ClickHouse大数据集群应用

腾讯思源计算平台

李俊飞 (微信: leyile1)

目录

- 我们的平台
- 数据仓库
- ClickHouse的优劣势
- 集群部署和管理
- 常见问题和解决方法

我们的平台

平台:
任务流，数据流

计算:
Spark, Tensorflow

数据仓库:
Hive，ClickHouse

底层:
Hadoop，Yarn，Kafka，etc.

数据仓库 – 实时查询需求

数据：
用户兴趣数据、推荐模型特征、业务日志

查询场景：
用户在页面自定义查询
需要在海量数据基础上，执行实时聚合查询

数据仓库 – 实时查询需求

数据量：

亿级日活
单个表240T
单库总数据量1PB
维度上百
实时自定义聚合查询
延时秒级，稳定可靠
强需求

数据仓库 – 实时查询需求

super fast

distributed

column-oriented

--

ClickHouse

ClickHouse的优势和劣势

优势:

- 速度快
- 分布式
- 丰富的接口、输入输出格式和函数支持
- 性能稳定
- 开发速度快，社区活跃

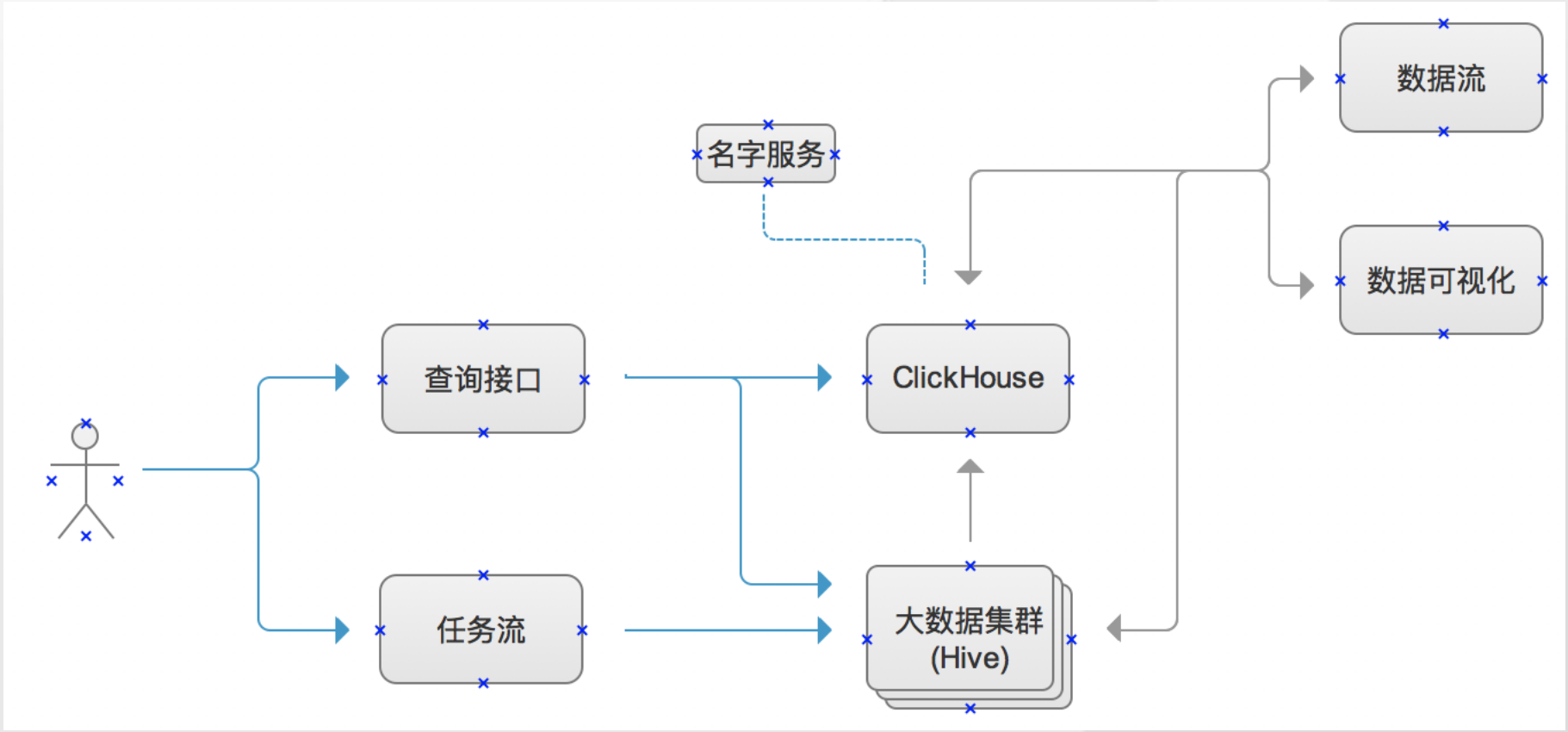
劣势:

