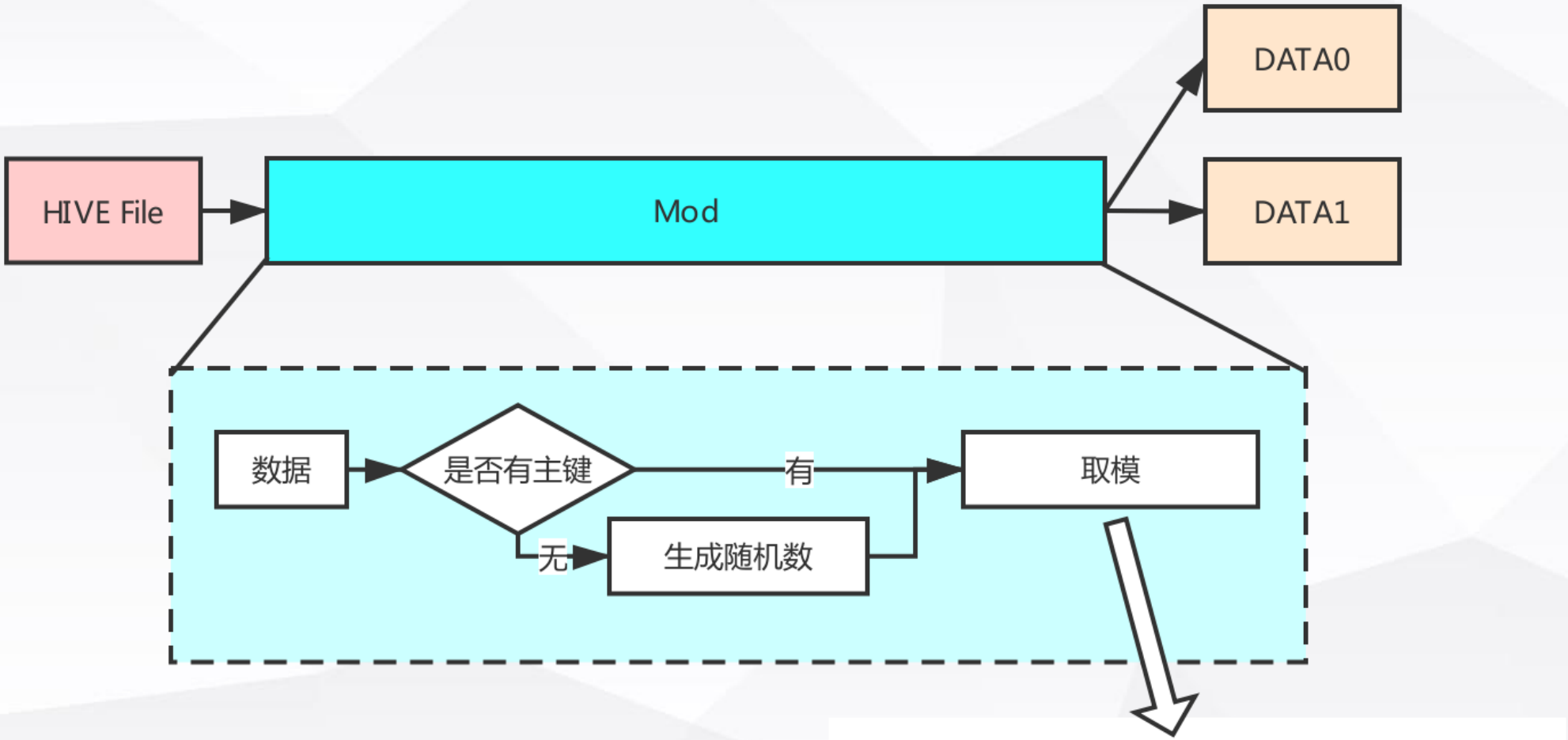
	2016	2019
乘客标签数据量	2.5TB	34TB
数据导入时间	2小时	8小时

新架构


HADOOP
Mapper
Reducer
Load Shard
HIVE File
HIVE File
Mod
Mod
DATA0
DATA1
DATA2
DATA3
HIVE Data -> Lucene Data
HIVE Data -> Lucene Data
HIVE Data -> Lucene Data
HIVE Data -> Lucene Data
Slice0
Slice1
Slice2
Slice3
Hadoop get
Shard Merge
Hadoop get
Shard Merge
ES
DataNode
shard0 diretory
ES
DataNode
shard1 diretory
个数：1000
个数：1680
个数：168

