应用研究·煤矿·

基于 HBase 与 Netty 的煤矿微震时序大数据存储优化

Coal Mine Microseismic Timing Big Data Storage Optimization Based on HBase and Netty

丁琳琳[1]，王智涵[1]，顾英豪[1]，王凯璐[1]，包鑫阳[2]

（1. 辽宁大学 信息学院，辽宁 沈阳 110036；2. 辽宁煤电产业控股有限公司红阳三矿，辽宁 辽阳 110101）

摘　要：在当前智能煤矿场景中，大量煤矿微震传感器所产生的时序数据呈爆炸式增长，进而对现有的存储系统及性能都有了更高的要求。目前已经存在基于分布式列族数据库 HBase 能够存储工业时序大数据的实例，但是由于没有考虑到特定业务场景中数据的特征关联问题，现有的策略仍然无法较好地满足煤矿微震波形时序数据的特定存储需求。针对上述问题，基于分布式存储系统 HBase，利用煤矿微震波形时序数据的特征，提出了基于 HBase 与 Netty 的煤矿微震时序大数据存储性能优化（CM^2TS-HBase），分成四个部分，分别为数据采集层、数据预处理层、数据中转层以及数据存储层。其中，数据采集层分为离线部分与实时部分，离线部分即存储在数据中心硬盘中的历史微震时序数据文件，实时部分即部署在某煤矿的多个微震波形传感器通过网络实时地向数据预处理层进行数据缓冲；数据预处理层实现对波形时序数据的文件进行对齐、解析以及序列化操作。根据煤矿微震波形时序数据特征提出了适用于微震波形时序数据的 HBase 数据表结构、预分区策略以及主键优化策略，有效地解决了数据存储过程中出现的数据热点问题以及数据分散问题；数据中转层提出了基于 Netty 与 Redis 的数据转发中间件平台为整个存储体系提供异步处理机制，较好地解决了高并发存储问题；数据存储层是基于分布式数据库 HBase 作为存储体系的底层存储媒介。最终根据真实数据集的存储耗时证明了相较于原生存储方法（HBase API）与基于金融时序数据存储优化（FTBase），CM2TS-HBase 在煤矿微震时序数据的存储性能有了明显提高。

关键词：煤矿微震大数据；时序数据；分布式存储；高并发事务处理；预分区策略

中图分类号：TP392　　**文献标志码**：A　　**文章编号**：1672-609X(2023)05-0029-07

Abstract: In the current intelligent coal mine scenario, the time-series data generated by a large number of coal mine microseismic sensors is growing explosively, which in turn has higher requirements for the existing storage system and performance. There are already instances where HBase, a distributed column family database, can store industrial time-series big data, the existing strategies are still unable to better meet the needs of coal mine microseismic waveform time-series data because they do not consider the feature association of data in specific business scenarios. specific storage requirements. In view of the above problems, based on the distributed storage system HBase, using the characteristics of coal mine microseismic waveform time series data, a coal mine microseismic time series big data storage performance optimization (CM2TS-HBase) based on HBase and Netty is proposed, which is divided into four parts, namely data acquisition layer, data preprocessing layer, data transfer layer, and data storage layer. Among them, the data acquisition layer is divided into offline part and real-time part. The offline part is the historical microseismic time series data files stored in the hard disk of the data center, and the real-time part is the multiple microseismic waveform sensors deployed in a coal mine. The layer performs data buffering; the data preprocessing layer implements alignment, parsing, and serialization operations on files of waveform time series data. According to the characteristics of coal mine microseismic waveform time series data, the HBase data table structure, pre-partition strategy and primary key optimization strategy suitable for microseismic waveform time series data are proposed, which effectively solves the data hot spot problem and data dispersion problem in the data storage process; data transfer layer A data forwarding middleware platform based on Netty and Redis is proposed to provide an asynchronous processing mechanism for the entire storage system, which better solves the problem of high concurrent storage; the data storage layer is based on the distributed database

[作者简介] 丁琳琳(1983—)，女，教授，从事大数据管理及图数据管理等方面研究。

[基金项目] 国家自然科学基金项目(62072220)；国家重点研发计划项目(2022YFC3004603)；辽宁省自然科学基金计划项目(2022-KF-13-06)

[引用格式] 丁琳琳，王智涵，顾英豪，等. 基于 HBase 与 Netty 的煤矿微震时序大数据存储优化[J]. 中国矿山工程，2023，52(5)：29-35.

