ClickHouse 在头条内部技术演化

陈星

自我介绍

- 10 year in database kernel(OLAP, warehouse) RD
- Worked on DB2 LUW, DB2 BLU(dashDB), BigSQL(SQL on Hadoop)
- 1+ years in Bytedance, and incubate ClickHouse development and deployment there

目录

ClickHouse 简介

1. Developed by Yandex, and open source since 2016

2. 查询性能优越的分析型引擎

3. 主要特点（not new)
   - Column oriented + vector execution
   - Local attached storage (not Hadoop ecosystem)
   - Linear scalable & Reliable(shard + replication)
   - SQL interface
   - Fast

ClickHouse 简介 - 性能优越的因素

1. Data Skipping
   - 分区以及分区剪枝
   - 数据局部有序（LSM-like engine, zone map)

2. 资源的垂直整合
   - 并发 MPP+ SMP（plan level)
   - Tuned执行层实现（multi-variant agg implementation..., SIMD)

3. C++ Template Code

ClickHouse 简介 - 适用场景与不足

1. 适用场景
   - 单表分析 或 colocate join case
   - distributed join 性能并不出色
2. 不足
   - no transaction
   - batch data ingest
   - weak update/delete support
   - weak optimizer & query rewrite

Bytedance 如何使用ClickHouse

1. 选择ClickHouse的原因

产品需求

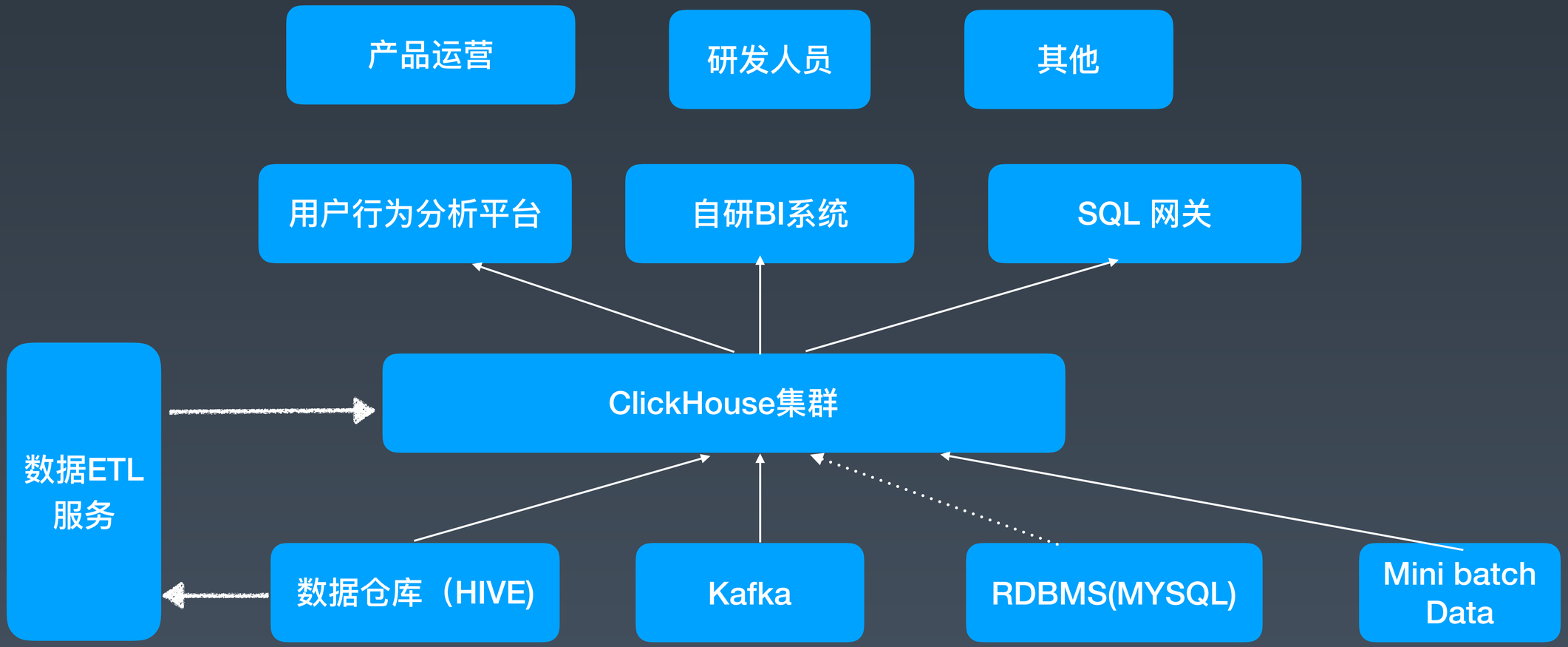
- 交互式分析能力 (in seconds)
- 查询模式多变
- 以大宽表为主
- 数据量大