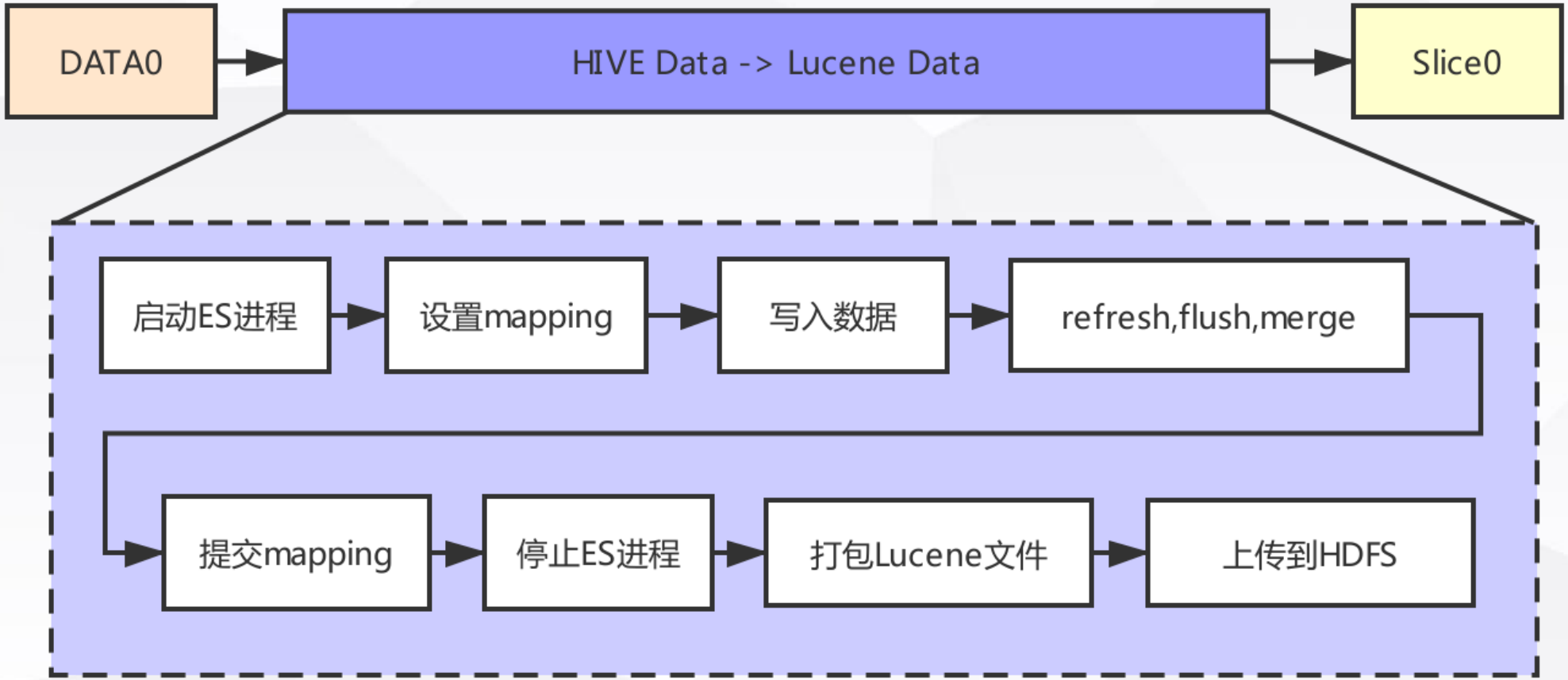
架构与实现

Mapper

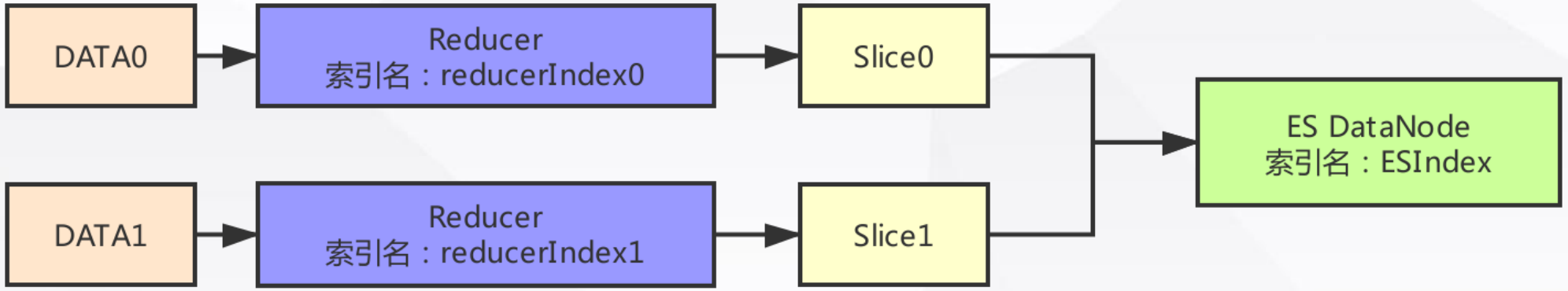



```
int hash = murmur3hash(主键);
return mod(hash, slice个数);
```

Reducer


DATA0
HIVE Data -> Lucene Data
Slice0
启动ES进程
设置mapping
写入数据
refresh,flush,merge
提交mapping
停止ES进程
打包Lucene文件
上传到HDFS

