文章编号：2096-1472(2022)-10-50-04
DOI:10.19644/j.cnki.issn2096-1472.2022.010.011

基于Hadoop的学习行为数据云存储平台的设计与实现

蔡春花，黄思远，高继梅

(黄河交通学院，河南 焦作 454000)
✉786803383@qq.com; huangsiyuan924@gmail.com; 2011080200@zjtu.edu.cn

摘　要：文中设计一个用于存储的平台，通过虚拟化技术实现了服务器，规划了平台的存储集群规模及服务；对存储集群架构、请求处理系统、数据库进行了设计；使用Spring Boot和Thymeleaf实现了前台用户功能模块，包括用户注册、用户登录、文件上传及文件相关操作、用户关注和用户分享等功能；最后对HBase库亿级大数据查询性能和Hadoop集群输入/输出(Input/Output，I/O)性能进行了测试。测试结果表明查询性能可以做到百毫秒级查询，写性能平均I/O速率为91.73 Mb/s，读性能平均I/O速率为348.56 Mb/s，能够满足用户需求。

关键词：云存储；学习行为；Hadoop集群；查询性能；I/O性能
中图分类号：TP391　**文献标识码**：A

Design and Implementation of Cloud Storage Platform for Learning Behavior Data based on Hadoop

CAI Chunhua, HUANG Siyuan, GAO Jimei

(*Huanghe Jiaotong University*, *Jiaozuo* 454000, *China*)
✉786803383@qq.com; huangsiyuan924@gmail.com; 2011080200@zjtu.edu.cn

Abstract: This paper proposes to design a storage platform which realizes server through virtualization technology and plans the storage cluster scale and services of the platform. The storage cluster architecture, request processing system, and database are also designed. Spring Boot and Thymeleaf are used to achieve function modules of front-end users, such as user registration and login, file upload and file-related operations, user attention and sharing. Finally, the billion-level big data query performance of HBase library and the Input / Output (I/O) performance of Hadoop cluster are tested. Test results show that the query performance can reach hundred milliseconds-level query, the average I/O rate of write performance is 91.73 Mb/s, and the average I/O rate of read performance is 348.56 Mb/s. The proposed platform can meet users' need.

Keywords: cloud storage; learning behavior; Hadoop cluster; query performance; I/O performance

1 引言(Introduction)

在当下为数不多的云存储相关技术中，Hadoop的HDFS(Hadoop Distributed File System)文件系统是所有互联网公司中使用最广泛的分布式文件系统。HDFS是Google在2003 年提出的GFS(Google File System)的一个开放源码的实现，不过在Google的设计中，为了降低数据存储与数据管理的耦合，GFS仅对数据的存储负责而不提供类似数据库查询的方案[1]。也就是说，GFS只存数据，而对数据的具体内容一无所知，自然也就不能提供基于内容的检索功能。所以，更进一步，Google开发了Bigtable作为数据库，向上层服务提供基于内容的各种功能，而HBase是Bigtable的实现[2]。与原始的存储模式相比，云存储具有访问方便、可靠性高、成本低、可扩展性好等优点。目前，云计算技术在中国的应用已经从几年前仅在国家高端领域推广到民用技术等领域，云存储已成为未来存储的趋势。由于一些相关云存储技术的开源性质，云存储的应用领域变得越来越广泛。云存储所用的存

基金项目：河南省科技攻关项目“MOOC的大数据学习行为与成效分析技术研究”(222102320218)；黄河交通学院校级课程教学资源库建设项目“Hadooop分布式基础架构”(HHJTXY-2021kczyk091)；黄河交通学院校级一流课程建设项目“Hadooop分布式基础架构”(HHJTXY-2022ylkc49).

储结构不同于传统的网络体系，随着科技的发展，它将会取代传统模式的存储数据的方法。

在线学习行为的日志数据如学生登录、浏览资源、点击课件等操作产生的日志数据来源于在线学习平台，平台上的日志数据将会依据不同的数据格式，分别存储于与之对应的数据库中[3]。首先，网站中每天都可能产生较多关于学习者观看视频、学习课程等的日志数据；其次，学习平台的数据需要进行保留(至少保留一年时间)，以供跨年数据的分析及对比；而HDFS在Hadoop技术体系中负责分布式存储数据，一个文件存储在HDFS上时会被分成若干个数据块，每个数据块分别存储在不同的服务器上[4]，因此HDFS解决了存储容量以及数据安全的问题。

2　云存储平台设计(Design of cloud storage platform)