- CUBE模型
- 事务
- 修改/删除
- 管理工具少
- Zookeeper依赖

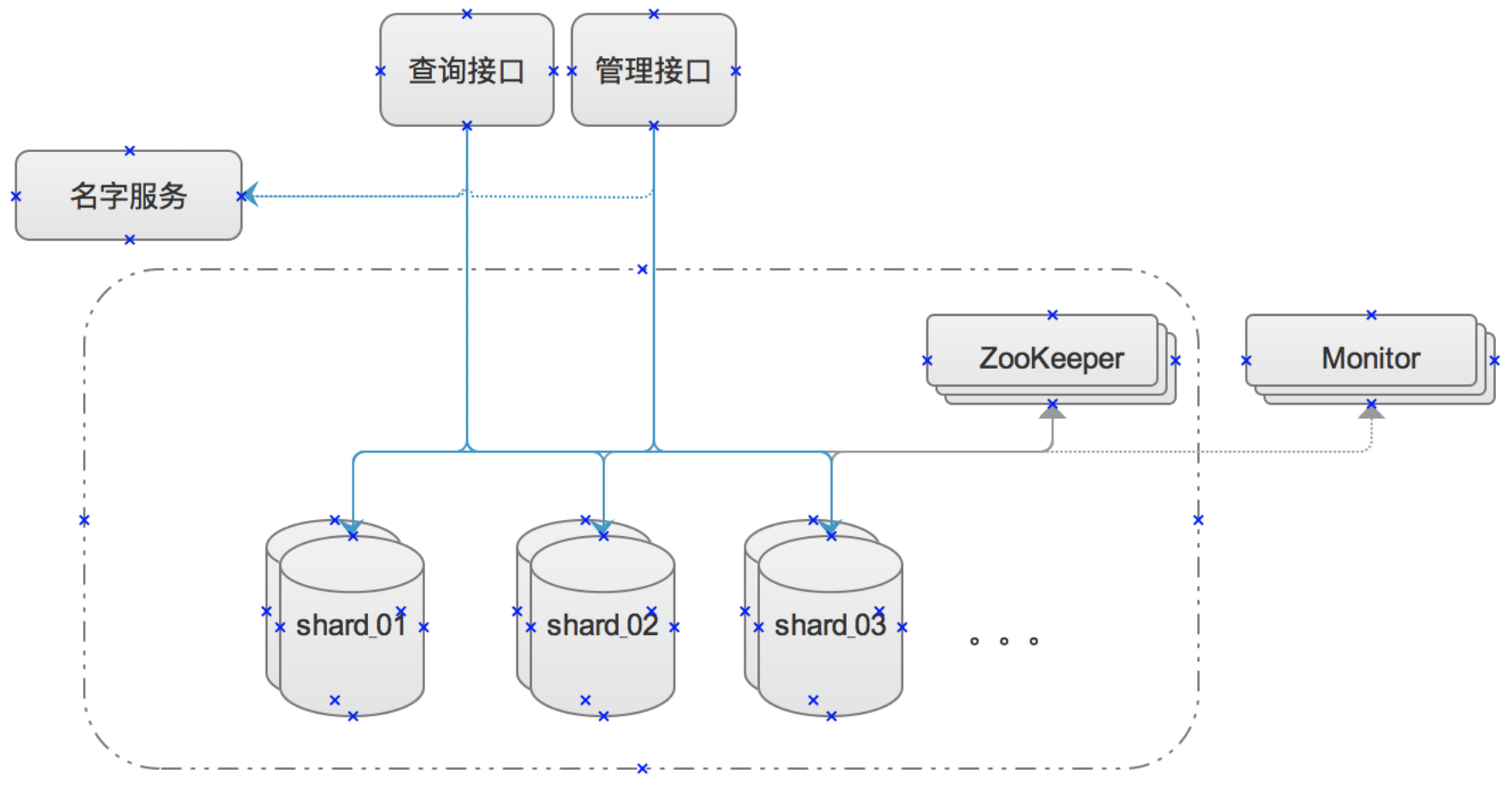
集群架构和管理 – 数仓架构

- 集群数量: 4
- 机器数: 130台
- 单机配置: Memory 128G
  CPU核数 24
  SATA 20T， RAID 5
  万兆网卡

