新浪微博
weibo.com

基于clickhouse数据高效分析

微博数据平台部-夏水军

新浪微博
weibo.com

关于我们

新浪微博
weibo.com

大　纲

1. 项目背景
2. 技术选型
3. 项目架构

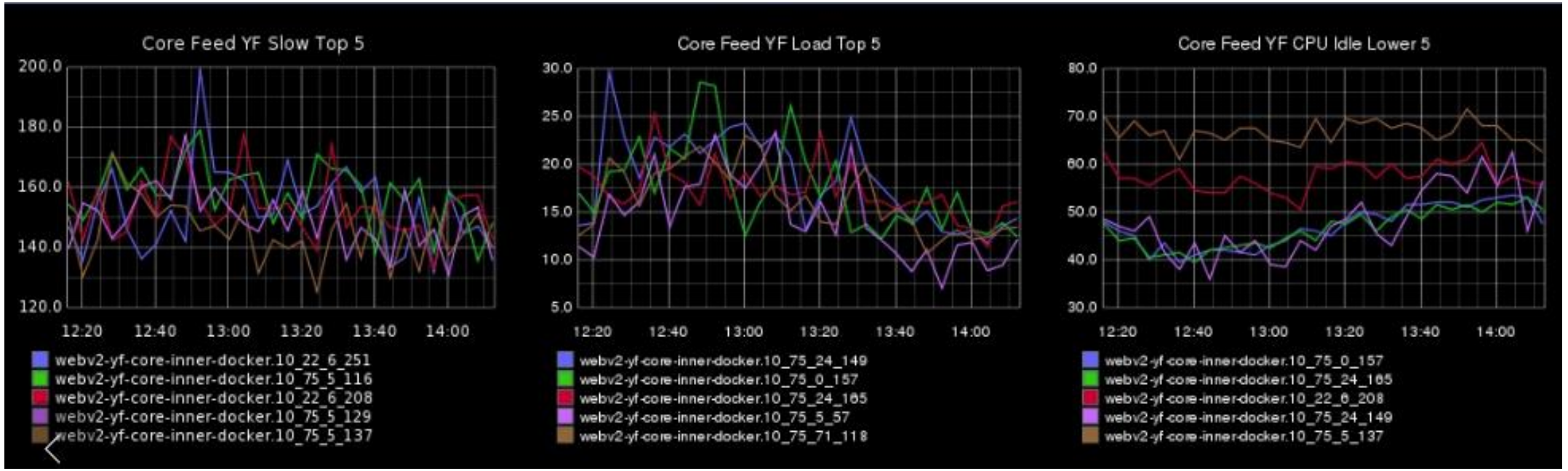
新浪微博
weibo.com

1 项目背景

- 并发量不太大
- 数据量的查询范围是百亿级
- 几千万不重复的key
- 响应速度需要在4s以内
- 日志数据
- 支持sql

新浪微博
weibo.com

实时feed日志监控:



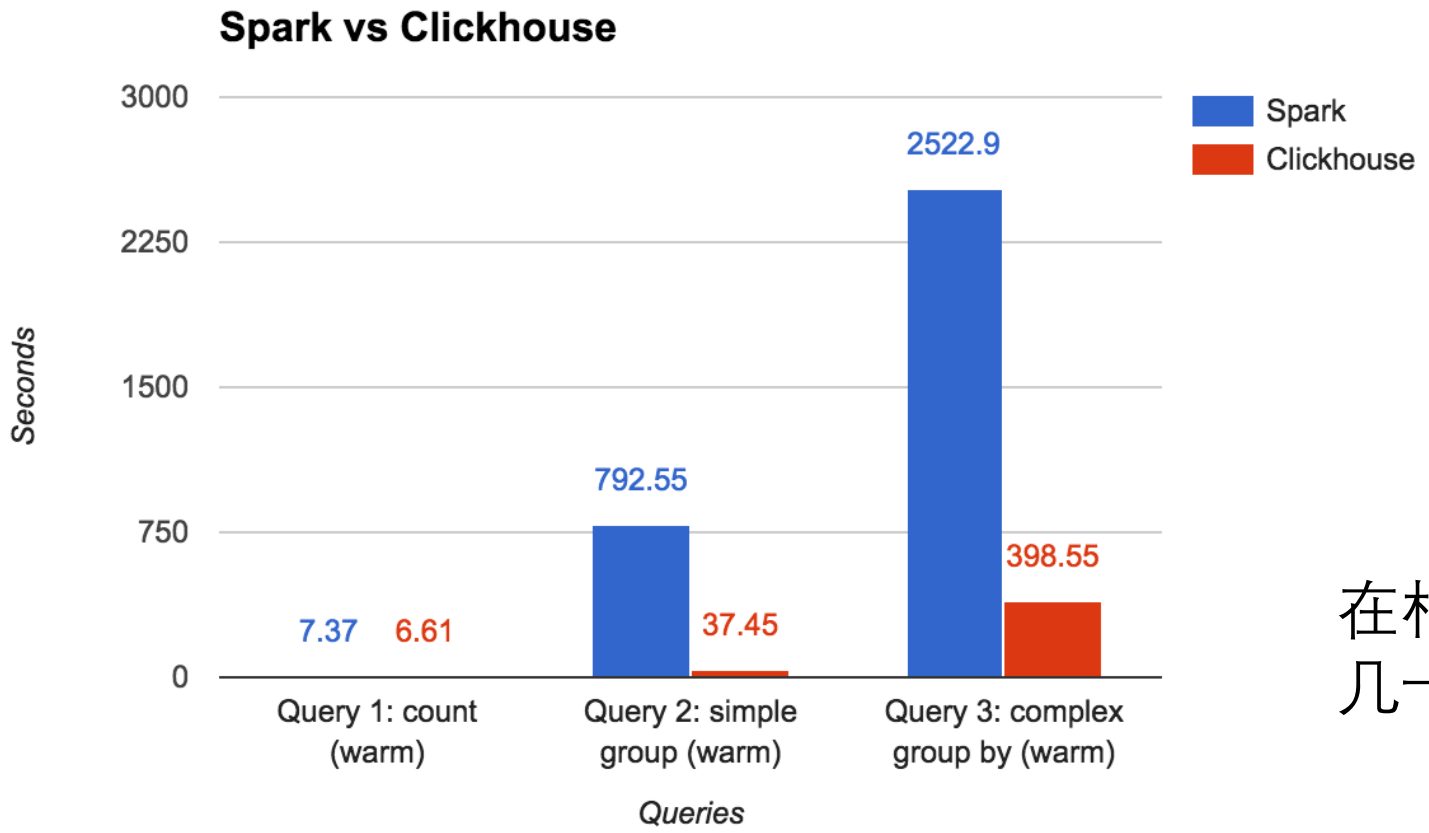
新浪微博
weibo.com

2 技术选型

数据特点:

- 数据量大，总的数据量几千亿
- 每4分钟写一次，一次写几百万
- 实时查询
- 响应时间 < 4s
- 可扩展性,存储和计算必须能应对数据量的增加
- 支持复杂的sql

数据分析：Sparksql vs Clickhouse



在相同条件下比spark快几十倍甚至几百倍

新浪微博
weibo.com

技术选型:

	elasticsearch	prometheus	druid	clickhouse	Influxdb	OpenTSDB
数据量	中等	小	大	大	中等	大
速度	快	快	中等	快	快	快
是否支持join	不	不	不	支持	不	不
可扩展性	好	不好	好	好	不好	好
支持sql	支持	不支持	支持	支持	不支持	不支持
函数	丰富	不丰富	丰富	丰富	不丰富	不丰富

新浪微博
weibo.com

clickhouse特性

优势:

- True column-oriented storage
- 向量化执行
- 数据压缩
- 并行执行
- 数据实时入库，实时查询
- 高可用
- 大量的sql支持
- 支持join
- 结构化数据
- 社区活跃
- 快,快,快

不足:

- 不支持事务
- 不支持多表join,需要通过子查询实现
- 支持局部的update和delete
- 不是标准的sql
- 高并发不够
- 运维困难

新浪微博
weibo.com

建表引擎

MergeTree

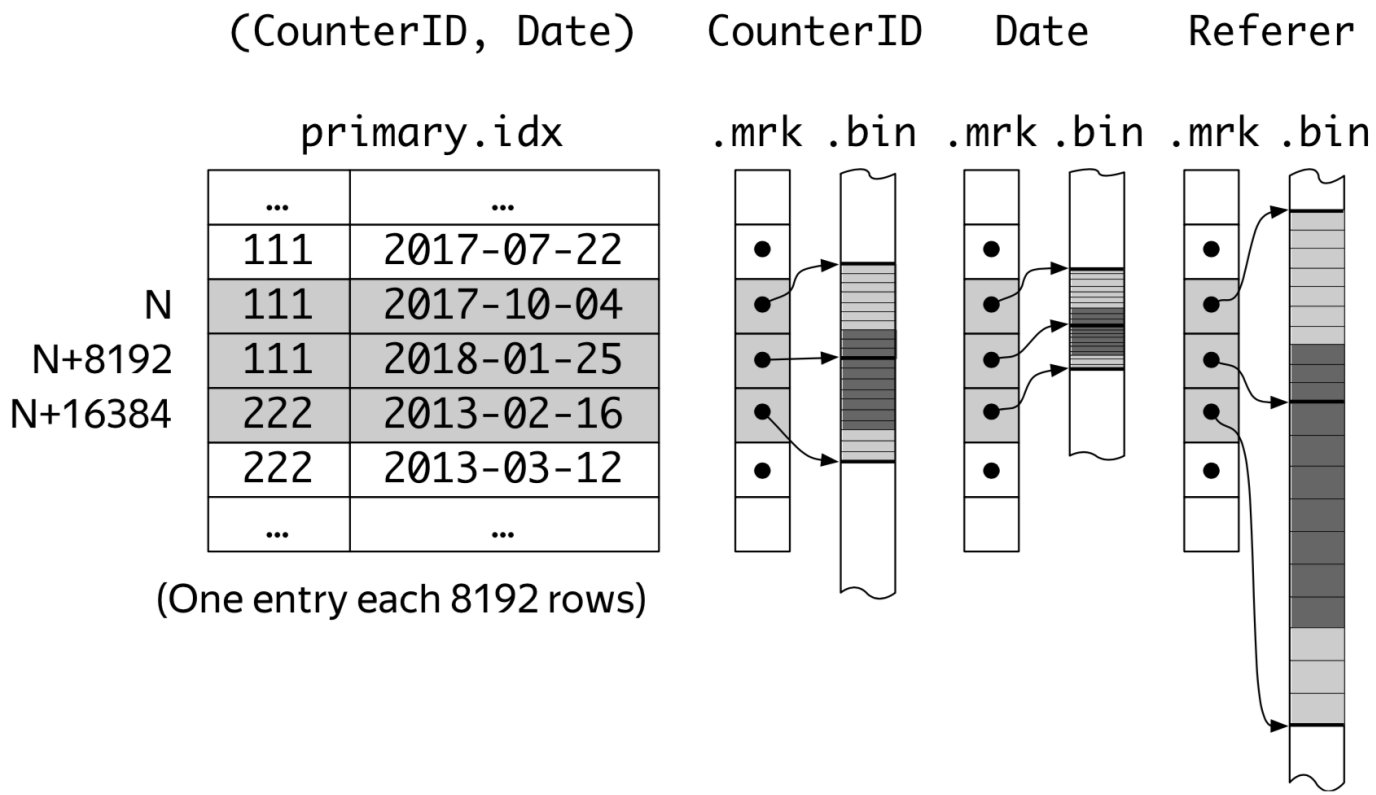
ReplicatedMergeTree

AggregatingMaterialized Views

Distributed

KafKa

Index internals



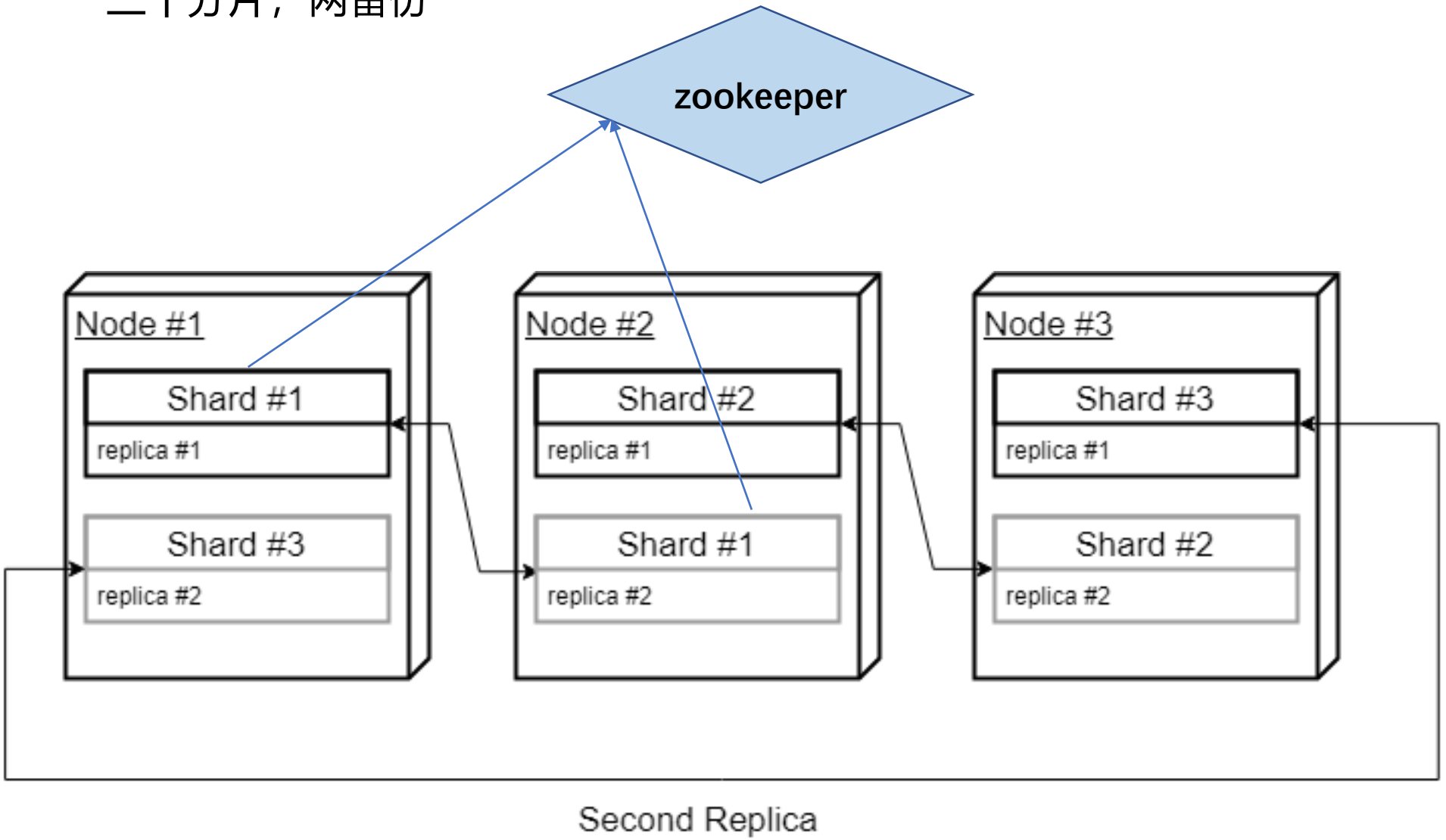
CREATE TABLE events (date DateTime, CounterId Int16,) ENGINE = MergeTree PARTITION BY date ORDER BY (CounterId,date) ndex_granularity = 8192