HBase as the underlying storage medium of the storage system. Finally, according to the time-consuming storage of real data sets, it is proved that compared with the native storage method (HBase API) and financial time-series data storage optimization (FT-HBase), CM2TS-HBase has significantly improved the storage performance of coal mine microseismic time-series data.

Key words: coal mine microseismic big data; time series data; distributed storage; high concurrent transaction processing; pre-partitioning strategy

1 前言

随着智慧矿山相关技术的发展,煤矿中众多微震传感器产生的时序数据呈现出爆炸式增长的态势。在时序数据规模逐渐增大的背景下,海量煤矿数据微震波形时序数据如何高效、合理地存储成为了大数据领域的亟待解决的问题,即煤矿微震波形时序大数据存储问题。时序数据具有时间序列化、时段密集化、单条数据高权重、数据产生高并发、数据总量巨大的特点[1]。煤矿微震时序波形数据作为工业时序数据中的一种,通常是由上百台工业设备的上万个传感器产生,并且各传感器之间存在着较为复杂的依赖关系,具有采样周期密集和强关联的特点[2]。

当前煤矿微震时序大数据的存储方案,通常可采用传统文件系统存储、关系数据库存储以及分布式存储三种方案。对于传统的文件系统存储方式,尽管操作简便,但需重复进行对齐操作和读取传感器波形文件操作以满足后续计算部分的数据需求,导致重复操作占据程序执行的大部分时间,并且无法对数据进行有效管理以及对数据的快速检索。对于关系型数据库存储方式,在存储煤矿微震波形时序数据时存在高并发事务场景下性能较差、扩展性差等缺点。对于分布式存储在存储时序大数据方面,尽管相对于关系型数据库已经有了一定的优化,但也存在一些缺陷,在煤矿微震波形时序大数据的存储场景下,需要考虑数据的特征关联问题、存储热点数据问题、存储分散以及海量数据存储数据阻塞问题,现有存储策略均无法较好解决。

因此,针对上述诸多问题,本文采用基于 Hadoop 分布式平台[3]的 NoSQL 非关系型数据库 HBase[4]作为底层存储介质,因为其在可扩展性、并发度、分布式以及面向列存储等方面均较为突出。并结合煤矿微震波形时序数据的高并发、时间序列化以及海量数据的特点,采用适用于煤矿微震波形时序数据的表结构设计策略、预分区策略以及行键优化策略对存储性能进行优化。同时,采用 Netty[5]网络通信框架编写的中间件集群作为数据中转层,对数据接收层流转而来的海量数据进行分流处理,有效避免数据阻塞问题。使用 Redis 内存数据库[6]的有序集合数据结构实现负载均衡算法的最小连接法[7],使 Netty 中间件集群中的每个节点都能够被合理地分配。最终,采用真实的传感器离线数据对实验进行了验证。

2 相关工作

基于 HBase 数据库实现的开源时序数据库 OpenTSDB[8]具有优秀的扩展性和伸缩性,可以轻松地水平扩展集群规模来处理大量数据。此外,文献[9]以时间序列数据的特点为中心,实现了分布式数据库存储海量时间序列数据的方法和应用。InfluxDB 是一个开源时序数据库,用于处理时序数据的高性能读写操作,InfluxDB 具有高性能、易扩展、数据可视化等优点[10]。基于 Cassandra[11]数据库构建的开源时序数据库 KairosDB 支持高效存储和查询时间序列数据,Kairos 还具有支持多种数据类型、提供丰富的查询接口、易于使用和部署等优点[12]。结合 Logstash 和 Elasticsearch 同样可以实现对时序数据的高效存储和查询,并具有良好的扩展性和灵活性[13]。