集群架构和管理 – 数仓架构



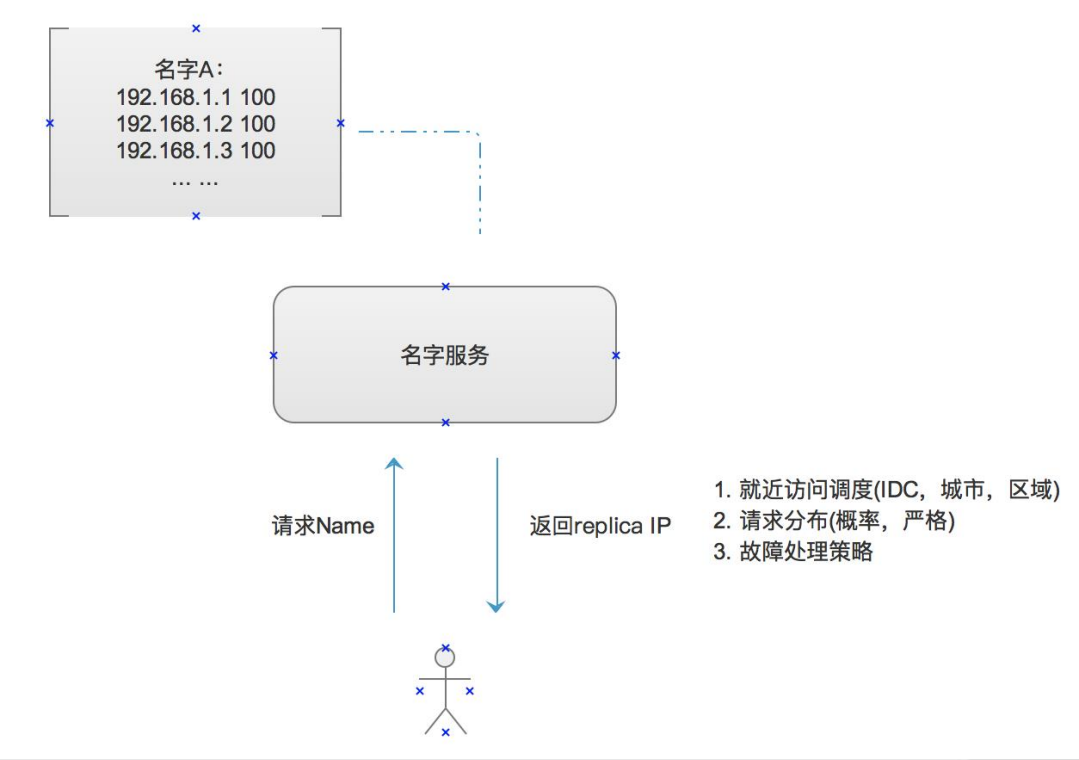
集群架构和管理 – 数仓架构



集群架构和管理 – 高可用

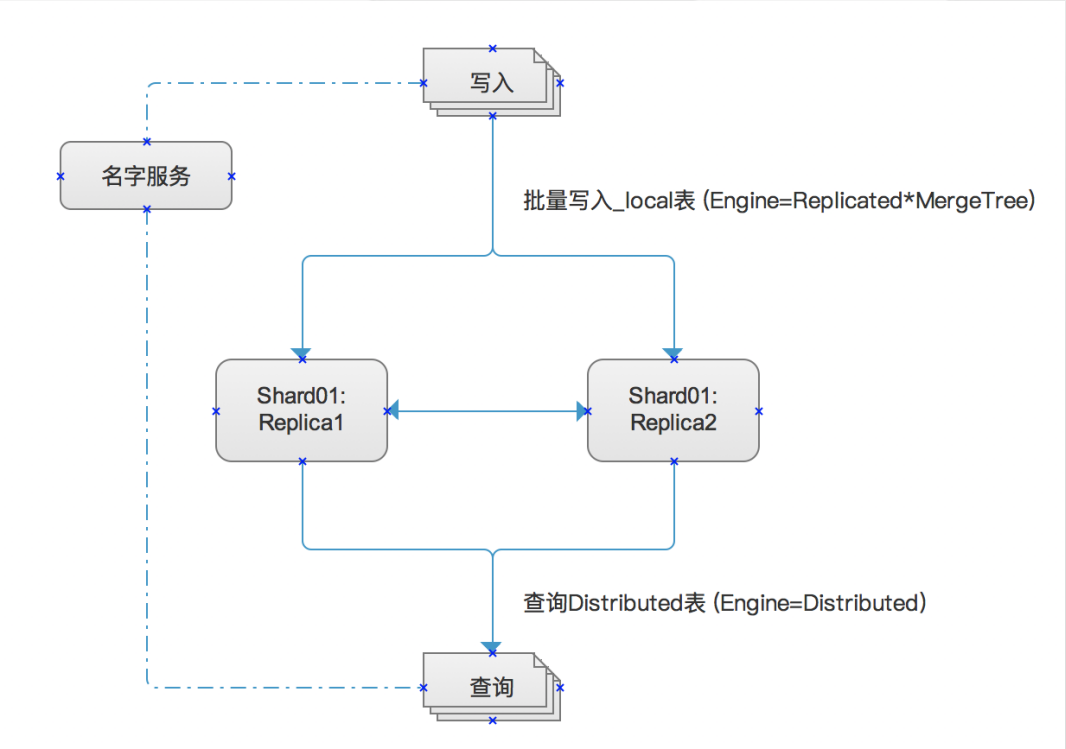
名字服务+复制：

- 高可用
- 跨机房容灾



集群架构和管理 – 复制

- 复制(ReplicatedMergeTree)
