基于Hadoop的电视编目系统的优化算法和性能测试

文/国家新闻出版广电总局监测数据处理中心 吴波 作者二 作者三

**摘要：**电视编目系统基于自动化的视频处理，为电视节目内容监管提供了技术支撑。针对内容监管范围变化大、时效性要求高的需求，本文提出了基于Hadoop的电视编目系统的优化算法，测试结果表明，该算法可以更充分的利用空闲的硬件资源，并在实现视频快速处理的同时，避免I/O瓶颈的出现。

**关键词：**Hadoop 大数据 内容监管 MapReduce 全局二值模式

**1 引言**

电视播出节目内容是政府监管的重点。基于广电统一监管平台建设的电视自动化编目系统（以下简称编目系统），利用视频样例库的视频指纹特征，自动识别电视节目，在电视内容监管中得到了广泛的应用，大大提高了电视内容监管的工作效率[[1](#_ENREF_1)]。然而内容监管多有临时、专项的任务需求，监管范围和规模变化大，时效性要求高，对编目系统资源调配及扩展能力提出了更高的要求。基于大数据技术的Hadooop计算框架，具备吞吐量大、高可靠、易于扩展、高容错等优点[[2](#_ENREF_2)]，可用于优化编目系统。目前，Hadoop已在文本数据分析[[3](#_ENREF_3); [4](#_ENREF_4)]和视频转码[[5-7](#_ENREF_5)]等领域得到了应用，然而编目系统有着特定的音视频处理过程，无法直接利用现有算法。文献[[8](#_ENREF_8)]提出了基于Hadoop的编目系统优化方案，但是需要在视频预处理阶段拼接关键帧文件，在实际应用时会成为系统性能的瓶颈。

本文先对现有编目系统的工作流程进行分析，然后基于Hadoop架构提出了优化的算法，并进行了性能对比测试，最后给出结论。该算法将原始录像文件切割成多个文件，再基于Hadoop完成转码和特征提取，以提高系统的效率。

**2 现有编目系统的分析**

**2.1 现有编目系统工作流程**

（1）监测前端将采集到的视频，以2小时一个文件，写入到共享存储中。

（2）服务器读取录像文件，将录像文件转码成标准格式，将转码后的文件写回到共享存储中。

（3）服务器读取转码后的文件，提取其视频指纹特征，将特征文件写回到共享存储中。

（4）服务器读取特征文件，与视频样例库进行匹配，识别出相应的节目。

（5）未识别的节目由人工进行识别，从而生成全天的节目单，同时人工将重点关注的节目视频段存入节目样例库。

在实际使用时，上述过程中的转码、特征提取、样例库匹配分别约占用视频处理总时长的45%、45%、10%。因此，本文主要优化编目系统的转码和特征提取算法。

**2.2 现有系统存在的问题**

（1）随着计算规模的扩大，大量的服务器频繁从存储读取文件，会产生I/O瓶颈。

（2）服务器之间按照频道固定划分任务，配置繁琐，在面临时效性要求高的临时、专项监管需求时，难以充分调度计算资源，扩展性差。

**2.3 Hadoop的存储和计算框架**

Hadoop作为分布式计算平台，其存储和计算的结构如所示[[9](#_ENREF_9)]。

（1）Hadoop分布式文件系统（HDFS）将文件切成若干数据块，以多副本的方式保存在不同的服务器上，可避免数据失效。

（2）MapReduce由HDFS的文件生成<key, value>键值对，拆分到不同的服务器并行完成Map Task，并在执行Reduce Task的服务器完成结果的合并。MapReduce可自动让各个服务器在计算时尽可能只读取本地的数据块，避免过多占用网络，提高了I/O效率。



图1 Hadoop的存储和计算框架

为了使得编目系统的算法能够适配MapReduce的计算框架，要求其具备可拆分和合并的特性。系统现有的转码算法基于FFMPEG开源库实现，FFMPEG提供了视频的切割和合并的功能[[10](#_ENREF_10)]；系统现有的特征提取算法提取的是视频各帧图像的全局二值模式（Global Binary Pattern，GBP）的指纹特征[[11](#_ENREF_11)]，各帧GBP特征的提取都是独立进行的，因此转码和特征提取算法都可以依据MapReduce的框架进行优化。

**3 算法实现**

**3.1系统结构**

算法基于Hadoop的MapReduce2.0框架实现，可克服传统MapReduce扩展性差、可靠性差、资源利用率低等缺点[[9](#_ENREF_9)]，框架如所示，包括Client、Master、Slave三类节点。

