|  |  |
| --- | --- |
| 内部版本号 | V1.3 |
| 编写人 |  |
| 评审人 |  |



技 术 文 件

技术文件名称：日志大数据量方案设计

技术文件编号：

江苏易安联网络技术有限公司

2024年7月8日

**修订记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **修订内容** | **修订人** | **修订时间** |
| V0.1 | 1-4内容填充 | 王成 | 2024-07-08 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**目 录**

[1 编写目的 3](#_Toc15398)

[2 术语、定义和缩略语 3](#_Toc2209)

[3 需求背景 3](#_Toc16237)

[4 解决思路 3](#_Toc27270)

[5 方案设计 4](#_Toc20015)

[5.1 架构设计 4](#_Toc14851)

[5.1.1 静态视图 4](#_Toc8459)

[5.1.2 部署视图 6](#_Toc24475)

[5.2 非功能设计 6](#_Toc12134)

[5.2.1 可测试性设计 6](#_Toc10817)

[5.2.2 性能设计 6](#_Toc5351)

[5.2.3 接口的兼容性 7](#_Toc13697)

[5.2.4 版本前向兼容性 8](#_Toc7042)

[5.2.5 安全性设计 8](#_Toc5575)

[5.2.6 可复用性设计 8](#_Toc22286)

[5.2.7 可扩展性设计 8](#_Toc9253)

[5.2.8 易用性设计 8](#_Toc26252)

[5.2.9 波及面分析 8](#_Toc19718)

[6 协作说明 8](#_Toc20600)

[6.1 流程a 9](#_Toc8620)

[7 存储设计 9](#_Toc23269)

[7.1 关系数据表 10](#_Toc10861)

[7.1.1 xxx表 10](#_Toc1328)

[7.2 Redis存储 10](#_Toc10248)

[7.2.1 xxx-Key设计 10](#_Toc14474)

[8 类设计 11](#_Toc9364)

[9 接口设计 11](#_Toc11700)

[9.1 Xxx接口 11](#_Toc30921)

[9.1.1 接口说明 11](#_Toc14743)

[9.1.2 入参 12](#_Toc20666)

[9.1.3 出参 12](#_Toc32671)

[10 任务分解 13](#_Toc21302)

[11 参考资料 13](#_Toc16394)

# 编写目的

本文档主要阐述大数据量场景下日志数据的存储方案。面向产品、测试、开发人员。

# 术语、定义和缩略语

|  |  |  |
| --- | --- | --- |
| 术语 | 释义 | 备注 |
| ElasticSearch | 基于lucence的分布式开源全文搜索引擎 | 以下以ES简称 |
| ClickHouse | ClickHouse是一款高性能、MPP架构、列式存储、具有完备DBMS功能的OLAP数据库 | 以下以CH简称 |

# 需求背景

为应对大局点 多用户，日志数据单日超过1kw（超过mysql单表即席查询存储上限）的的场景，需要寻找一个合适的数据存储方案，既要支持大数据量数据的存储，又要支持在这些大数据量场景下的即席查询。目标：支持单日1亿数据存储与检索查询。

# 解决思路

## 方案调研

由于我们系统需要针对日志存储实现即席查询，故传统的基于HDFS生态上的各种方案，如hive、hbase这里暂不介绍，搭建一个Hadoop平台也需要很多的资源，并且当前我们的业务范围不包含数据的处理。故当下我们主要以支持即席查询为优先考虑点。目前有如下两款开源产品：ES、CH

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 产品 | 存储方式 | CPU | 内存使用 | 磁盘使用 | 写入性能 | 查询性能 | 优缺点 |
| CH | 列式存储，高效压缩 | 并行查询，单查询也要占用很多的cpu | 由于压缩存储，相对ES较少的内存占用，列式存储的方式，更有利于内存的减少使用 | 官方：同等数据量，是ES的1/3 到 1/30 | 官方：是 ES 的 5 倍以上,对大量的小数据量实时写入不友好 | 官方：ClickHouse 的查询速度比 ES 快 5-30 倍以上 | 写入吞吐量大，无法动态添加字段，无法提供全文搜索，不支持高并发查询，不适合许多小数据高频插入。 |
| ES | json存储 | 内存查询，单查询下cpu占用相对CH可能低（和数据结构、查询复杂度挂钩） | 需要维护倒排索引，内存占用相对CH较多，并且在特殊场景下，如查询数据量大，可能导致OOM | 比CH占用更多 | 比CH慢，对实时写入较友好 | 比CH慢（和数据结构、查询复杂度挂钩）,但对实时查询、高并发查询友好 | 吞吐量相对较小，内存占用、cpu使用相对更多，支持实时检索、高并发查询，磁盘占用大很多。当下我们的场景较少，并且都是全字段查询，列式存储没有发挥完全的优势。需要一定的成本学习CH用法及其相关调优。 |

以上是根据网上的数据进行的对比，下面对两个产品做进一步的测试

## ES

使用 16核 32G 500G 虚机测试。数据入库使用 10个线程并发写入，单批次1000条web访问应用日志。由于磁盘空间及时间原因，入库了9亿数据

由于ES的机制，内存32G，实际ES服务程序只能分配到16G，另一半分配给lusence

### 入库性能

每批数据入库用时 200ms ~ 300ms

ES写入并发与cpu核数有关，TPS较好，在承畴范围内并发下，入库时间波动不大

并发度越高，cpu负载越高。

如10个线程并发时：cpu能打到300%；而30个并发，cpu能打到 1500%；但两者入库时间差不多。

---ES会(定时？)对倒排文件进行段合并，会占用一定的cpu资源，此时入库时间有波动

内存上，ES对内存要求有一定的要求，内存随着入库量的增加会有持续的提升，但ES在段合并之后会降下来，部分段只有使用时才会加载，类似于冷热数据的优化。在当前机器上内存未看到明显瓶颈。