Open sourced MPP OLAP engine - (performance, feature, quality)

Bytedance 如何使用ClickHouse

1. 几千个节点, 最大集群1200个节点
2. 数据总量 ~ 几十PB
3. 日增数据 ~ 100TB
4. 查询响应时间(mostly) ~ ms - 30s
5. 覆盖下列用户
   - 产品运营，分析师
   - 开发人员
   - 少量广告类用户
   - openapi

Bytedance 如何使用ClickHouse



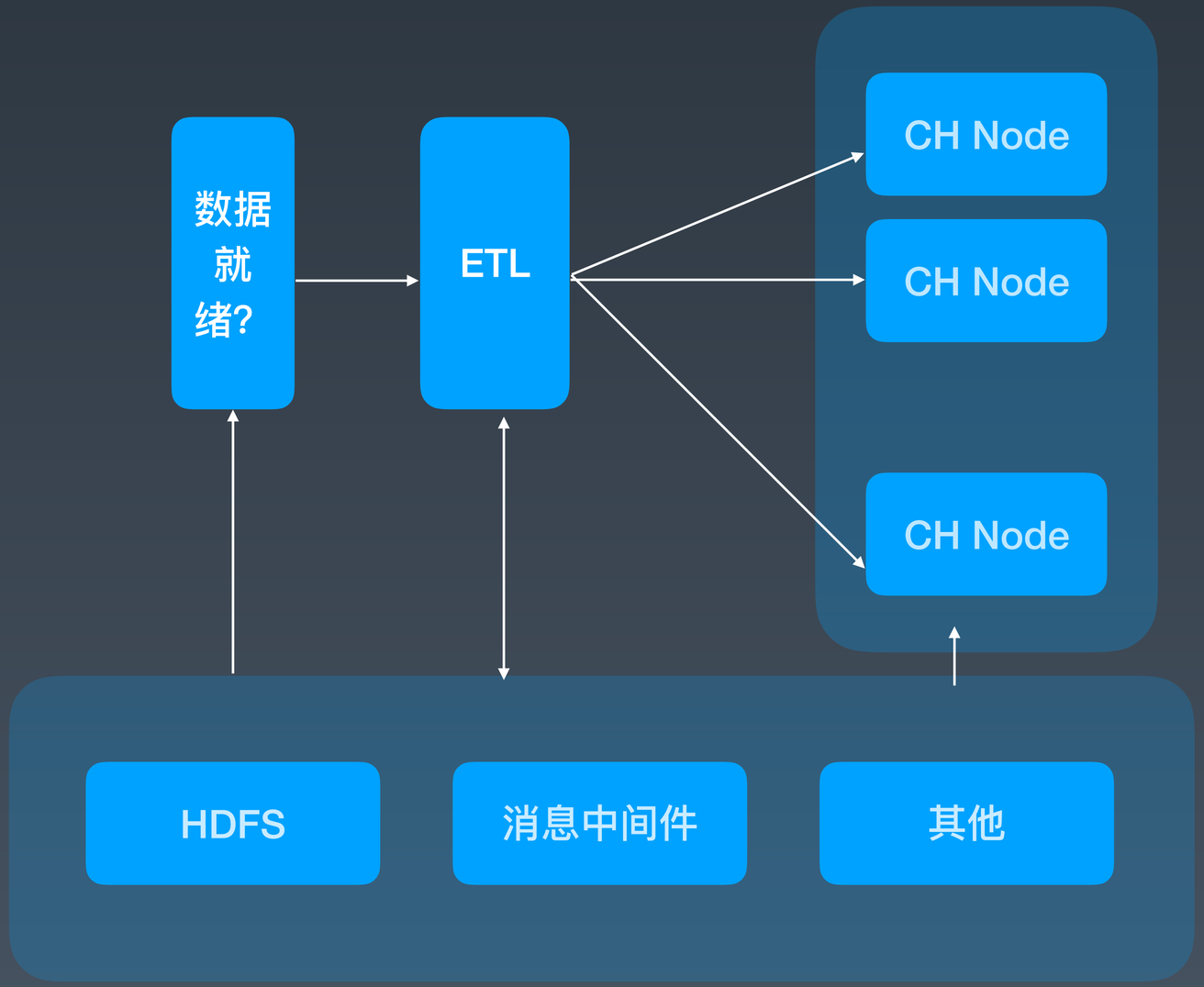
Bytedance 如何使用ClickHouse

- 多种数据源（离线 + 实时 + ...）
- 交互式分析
- 数据处理链路对业务方透明

满足数据中台对数据查询需求。

问题与解决方案

数据源->ClickHouse 服务化



服务化与自动化中的问题

1. HDFS 数据访问
2. 数据导入过程中Fail over
3. CH数据就绪速度(Part生成）

数据源->ClickHouse 服务化

1. 增加 HDFS 数据访问能力（HDFS client porting from HAWQ)
2. ETL服务维护外部事务保障数据一致性（fail over)
3. INSERT INTO LOCAL TABLE

4. 数据构建与查询分离 (experimental feature)

Map 数据类型 - 动态Schema

1. 客户端上报字段多变 (自定义参数)
2. 数据产品需要相对固定schema

Engine fix this Gap by Map type

性能需求： 访问MAP 键值需要与访问POD 类型的列速度一致

实现方式： LOB ？ Two-implicit column? Other

Map 数据类型 - 动态Schema

1. 数据特征： # keys 总量可控， 局部有限
2. 局部（ PART level)展平模型 (自描述 ）

{‘a’: 1, ‘b’:2 }　{‘a’:1，‘c’:3} 会在存储层表示为column_a, column_b, column_c 三个列，对应的值如下图所示：

column_a	column_b	column_c
1	2	N
1	N	3

Map 数据类型 - 动态Schema

1. MAP键访问自动改写
   e.g. “select c_map{‘a’} from table” will be rewrote to “select c_map_a from table”
2. MAP列访问 (代价较大)
   e.g. select c_map from table
3. Merge阶段优化 (无需重构MAP column)