新浪微博 weibo.com

ReplicatedMergeTree

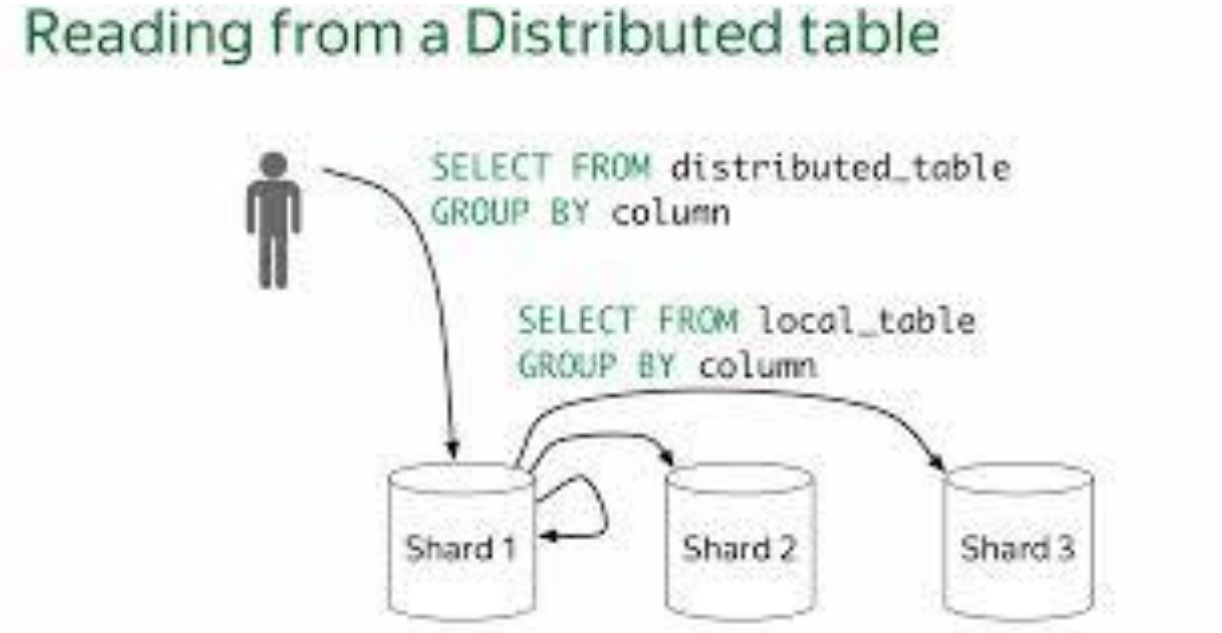
分片之间是基于zookeeper的异步复制

三个分片，两备份



新浪微博
weibo.com

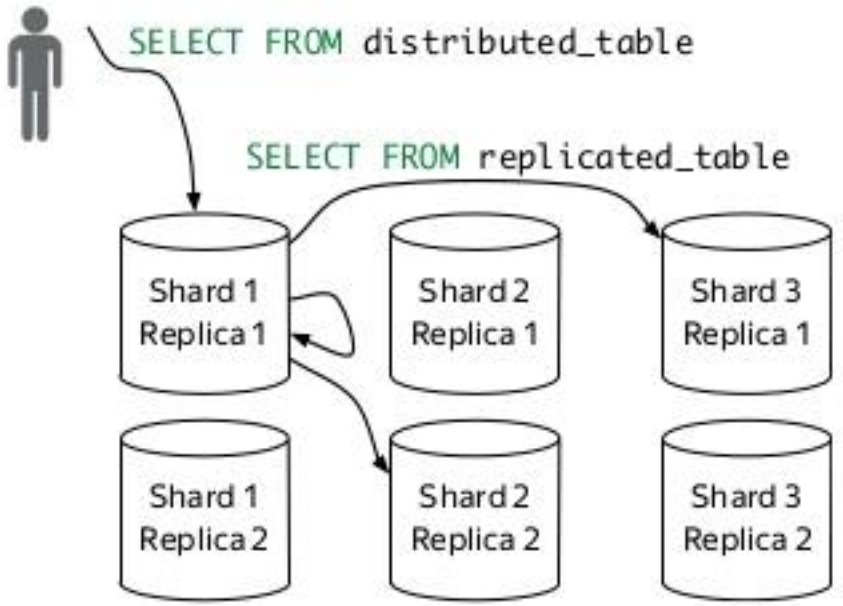
查询：Distributed + MergeTree



新浪微博
weibo.com

查询：Distributed + ReplicatedMergeTree

Putting it all together



新浪微博 weibo.com

Distributed + ReplicatedMergeTree模式建表

本地复制表：

```
CREATE TABLE users.events (
            user_id UInt32,
            time Date)
ENGINE=ReplicatedMergeTree( '/clickhouse/tables/users/events' , '{replica}' )
PARTITION BY time ORDER BY (time, user_id) settings index_granularity=8192;
```

分布式表：

```
create table events_all as events
ENGINE = Distributed( "cluster_name" , "database name" , "tablename" , rand())
```

新浪微博 weibo.com

Materialized Views

- 物化视图就是一个表，包含数据
- 基于Merge Tree创建物化视图表，插入MergeTree的数据实时更新到物化视图
- 支持AggregatingMergeTree,将聚合结果写入物化视图
- 增量聚合(uniq,sum,max,count)

