文章编号：2096-1472(2021)-08-16-04
DOI:10.19644/j.cnki.issn2096-1472.2021.08.004

一种分布式存储索引中间件的设计与实现

黄 静，别要祥，谢 宣

(浙江理工大学信息学院，浙江 杭州 310018)
✉syhj_sy@163.com; 1009214965@qq.com; 15951612952@163.com

摘 要：HBase(分布式存储数据库)是大数据存储领域的热点技术，为信息化快速发展带来的存储问题提供了有效的解决方案。针对HBase检索低效以及企业对系统的低耦合、高扩展性需求，通过分析HBase检索困难的原因，设计一个索引中间件。利用Lucene(全文检索引擎工具)技术构建二级索引，以统一接口的形式提供服务。经过实验验证，索引中间件在保证写入需求的情况下，有效地改善了查询性能，在千万级数据量下仍然达到毫秒级检索，并且耦合性低，易于部署，可以快速整合到已有系统中，具有较强的泛用性。

关键词：HBase；Lucene；中间件；索引
中图分类号：TP311.1 **文献标识码**：A

Design and Implementation of a Distributed Storage Index Middleware

HUANG Jing, BIE Yaoxiang, XIE Xuan

(*School of Information*, *Zhejiang Sci-Tech University*, *Hangzhou* 310018, *China*)
✉syhj_sy@163.com; 1009214965@qq.com; 15951612952@163.com

Abstract: HBase (Distributed Storage Database), a hot technology in the field of big data storage, provides an effective solution to the storage problems brought about by the rapid development of information technology. Aiming at the low efficiency of HBase retrieval and the enterprise's needs for system with low coupling and high scalability, this paper proposes to design an indexing middleware by analyzing reasons for the difficulty of HBase retrieval. Lucene (a full-text search engine tool) technology is used to build a secondary index to provide service in the form of a unified interface. The experimental verification shows that the proposed indexing middleware can effectively improve the query performance while ensuring the writing requirements. It can still reach the millisecond level of retrieval under data volume of tens of millions. Besides, it has low coupling, easy deployment, and can be quickly integrated into the existing system with strong versatility.

Keywords: HBase; Lucene; middleware; index

www.rjgcs.com

1 引言(Introduction)

在云时代的背景下，计算机需要管理的数据呈指数级增长[1]，庞大的数据规模给存储和处理带来了极大的挑战。受单机存储的性能瓶颈以及单点故障影响，分布式存储应运而生[2]。HBase是一个面向列的高性能分布式存储系统[3]，其优异的高并发读写性能受到众多企业的青睐，甚至作为企业核心数据库使用。但在大数据存储情况下，HBase存在查询慢的问题[4]，并且相关检索优化措施实现复杂，容错性能低，与系统业务耦合紧密，不利于系统的扩展和升级。

本文从低耦合性与高扩展性出发，设计索引中间件CFH(Coordinator for HBase)。该中间件作为一个协调器，承接客户端HBase操作请求的处理与转发，通过Lucene技术构建二级索引，在保证高效写入与查询的同时，达到业务低侵入性、易于快速扩展与部署的目的。

2 HBase检索低效与优化研究(Research on inefficiency and optimization of HBase retrieval)

HBase检索低效的原因可以归纳为三种：

(1)HBase以键值对的方式存储数据，仅提供B+树构造的

基金项目：浙江省重点研发计划项目(2021C01048)．

一级索引(也称为主键)。因此，基于主键的查询具有很高的效率，对非主键的查询效率很低。虽然，针对非主键字段查询，HBase通过MapReduce(分布式计算)进行检索，但这种方式的硬件资源消耗较高，时间损耗也不尽人意。

(2)海量的数据对检索也会产生影响。大规模数据会建立更多的索引表，同时，非索引查询会触发全文扫描，对海量数据进行全文检索，资源消耗非常巨大[5]。

(3)HBase在检索时，需要将数据加载到内存中，与CPU的处理速度相比，磁盘的读写操作非常慢。索引表过大、过多，可能会造成频繁的磁盘I/O操作，这会极大增加查询的时间[6]。

