Universe Scientific Publishing
Computer System Networking and Telecommunications
计算机系统网络和电信，2019,1(2)
ISSN:2661-3719 (Print); 2661-3727 (Online)

Based on Hadoop Platform Analysis User Music Playback Log Implementation

Siping WANG Zhengde BAO Chenxi LI
School of Computer and Software, Jincheng College, Sichuan University, Chengdu, 611731

Abstract
With the rapid development of big data technology, the application of big data technology to deal with relevant data in enterprises and society has become more and more common. This paper starts from the social development to analyze the demand of Internet music company for data processing, and develops a music log statistics system based on hadoop platform. The system is mainly composed of the log acquisition component and the calculation component. The log data generated in the process of playing music by the user is counted on the basis of year and month. Finally, the results of the calculated components are put into the Database. The system is developed to help enterprises quickly analyze massive logs and improve the low efficiency of traditional database analysis.

Key Words
Big Data Technology; Music; Log Analysis; Hadoop Platform
DOI:10.18686/jsjxt.v1i2.692

基于 Hadoop 平台分析用户音乐播放日志的实现

王思平 鲍正德 李晨曦
四川大学锦城学院计算机与软件学院，四川成都，611731

摘 要
伴随大数据技术的快速发展，使用大数据技术处理相关数据在企业和社会中的应用愈加普遍。本文从社会发展开始分析互联网音乐公司对数据处理的需求，并且开发了一个基于 hadoop 平台的音乐日志统计系统。该系统主要由日志采集组件和计算组件组成，对用户播放音乐过程中产生的日志数据进行了关于年月为单位的统计。最后把计算组件的结果放入了 Database 中。该系统开发是为能助力企业快速分析海量日志，改善传统数据库分析方式效率低下的问题。

关键字
大数据技术；音乐日志分析；Hadoop 平台

1.引言

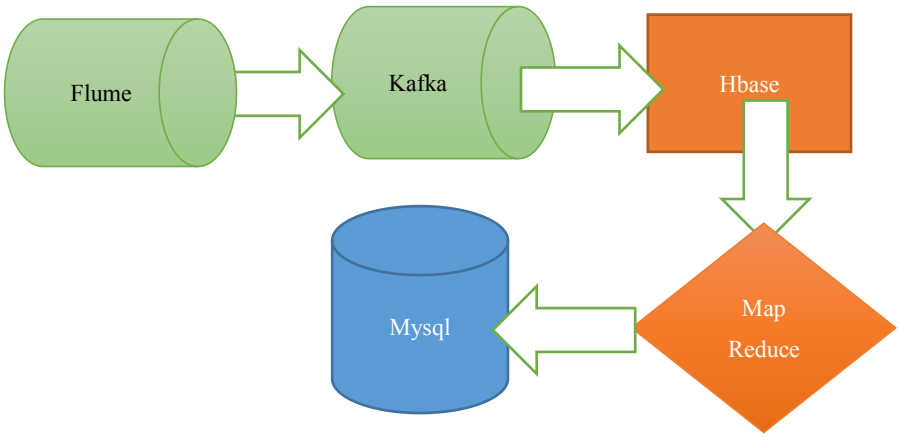
互联网特别是移动互联网的发展，加快了信息化向社会经济各方面、大众日常生活的渗透[1]。人们播放歌曲的方式也在不断地更新,从最初的 DVD+音响到 MP3+耳机，到现在的智能手机。人们能够随心所欲播放自己想听的歌曲，不用像以前要在电脑上下载或者购买 CD 等。这些播放方式的改变也推动了在线音乐的蓬勃发展。人们听歌的次数增加，收听音乐的时间也在增长。用户使用 Music software 产生的日志数量愈加增多。快速处理并从庞大的数据量中分析出有效的数据结果，能够帮助企业更好的发展。音乐公司可以根据使用次数等信息分析出用户是否是“忠实粉丝”，也可以分析用户的浏览信息等，为用户挖掘出更加与其爱好符合的音乐。本文利用大数据系统分析用户音乐日志，统计了一些用户使用信息。

2.用户音乐播放日志系统目标

在大数据分析领域最基础的应用是统计相关数据，这些数据不仅体量巨大而且大部分为非结构化数据，直接利用传统的 db 数据库存储很难做到，所以要先用 Hadoop 分析平台对这些数据实行统计，最后将系统计算结果写入进传统 Database，减轻数据库的压力。本系

统主要实现目标包括（1）统计每位使用者在 Music software 中的年度音乐收听次数和年度音乐收听时长；（2）统计每位使用者在 Music software 中的月度收听次数和月度收听时长；（3）统计不同歌曲在 Music software 中的年度收听频率和年度收听时长。这三个目标是对用户数据的基本分析。有利于音乐公司快速了解每个用户的使用频率和每首歌曲的详细信息。