1. 创建mergeTree create table device (userid Int32,url String ,device String, date Date) engine= MergeTree() partition by date order by (date,device) SETTINGS index_granularity=8192;
2. 创建Materialized Views : CREATE MATERIALIZED VIEW device_agg_day ENGINE = AggregatingMergeTree() partition by date order by (date,device) SETTINGS index_granularity=8192 POPULATE AS SELECT date, device, countState(*) AS total FROM device GROUP BY date,device;
3. **查询: SELECT countMerge(total),device FROM device_agg_day GROUP BY device;**

新浪微博 weibo.com

Figure 1

```
test :) select date,device from device;

SELECT
    date,
    device
FROM device
```

date	device
2018-12-14	huawei

date	device
2018-12-12	apple

date	device
2018-12-12	apple

date	device
2018-12-13	huawei

date	device
2018-12-14	huawei

Figure 2

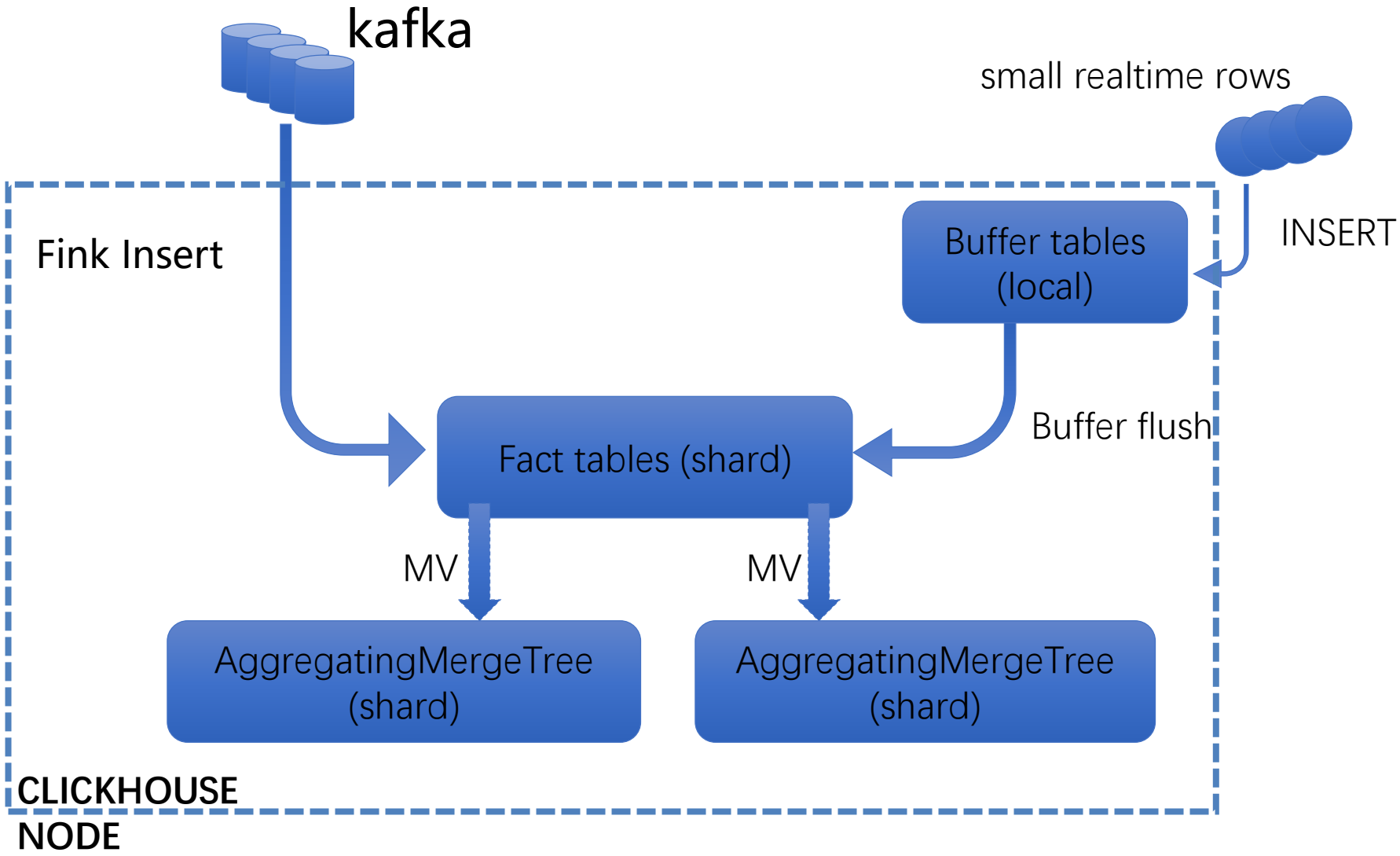
```
test :) SELECT countMerge(total),device,date FROM device_agg_day GROUP BY device,date;

SELECT
    countMerge(total),
    device,
    date
FROM device_agg_day
GROUP BY
    device,
    date
```