4. 收益：
   - 自动化接入
   - Table schema 简化
   - 极大简化数据构建(ETL)逻辑

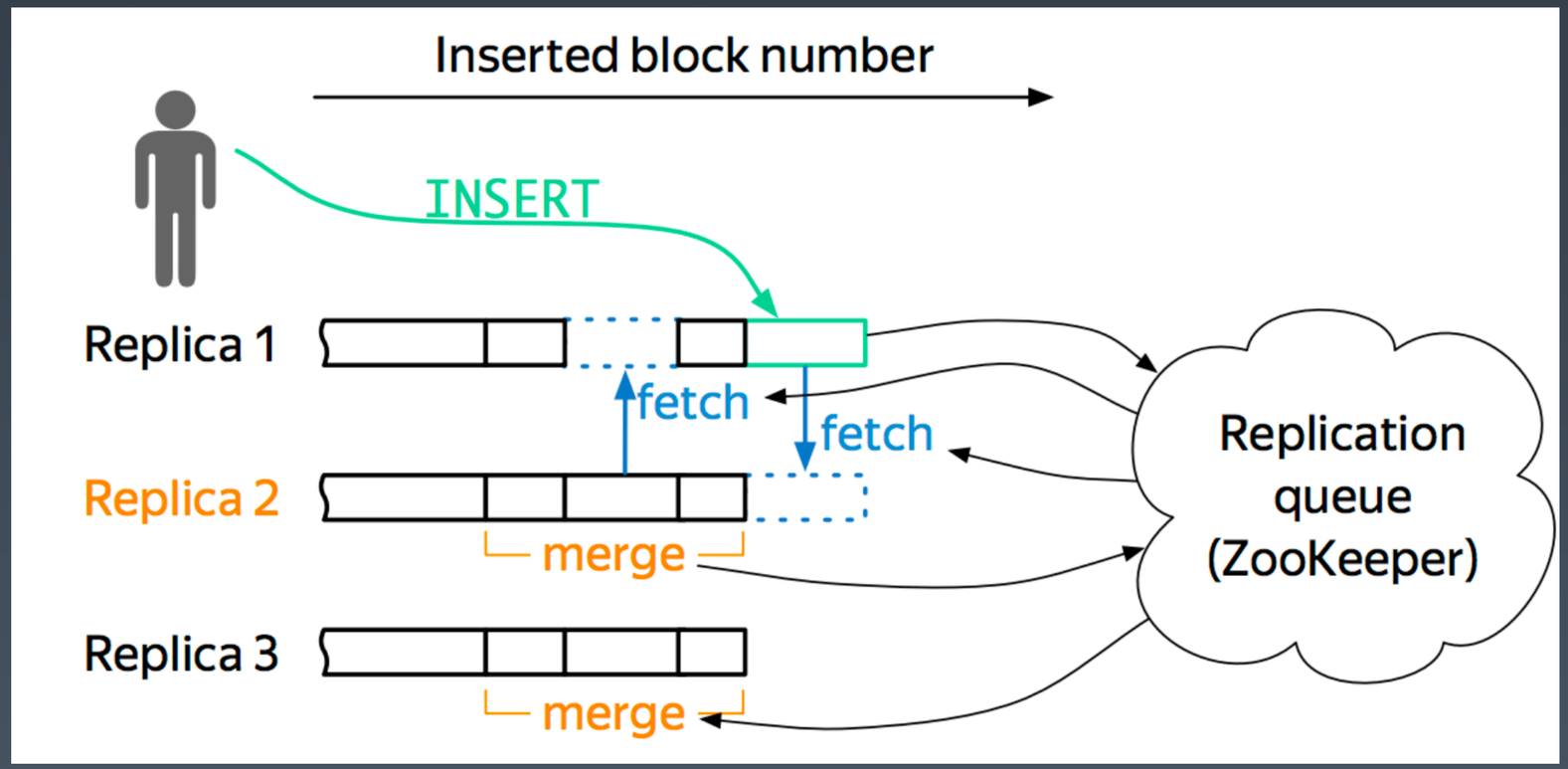
Map 数据类型 - 动态Schema

语法：

- Create table t(c1 UInt64, c2 **Map(String, UInt8)**) ENGINE=MergeTree….
- insert into t values(1, {'abc':1, ‘bcd':2})
- Select c2{‘abc’} from t

High Volume Data & High Availability (zookeeper 使用问题)

1. 两副本保障数据/服务
2. ReplicatedMergeTree in ClickHouse（ Issue)
   - Async Master-Master replication



High Volume Data & High Availability (zookeeper 使用问题)