（1）Client

Client将任务提交到集群，将录像文件上传到HDFS。

（2）Master

Master负责集群内资源的调度和管理。收到Client提交的任务时，依据资源调度策略在选择一个Slave节点创建ApplicationManager，并向ApplicationManager分配资源，监控其运行并在其失败时进行重启。

（3）Slave

运行ApplicationManager的Slave根据从Master获取的资源分配信息将资源分配给相应的Map Task和Reduce Task，监控其运行并在其失败时进行重启。

**3.2系统逻辑关系**

Hadoop提供了相应的编程模型以实现分布式存储和计算，模型中对象的逻辑关系如所示，处理的流程如下。

（1）任务提交

Client将任务提交到Hadoop集群。

（2）任务拆分

InputFormat对象读取待处理的文件，并将文件分成一个或多个InputSplit，InputSplit记录了待处理文件的路径、文件中的开始位置、处理文件的长度；RecordReader根据InputSplit生成<key, value>，交给Mapper对象进行处理。

（3）任务处理

Mapper对象各自执行Map Task，将结果<key, value>交给Reducer对象；Reducer对象自动按照key排序输入的<key, value>，完成Reduce Task，将结果保存到HDFS中。



图2 系统结构



图3 系统逻辑关系

**3.3 实现方法**

在实现中，仅需依据对相关对象进行扩充和定义即可。

**3.3.1 任务提交**

Hadoop默认文件分块大小为64MB或者128MB，会自动切割超过分块大小的文件。如果将录像文件上传到HDFS上，按照HDFS默认的切割会导致文件内部的帧结构被破坏，使得节点无法处理被切割的文件。为此，可以利用FFMPEG预先将录像文件切割成低于分块大小的文件，保证切割后的文件仍是完整的视频文件，防止文件在HDFS中被再切割。

切割文件和上传文件可以由切割线程和上传线程并行完成，提高系统的效率，如所示，其处理流程如下。

（1）有新录像文件到达时，将录像文件分配给其中一个切割线程。

（2）切割线程拆分录像文件，将拆分后的文件路径信息顺序写入到队列中；同时，上传线程从队列中依据路径信息读取已经完成处理的文件，上传到HDFS。

（3）上传线程读到文件结束信息时，向Hadoop提交任务信息。

切割线程和上传线程之间可以利用“生产者-消费者”模型实现对队列访问的互斥[[12](#_ENREF_12)]。

图4 任务提交的设计

**3.3.2 任务拆分**

（1）配置InputFormat的isSplitable属性为false，保证每一个输入文件只有一个InputSplit，这样Mapper对象就能处理完整的一段文件。

（2）配置<key, value>为<文件开始时间，文件在HDFS的路径>，从而Mapper对象能找到待处理的文件。

**3.3.3 任务执行**

Mapper对象调用FFMPEG完成转码和视频特征的提取，Reducer对象合成各Mapper输出的视频特征。任务执行的过程所示，在Mapper上，由于FFMPEG不能直接处理HDFS中的文件，因此需要将HDFS的文件下载到Mapper所在服务器的本地文件系统上，再转码和提取特征，输出<key,value>为<文件开始时间，GBP特征>。在Reducer上，由于视频特征结果按照文件开始时间进行了自动排序，因此仅需要顺序将Mapper传递的GBP特征进行合并，就可以得到原始视频的特征。



图5 任务执行的流程

**4 性能测试**

我们利用9台服务器对算法进行了验证和比较，配置如表1所示。

表1 服务器配置

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| IP地址 | 节点类型 | 运行环境 | CPU | 内存 | 硬盘 | 网络连接 |
| 10.242.2.49 | Master、Slave、Client | Suse Linux Enterprise Server 11 SP2,JDK 7u80,Hadoop 2.6.0 | 2颗Intel E5620 4核处理器 | 4条4GB DDR 1333MHz内存 | 3个3.5寸300GB 15000rpm SAS硬盘，配置为RAID5 | 千兆交换机 |
| 10.242.2.50 | Slave |
| 10.242.2.51 | Slave |
| 10.242.2.52 | Slave |
| 10.242.2.53 | Slave |
| 10.242.2.54 | Slave |
| 10.242.2.55 | Slave |
| 10.242.2.56 | Slave |
| 10.242.2.57 | 共享存储 |