针对HBase查询慢的问题，华为技术公司基于Coprocessor(协处理器)开发的hindex框架支持二级索引，但hindex基于HBase 0.94版本开发，如今版本老旧，也不再维护。徐江峰等[7]提出New-Grid框架，此方案扩展性较差，也未作负载均衡，容易产生性能瓶颈。党鹏[8]提出LBIndexer机制，利用最小均方差(Least Mean Square，LMS)模型管理索引数据，增强了索引的可用性，但基于LMS设计的索引，其性能易受长久不使用的大索引文件影响，会降低索引文件的读写与查询速度。王惟一[9]也基于LMS提出CFIDM数据模型，根据列族存储特点缩小待查询的存储文件数量，从而提高查询效率。黄璨等[10]通过分片计数布隆过滤器，利用哈希映射的方式建立二级索引，但在分片数据量大的情况下优化效果并不明显。叶奇明等[11]将Hive(数据仓库工具)作为HBase的查询入口，然而在获得查询效率提升的同时增加了系统的不稳定性。朱松杰等[12]通过协处理器实现二级索引的快速构建，并根据HBase表的变化自动更新索引，由于该方案对所有列建立索引且索引文件与HBase处于同一服务器，这会增大内存资源的消耗，严重时甚至可能使服务器瘫痪。

HBase查询优化的方案有很多，其探索脚步也从未停止，但实现方式大多较为复杂，局限性大，并且代码耦合性强，无法进行广泛应用，造成资源的重复建设。

3　CFH设计(Design of CFH)

建立一个独立于业务系统、支持多级索引构建的协调机制，对HBase查询优化与系统维护具有十分重要的意义。为了达到这样的目的，本节基于Lucene技术设计索引中间件CFH。

3.1　CFH介绍

CFH服务端基于Netty(网络应用程序框架)开发，Netty的NIO(Non-blocking，同步非阻塞I/O)机制以及零拷贝技术具有优异的性能[13]。服务端打包后通过jar文件启动，仅在配置文件中进行HBase链接地址等少量设置，即可快速启动，易于部署。通过CFH，可以按需建立索引，区分出冷热数据，加速查询。下面从不同的数据操作类型阐述CFH的业务逻辑，主要是针对二级索引、中间件进行的辅助操作。

(1)查询操作。CFH会判断查询操作是否基于索引，如果不是，则先通过Lucene查询出符合条件的主键值，然后创建新的查询语句，通过主键值在HBase中查询数据。调用Lucene技术时，如果发生索引覆盖，则直接返回查询结果。查询流程如图1所示。

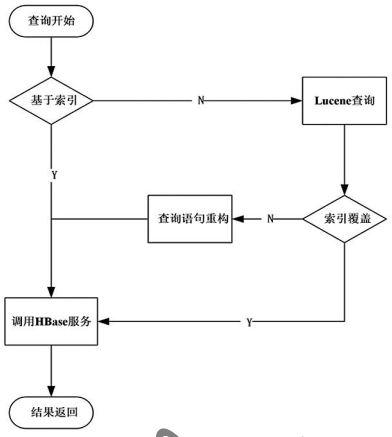

图1 CFH插入流程示意

Fig.1 Insert flow diagram of CFH

(2)修改操作。重点在于判断索引字段是否发生修改，如果未修改索引字段，则无须操作二级索引文档，仅调用HBase服务处理原数据。

(3)新增操作。此操作需要客户端告知中间件建立索引的字段。除了把新数据添加到HBase中，HBase还会将需建立索引的字段与数据的主键添加到二级索引文档中，便于后续检索。写入流程如图2所示。