1. ReplicatedMergeTree的问题
   - ZooKeeper压力大，znode太多
   - 400 Nodes 集群， 半年数据， 800万znodes
2. ReplicatedMergeTree use ZK to store:
   - Table schema
   - 副本状态 (part info & log info)
   - 分片 (shard) 状态

Catalog service + mini log service + coordinate service

High Volume Data & High Availability (Zookeeper问题)

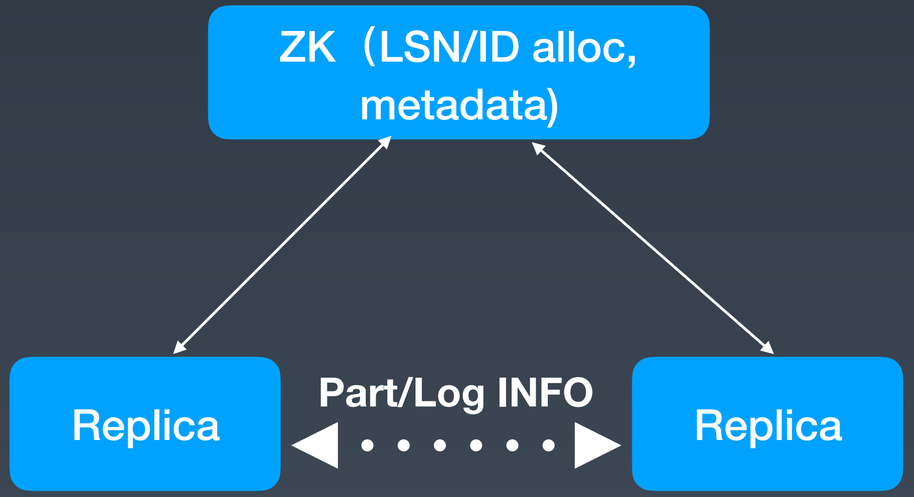
1. 数据继续增长会导致ZK无法服务

2. 社区mini checksum in zk能缓解内存使用，但不能解决问题

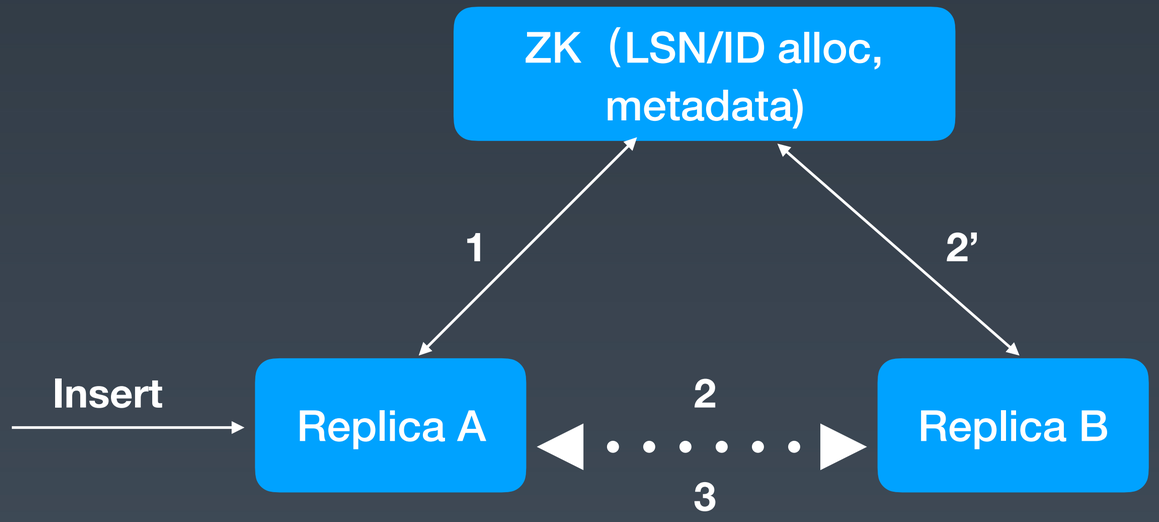
3. 基于MergeTree开发HA 方案

High Volume Data & High Availability - HaMergeTree

1. ZooKeeper只用作coordinate
   - Log Sequence Number(LSN) 分配
   - 数据Block ID 分配
   - 元数据管理
2. 节点维护local log service (action log)
3. Log 在分片内部节点间通过Gossip协议交互
4. 数据信息（parts) 按需交互
5. 外部接口与社区兼容（例如：multi-master写入）