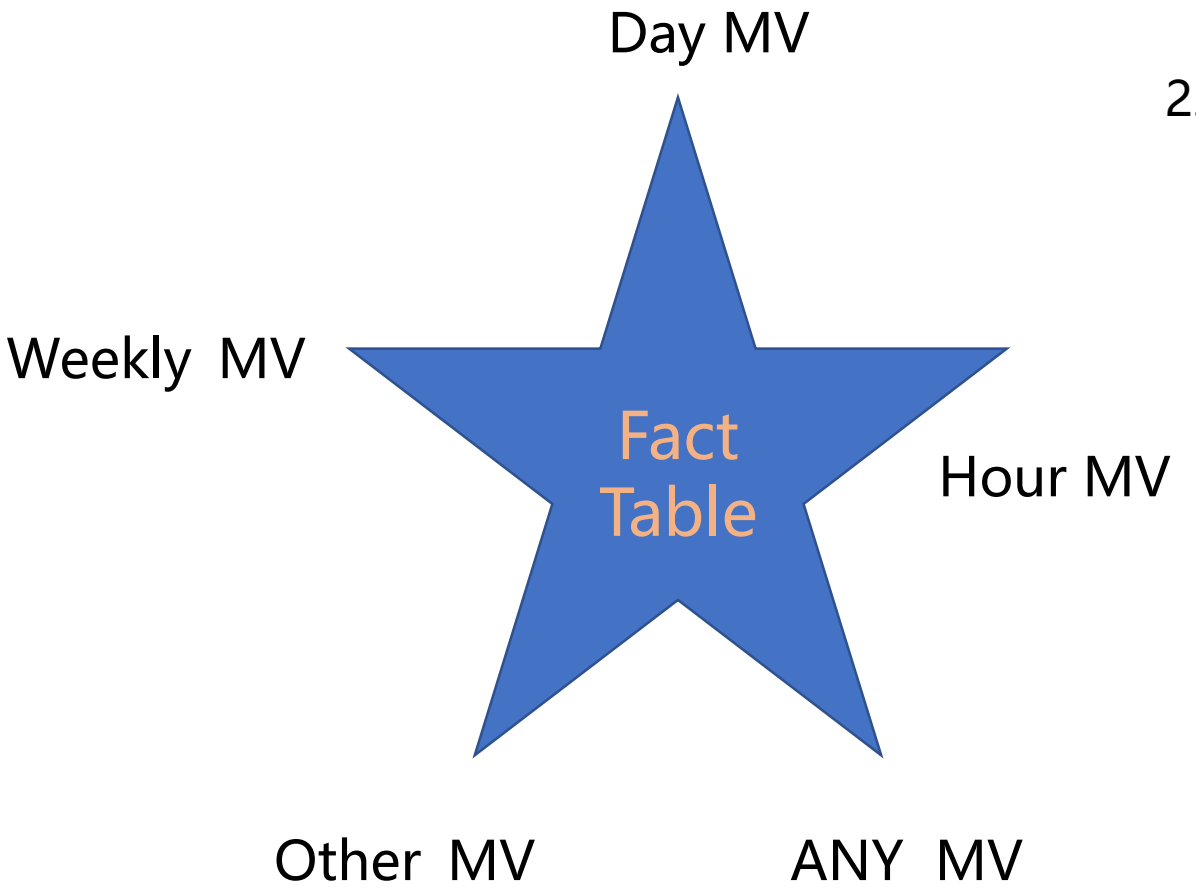
countMerge(total)	device	date
1	huawei	2018-12-13
2	huawei	2018-12-14
2	apple	2018-12-12

Figure 1:原始表数据

Figure 2: 物化视图

3 项目架构


kafka
small realtime rows
INSERT
Fink Insert
Buffer tables
(local)
Buffer flush
Fact tables (shard)
MV
MV
AggregatingMergeTree
(shard)
AggregatingMergeTree
(shard)
CLICKHOUSE
NODE



1. 创建事实表：CREATE TABLE IF NOT EXISTS events (date DateTime, counterId Int16) ENGINE = MergeTree PARTITION BYdata ORDER BY (date, counterId);
2. 基于AggregatingMergeTree创建视图表： CREATE MATERIALIZED VIEW IF NOT EXISTS event_view ENGINE = AggregatingMergeTree PARTITION BY toMonday(Period) ORDER BY (Period, CounterId) POPULATE AS SELECT date,counterId,uniqState(cash) AS Uniq from event;