2.1　存储集群规模及服务的规划

云存储业务主要面向海量数据和文件的存储和计算，强调单节点存储容量和成本，因此配置相对廉价的串口硬盘，满足成本和容量的需求[5]。

测试环境为了控制成本，采用虚拟化技术虚拟出三台Linux服务器构成集群，主机名分别设为Hadoop101、Hadoop102、Hadoop103，集群配置Hadoop101 CPU核心数4，磁盘空间50 GB，内存大小8 GB；Hadoop102 CPU核心数2，磁盘空间50 GB，内存大小4 GB；Hadoop103 CPU核心数2，磁盘空间50 GB，内存大小4 GB。

集群服务规划：Hadoop101部署的服务有NameNode、DataNode、QuorumPeerMain、HMaster、HRegionServer、SecondaryNameNode；Hadoop102部署的服务有DataNode、QuorumPeerMain、HRegionServer；Hadoop103部署的服务有DataNode、QuorumPeerMain、HRegionServer、SecondaryNameNode。

2.2　请求处理系统设计

云存储平台是指Spring Boot应用服务器，可以为用户直接提供服务，比如进行文件的上传或下载。基于Hadoop的学习行为数据云存储平台用户功能模块主要包含用户注册登录、文件操作、用户关注和文件分享等。用户功能模块如图1所示。

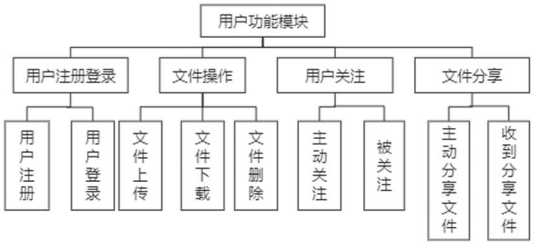

图1 用户功能模块框架图

Fig.1 Framework diagram of the user function module

2.3　HBase数据表设计

传统关系型数据库是可以支持随机访问的，但关系型数据库却不能很好地适用于存储海量的数据，比如关系型数据库MySQL，单表在存放约500万条的数据时，性能会大大降低。在这种情况下，必须有一种新的方案来解决海量数据存储和随机访问的问题，HBase就是其中之一，HBase的单表可存储百亿级的数据且不影响查询效率。

HBase的表结构设计与关系型数据库有很多不同，主要HBase有行键(Row Key)和列族(Column Family)、时间戳(Time Stamp)这几个全新的概念，如何设计表结构就非常重要[6]。HBase就是通过Table、Row Key、Column Family、Time Stamp确定一行数据。这与MySQL、Oracle等关系型数据库管理系统(Relational Database Management System，RDBMS)完全不同。大数据云存储平台涉及的主要表有file表、id_user表、user_file表和share表等。

在设计HBase列族时，使用文件的编号(Identity Document，ID)来作为每一个行键，使用file来当列族，只设计一个列族，file中包含的各种字段都是该列族的列，使用Row Key和Column Family可以确定一个存储单元(Cell)，使用Time Stamp可以确定最新版本，如表1所示。

表1 文件表

Tab.1 File table

行键	列族
rk01	file:originName
	file:name
	file:isFile
	file:isDir
	file:viewflag

3　云存储平台实现(Implementation of cloud storage)

3.1　云存储平台的搭建

(1)Hadoop集群的安装及配置

假设已经配置好服务器集群的静态网址，并且Java开发环境都配置正确，集群之间可以进行免密登录[7]。下面对Apache Hadoop的具体实现步骤做简要阐述。

第一步，在Apache Hadoop的官方文档中下载Apache Hadoop3.1.3的安装包到hadoop101，并解压到/opt/app/hadoop，再将Hadoop添加到环境变量中。

第二步，在HADOOP_HOME下的etc/hadoop中修改hadoop-env.sh配置文件，其中添加JAVA_HOME的路径。

第三步，在HADOOP_HOME下的etc/hadoop中修改core-site.xml配置文件，添加NameNode的地址

hadoop101:8020，并指定Hadoop数据的存储目录。

第四步，在HADOOP_HOME下的etc/hadoop中修改hdfs-site.xml配置文件，指定NameNode的网页端访问地址为hadoop101:9870，指定SecondaryNameNode的WEB端访问地址为hadoop103:9868。

第五步，在HADOOP_HOME下的etc/hadoop/workers中添加节点，共三行，分别是hadoop101、hadoop102、hadoop103。

第六步，分发Hadoop和环境变量配置文件到hadoop102、hadoop103。

至此，Hadoop集群的安装及配置已经完成，可以使用sbin/start-dfs.sh来启动HDFS集群，在浏览器访问hadoop101:9870，可以查看Hadoop集群的节点状态。