High Volume Data & High Availability - HaMergeTree



1. A 获取LSN和Block ID
2. A Push log to active replica B
2’. B get its log lags from ZK and pull from A
3. B redo the log and get Block from A

High Volume Data & High Availability - HaMergeTree

1. ZooKeeper压力不会随着数据量增长

2. ~ 3M znodes in ZK

3. 保障数据&服务高可用

String 类型处理效率 - Global Dictionary

1. String 类型的滥用（from HIVE), 处理低效

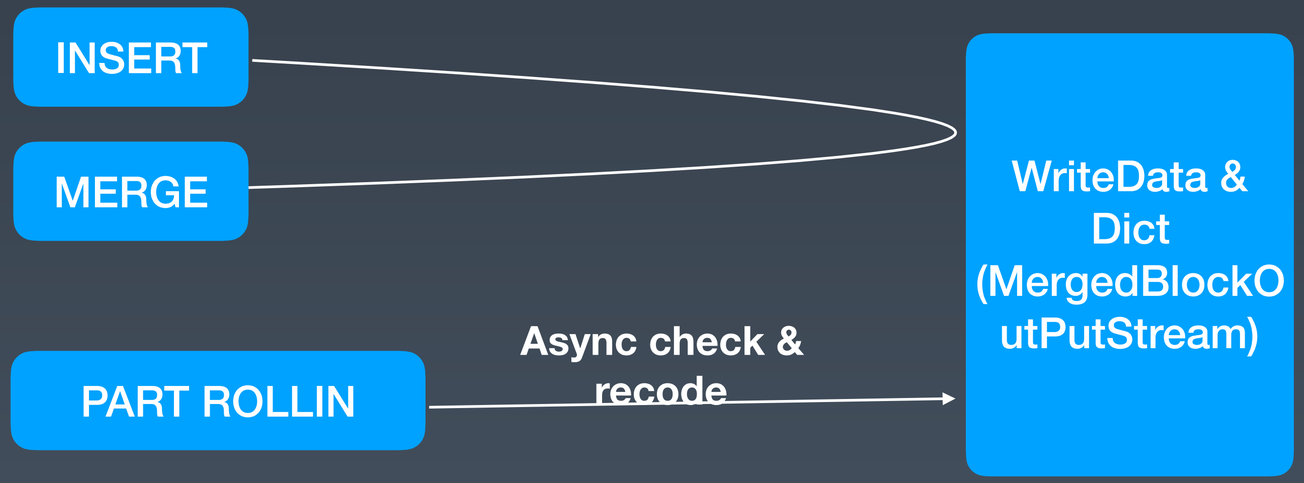
2. Why not LowCardinalityColumn ?

3. 算子尽量在压缩域上执行（actionable compression)
  - pure dictionary compression
  - predication (equality family)
  - group by (single/composite keys)