星型模型:其他各个维度的统计表

现在clickhouse集群：

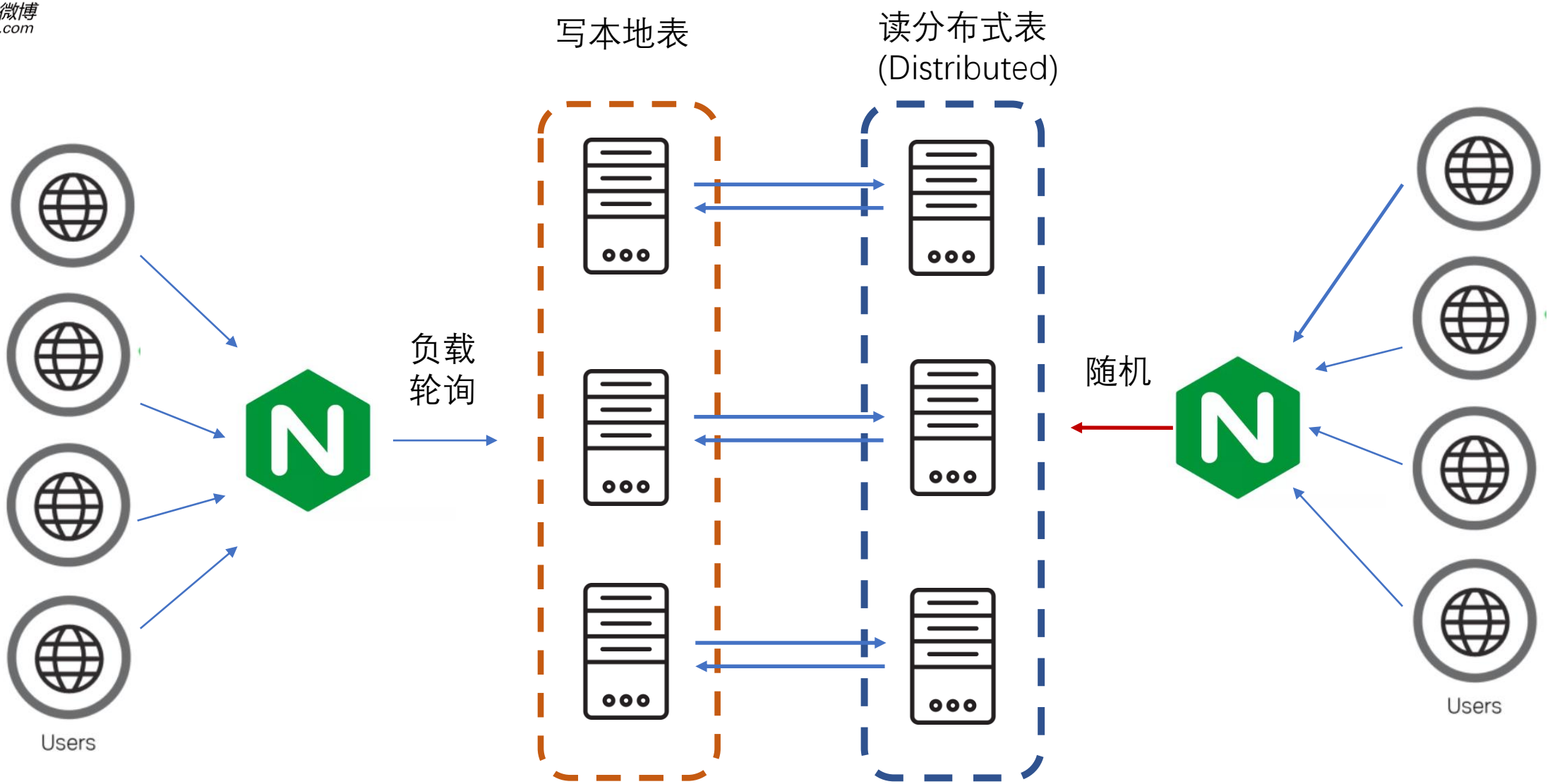
6台，三台查询，三台备份

12cores,48g

数据量：一个月的数据大概2400亿

响应时间: < 4s

新浪微博
weibo.com



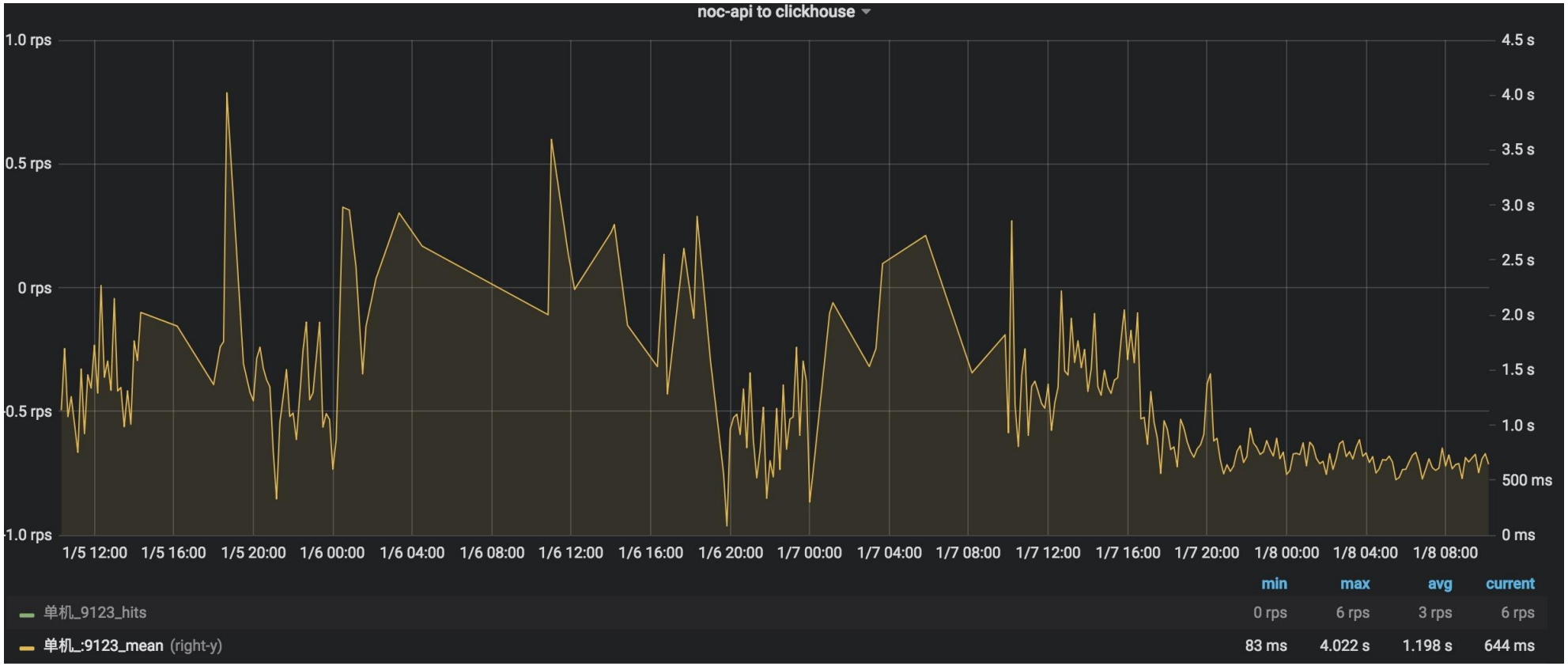
新浪微博
weibo.com

8天数据量：

```
[redacted] :) select count() from graphite_rep_all;

SELECT count()
FROM graphite_rep_all

┌─count()─┐
│ 77511896715 │
└─────────┘