(2)Zookeeper集群的安装及配置

Zookeeper的作用在前文已经介绍过，这里是Zookeeper的集群安装和安装过后的配置过程。下面对Apache Zookeeper的具体实现步骤做简要阐述。

第一步，在Apache Zookeeper官方下载Apache Zookeeper3.5.7的安装包，并上传到hadoop101，解压到/opt/app/zookeeper，再将Zookeeper添加到环境变量中。

第二步，将ZOOKEEPER_HOME/conf/下的zoo_sample.cfg文件重命名为zoo.cfg，并在其中添加Zookeeper的数据存储路径，存放在ZOOKEEPER_HOME/zkData，添加hadoop101、hadoop102、hadoop103的Zookeeper监听地址。

第三步，在ZOOKEEPER_HOME下创建zkData文件夹，并在zkData文件夹下创建myid文件，并写入序号1。

第四步，分发Zookeeper到hadoop102、hadoop103，并修改myid文件为不同的ID序号，在hadoop102中写入2，在hadoop103中写入3，保证ID序号不同即可。

至此，Zookeeper集群的安装及配置已经完成，可以使用bin/zkServer.sh start命令进行Zookeeper服务的启动，要注意的是需要在三台服务器上都运行该命令。启动完成后可以使用bin/zkServer.sh status查看Zookeeper是否启动成功。

(3) HBase集群的安装及配置

下面对Apache HBase的具体实现步骤做简要阐述。

第一步，在Apache HBase官方下载Apache HBase2.0.5的安装包，并上传到hadoop101，解压到/opt/app/hbase，再将HBase添加到环境变量中。

第二步，在HBASE_HOME/conf下的hbase-env.sh配置文件中将HBASE_MANAGES_ZK的属性值设置为false，代表使用我们自己配置的Zookeeper集群而不是使用HBase自带的Zookeeper。

第三步，在HBASE_HOME/conf下的hbase-site.xml中添加HBase的数据在HDFS存储的位置，并指定HBase以分布式的方式工作，再指定使用自己的Zookeeper集群。

第四步，在HBASE_HOME/conf下的regionservers中添加三行记录，分别为hadoop101、hadoop102、hadoop103，代表HBase节点。

第五步，分发HBase到hadoop102、hadoop103。

至此，HBase集群的安装及配置已经完成，可以使用bin/start-hbase.sh来启动HBase集群，并且可以使用浏览器访问hadoop101:16010查看HBase集群的状态。

3.2 请求处理系统

用户进行成功注册验证后，使用正确的信息即可登录到系统。在Controller层进行用户认证后如果用户信息没问题，那么即可放行用户进入平台，如果用户信息不正确，会提示用户。

其中文件分享部分关键代码：

```
for (int i=0; i<id.length; i++) {
    File file=fileService.getFileInfoById(Long.parseLong(id[i]));
    Share share=new Share();
    share.setPath(file.getPath());
    share.setOriginalPath(file.getOriginalPath());
    share.setType(file.getType());
    share.setName(file.getOriginalName());
    share.setSharetime(DateUtil.DateToString("yyyy-MM-dd HH:mm:ss", new Date));
    for(int j=0; j<name.length; j++) {
        shareService.addShare(user, share, name[j]);}}
    result.put("errres", true);
    result.put("errmsg", "分享成功！");
```

关注某用户关键代码：

```
@RequestMapping("/followUser")
    public ModelAndView followUser(HttpSession httpSession, HttpServletResponse response,
        @RequestParam(value="name") String followName) {
        User user=(User) httpSession.getAttribute(Constants.currentUserSessionKey);
    JSONObject result=new JSONObject();
```

```
        try {
            followService.addFollowUser(user, followName);
            result.put("errres", true);
            result.put("errmsg", "关注成功！");
        } catch (Exception e) {
            result.put("errres", false);
            result.put("errmsg", "关注失败！");
            e.printStackTrace();
        }
        ResponseUtil.write(response, result);
        return null;
    }
```

4 性能测试(Performance testing)