String 类型处理效率 - Global Dictionary

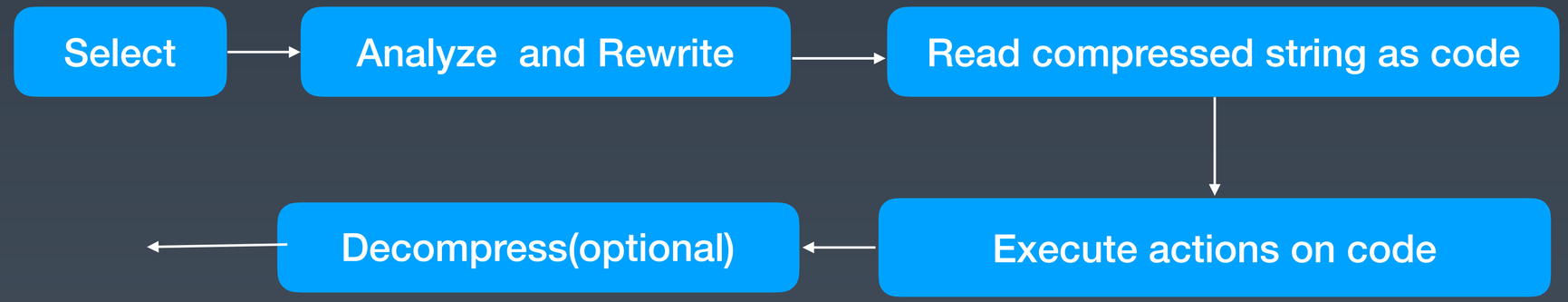
1. Per replica字典(异步)构建
   - why not cluster level/shard level?