  异步master-master复制
  表级复制



- 读写模式

批量写入:

根据IO能力和行长: Batchsize= 10w ~ 100w

集群架构和管理 – Engine

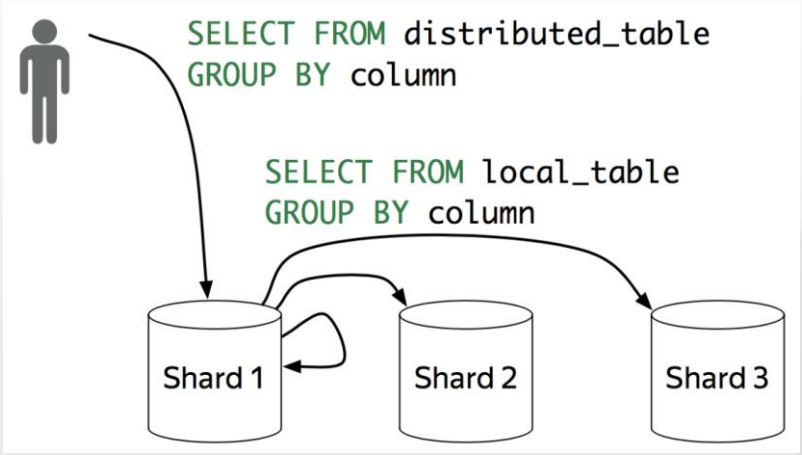
In memory:
– Memory
– Buffer
– Join
– Set