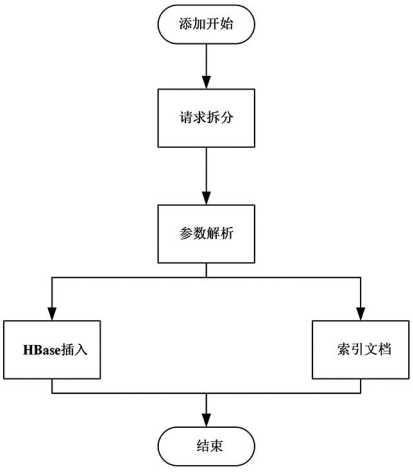

图2 CFH新增流程示意

Fig.2 New process schematic of CFH

(4)删除操作。若删除的数量比较大，CFH会检查索引记录数，如果剩余索引数据比较少，CFH会将剩余数据建立新的索引，并启动定时任务，在不影响业务运转的情况下，定期删除数据。若删除的数据较少，则直接删除Lucene索引文档中的数据。

3.2　CFH架构设计

CFH涉及服务端与客户端，建立一套有效的通信机制非常重要。CFH的工作流程可以划分为八个步骤，包括：

(1)客户端与服务器端建立连接；

(2)客户端对数据序列化并传输；

www.rjgcz.com

(3)服务端接收数据，完成反序列化并验证操作权限；

(4)服务端根据请求类型，选择对应的解析器解析请求参数；

(5)对请求进行拆分，完成请求重写；

(6)本地查询以及通过RPC(远程过程调用)调用Lucene或HBase服务；

(7)封装数据、序列化以及回写客户端；

(8)客户端接收数据，反序列化。

中间件的服务端采用Zookeeper(分布式协调工具)进行集群部署，采用一主多从结构，CFH集群的主节点负责处理HBase写操作，读操作由从节点进行。CFH架构示意图如图3所示。

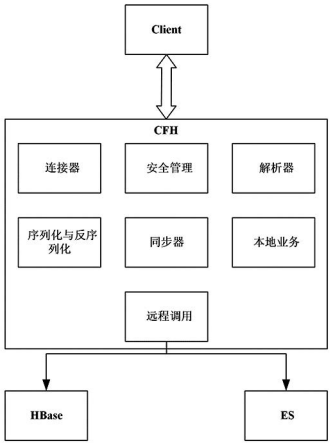

图3 CFH架构

Fig.3 Architecture of CFH

由图3可知，CFH服务端包含七个部分，各部分功能如下：

(1)连接器：管理与客户端的连接，通信方式采用Http(Hypertext Transfer Protocol，超文本传输协议)+JSON(JavaScript Object Notation，JS对象简谱)的形式，网络I/O模型为Netty。

(2)安全管理：负责数据信息的加密以及连接权限的管理。

(3)序列化与反序列化：数据传输前后，控制对象与字节序列之间的转化，CFH采用FastJson工具包进行JSON数据的解析与生成。

(4)解析器：解析请求，针对不同的请求类型，调用相关解析器以及本地业务。

(5)本地业务：业务逻辑增强，对客户端请求进行重构以及生成本地索引。

(6)同步器：采用原生Lucene服务包建立索引时，集群节点进行索引文档同步。

(7)远程调用：调用HBase等服务、操作索引以及获取客户端需要的数据。

为了增强可用性，服务端设置了两种创建索引的方式。第一种通过CFH原生集成的Lucene服务包实现，此方法生成的索引文件在中间件所在的服务器上。由于CFH可以进行分布式部署，集群节点需要对索引进行同步，从而保证数据的一致性，这在一定程度上会增加系统资源消耗。索引同步通过操作日志进行，CFH集群中非查询操作由主节点处理，将操作记录保存到同步日志中，主节点会定时往其他节点发送同步日志，从节点接收到日志文件后更新索引文档。第二种是利用ES(Elasticsearch)。ES是一种基于Lucene技术的分布式搜索引擎，有着较为成熟的索引构建以及检索方法，但这种方式需要更多的服务器资源。如果通过ES构建二级索引，需要在服务端启动之前，在配置文件中做出相应的设置。采用ES的情况下，可以经Kibana(可视化平台)工具对索引进行可视化管理。在代码实现中，根据配置参数，利用JNDI(Java Naming and Directory Interface，Java命名和目录接口)机制，CFH会加载不同的索引创建工具包，即采用本地生成或者ES生成索引的方式。