在 HBase 分布式数据库底层存储原理方面,文献[14]利用 JavaNIO 技术设计了一种 HBaseRPC 客户端的非阻塞通信模型,文献[15]则提出了一种存储大规模空间向量数据的模型,适用于处理大规模数据的应用。另外,文献[16]基于 HBase 与 Redis 高性能缓存,为图片数据的查询和存档性能做出了客观的贡献。在存储优化架构方面,文献[17]提出了四层结构,并使用 Netty 网络通信框架作为数据缓存中间件,该架构在金融时序数据的高并发存储场景中得到了可观的优化效果。最后,文献[18]设计并实现了三层存储架构,并将数据缓存中间件集群化处理,有效避免了高并发场景下海量传感器数据阻塞的问题。

以上述研究工作为基础,本文设计并实现了一

种基于 HBase 与 Netty 的煤矿微震时序数据存储优化方案。该方案综合了多个方面的优化措施,成功解决了煤矿微震时序数据存储分散以及存储热点等问题。同时,该方案在高并发事务处理方面也有可观的优化效果,大幅提升了存储效率,为煤矿微震时序数据的存储和处理提供了有力的支撑和保障。

2　CM2TS－HBase 存储框架

本文设计并实现了一种基于 HBase 与 Netty 的煤矿微震时序波形数据存储优化框架。该框架针对煤矿微震时序波形数据的特征,解决了热点数据、存储分散以及存储过程中的高并发数据阻塞等问题,从而大幅提升了煤矿微震时序波形数据的存储效率和性能。

2.1　存储框架整体架构

为了应对微震监测传感器分布广、数据总量大的特点,以及高并发处理、热点数据和存储分散的挑战,本文开发了一种名为 CM2TS－HBase(Coal Mine Microquake Time Series data-HBase)的煤矿微震时序波形数据存储框架,该存储框架基于 HBase 与 Netty 技术实现。图 1 所示为该存储框架的详细架构图。

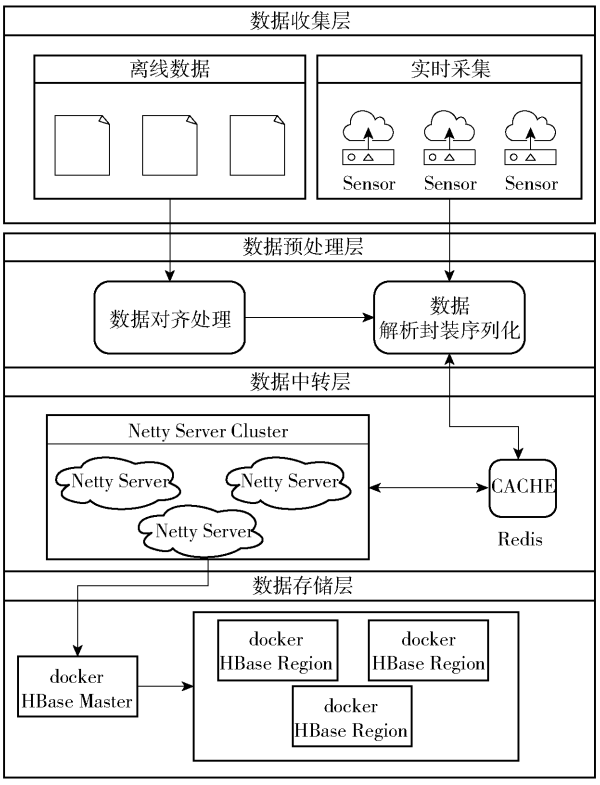

图 1　CM2TS－HBase 存储框架架构图

存储引擎整体分为以下四部分。

(1)数据收集层:该层分为离线和实时两部分。离线数据就是数据中心存储在硬盘的二进制原始波形文件;实时数据就是在实际应用环境下部署在矿区的众多传感器实时产生的时序微震波形数据。下面将离线状态中的每个工作线程以及实时状态中的每个传感器统称为客户端。