4.1 HBase库亿级大数据查询性能测试

在云存储平台HBase的所有表中，只有与文件file相关的表的数据会比较大，因此只需对file表进行性能测试[8]。下面对查询性能测试步骤做简要概述。

第一步，模拟生成file表的1 个列族，10 列字段，目标数据量1亿多条。

第二步，使用Java客户端连续根据Row Key查5 条记录，将查询耗时记录下来，得到表2。

表2 HBase查询性能测试

Tab.2 Tests of HBase query performance

次数	耗时/ms
1	8,825
2	2,324
3	921
4	452
5	321

通过表2可以看出，使用Row Key进行查询时，初次查询较慢，需要 8 秒左右的时间，但之后查询很快，基本可以做到百毫秒级查询。当然，测试环境配置不高，如果在5 个节点，16核CPU，128 GB内存的配置下可以实现几十毫秒甚至几毫秒级查询。这得益于HBase适合存储PB级别的海量数据(百亿千亿量级条记录)，能在几十到百毫秒内返回数据，并且不会因为数据量的剧增而导致查询性能下降[9]。

4.2 Hadoop集群性能测试

(1)写性能I/O测试

使用Hadoop自带的基准测试工具进行HDFS写性能I/O测试，运行如下命令hadoop jar/opt/app/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-client-jobclient-3.1.3-tests.jar TestDFSIO-write-nrFiles 10-fileSize 128 MB，其中向HDFS写入10 个128 MB的文件，测试结果如图2所示。

```
2022-05-05 08:54:13,468 INFO fs.TestDFSIO: ----- TestDFSIO ----- : write
2022-05-05 08:54:13,468 INFO fs.TestDFSIO:            Date & time: Thu May 05 08:54:13 UTC 2022
2022-05-05 08:54:13,468 INFO fs.TestDFSIO:        Number of files: 10
2022-05-05 08:54:13,468 INFO fs.TestDFSIO: Total MBytes processed: 1280
2022-05-05 08:54:13,469 INFO fs.TestDFSIO:      Throughput mb/sec: 73.64
2022-05-05 08:54:13,469 INFO fs.TestDFSIO: Average IO rate mb/sec: 91.73
2022-05-05 08:54:13,469 INFO fs.TestDFSIO:  IO rate std deviation: 47.73
2022-05-05 08:54:13,469 INFO fs.TestDFSIO:     Test exec time sec: 20.63
```

图2 HDFS写性能测试图

Fig.2 Test plot of HDFS write performance

如图2所示，向HDFS写入了10 个文件，一共处理1,280 MB的数据，每秒的吞吐量为73.64 MB，平均I/O速率为91.73 Mb/s，执行时间为20.63 s。

(2)读性能I/O测试

接下来进行HDFS读性能测试，使用Hadoop自带的基准测试工具，运行如下命令hadoop jar/opt/app/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-client-jobclient-3.1.3-tests.jar TestDFSIO-read-nrFiles 10-fileSize128 MB，其中读取HDFS的10 个128 MB的文件，测试结果如图3所示。

```
2022-05-05 09:03:04,615 INFO fs.TestDFSIO: ----- TestDFSIO ----- : read
2022-05-05 09:03:04,615 INFO fs.TestDFSIO:            Date & time: Thu May 05 09:03:04 UTC 2022
2022-05-05 09:03:04,615 INFO fs.TestDFSIO:        Number of files: 10
2022-05-05 09:03:04,615 INFO fs.TestDFSIO: Total MBytes processed: 1280
2022-05-05 09:03:04,616 INFO fs.TestDFSIO:      Throughput mb/sec: 147.13
2022-05-05 09:03:04,616 INFO fs.TestDFSIO: Average IO rate mb/sec: 348.56
2022-05-05 09:03:04,616 INFO fs.TestDFSIO:  IO rate std deviation: 312.91
2022-05-05 09:03:04,616 INFO fs.TestDFSIO:     Test exec time sec: 19.48
```

图3 HDFS读性能测试图

Fig.3 Test plot of HDFS read performanc

如图3所示，向HDFS读取了10 个文件，一共处理1,280 MB的数据，每秒的吞吐量为147.13 MB，平均I/O速率为348.56 Mb/s，执行时间为19.48 s。

5 结论(Conclusion)

本论文重点研究以Hadoop和HBase集群实现的大数据云存储平台，成功实现了一个自动化、高效率的大数据云存储平台。各部分的功能都能实现后，搭建合适的环境进行功能的测试，通过测试结果可见各模块的功能需求可以满足使用条件，可用性很高。

在平台高峰时请求量非常大，如果不进行“流量削峰”，服务器可能因无法承受而宕机，因此接下来可使用Kafka来进行“流量削峰”。在存储成本方面，尽管存储设备都是低廉设备，但是由于备份机制，存储成本也需要进一步控制。因此在接下来的平台版本升级中，会使用压缩技术对文件进行压缩，并且对不同用户的相同文件进行“文件指纹技术”，对于重复文件只保留一份。

(下转第49页)