2. Support xxMergeTree only



String 类型处理效率 - Global Dictionary

1. 压缩域执行



String 类型处理效率 - Global Dictionary

1. 分布式表字典（ per shard, per replica)
2. 分布式表压缩域执行
3. 性能提升约 20% ~ 30%

特定场景内存OOM - Step-ed Aggregation

1. Query：60天内用户转化率/行为路径，以及对应每天转化率

2. 内存使用量大，OOM对服务稳定性影响

3. Aggregator无法感知底层数据特性

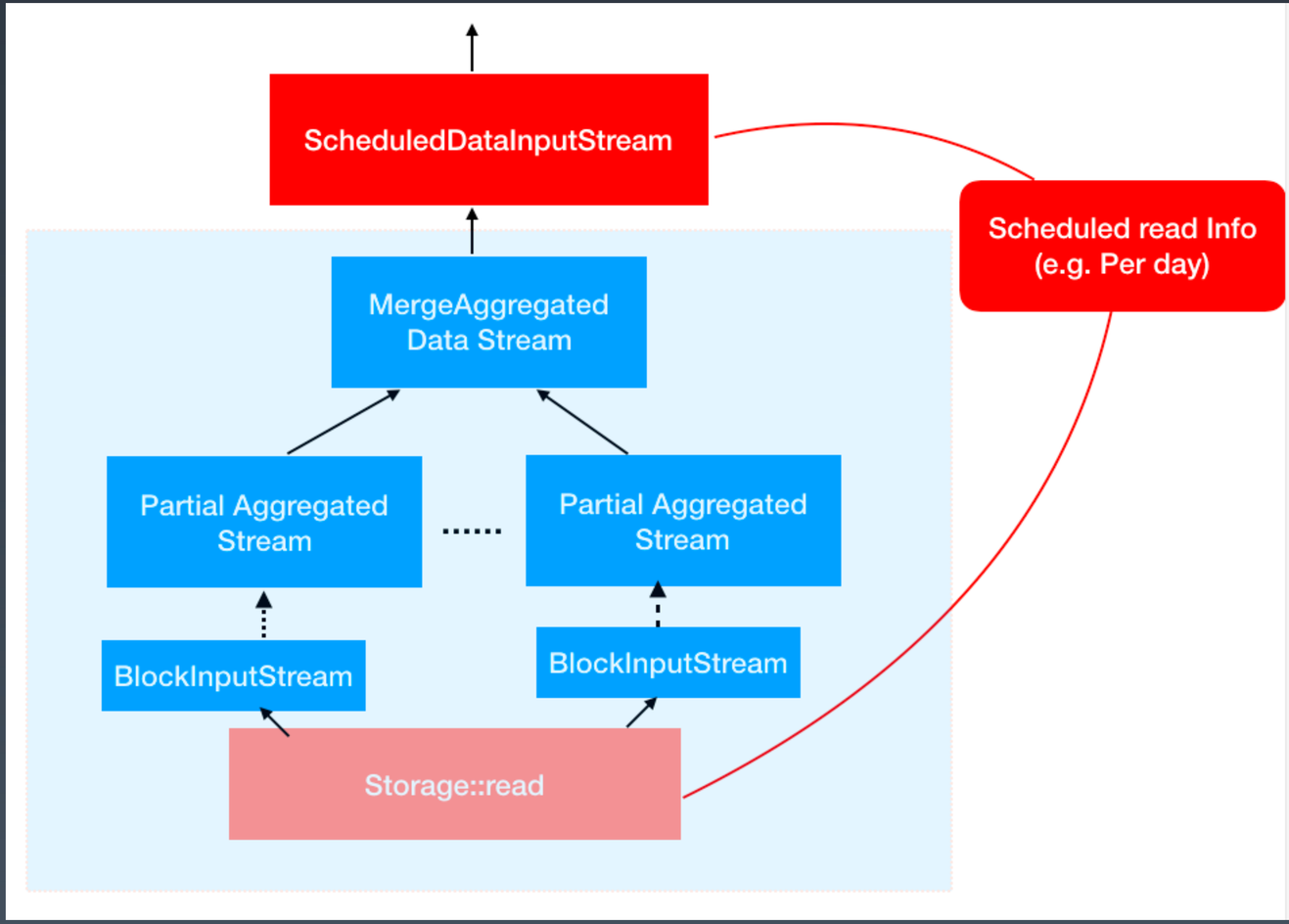
特定场景内存OOM - Step-ed Aggregation

1. Aggregator 由执行HINT控制

2. HINT 感知数据分区/指标语义

3. Blocked Aggregator 按partition pipeline计算指标。

收益：内存使用比默认方式降低约五倍



Array类型处理 - BloomFilter & BitMap index

1. Array类型用来表示实验ID

2. Query：命中某些实验的用户指标

3. 单条记录Array（实验）~ 几百 or 上千

Array类型处理 - BloomFilter & BitMap index

1. 需要辅助信息减少 Array column materialize
 - Two scale BloomFilter (Part level, MRK range level)

 2. 减少Long Array column in Runtime Block
  - Transform hasAny into BitMap index OR-ing