磁盘上，ES对磁盘的使用较大，一个索引在写入过程中会生成很多小段，以一个索引1kw数据为例，写入过程中最高能达到 41g，这个也不一定是峰值，但最终会进行段合并及压缩，压缩后有 25.8g。也是很大，故当前测试虚机，只跑了9亿的数据量

所以磁盘上若我们要存储半年数据（180天），按照每天1亿数据量为例，每天需要磁盘50G(根据数据差异度有波动)，180天的存储需要 9T，需要冗余，至少10T的磁盘空间

### 查询性能

以天为索引划分依据，一天一亿数据量，单索引分组聚合查询、分页查询都是毫秒级

但是跨天多索引查询是，此时基于ES的api，选择的是模糊索引方式，即可能查询全部索引，会慢很多，尤其是第一次查询。基本上是20s内，大部分执行都是秒级。

## CH

使用 8核 24G 200G 虚机测试。数据入库使用 10个线程并发写入，单批次1000条web访问应用日志。

### 入库性能

CH对TPS 不太友好，单线程入库，每1k数据库用时200ms以下，但10个线程，入库速度到580ms左右，20个线程入库，速度更是能到1.3s。

入库上，CH对cpu的使用很少，20个线程，cpu也很少打到100%，可能和机制有关。

内存占用也比较少 20个线程并发入库，内存基本都在1g以下。

磁盘上，CH 一亿条数据，压缩后只占用 2.13 GiB， 过程中也未发现有超过4G的情况。30亿数据，最终入库完成后，磁盘占用67G

考虑到中间过程的磁盘使用，每一亿按照5G计算，若存储180天，则需要900G，再冗余一下，需要1T的磁盘空间。

### 查询性能

CH对QPS 不太友好，单条sql，也会尽可能占用全部的cpu资源。以下面这条sql为例

|  |
| --- |
| select formatDateTime(access\_time, '%Y%m%d') as node, count(\*) as value from web\_access\_log where access\_time>='2024-07-01 00:00:00' and access\_time<='2024-08-01 00:00:00' and user\_account='user\_add\_458' and mobile like '1755103%' group by node; |

当前数据量20亿，单查询用时30s，但两个sql同时查询，则就会增加到60s。

单查一天速度很快，并发上相对更友好

单查询，24亿数据时，用时35s。

同步查看cpu情况，会发现，直接打满。

列表分页查询超级快（几十毫秒），无论数据量1亿还是30亿！！！

## 方案比较

我们当前的诉求只有：按条件分组统计、按条件分页查询。其实都没有发挥两款产品的优势。基于这两点诉求，当下两个方案：ES查询“可能”快一点，但资源使用较多，查询性能依赖当下的资源情况。而CH查询（分组）稍微慢点，但其对资源使用较少，并且有稳定的可预测性的资源预判，CH的测试虚机若是有16核，性能肯定更好。但CH对QPS/TPS不太友好，特别是对我们这种小批量频繁插入场景。

若不考虑资源使用，以日志能快速体现到管理页面优先的话，ES更好。

若不考虑日志实时性，以资源使用及后续可能的日志分析演进方向优先的话，CH更好。

## 问题：

1. 日志存储高可用怎么做？

# 方案设计

## 架构设计

不涉及，参考《易安联SDP日志优化设计方案》

### 静态视图

不涉及，参考《易安联SDP日志优化设计方案》

### 部署视图

部署上，无论是ES和CH都需要独立的一台机器进行分离部署，因为两者对资源的占用有“瞬时极高”的情况，若部署在一起会影响正常的业务使用。

从数据量上，按照每天1亿的数据量

ES最低配置：16核 32G 10T

CH最低配置：8核 16G 1T

从业务量上，以2w个用户为例，每个页面访问产生10-20条日志为例，每个用户进行250 ~ 500个页面访问即可达到。

#### 单机合一

#### 主备

#### 高可用

#### 异地双活

## 非功能设计

不涉及，参考《易安联SDP日志优化设计方案》

### 可测试性设计

### 性能设计

#### 接口交互

#### 资源使用检查

* CPU占用是否合理。例如，是否可能存在以下情况：
* 死锁
* 计算密集型任务
* 频繁的垃圾回收
* 数据库表设计、索引逻辑不当
* 内存占用是否合理。
* 同时加载大量数据（大文件、数据库、Redis）
* 数据导入导出
* IO占用是否合理。
* 不合理的数据库访问（缺少索引设计、频繁读写大量数据）
* 大文件读写
* 线程占用是否合理
* 线程执行长时间不结束（计算密集型任务、死锁、大量IO）

### 接口的兼容性

### 版本前向兼容性

### 安全性设计

### 可复用性设计

### 可扩展性设计

### 易用性设计

### 波及面分析

# 协作说明

## 流程a

# 存储设计

## 关系数据表

### xxx表

#### 数据表说明

#### 数据字段定义

#### 数据生命周期

#### 其它说明

## Redis存储

### xxx-Key设计

#### Key说明

#### key生命周期

# 类设计

# 接口设计

### 出参

# 任务分解

# 参考资料

<要求：列出编写本文件时的参考资料。>