1 rows in set. Elapsed: 31.165 sec. Processed 77.51 billion rows, 155.02 GB (2.49 billion rows/s., 4.97 GB/s.)
```

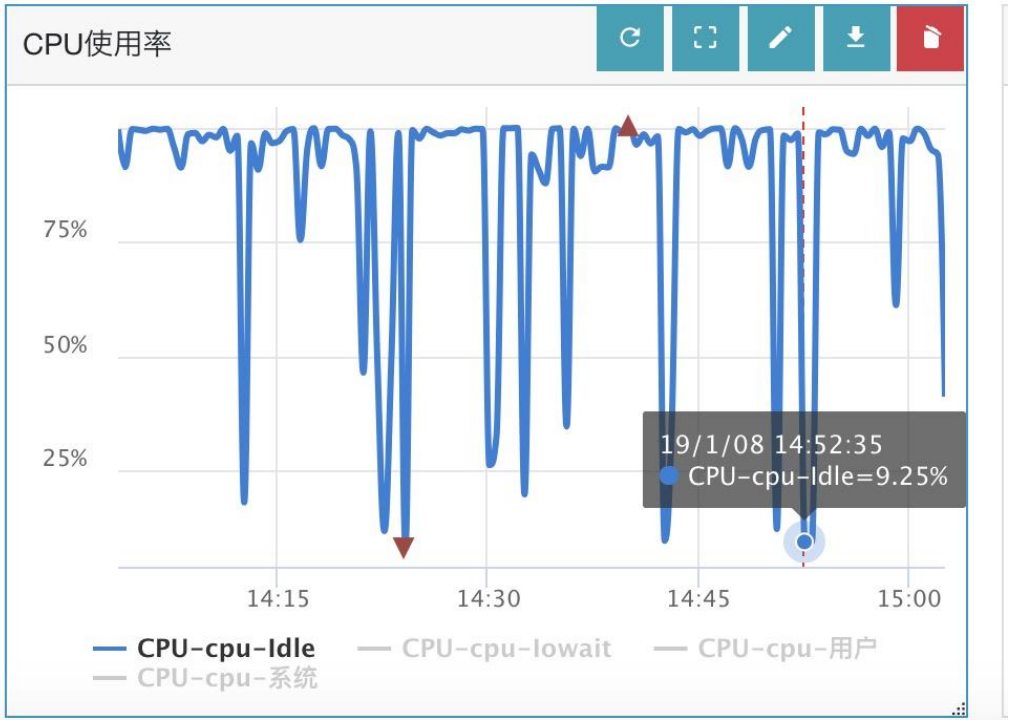
新浪微博
weibo.com

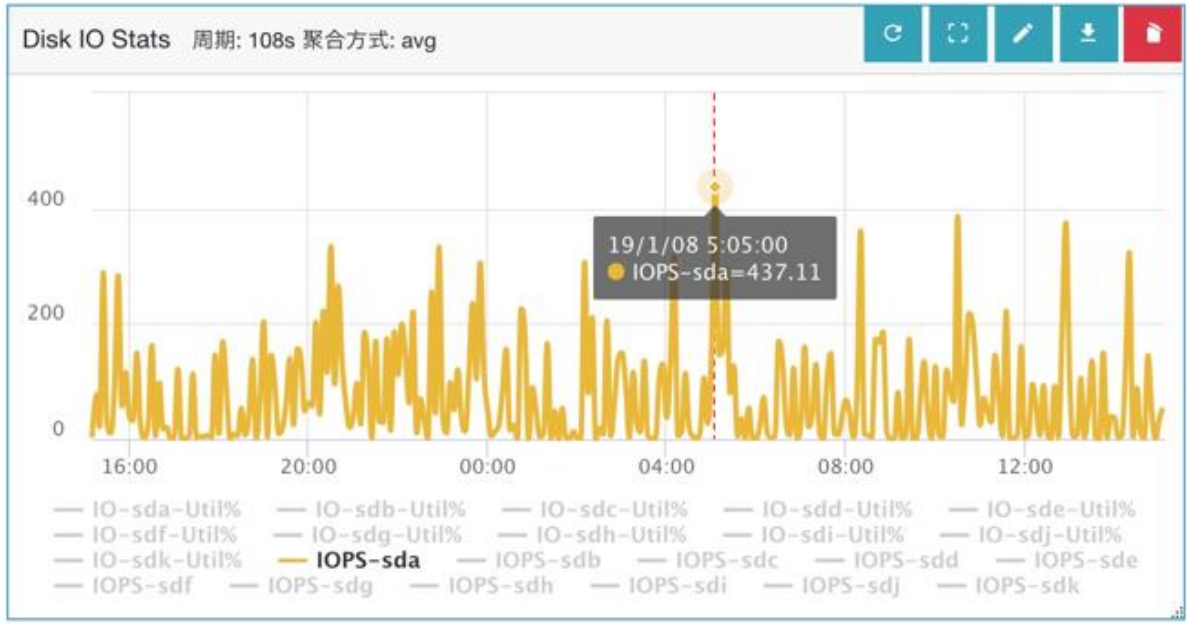
查询响应时间:



新浪微博
weibo.com

cpu和硬盘实时状态:





新浪微博
weibo.com

1.写入

- 先排序，再写入
- 批量,一次100000rows 或者 50-200M
- max_insert_block_size = 1,048,576
- 分区的粒度可以大点，设计是按月分区