(2)数据预处理层:对于离线数据需要对原始波形文件进行对齐操作,找到众多文件中时间重叠的部分进行解析并序列化,最后在多线程并发环境下将数据通过 Http/3 传输至数据中转层;实时传感器数据则可以直接进行解析封装序列化操作然后同样通过 Http/3 传输至数据中转层。

(3)数据中转层:数据中转层是基于 Netty 与 Redis 的数据转发中间件,可以对高并发事务进行优化处理,利用负载均衡思想将单一客户端承受的压力均衡地分布给所有承担存储任务的服务器。

(4)数据存储层:基于分布式数据库 HBase 作为存储体系的底层存储媒介。在实验环境下,数据存储层的 HBase 分布式存储节点被部署在云服务器的 Docker 虚拟化容器中。负责存储数据中转层传来的序列化数据。

2.2　主键优化策略

HBase 是一个由众多节点组成的集群架构。优秀的主键设计可以显著提升 HBase 的读写效率,而且可以将一段时间内存储的数据放置在连续的物理空间内,这样也能有效地解决数据分散的问题。

主键的设计原则有四点,分别是长度原则、唯一原则、排序原则以及散列原则。本文设计了适用于时序微震波形数据特点并结合主键设计四原则的主键优化策略。主键结构示意图如图 2 所示。

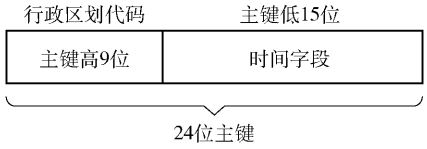

图 2　主键结构示意图

基于主键长度原则,将主键长度设置为 24 位,由于目前大多数服务器是 64 位操作系统,其内存均按照 8 字节对齐。因此主键设置为 24 位可以提高寻址效率。其中主键高 9 位表示行政区划代码,最小可以精确到乡级行政区;低 15 位是数据的时间字段,单位可以精确到秒级,基于上述主键设计,HBase 在存储时首先会按照高 9 位进行排序,此时具有相同地区编号的数据就会被存储在连续的物理空间中;若高 9 位相同,就会根据低 15 位的时间字

段按照写入时间顺序进行存储。

2.3 预分区策略

HBase 分布式数据库存储数据的基本单位被称为分区(Region),HBase 默认建表时设置一个 Region,并且这个 Region 的主键是无边界的,因此在数据写入时,所有数据都会写入到这个默认 Region,但随着数据量的不断增加,HBase 会进行 Split 操作分割成 2 个 Region。在这个过程中就会出现两个问题:数据不停向一个 Region 写入会造成热点数据问题;Split 操作会消耗宝贵的集群 I/O 资源。

因此本文在建表时就基于上述主键特点创建了多个空的 Region,并确定了每个 Region 的起始和终止主键,这样可以使每条数据均匀地命中各个 Region,从而避免了热点数据的产生并降低了 Split 的发生概率。

预分区的前提是有明确的主键结构,基于上述主键结构,根据高 9 位的行政区划代码进行分区操作。根据地区的不同进行分类,并按照各个地区的煤矿矿区数量动态分配 Region 数量,分区分类见表 1。本表数据来源于中华人民共和国行政区划代码 1 983 版本,仅以东北、华北以及华东为例。

表 1 分区分类表

煤矿所属地区	地区代码区间	Region 数量
华北	1xxxxxxxx	5
东北	2xxxxxxxx	10
华东	3xxxxxxxx	3

根据上述预分区策略对 HBase 数据库进行预分区操作,可将数据均匀地进行分布式存储,基本解决了煤矿微震波形时序数据在 HBase 分布式数据库存储时的数据热点问题。