3.用户音乐播放日志系统结构



4.系统功能实现

4.1 搭建 hadoop 高可用集群和安装 mysql 数据库

音乐日志统计系统使用环境：Centos6.8 系统。在本系统中建立了多台 Hadoop 环境节点，分别命名为 linux01，linux02，linux03 并且这三台 linux 节点可以相互通信。在集群中一共创建两个 NameNode，保证当其中一个 NameNode 出现错误时，另一个可以接着工作。在分布式集群平台中，每一台服务器可视为分布式环境中的一个节点[2]。运行节点的数量越多，分布式集群平台的运行效率越高[2]。

4.2 搭建日志采集组件

本系统中的日志采集组件是通过 Flume+kafka+Hbase 组成。Flume 是一个非常优秀的日志聚合系统，适合采集分布式系统中零散的日志，且具有高可用、高可靠和可扩展等特性，同时 flume 接收的数据可以是来自于多样化的、自定义的数据发送方[3]。Apache Kafka 是由 LinkedIn 公司开发一种分布式的发布-订阅消息系统，是一种快速的、可扩展的、分布式的、分区的和可复制的提交日志服务机制[4]。Kafka 核心组件有 Producer、Consumer 、Topic（可扩展性高） 。kafka 在企业开发中常常被用于消息中间件用途广泛，起一个缓冲数据和应对突发高流量的情况。日志采集组件使用 Flume+kafka 来缓冲和拉取数据，并且将数据写入到 HBase 中。HBase 使用 HDFS 作为底层文件存储系统[5]，本组件中用于存储音乐日志中的结构化信息。Hbase 能够在低延时情况下稳定地存储大量结构化数据，适合于本组件的搭建需求。将 Flume,kafka,Hbase 的安装包分别上传到三台 linux 服务器上并解压。

（1）创建 music.conf 用于运行 Flume 时指定其 sources，sinks 等运行信息。配置文件部分如下

```
# define
a1.sources = r1
a1.sinks = k1
a1.channels = c1
# source
a1.sources.r1.type = exec
a1.sources.r1.command = tail -F -c +0 /home/wangzi/music/log.csv
a1.sources.r1.shell = /bin/bash -c
# sink
a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink
a1.sinks.k1.brokerList = linux01:9092,linux02:9092,linux03:9092
a1.sinks.k1.topic =music
a1.sinks.k1.batchSize = 20
```

a1.sinks.k1.requiredAcks = 1

（2）更改 kafka 的 conf 文件。更改三台服务器上的 broker.id，保证彼此不重复，并且修改日志存储的地方。

（3）原始的日志文件 173azc31,2003-06-09,15,2018-10-03 09:52:06,0355（举例）分别是用户 id，用户生日，对应的音乐 id，歌曲时间，歌曲时长。（本数据是由代码生成的测试数据，非企业实际生产数据）在该组件中需要通过 Java api 从 kafka 中获取数据并且在 Hbase 中创建表，且写入到表中。在 Hbase 表设计中采用了预分区键对 Hbase 存储进行优化，对于 Hbase 而言，若使用者不预先对表进行分区操作，则数据将会全部被放入到创建表时的 region 中，当这个预先创建的 region 记录的列过多时，Hbase 将采取主动将 region 进行 split 的措施从而避免分区表过大，然而这个过程将消耗大量节点资源。所以需要在 Hbase 中建立预先分区的存储表，提高 Hbase 的读写能力。在 music:some(创建的表名称)中一共预设立了 10 个分区，分区号规则是按用户出生的年数值取余运算获得分区号。该分区规则有利于将用户数据均匀地存储在创建的 10 个 region 中。

（4）在该表的 RowKey 设计中是通过拼接分区号，用户 id，用户年龄，音乐 id，播放时间，播放时长组成的。部分 HbaseInfo 类（预分区键，rowkey）代码如下。

```
//获取预分区号
public static String getSplitNum(int region, String age) {
        String yearAge = age.substring(0, 4);
        Integer yearAge_int = Integer.valueOf(yearAge);
        int num =(yearAge_int % (region+1))-1;
        return String.valueOf(num);
    }
// 拼接 rowkey 9_173azc33_1999-01-15_20_2018-01-21 14:18:16_0211
public static String getRowKey(String getSplitNum, String uId, String uAge, String mId, String playTime, String playDuration) {
    return getSplitNum + "_" + uId + "_" + uAge + "_" + mId + "_" + playTime + "_" + playDuration;
    }
```

（5）在日志组件中通过 kafka_cosumer.class 拉取 kafka 生产者中的数据并且写入到 Hbase 表中。kafka_cosumer 类部分代码如下

```
while (true) { //不断拉取数据
ConsumerRecords<String, String> pollInfo = kc.poll(10);
for (ConsumerRecord<String, String> record : pollInfo) {
hbaseStart.put_data(record.value());}
```