CFH客户端包括连接器、序列化与反序列化以及转发器三部分。连接器会从Zookeeper中获取CFH的集群信息，转发器则根据读写请求选择CFH集群节点。客户端操作方式不做过多新的设计，在已有开发包的基础上进行封装。CFH客户端操作基于<groupId>org.apache.HBase</groupId><artifactId>HBase-client</artifactId>包中提供的API，对其中的部分API进行了扩展，通过新的API告知CFH是否需要操作二级索引，默认服务端会对所有字段建立索引。客户端利用Spring Boot(服务开发框架)的自动装配机制降低配置复杂度，通过系统的配置文件导入CFH外部配置。

3.3 CFH故障处理

引入中间件会增加系统的不稳定性。如果中间件服务发生故障，会导致该链路服务不可用[14]。为了避免中间件故障造成的服务不可用，CFH服务端以Zookeeper作为注册中心进行集群部署。通过集群部署可以降低服务不可用的概率，同时增强系统的吞吐量。此外，在CFH客户端设置了故障、超时重试机制。如果重试达到一定次数仍未获得响应，客户端将越过中间件直接访问HBase，在HBase正常的情况下保证提供服务，但客户端对HBase服务器信息的获取仍通过中间件。客户端会设置定时任务，持久化最新的服务信息到本地，客户端本身不需要对HBase进行配置。

4 性能测试(Performance testing)

为了方便查看二级索引信息，实验所采用的索引构建方式为ES，通过Kibana可视化界面操作索引，构建索引时采用的分词器为IK Analyzer(中文分词工具包)。测试用的主机为CentOS系统，CPU为Intel(R) Core(TM) i9-7900X CPU @ 3.30 GHz，64 GB内存，通过VMWare15软件搭建三台CentOS7虚拟机，每台分配8 GB内存。测试软件采用集群方式部署，版本信息如表1所示。HBase与ES内存分配为4 GB。测试软件未做其他优化措施，均为默认设置。客户端在另一台物理机上，Windows 10操作系统，配置为Intel(R) Core(TM) i5-9600K CPU @ 3.70 GHz，16 GB内存。实验准备过程中，利用Python脚本批量生产了1,000万条实验数

www.rjgcz.com

据，每条数据约76 字节，包含用户个人信息共7 个字段，分别为姓名、年龄、性别、学历、电话、邮箱和国籍，每个字段由字母、数字或特殊字符组成。

表1 软件版本信息

Tab.1 Information of software version

名称	版本号
Hadoop	2.8.5
HBase	2.1.9
Kibana	7.0.0
Elasticsearch	7.0.0
Zookeeper	3.1.4

表2表示数据量与所需建立二级索引字段数不同组合情况下的写入耗时(字段值为0表示无任何字段需要建立二级索引，但数据仍经CFH中转)，单位为秒。

表2 数据写入性能

Tab.2 Data writing performance

数据量/万条	0 字段/秒	1 字段/秒	4 字段/秒	7 字段/秒
1	1.454	2.252	2.544	2.665
10	8.289	15.733	17.614	19.055
100	81.326	150.399	166.782	191.958
1,000	851.086	1,480.764	1,600.565	1,732.937

从表2可以看出，建立索引以及需建立索引的字段数增加都会增加存储耗时，因为中间件重构请求并维护索引数据结构需要消耗额外的时间。此外，随着数据量的增加，写入时间也以近似线性的趋势增加。表2中的实验均为10,000 条数据的批写入，由于未作其他优化，高频提交小数据会导致HBase频繁创建线程并进行资源回收，浪费服务器资源，严重时会出现宕机，因而采用批写入的方式。当CFH不可用时，客户端直连HBase，此时写入效率为1.3 秒/万条，未建立任何二级索引。