2.4 数据表设计与数据对象序列化操作

基于煤矿微震波形时序数据特点以及多传感器网络的场景,本文设计了一种适用于此类特殊场景的数据表结构,数据表结构示意图如图 3 所示。

HBase 分布式数据库在进行查询操作时的规则是以主键为索引进行的,主键可以确定唯一一行数据,但无法确定一个具体的 Cell。本文将列族设置为具体的煤矿矿区名称,以下不同的列分别表示部署在该煤矿中的传感器代号,这样在后续的计算任务中便可以进行多维度多条件的查询。

由图 3 可见数据行中存储的是序列化数据,序列化的定义是为了将数据对象转换为更易进行网络传输和保持格式的过程。在存储引擎当中客户端与 Netty Server 之间的通信在宏观的角度看就是两个进程之间的远程交互,客户端会根据时序波形的特征将解析后的原始数据封装成固定格式的数据对象,使序列化后的数据在空间上被大幅度压缩,提高双方在远程传输数据过程的通信效率。

主键	列族:煤矿矿区名称					
rowkey	列:S	列:Y	列:U	列:Z	列:V	列:T
	序列化数据	序列化数据	序列化数据	序列化数据	序列化数据	序列化数据
. . . . .						

图 3 煤矿微震波形时序数据表结构示意图

2.5 基于 Netty 与 Redis 的异步数据缓存中间件

本节使用了两个非常流行的组件,分别是 Netty 框架与 Redis 内存数据库。该模块的架构图如图 4 所示。

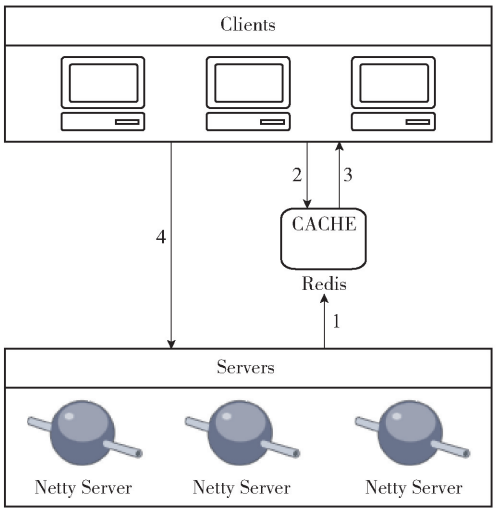

图 4 基于 Netty 与 Redis 的异步数据缓存中间件架构图

客户端(Clients),在实时状态下,每个微震波形传感器都可以被认为是一个客户端;在离线状态下,线程池中的每个发起存储请求的线程任务也可以被设定为客户端。图中步骤 2、3 表示客户端在发起存储请求前会从 Redis 缓存中查找最小连接服务器节点。图中步骤 4 表示客户端根据查询到的结果向当前最小连接服务器发送存储请求。

Redis 缓存,该部分用于实现对中间件服务器集群的负载均衡调度。本文采用负载均衡算法中最为流行的最小连接法,使用 Redis 数据库中的有序集合数据结构,基于所有当前正在运行服务器的连接