Reducer - mapping冲突

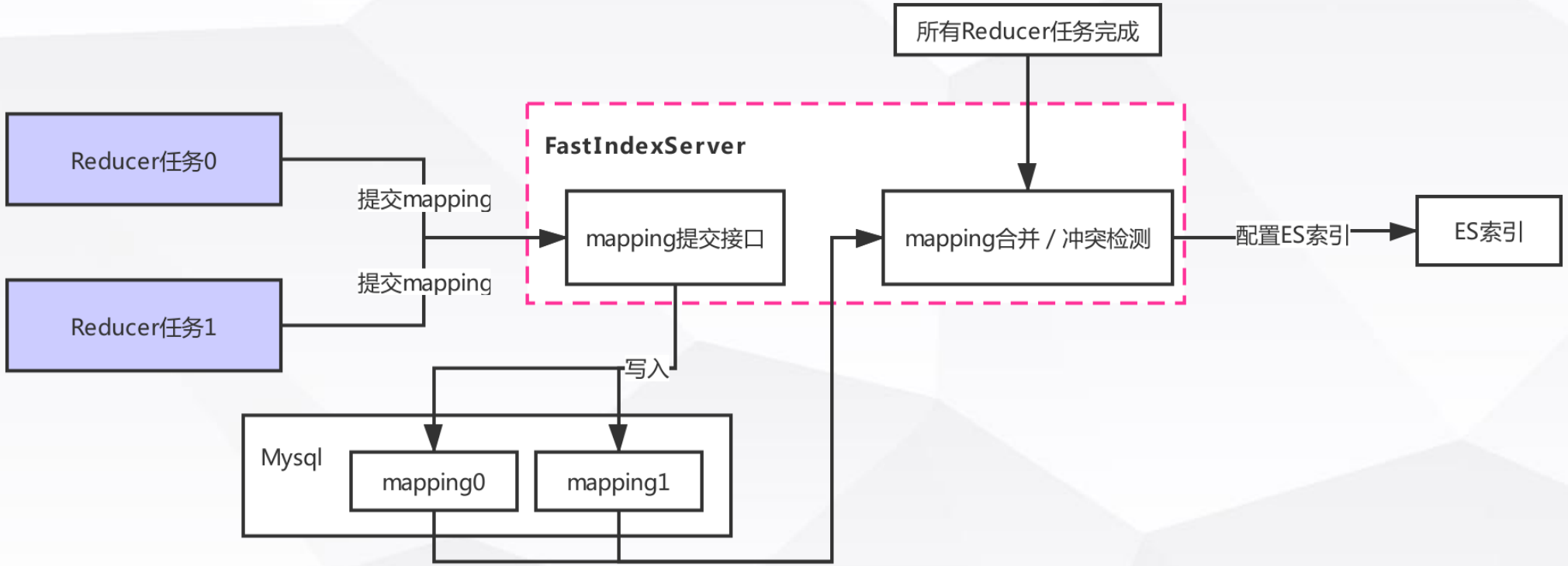


正常情况：reducerIndex0.mapping == reducerIndex1.mapping &&
reducerIndex1.mapping == ESIndex.mapping

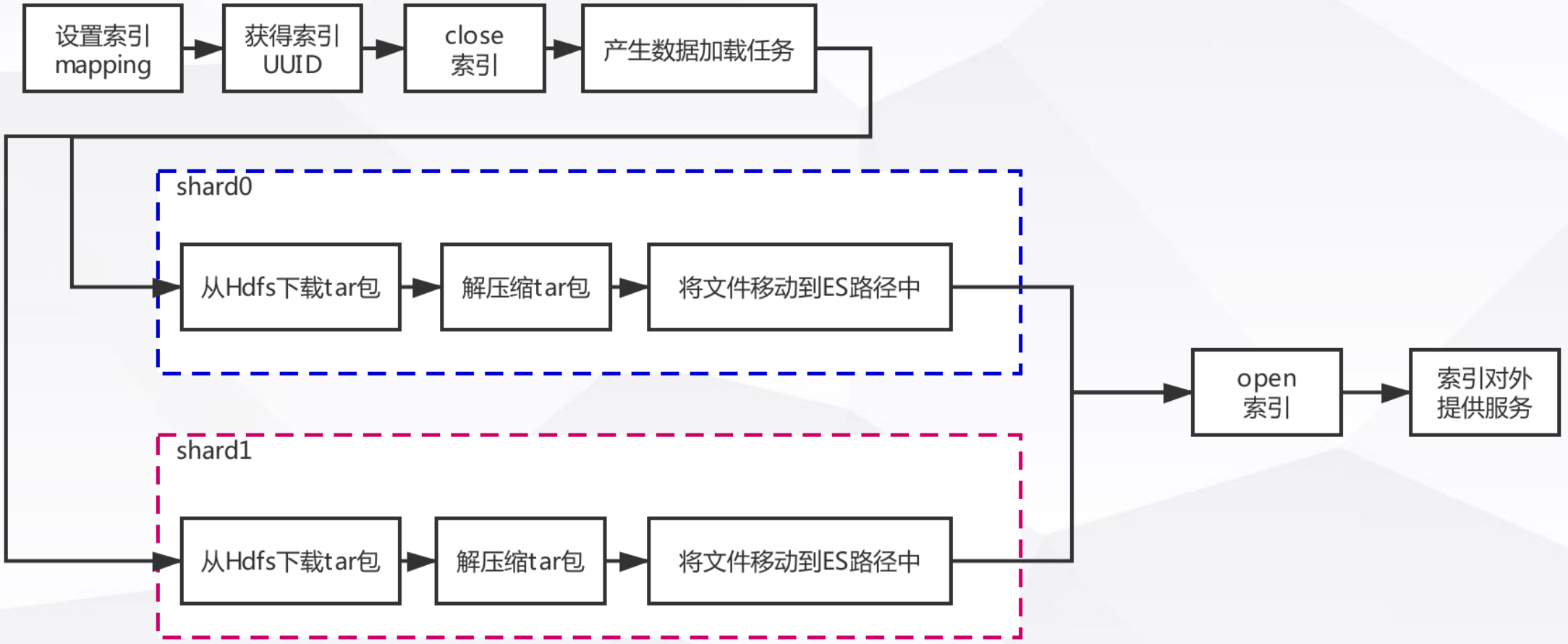
Mapping冲突：reducerIndex0.mapping ！= reducerIndex1.mapping ||
reducerIndex1.mapping ！= ESIndex.mapping