4.3 搭建目标 1 和目标 2 的 MapReduce 计算组件并写入到 mysql 表中

(1) 在 msyq 表中创建 user 和 user_music 表分别用于存储 mapreduce 计算出的年度信息和月度信息。

(2)在本组件中一共创建了两个类 userMusicInfo 和 countValue 来封装传入和传出的字段。部分代码如下

```
public class userMusicInfo implements WritableComparable<userMusicInfo> {
    private String userId;
    private String uAge;
    private String uName;
    private String playTime_year;
    private String playTime_month;
public class countValue implements WritableComparable {
    private int count;
    private int playDuration;
```

(3)编写 MapTask 工作阶段。创建 user_Mapper 类继承 TableMapper 类并且重写 map 方法，key 中获取从 hbase 读的每列数据，并且进行切割，然后将这些值封装在 userMusicInfo 对象中，将该对象作为 map 的 KEYOUT，reduce 的 KEYIN。将播放时长作为值封装在 Text 类中，作为 map 的 VALUEOUT。

map 方法中部分源码如下

```
Text val = new Text();val.set(playDuration);//播放时长
userMusicInfo Info1 = new userMusicInfo(); //按年算
Info1.setUserId(userId);Info1.setuAge(uAge);Info1.setuName(uName);
Info1.setPlayTime_year(playTime_year);Info1.setPlayTime_month("year");
context.write(Info1,val);
```

```
userMusicInfo info2 = new userMusicInfo(); //按月算
info2.setUserId(userId);info2.setuAge(uAge);info2.setuName(uName);
info2.setPlayTime_year(playTime_year);info2.setPlayTime_month(playTime_month);
context.write(info2, val);
```

(4) ReduceTask 任务准备阶段。重写 reduce 方法，从 VALUEIN(Text 类)中获取播放时长，循环累加，并且定义一个计数变量用于累加用户播放音乐次数。

reducer 阶段部分源代码如下

```
countValue cValue = new countValue();
int count = 0;    int pD= 0;
   for (Text text : values) {
    count++;                    pD=          pD+
Integer.valueOf(text.toString()); }
 cValue.setCount(count);cValue.setPlayDuration(pD);
   context.write(key, cValue);
```

(5)自定义 OutputFormat 类并且判断年度信息和月度信息。在该 mapreduce 中需要输出到多张 mysql 表中，所以采用自定义 OutputFormat 类的模式重写数据输出的格式和目的地。创建 mysqlRW 类继承 RecordWriter 类并且重写 write 方法，这是将计算结果写入 mysql 中的核心方法。通过比较 playTime_month 字段来判断是年度信息还是月度信息，如果 playTime_month 字段值是”year”，则执行插入年度信息的 sql 语句，否则执行插入月度信息的 sql 语句。

write 方法部分源代码如下

```
 //说明只带年份不带月份
 if (playTime_month.equals("-1")) {
 primary_key = userId + age + name + playTime_year;
 try {
     ps = connection.prepareStatement("insert into user values(?,?,?,?,?,?,?)");
     ps.setString(1,primary_key);ps.setString(2,userId);ps.setString(3,age);
     ps.setString(4,name);ps.setString(5,playTime_year);ps.setInt(6,count);
     ps.setInt(7, playDuration);ps.executeUpdate();ps.close();
  } catch (SQLException e) {    e.printStackTrace();}
} else {
primary_key = userId + playTime_year + playTime_month;//说明既带年份，也带月份
try {
ps=connection.prepareStatement("insert into user_music values(?,?,?,?,?,?)");
ps.setString(1,primary_key);ps.setString(2,userId);ps.setString(3,playTime_year);
ps.setString(4,playTime_month);ps.setInt(5,count);ps.setInt(6,playDuration);
ps.executeUpdate();ps.close();}
catch(SQLException e) {    e.printStackTrace(); } }
```

4.4 实现目标 3 的 MapReduce 计算组件并写入到 mysql 表中

(1)创建 mysql 表。创建 musicInfo 表，创建 mId,mName,count,playDuration 四个字段。

(2)创建 musicBean 类。因为本组件只需输出到一张 sql 表中，所以采用 hadoop 自带的 DBWritable 组件用于封装 bean 数据。musicBean 类继承了 WritableComparable 和 DBWritable 类，封装了 mId,mName,count,playDuration 四个字段，分别代表音乐 id，音乐名称，播放次数，播放时长。

部分 musicBean 类源代码如下

```
public class musicBean implements WritableComparable<musicBean>, DBWritable {
    private String mId;
    private String mName;
    private int count;
    private int playDuration;
```