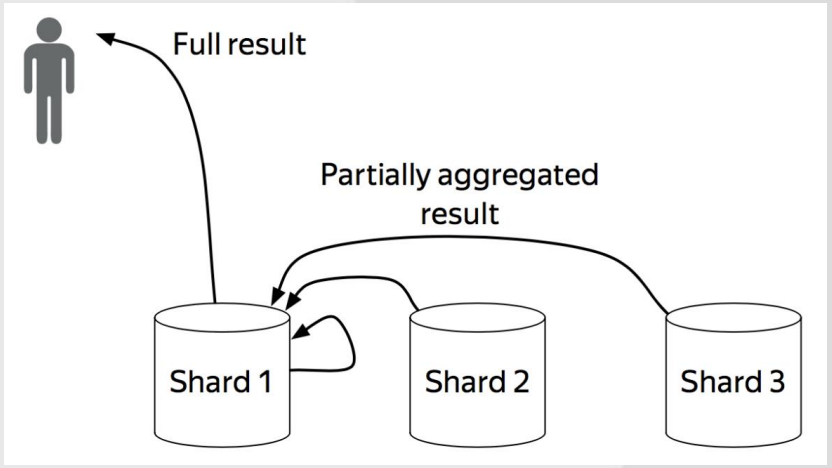
On disk:
– Log,TinyLog
– MergeTree family

Virtual:
- Merge
- Distributed
- Dictionary
- Null

Special purpose:
- View
- Materialized View

集群架构和管理 – 分布式


SELECT FROM distributed_table
GROUP BY column
SELECT FROM local_table
GROUP BY column
Shard 1
Shard 2
Shard 3


Full result
Partially aggregated result
Shard 1
Shard 2
Shard 3

集群架构和管理 – 扩缩容

按业务需求，线性扩展：

扩容：
1. 安装新shard
2. 批量修改配置文件（ck管理工具需求）
3. 新shard上创建表结构
4. 名字服务添加节点

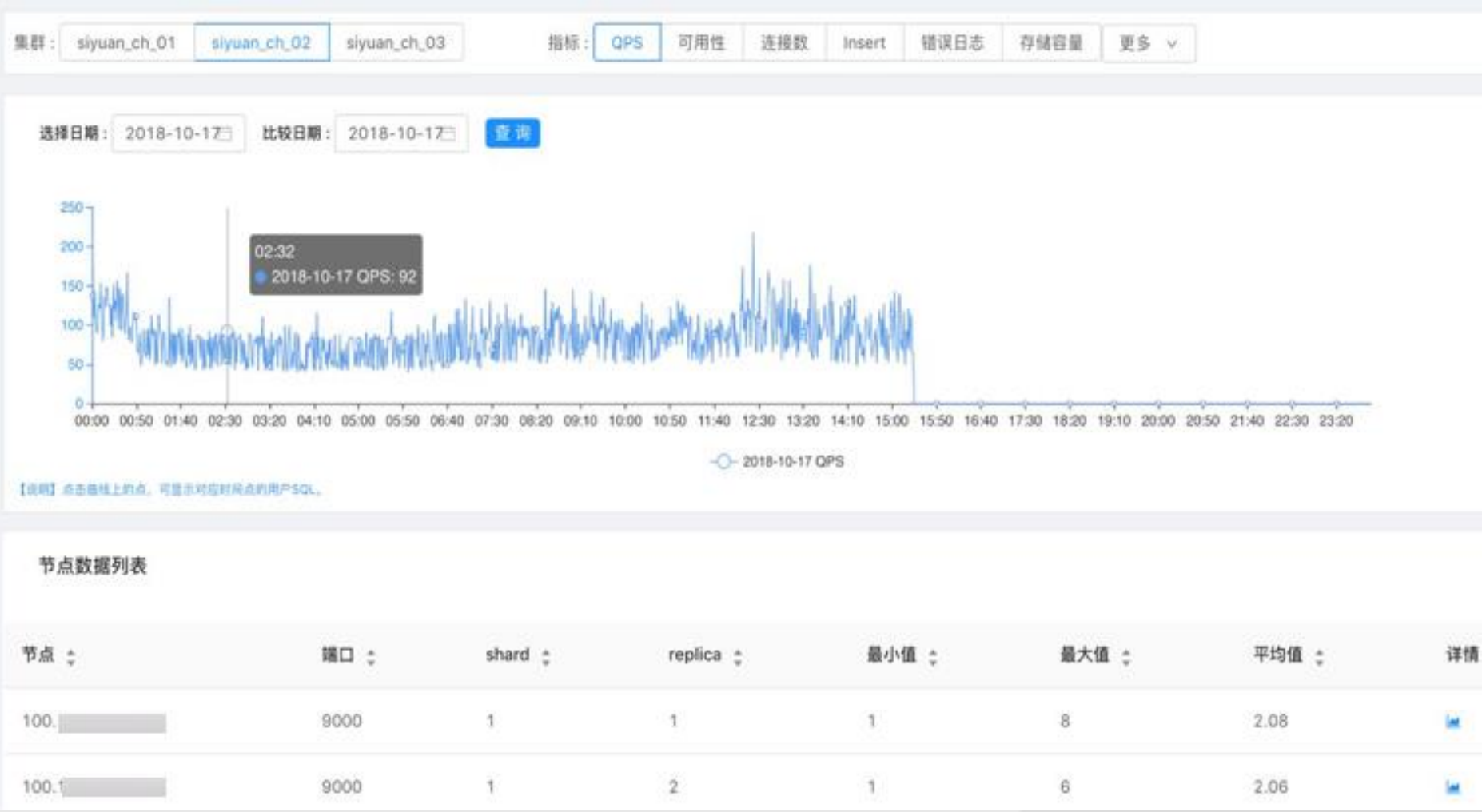
缩容：
1. 名字服务摘除节点
2. 迁移数据到其他节点
3. 删除shard表
4. 批量修改配置文件