Reducer – mapping字段映射

Reducer – mapping冲突检测



Load Shard



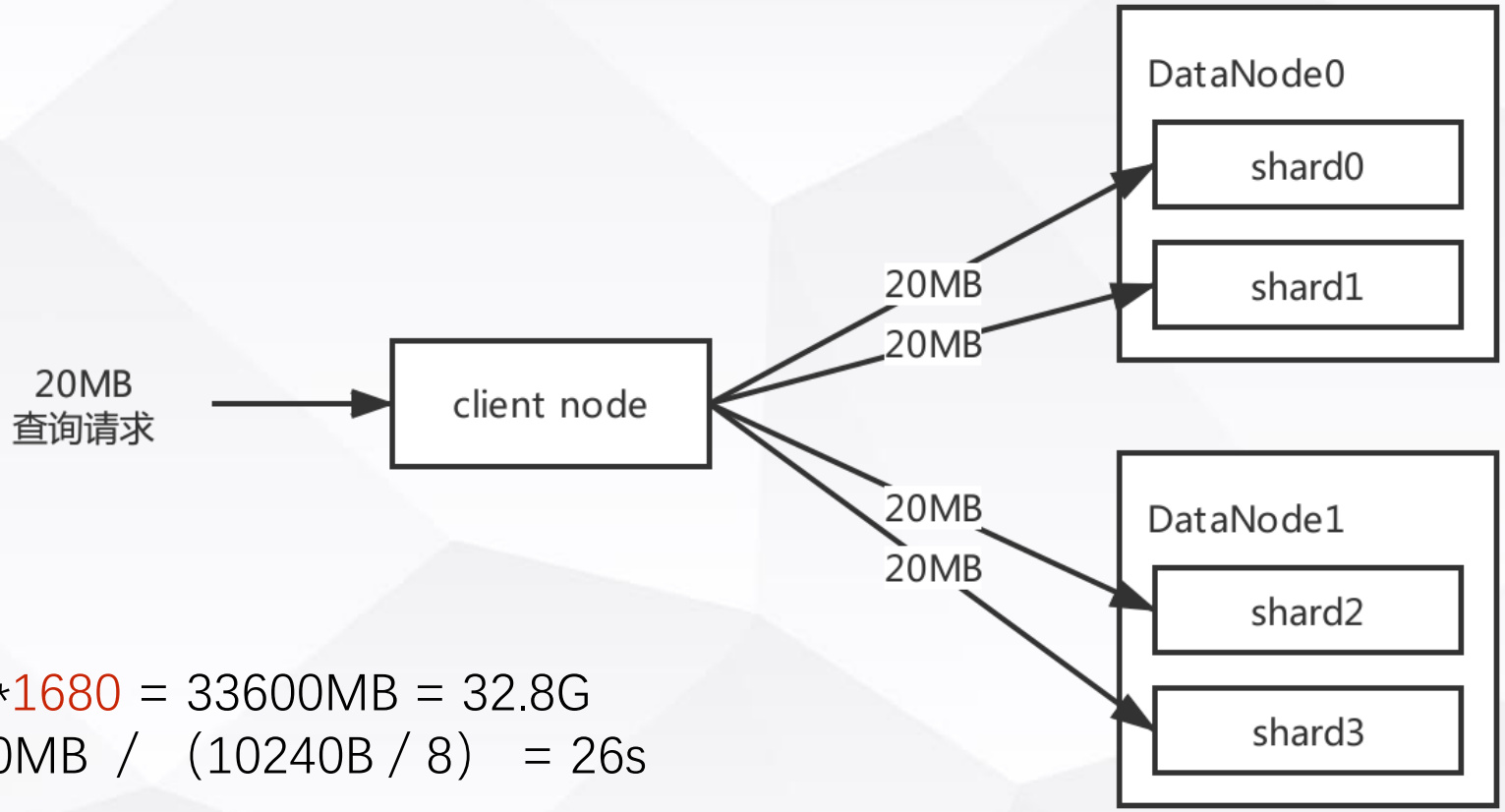
ES路径：{elasticsearch路径}/nodes/0/indices/{索引|UUID}/{索引|shardID}/index/

优化与踩坑

优化

- 性能
  - 单个任务，Lucene文件大小保持在2GB以下，merge时间为4分钟
  - 多线程写入，reducer时间从60分钟降低为30分
- 限速
  - 控制整体数据加载速度
  - 控制单个ES节点数据加载速度
- 稳定性
  - Mapper任务和reducer任务重试5次才返回失败
  - 数据加载任务配置20分钟重试

踩坑



1680个shard:

带宽：20MB*1680 = 33600MB = 32.8G

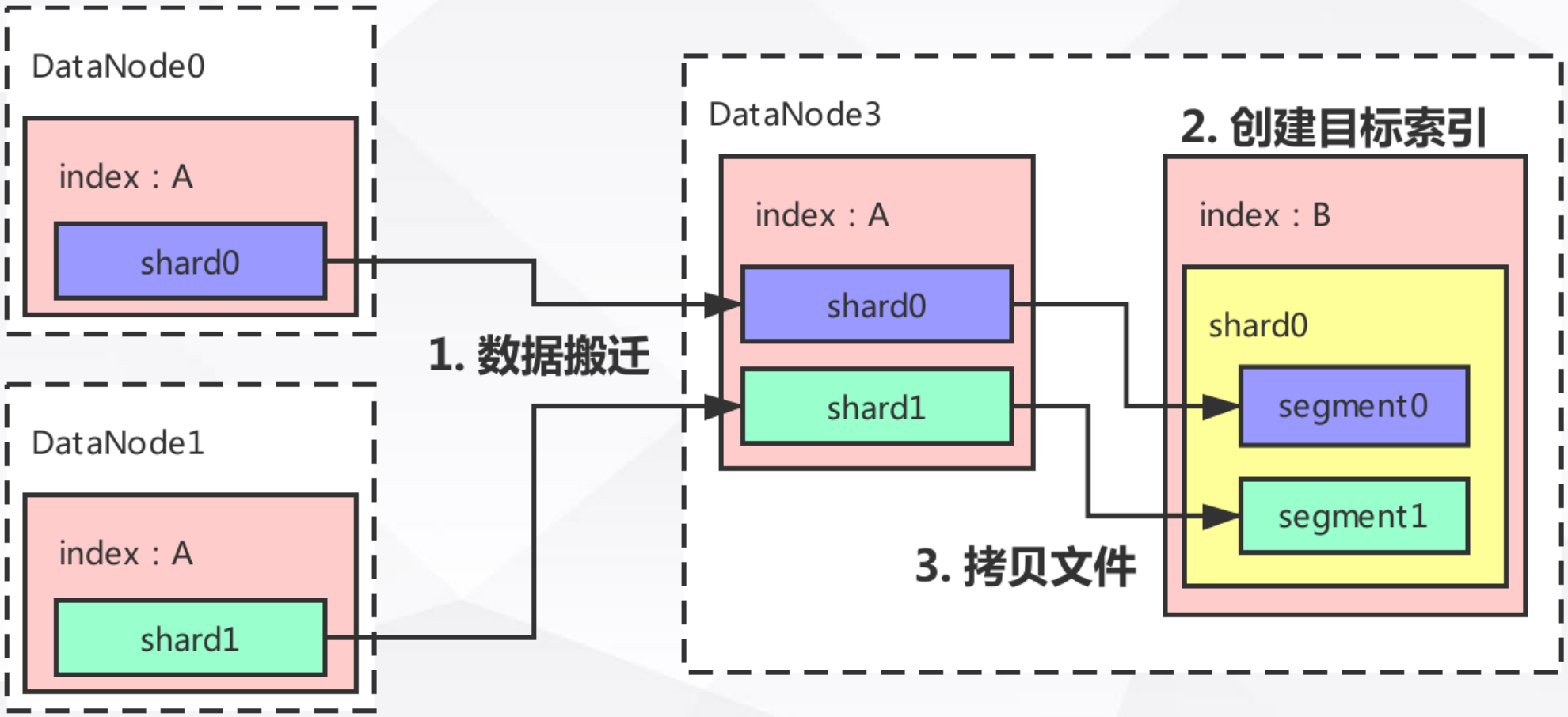
耗时： 33600MB / （10240B / 8） = 26s

168个shard:

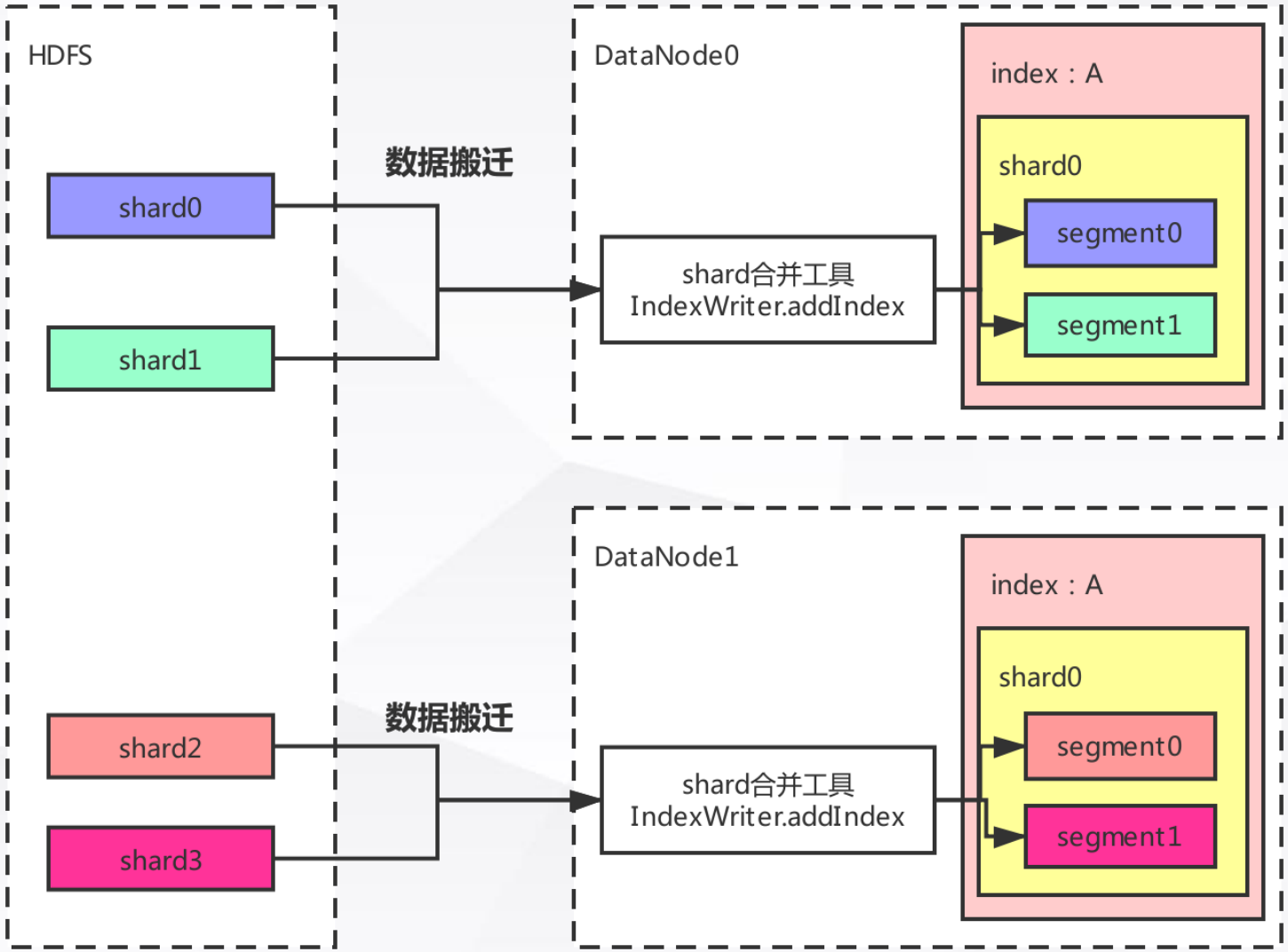
带宽：20MB*168 = 3360MB = 3.28G

耗时： 3360MB / （10240B / 8） = 2.6s

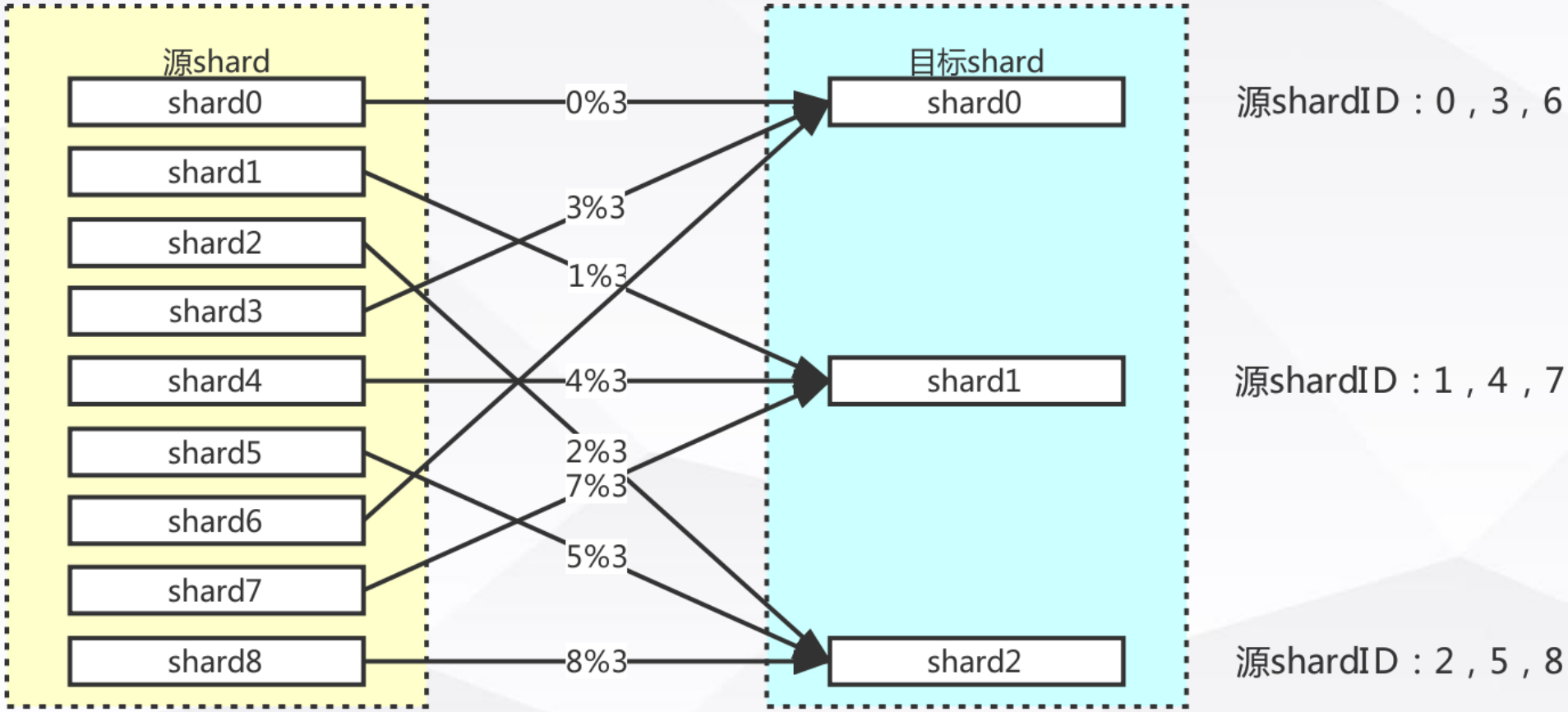
踩坑 – shrink shard功能



踩坑 - shard合并


HDFS
shard0
shard1
shard2
shard3
数据搬迁
数据搬迁
DataNode0
index：A
shard0
segment0
segment1
shard合并工具
IndexWriter.addIndex
DataNode1
index：A
shard0
segment0
segment1
shard合并工具
IndexWriter.addIndex