数进行排序,使每个发起存储请求的客户端都能获取到当前压力最小的 Netty Server。

Netty Server,基于 Netty 框架开发的中间件服务器,并以集群的形式分布式地向 HBase 发送待存储数据。

3　数据存储过程

CM2TS-HBase 数据存储过程如图 5 所示。

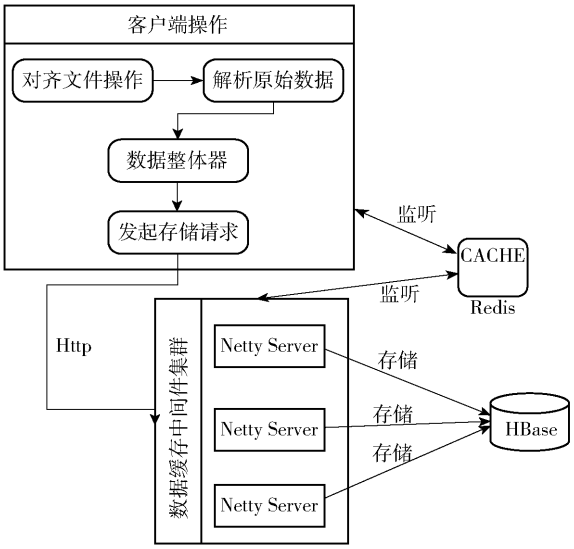

图 5　数据存储过程

原始文件解析主机开启多线程并发解析原始波形文件,并在此过程中调用数据整理器对主键进行调整并对波形数据对象进行序列化整理,使其便于网络传输。

携带整理完毕的数据发起存储请求,在请求前需要根据 Redis 缓存存储的中间件服务器集群中各个服务器的实时连接状态,选取最优状态的服务器进行存储。同理,每当中间件服务器开启都会将本节点的信息存入 Redis 中向存储请求线程提供服务;同时服务器的关闭与宕机也会进行更新操作。

存储请求线程得到最优节点信息后尝试与服务器建立测试通信,如果成功便更新节点信息并携带序列化波形数据对象发送存储请求,中间件服务器接收到请求后根据预分区策略将数据存储到对应 Region 中,反之将报错信息返回给客户端并记录日志,最终所有存储请求传输完成后关闭连接,并更新对应节点信息。

CM2TS-HBase 存储过程算法如算法 1 所示。该算法时间复杂度为 $O(n)$,空间复杂度为 $O(n)$。

算法 1:CM2TS-HBase 存储过程算法

Input:待存储数据 data

OutPut:存储完成状态 R

Begin

1:*nserver*← *zrange nserset* 0 0

2:*boolean connected* ←*doConnected*(*nserver*);

3:*if connected* =*false*

4:　　*zrem nserset nserver*;

5:*else*

6:　　*zincrby nserset* 1 *nserver*;

7:*while data*←*hasNext*( )

8:　　*R*←*doWrite*(*data*);

9:　　*if*(*R* = *False*)

10:　　*emit R*;

11:　　*end while*

12:　*else*

13:　　*continue*;

End.

算法第一行通过 Redis 命令 zrange 0 0 获取到当前 Netty Server 服务器集合中连接数最小的服务器 nserver,算法第 2 行到第 6 行表示对选中的 Netty Server 进行连接测试,如果连接失败,通过 Redis 命令 zrem nserset nserver 删除该服务器,如果连接成功,则通过 Redis 命令 zincrby nserset 1 nserver 将 nserver 的连接数加一,算法中第 7 行到第 13 行表示进入存储循环过程,通过 doWrite 方法向 HBase 数据库写入数据,并实时返回存储状态 R,如果存储出现问题,R 将携带报错信息返回控制台,如果存储过程未出现报错问题,则存储过程将继续循环,存储下一条数据,直到此轮存储请求结束。

4　实验分析

4.1　实验环境

实验在 HBase 伪分布式环境下进行。硬件环境为一台 2 核 4G 云服务器,通过 Docker 虚拟容器化后的 4 台虚拟主机组成的伪分布式集群。原始数据解析主机为一台 Intel 酷睿 i5 8250U 8 核 1.6 GHz 16 GB RAM 主机对历史微震数据文件进行多线程解析存储。

软件平台为 CentOS 7.6-64 位、JDK-1.8、HBase-2.3.6、Zookeeper-3.4.10、Hadoop-3.1.3。服务器节点信息见表 2。

本文实验设计了 3 种存储方案,通过对比 3 种方案在存储煤矿微震波形时序数据的性能表现得出最终结论。3 种方案分别为:通过 HBase 原生客户

端 Put 方法存储 HBase(HBaseAPI);基于 HBase 的金融时序数据存储系统思想存储煤矿微震波形时序数据(FTBase);基于 HBase 与 Netty 的煤矿微震时序大数据优化策略存储煤矿微震时序数据(CM2TS-HBase)。