CSV 227 Gb, ~1.3 bln rows

```
SELECT passenger_count, avg(total_amount)
FROM trips GROUP BY passenger_count
```

Shards	1	3	140
Time, s.	1,224	0,438	0,043
Speedup		x2.8	x28.5

集群架构和管理 – 监控



监控采集：

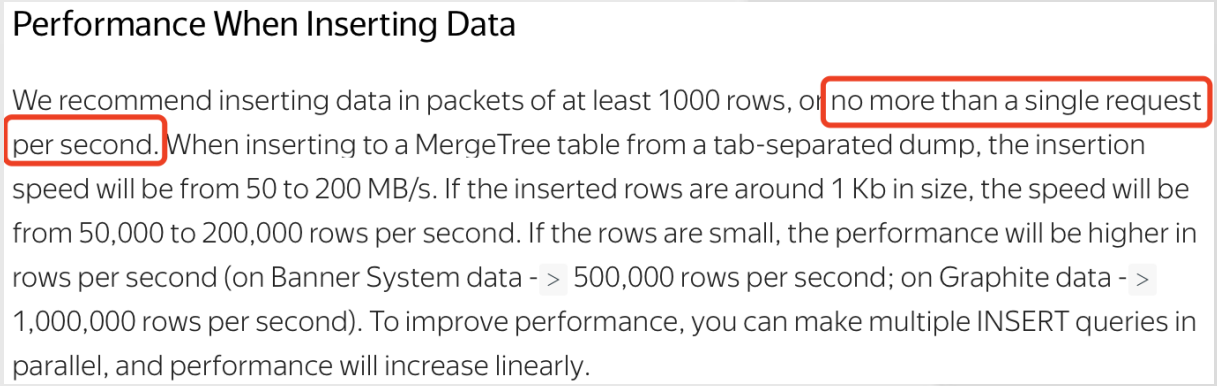
System schema, clickhouse log, OS-related, etc.

问题和解决方法(1)

首先，避免频繁写入

问题：

- DB::Exception: Too many parts (302). Merges are processing significantly slower than inserts.