踩坑 - shard id映射



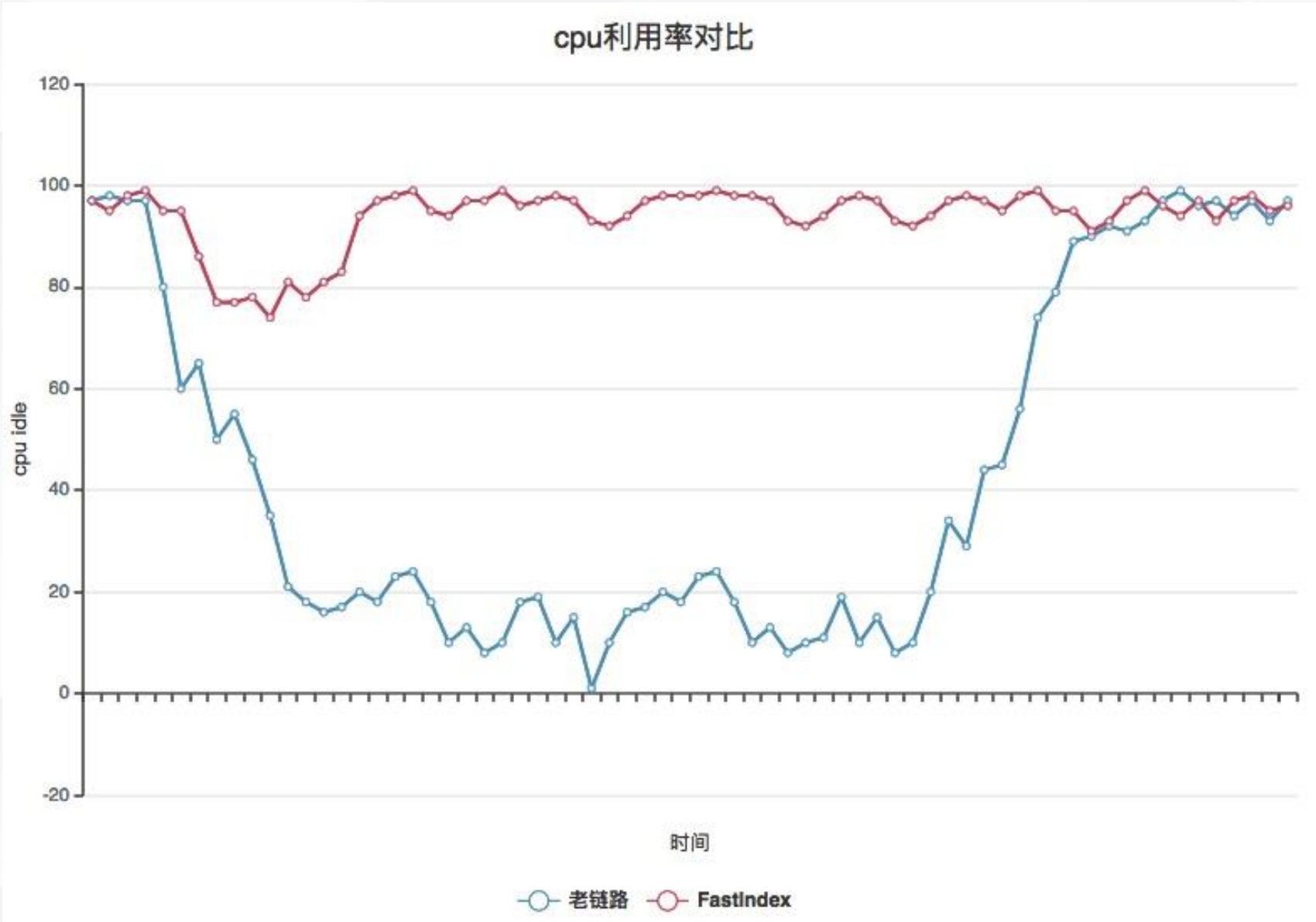
目标shardID = 源shardID % 目标shard个数

源shard个数 = 目标shard个数*N

收益

	老链路	FastIndex
成本	7W/月	3W/月
写入时间	8小时	1.5小时

收益

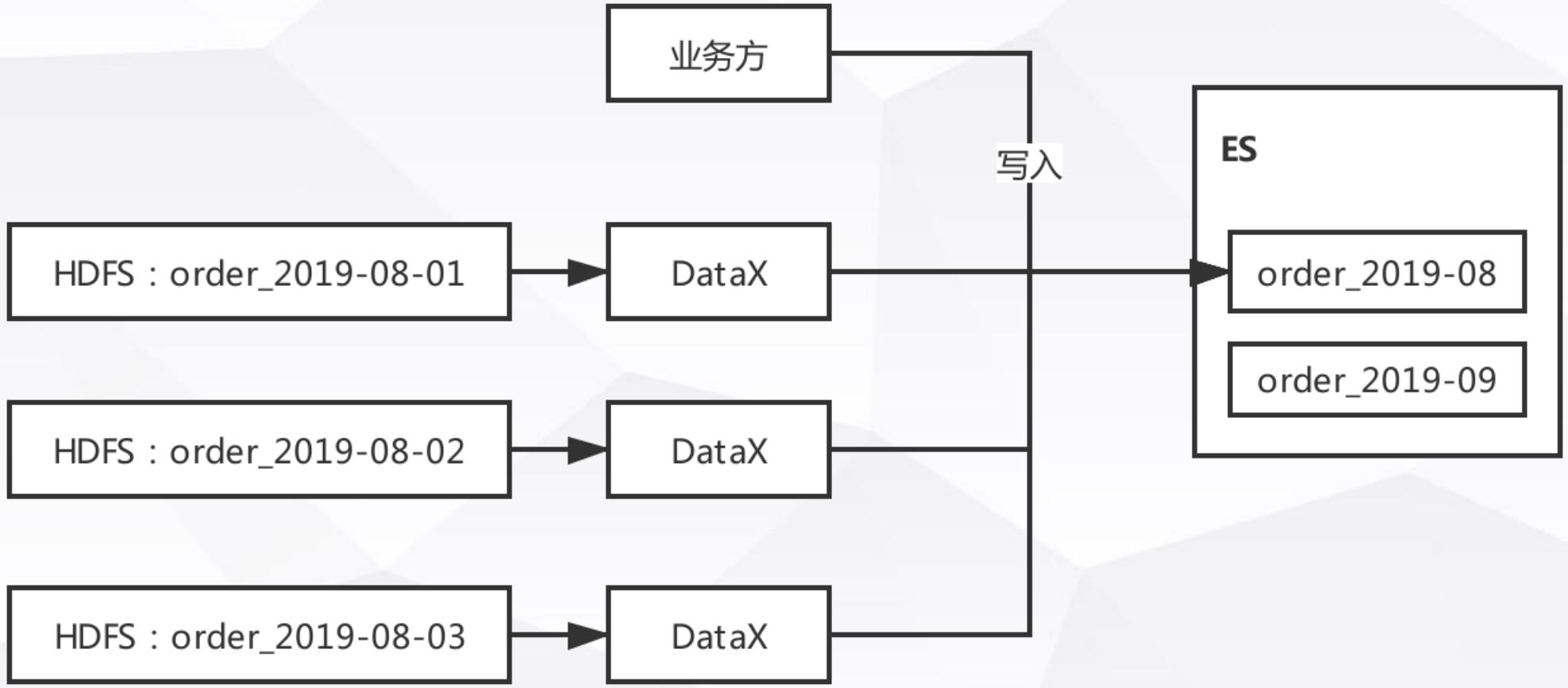


规划

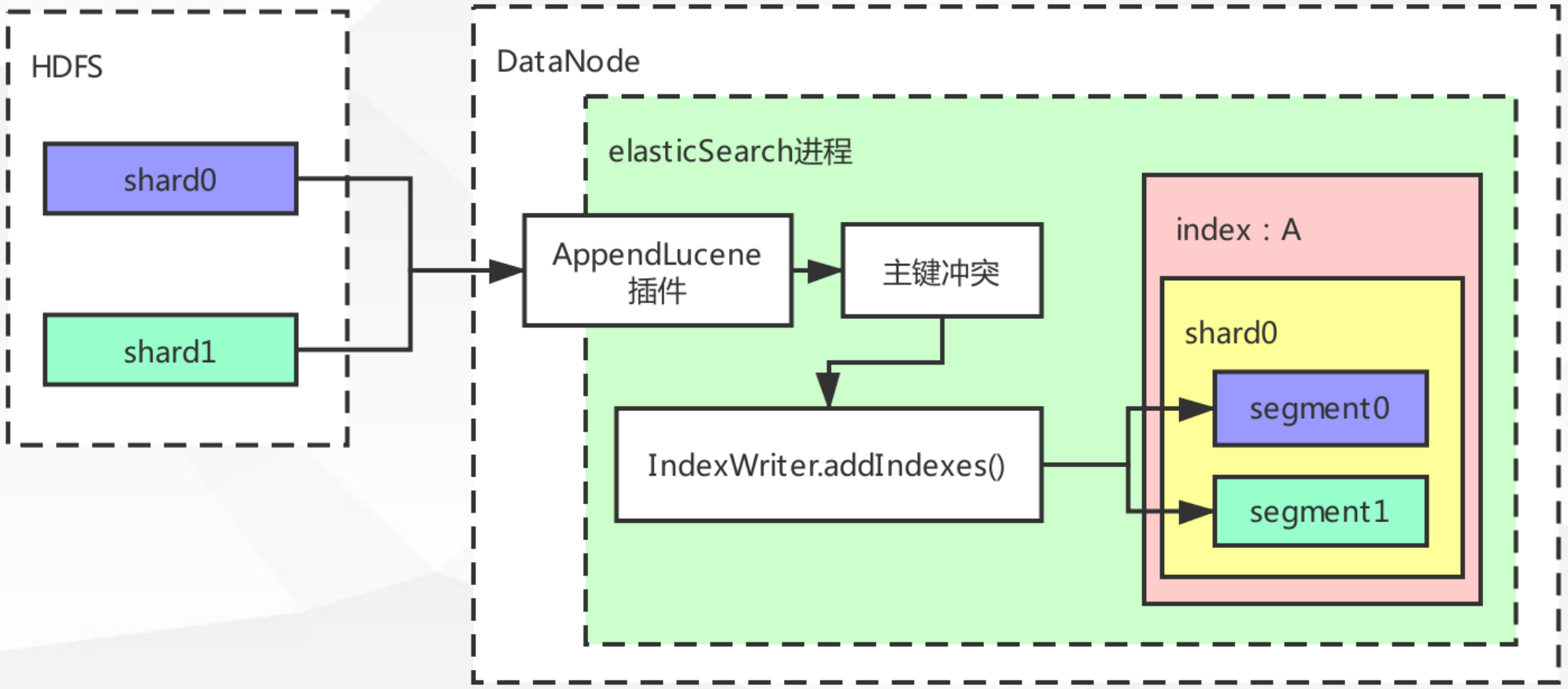
对比 - DataX

	FastIndex	DataX
成本	低	高
导入时间	分钟级	小时级
资源利用率	高	低
查询稳定性	导入过程不使用 es cpu资源	导入过程使用 大量cpu资源

规划-实时写入


业务方
写入
ES
HDFS：order_2019-08-01
DataX
order_2019-08
order_2019-09
HDFS：order_2019-08-02
DataX
HDFS：order_2019-08-03
DataX

规划


HDFS
shard0
shard1
DataNode
elasticSearch进程
AppendLucene
插件
主键冲突
IndexWriter.addIndexes()
index：A
shard0
segment0
segment1

THANKS