表 2 服务器节点信息表

ip	hostname	character	Netty-Server num	port
172.19.0.3	dadoop01	Master	3	45 889
172.19.0.2	dadoop02	Region	2	45 890
172.19.0.4	dadoop03	Region	2	45 891
172.19.0.5	dadoop04	Region	2	45 892

4.2 数据集

实验所用数据为 2019 年辽宁某煤矿真实部署传感器监测到的历史微震波形文件,每条数据包含采集时间、波形空间坐标等内容。传感器数据采集频率为 5 000 条/s,实验设置 3 个组别,分别是 6 文件组、12 文件组以及 24 文件组进行并发存储,每个文件大小约 250 MB。

4.3 评估指标

针对煤矿微震波形时序数据的存储性能优化实验,采用 2 项指标作为最终评估标准。分别是:根据固定存储文件数量统计存储耗时情况以及单位时间节点存储数据量的对比。

(1) 固定文件数量统计存储耗时情况,在离线状态下,可通过离线波形文件以多线程的方式模拟出煤矿微震传感器的实时存储数据场景。同时可以统计出高并发状态下各个实验方案的存储性能。因此在单位文件数量的前提下,存储耗时越小即表示存储性能更佳,即对高并发场景的存储性能进行了有效地提升。

(2) 单位时间节点存储数据量对比,实验根据单位文件数量下存储持续时间设置实验组,分别在各实验组的时间节点统计存储的数据量。存储数据量越大就说明性能更佳。

4.4 实验结果

通过对比存储总耗时区分二者的性能差距,存储总耗时实验对比结果见表 3。当实验设置文件数量为 6 个和 12 个时,3 种方案的实验结果相差不大。当实验将文件数量提升至 24 个时,HBaseAPI 的处理时间明显变长,延长至 167 s;同时,CM2TS-HBase 存储耗时为 97 s,FTBase 存储耗时为 78 s。CT2MS-HBase 存储耗时明显低于 HBaseAPI 与 FTBase,存储耗时对比如图 6 所示。

表 3 实验耗时结果表

客户端数量	HBaseAPI	FTBase	CM2TS-HBase
6	38s	54s	61s
12	89s	83s	78s
24	167s	124s	97s

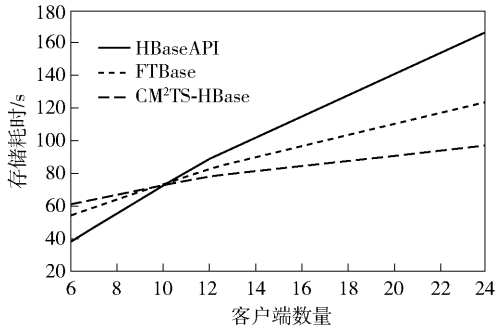

图 6 存储耗时对比图

单位时间节点存储数据量实验结果如图 7 所示。在实验过程中,HBaseAPI 的每秒钟存储数据量由持续时间 20 s 时的 31.6×104 条/s 提升至持续时间 120 s 时的 86.2×104 条/s,性能有大幅度的提升。同时,通过与 FTBase 的对比中也可以看出负载均衡算法的引入对中间件集群的资源分配进行了合理地调整,进而提升了整体存储系统的性能。因此,在高并发场景下,CM2TS-HBase 的表现更好。

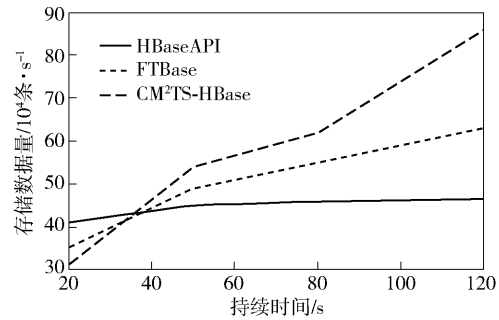

图 7 单位时间存储数据量对比图

5 结论

本文探讨了如何基于 HBase 分布式数据库、Netty 网络通信框架以及 Redis 内存数据库等技术来存储煤矿微震时序大数据。在实践中发现,由于 HBase 分布式数据库的自身缺陷和煤矿微震时序大数据的特点,需要采取特殊的策略来进行预分区、主键优化和数据表结构的设计。通过本文提出的策略,可以有效地提高煤矿微震时序大数据的存储效率和查询性能。