(3)MapTask 任务准备阶段。创建 musicMapper 类继承 TableMapper 类便于从 hbase 中获取数据，将 map 方法中的 key 切割，获得音乐 id 和其他属性。在类中定义名为 musicInfo 的 HashMap 集合，存储音乐 id 和音乐名称，并且通过音乐 id 映射音乐名称。

部分 musicInfo 类源代码如下

```
Text val = new Text();//拼接键
musicBean musicBean = new musicBean();musicBean.setmId(mId);
musicBean.setmName(mName);musicBean.setCount(0);musicBean.setPlayDuration(0);
val.set(playDuration);//拼接 value
```

ISSN:2661-3719 (Print); 2661-3727 (Online)

context.write(musicBean,val);

(4)ReduceTask 任务准备阶段。在本组件中将全部字段信息封装在 musicBean，所以 VALUEOUT 不需要信息，VALUEOUT 用 NullWritable 类代替

部分 reducer 阶段源代码如下

```
int count = 0; int pD= 0;
for (Text value : values) {
        ++count;
pD= pD+ Integer.valueOf(value.toString()); };
key.setCount(count);key.setPlayDuration(pD);context.write(key, NullWritable.get());
```

(5)组合 map 和 reduce 的任务驱动类。新建 musicDriver 并且实现 Tool 接口，重写 run 方法，便于后续 ToolRunner 调用 musicDriver 中的 run 方法。Hadoop 框架通过调用该类找到各阶段功能实现所需的正确的类。

部分 musicDriver 类源代码如下

```
DBConfiguration.configureDB(conf,"com.mysql.jdbc.Driver", "jdbc:mysql://192.168.75.10:3306/music", "music", "music123456");
Job job = Job.getInstance(conf);
job.setJarByClass(musicDriver.class);
Scan scan = new Scan();scan.addFamily(Bytes.toBytes("f1"));
//设置 Mapper 相应属性
TableMapReduceUtil.initTableMapperJob("music:some",scan,musicMapper.class,musicBean.class,Text.class,job);
//设置 reduce 的类
job.setReducerClass(musicReducer.class);
job.setOutputFormatClass(DBOutputFormat.class);// 设置输出格式
job.setOutputKeyClass(musicBean.class);
job.setOutputValueClass(NullWritable.class);
DBOutputFormat.setOutput(job,"musicInfo","mId","mName","count","playDuration";
```

5.测试验证

经过建立分析平台，调用日志采集组件，以及运行计算组件代码，查看数据库对应的表数据生成。

user 表

primary_key	userId	age	name	playTime_year	count	playDuration
173azc011999-01-01tom2018	173azc01	1999-01-01	tom	2018	61	17145
173azc011999-01-01tom2019	173azc01	1999-01-01	tom	2019	59	16210
173azc022002-02-21lili2018	173azc02	2002-02-21	lili	2018	52	13453

user_music 表

primary_key	userId	playTime_year	playTime_month	count	playDuration
173azc01201801	173azc01	2018	01	2	559
173azc01201802	173azc01	2018	02	4	1077
173azc01201803	173azc01	2018	03	5	1361

musicinfo 表

mId	mName	count	playDuration
11	卡拉永远 OK.mp3	181	47733

6.总结与展望

本文联系音乐公司在数据分析领域的具体问题。分析了传统数据统计方式的弊端和音乐日志分析平台的架构，并且实现了具有多个部件音乐日志统计系统。扩展了音乐企业对用户数据分析的方式。通过测试验证，该系统能够对大量用户音乐日志做有效的数据统计，提高数据分析的效率。由于本系统只是将统计结果写进mysql 数据库中，未展示数据图表，后续进一步研究工作将围绕数据可视化展开。

参考文献

[1] 邬贺铨.大数据时代的机遇与挑战[J].求是,2013(04):47-49

[2] 郑文青.基于 hadoop 的大数据分布式集群平台搭建的研究[J].计算机产品与流通,2017(12):143.

[3]蒲志明. 云平台中日志管理模块的研究与实现[D].电子科技大学,2017.

[4]张海华,杨秀波,张非,钟磊.基于 Kafka 的在线教学平台事件中心设计与实现[J]. 数字通信世界,2019(02):117-118+129.

[5]康毅. HBase 大对象存储方案的设计与实现[D].南京大学,2013.

作者简介

第一作者：王思平（1999-），男，汉，四川省内江市，本科，四川大学锦城学院，研究方向：J2EE 和大数据。

第二作者（通讯作者）：鲍正德（1989-），男，汉，黑龙江哈尔滨，研究生，四川大学锦城学院，研究方向：电子商务。

第三作者：李晨曦（1998-7），男，汉，贵州贵阳，本科，四川大学锦城学院，研究方向：大数据技术开发