8台服务器用于完成文件的转码和特征提取，1台服务器用于存储待处理的文件。文件为从北京歌华有线数字前端采集到的BTV-新闻频道1天共12个录像文件，每个文件时长2小时，MPEG-TS格式，码率约800kbps。任务提交时按照20分钟一段进行切分，视频大小为800×20×60/8/1024=117MB，可以保证切片文件大小小于Hadoop2.6.0默认的分块大小128MB。HDFS文件副本数配置为3。

我们比较了原有算法和本文所提出的算法转码和提取视频特征所需的总时长。后者分别利用了Hadoop的FIFO、Fair、Capacity调度器[[9](#_ENREF_9)]对任务进行了调度。

图6给出了只处理1个文件所需时长。原有算法只能在1个服务器进行处理，无法利用空闲资源提高处理速度。本文算法则可以将视频文件拆成6段，分配到不同的服务器和不同的线程进行处理，比原有算法提高了约有3倍的速度。



图6 处理1个录像文件所需时长的比较

给出了处理1天12个文件所需时长。原有算法在处理1天文件时，可以将12个文件尽可能平均分配到不同的服务器（如表2所示），每个服务器针对每个文件开启独立的处理线程，以提高处理速度。

表2 原有算法处理1天文件的分配方法

|  |  |
| --- | --- |
| **服务器数** | **文件分配** |
| 1 | 单服务器处理12个文件 |
| 2 | 每个服务器处理6个文件 |
| 3 | 每个服务器处理4个文件 |
| 4 | 每个服务器处理3个文件 |
| 5 | 2个服务器各处理3个文件，3个服务器各处理2个文件 |
| 6 | 每个服务器处理2个文件 |
| 7 | 5个服务器各处理2个文件，2个服务器各处理1个文件 |
| 8 | 4个服务器各处理2个文件，4个服务器各处理1个文件 |

服务器数小于3时，本文算法的处理速度比原有算法慢。这是由于在服务器数较少的情况下，资源几乎都已经被用满，而本文算法额外的视频切割和多副本管理策略会影响处理速度。随着服务器数的增加，本文算法可以更加充分的利用系统的资源，处理速度比原有算法提升的更快，当服务器数达到8时，处理时间仅为原有算法时间的2/3。另外，原有算法在8个服务器下的处理时间比7个服务器下处理时间要长，这是由于原有算法在服务器数增多后大量访问共享存储，会导致性能的下降；而本文算法则可以在计算时尽可能只读取存储在本地的数据，减少了网络访问，从而可以随着服务器数的增加提高系统的处理速度。



图7 处理1天录像文件所需时长的比较

**4 结论**

电视节目内容监管多有时效性要求高的临时、专项的监管任务的要求，现有编目系统的资源调配和扩展能力仍有限。为此，本文利用Hadoop优化了编目系统的转码和特征提取算法，测试结果表明，本文算法在有空闲资源的情况下处理速度比现有的算法快，且可避免大量服务器访问存储导致的I/O瓶颈的出现，因此适用于时效性要求高的内容监管需求。

**参考文献：**

[1] 王婧.基于电视自动编目技术的广告监管自动化系统[J].广播与电视技术,2014 (1): 68-72.

[2] White T. Hadoop: The definitive guide[M].O'Reilly Media, Inc., 2012.

[3] 胡光民,周亮,柯立新.基于Hadoop的网络日志分析系统研究[J].电脑知识与技术,2010 (22): 6163-6164+6185.

[4] 王德文, 孙志伟.电力用户侧大数据分析与并行负荷预测[J].中国电机工程学报,2015(3): 527-537.

[5] 杨帆,沈奇威.分布式系统Hadoop平台的视频转码[J].计算机系统应用, 2011 (11): 80-85.

[6] 张翰,石小明.一种基于云技术的视频转码平台设计[J].有线电视技术, 2015 (1): 35-37.

[7] 张浩,孙淑霞.Hadoop云计算平台在视频转码上的应用[J].电脑与电信,2011 (12): 36-37+40.

[8] 吴波.基于Hadoop的广播电视节目编目系统的优化[C].第20届国际广播电视技术讨论会论文集, 2015: 44-49.

[9] Kulkarni A P,Khandewal M.Survey on Hadoop and Introduction to YARN[J].International Journal of Emerging Technology and Advanced Engineering, 4(05): 82-87.

[10] Tomar S. Converting video formats with FFmpeg[J]. Linux Journal, 2006(146): 10.

[11] 徐科,宋畅.基于全局二值模式的特征提取方法及其应用[J].模式识别与人工智能,2013 (09): 872-877.

[12] 崔立剑,吴平. Java多线程设计模式研究[J].计算机与现代化, 2006 (11):92-94+98.