 Array Column —> value+BitMap 集合

 has(array, value) —> get BitMap (执行层自动改写)

其他问题与改进

1. HaKafka engine (主备容错的kafka engine)
2. 轻量级update/delete支持（基于delta表的方案)
3. 多尺度分区（小文件读取问题）
....

Bytedance ClickHouse TODO

1. 控制面开发，简化运维

2. Query cache支持

3. 数据导入原子性支持

4. 物化视图增强

5. 分布式Join

…

Geekbang. 极客邦科技 InfoQ

极客邦科技
会议推荐2019

5月

QCon 北京
全球软件开发大会
大会：5月6–8日
培训：5月9–10日

QCon 广州
全球软件开发大会
培训：5月25–26日
大会：5月27–28日

6月

GTLC GLOBAL TECH LEADERSHIP CONFERENCE 上海
技术领导力峰会
时间：6月14–15日

GMTC 北京
全球大前端技术大会
大会：6月20–21日
培训：6月22–23日

7月

ArchSummit 深圳
全球架构师峰会
大会：7月12–13日
培训：7月14–15日

10月

QCon 上海
全球软件开发大会
大会：10月17–19日
培训：10月20–21日

11月

GMTC 深圳
全球大前端技术大会
大会：11月8–9日
培训：11月10–11日

AiCon 北京
全球人工智能与机器学习大会
大会：11月21–22日
培训：11月23–24日

12月

ArchSummit 北京
全球架构师峰会
大会：12月6–7日
培训：12月8–9日

THANKS!

QCon 10th