本文所述的设计思想从实际问题出发,针对煤矿微震时序大数据的存储问题提供了有效的解决方案。这对于工业界使用传感器产生的时序数据进行生产或安全维护具有重要的参考价值和实用意义。此外,本文所介绍的技术和策略也可以为其他领域的时序数据存储问题提供一些借鉴和参考。

[参考文献]

[1] 杨力,陈建廷,向阳. 基于 HBase 的工业时序大数据分布式存储性能优化策略[J]. 计算机应用,2023,43(3):759 - 766.

[2] 李晓根. 基于 Hadoop 的工业大数据监测分析平台技术实现[D]. 北京:北方工业大学,2019.

[3] Amenenh Zarei, Sohahla Safari, Mahmaod Ahmadi, et al. Past, Present and Future of Hadoop: A Survey [C]. arXiv: 2202. 13293,2022.

[4] LUNTOVSKYY. From Big Data to Smart Data: Best Practices for Data Analytics [A]. Highly-Distributed Systems: IoT, Robotics, Mobile Apps, Energy Efficiency, Security[C]. Cham: Springer, 2022.

[5] YANG S. Performance improvement of apache storm using infiniband RDMA [J]. The Journal of Supercomputing , 2019(75): 6804 - 6830.

[6] XU H J. Research on mass monitoring data Retrieval Technology based on HBase [C]. Journal of Physics: Conference Series,2021.

[7] ALANKAR B. Experimental setup for investigating the efficient load balancing algorithms on virtual cloud [J]. Sensors, 2020: 7342.

[8] SUN P. Design of containerized marine knowledge system based on IoT-Cloud and LoRaWAN [J]. Personal and Ubiquitous Computing, 2022: 1 - 13.

[9] OCHIAI H. Facility information management on hbase: Large-scale storage for time-series data [C]. 2014 IEEE 38th International Computer Software and Applications Conference Workshops. IEEE, 2014: 306 - 311.

[10] CALATRAVA. Introducing Polyglot-Based Data-Flow Awareness to Time-Series Data Stores [J]. IEEE access, 2022: 69398 - 69411.

[11] KAUSAR A. A Study of Performance and Comparison of NoSQL Databases: MongoDB, Cassandra, and Redis Using YCSB [J]. Indian J. Sci. Technol 15, 2022: 1532 - 1540.

[12] BIRD M. Real-world implementation and cost of a cloud-based MPC retrofit for HVAC control systems in commercial buildings [J]. Energy and Buildings 270, 2022: 112269.

[13] CHEN Y F. Deep Attentive Anomaly Detection for Microservice Systems with Multimodal Time-Series Data [C]. 2022 IEEE International Conference on Web Services (ICWS). IEEE, 2022: 373 - 378.

[14] 胡波,谭良. HBase 架构中 RPC 客户端的通信性能优化[J]. 计算机科学,2016,43(4): 97 - 101.

[15] WANG Y. HBase storage schemas for massive spatial vector data[J]. Cluster Computing,2017: 3657 - 3666.

[16] ZHOU L J. Data cache optimization model based on hbase and redis [C]. Proceedings of the 3rd International Conference on Data Science and Information Technology, 2020: 31 - 35.

[17] 刘博伟,黄瑞章. 基于 HBase 的金融时序数据存储系统[J]. 中国科技论文,2016,11(20): 2387 - 2392.

[18] 李朝奎,王露瑶,周新邵,等. 基于 HBase 的矢量空间数据存储与查询方法及其应用[J]. 地理科学,2022,42(7): 1146 - 1154.