Performance When Inserting Data

We recommend inserting data in packets of at least 1000 rows, or no more than a single request per second. When inserting to a MergeTree table from a tab-separated dump, the insertion speed will be from 50 to 200 MB/s. If the inserted rows are around 1 Kb in size, the speed will be from 50,000 to 200,000 rows per second. If the rows are small, the performance will be higher in rows per second (on Banner System data - > 500,000 rows per second; on Graphite data - > 1,000,000 rows per second). To improve performance, you can make multiple INSERT queries in parallel, and performance will increase linearly.

解决：

降低写入频率，根据IO能力， 1k行长，单次写入行数10万+

频繁小查询是否合理？降低并发度

频繁写入也会导致zxid消耗太快, zk不稳定

问题和解决方法(2)

问题：

- DB::Exception: Memory limit (for query) exceeded: would use 9.78 TB (attempt to allocate chunk of 393216 bytes), maximum: 9.09 GB.

解决：

聚合结果大小受限内存，启用外部磁盘存储

max_bytes_before_external_group_by
max_bytes_before_external_sort
max_memory_usage

问题和解决方法(3)

定义合适的分区键和主键索引

1. 查询务必使用分区字段
2. 数据是按索引顺序组织的
3. 客户端设置合理的查询超时阈值

问题和解决方法(4)

问题定位

```
2018.10.23 15:03:06.914270 [ 6735133 ] <Debug> executeQuery: Query pipeline:
Expression
 Expression
  MergingAggregated
   Union
    Materializing
     ParallelAggregating
      Expression × 12
       Filter
        MergeTreeThread
    Remote × 12

2018.10.23 15:03:07.121715 [ 6735133 ] <Trace> Aggregator: Read 13 blocks of partially aggregated data, total 13 rows.
2018.10.23 15:03:07.121758 [ 6735133 ] <Trace> Aggregator: Merging partially aggregated single-level data.
2018.10.23 15:03:07.122076 [ 6735133 ] <Trace> Aggregator: Converted aggregated data to blocks. 1 rows, 0.000 MiB in 0.000 sec. (91937.115 rows/sec.
2018.10.23 15:03:07.122168 [ 6735133 ] <Trace> UnionBlockInputStream: Waiting for threads to finish
2018.10.23 15:03:07.122397 [ 6735133 ] <Information> executeQuery: Read 9637464 rows, 726.80 MiB in 0.223 sec., 43145363 rows/sec., 3.18 GiB/sec.
2018.10.23 15:03:07.122437 [ 6735133 ] <Trace> virtual DB::MergingAndConvertingBlockInputStream::~MergingAndConvertingBlockInputStream(): Waiti
2018.10.23 15:03:07.122902 [ 6735133 ] <Debug> MemoryTracker: Peak memory usage (for query): 72.15 MiB.
2018.10.23 15:03:07.123235 [ 6735133 ] <Information> HTTPHandler: Done processing query
```

1. 报错信息
2. 分析日志
   执行计划
3. 资源使用情况
   system.processes -> slowlog
4. 监控告警

```
:) select address,user,query_id,read_rows, total_rows_approx, formatReadableSize(read_bytes) read, formatReadableSize(memory
tem.processes;

┌address─┬─user────┬─query_id─────────────────────────────┬─elapsed─┬─read_rows─┬─total_rows_approx─┬─read─────┬─memory_u
│ 100    │ default │ e5f86c4a-7fc1-4f2a-af84-e52d56a7420a │ 0.055565177 │   3317202 │          13656064 │ 245.72 MiB │ 37.38 Mi
```

问题和解决方法(5)

写入数据量均衡问题：

1. 批量写入Replicated*MergeTree本地表（写Distributed分布式表，影响写入性能，且数据分布容易不均衡）
2. 每次写入前，从名字服务获取节点IP，由名字服务提供数据量概率分布或精确分布
3. 频繁小批量写的表，可以添加Buffer表；可靠性要求低的表，考虑不使用Replicated系列表

问题和解决方法？(6)

总节点数：30

countDistinct(uid)	Elapsed
100 000 000	13 s
500 000 000	69 s
1 000 000 000	137 s

Cube需求：

计算UV：select countDistinct(uid) from ...

问题：
1. 结果集受限于节点内存，内存溢出落地磁盘选项不支持？
2. 类似Kylin的预计算Cube需求？

谢谢大家！