查询时，提取数据所有字段信息，即先通过ES获取主键，继而经主键查询数据信息，不考虑索引覆盖的优化效果。表3为不同条件下经CFH的查询耗时(模糊匹配字段数与HBase数据量的组合)，单位为秒。

表3 数据查询性能

Tab.3 Data query performance

数据量/万条	1 字段/秒	4 字段/秒	7 字段/秒
10	0.457	0.508	0.518
100	0.497	0.528	0.543
1,000	0.527	0.596	0.684

检索时，测试100 条记录取均值。由表3可知，数据量与模糊匹配的字段数增加，都会增加查询的时间，但经CFH与ES的配合，CFH检索仍达毫秒级查询。由此可见，此方案实现了快速查询。

5 结论(Conclusion)

本文设计了分布式存储索引中间件CFH，通过Lucene技术构建二级索引，实现快速检索。经过实验验证，CFH在保证大规模数据并发写入需求的同时，有效地改善了HBase的检索效率，对千万级数据查询可达毫秒级检索。CFH独立于业务系统，基于已有的HBase客户端工具包进行开发，易于整合到已有系统中，支持集群部署。然而，中间件对复杂查询的支持还不够完善，采用原生Lucene工具包构建索引，集群节点同步也可能会发生数据丢失的问题。

参考文献(References)

[1] 周淳,李贤慧.一种实时数据库的分布式存储方法[J].计算机系统应用,2019,28(04):125-130.

[2] 周逸文.分布式存储技术和应用浅析[J].数码世界,2017(12):49.

[3] 田胜利.针对HBase的MapReduce数据访问方式的优化[D].长沙:国防科学技术大学,2012.

[4] YE F, ZHU S J, LOU Y S, et al. Research on index mechanism of HBase based on coprocessor for sensor data[C]// REISMANS. 2019 IEEE 43rd Annual Computer Software and Applications Conference(COMPSAC). Los Alamitos, CA: IEEE Computer Society, 2019: 598-603.

[5] 吴国泉.基于HBase的全文索引及检索技术的研究[D].武汉:华中科技大学,2015.

[6] 石磊,黄高攀,乔雄.基于内存数据库的索引算法研究[J].信息技术,2016,40(11):139-142.

[7] 徐江峰,谭玉龙.基于HBase的多维索引查询机制的优化[J].计算机应用,2020,40(02):571-577.

[8] 党鹏.HBase层次化辅助索引系统的设计与实现[D].西安:西安电子科技大学,2019.

[9] 王惟一.基于存储模型的HBase查询优化技术研究[D].南京:南京大学,2018.

[10] 黄璨,方旭昇,张朝泉.分片计数布隆过滤器及其在HBase二级索引的应用[J].计算机系统应用,2016,25(03):119-123.

[11] 叶奇明,陈俊宇.Hive over HBase框架查询性能研究[J].广东石油化工学院学报,2018,28(06):36-38,42.

[12] 朱松杰,娄渊胜,叶枫,等.基于协处理器的HBase内存索引机制的研究[J].计算机工程与应用,2020,56(01):98-105.

[13] 朱炯名.基于智慧水务的供水大数据采集架构分析研究[J].软件工程,2018,21(09):5-7.

[14] CIAVOTTA M, ALGE M, MENATO S, et al. A microservice-based middleware for the digital factory[J]. Procedia Manufacturing, 2017, 11(05):931-938.

作者简介：

黄　静(1965-)，女，博士，教授.研究领域：通信工程，大数据，深度学习.

别要祥(1996-)，男，硕士生.研究领域：软件工程，大数据.

谢　宣(1995-)，男，硕士生.研究领域：软件工程，图像处理.

www.rjcxz.com