2.查询

- max_threads: 单次查询最多使用的线程数
- background_pool_size： merge的线程数
- compile_expressions： 编译运行，对cpu敏感
- max_execution_time：最大执行时间
- max_ast_depth:
- max_memory_usage：单个查询的使用内存
- max_memory_usage_for_user：对于某个用户最大的使用内存
- max_memory_usage_for_all_queries：并发查询最大时
- max_bytes_before_external_sort： 最大使用排序
- max_bytes_before_external_group_by：聚合最大内存
- 使用prewhere

3. merge_tree_settings

- replicated_max_parallel_fetches
- replicated_max_parallel_fetches_for_table:

新浪微博
weibo.com

```
bdp01 :) select count(*) from ti_mdzt_all where eventdate between '2018-12-20' and '2018-12-29' and contacter like '%先生%'

SELECT count(*)
FROM ti_mdzt_all
WHERE ((eventdate >= '2018-12-20') AND (eventdate <= '2018-12-29')) AND (contacter LIKE '%先生%')

┌─count()─┐
│ 1901838 │
└─────────┘

1 rows in set. Elapsed: 12.918 sec. Processed 2.09 billion rows, 28.77 GB (162.11 million rows/s., 2.23 GB/s.)
```

前

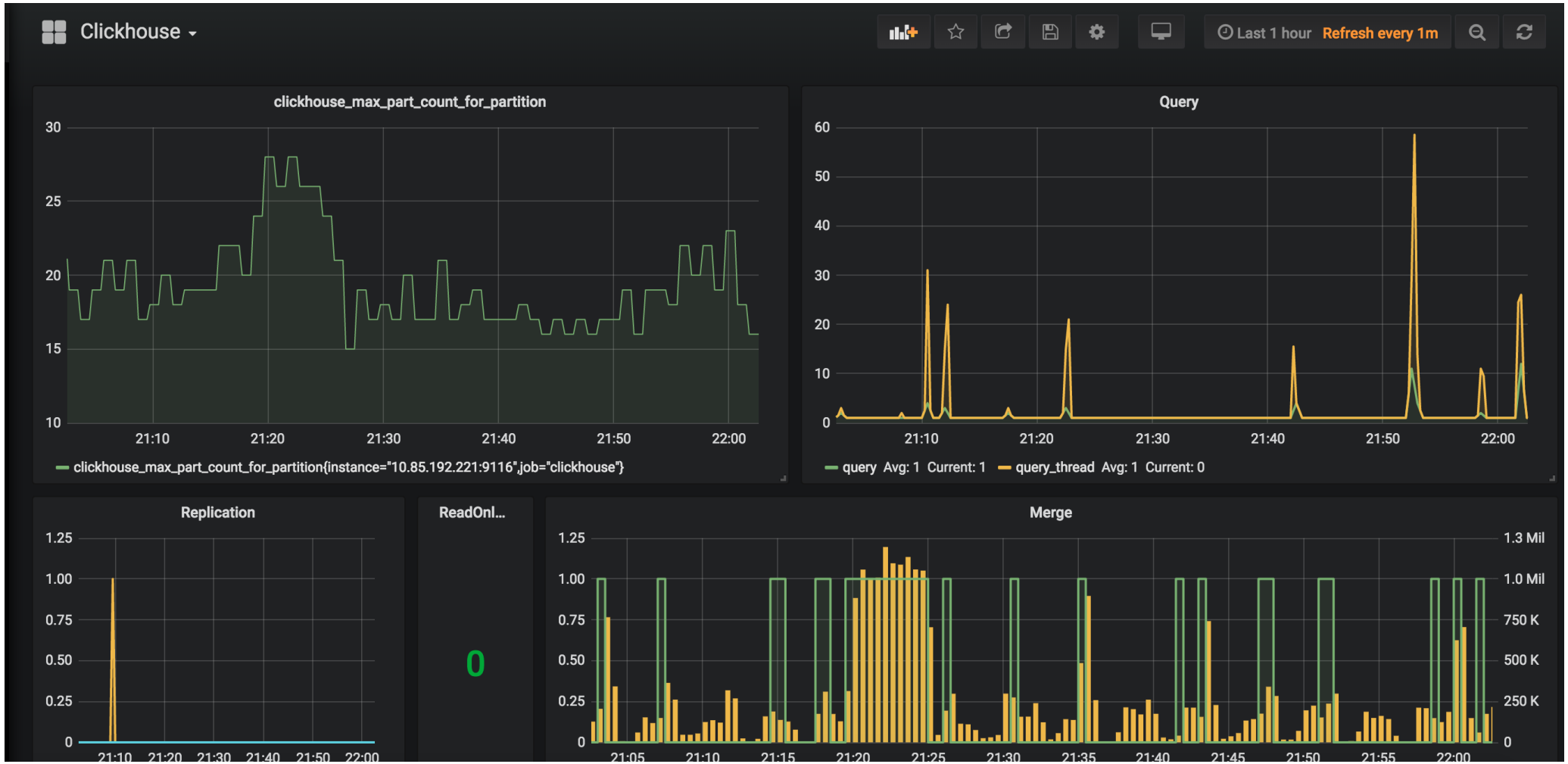
使用perwhere

后

```
bdp01 :) select count(*) from ti_mdzt_all prewhere eventdate between '2018-12-01' and '2018-12-31' where contacter like '%先生%'

SELECT count(*)
FROM ti_mdzt_all
PREWHERE (eventdate >= '2018-12-01') AND (eventdate <= '2018-12-31')
WHERE contacter LIKE '%先生%'

┌─count()─┐
│ 3091100 │
└─────────┘

1 rows in set. Elapsed: 3.858 sec. Processed 3.51 billion rows, 55.48 GB (910.71 million rows/s., 14.38 GB/s.)
```

新浪微博
weibo.com

监控：https://github.com/f1yegor/clickhouse_exporter



新浪微博 weibo.com

Community

- github: https://github.com/yandex/clickhouse
- jdbc: https://github.com/yandex/clickhouse-jdbc
- download: https://packagecloud.io/Altinity/clickhouse
- blog: http://jackpgao.github.io/

新浪微博
weibo.com

扫描二维码，获取更多交流