基于 Hadoop 云计算平台的分布式转码方案①

孙建伟[2], 付 雷[1,2], 于 波[2]

[1](中国科学院大学, 北京 100049)
[2](中国科学院 沈阳计算技术研究所, 沈阳 110168)

摘 要: 在新媒体视频业务快速发展的今天, 传统单机视频转码能力已经出现瓶颈. 在 Hadoop 云计算平台的研究基础上, 结合当前主流的音视频处理工具 FFmpeg, 提出了一种新的视频转码方案. 该方案通过使用 Hadoop 两大核心: HDFS(Hadoop Distributed File System)和 MapReduce 编程思想, 进行分布式转码. 同时, 还详细地介绍和设计了分布式转码的具体流程. 最后实验结果表明, 该分布式转码方案在效率上有较大提高. 在实验中, 视频的分段大小也影响着视频转码的时间. 随着分段大小从小到大, 同样的视频转码时间变化却是由高降低再升高. 从实验数据来看, 相对于其他的分段, 分段大小为 32M 的时候, 转码时间最佳.

关键词: 视频业务; Hadoop; MapReduce; FFmpeg; 分布式转码

Distributed Transcoding Scheme Based on Hadoop Cloud Computing Platforms

SUN Jian-Wei[2], FU Lei[1,2], YU Bo[2]

[1](University of Chinese Academy of Sciences, Beijing 100049, China)
[2](Shenyang Institute of Computing Technology, Chinese Academy of Sciences, Shenyang 110168, China)

Abstract: With the rapid development of new media video services today, traditional standalone video transcoding capability has been a bottleneck. This paper proposes a new video transcoding scheme based on the research on Hadoop cloud computing platform and the current mainstream audio and video processing tool FFmpeg. This scheme fulfills distributed transcoding method by using two core components in Hadoop: HDFS(Hadoop Distributed File System) and the programming ideas of MapReduce. Meanwhile, the paper also describes in detail the specific processes and design of distributed transcoding. Finally, experimental results show that The distributed transcoding scheme has greatly improved the efficiency. Simultaneously in the experiment, the segment size of each video file also impacts the time of video transcoding process. The experiment shows that as the segment size goes from small to large, time consumed by the transcoding process index experiences a curve: firstly it changes from large to small, then drops to its lowest point and gradually rises. The lowest point exists when the segment size is adjusted to 32M.

Key words: video service; Hadoop; MapReduce; FFmpeg; distributed transcoding

近年来随着网络技术与新媒体的融合发展, 围绕视频应用的服务越来越丰富. 在这过程中一些热门的视频业务深受用户喜爱, 例如: 国外的有 Youtube、Spreecast、Vaobab 等, 国内的有 Youku、YY 直播、乐视网、美拍等. 新的市场需求对视频业务处理能力提出了更高的要求. 当前市场上, 由于终端类型的多样化、各种视频媒体格式以及网络环境的差异性[1], 这就需要一种能够解决视频转码、实时快速处理视频的技术来满足市场需求.

为了推动研究科室新媒体业务, 提高研究科室新媒体互动广播项目的视频转码业务能力, 研究新的视频转码技术已经刻不容缓. Hadoop 平台是目前运用最广泛的云计算技术的开源框架, 支持分布式计算和海量数据处理. 因此, Hadoop 开源框架的出现, 使得解决面向多种终端设备、多种视频格式、多种码流要求的实时视频转码问题成为可能.

① 收稿时间:2015-12-02;收到修改稿时间:2016-01-15 [doi:10.15888/j.cnki.csa.005286]

目前主流的转码技术有以下三种[2-5]:

(1)单机转码: 采用单一的转码服务器进行视频转码. 首先将视频上传至转码服务器, 转码服务器进行转码并返回转码后的视频. 这种方式的主要优点是实现起来比较简单, 但是业务能力比较低, 不能承受高并发转码任务, 同时单一的转码服务器的性能效率相对较低.

(2)面向移动端的实时转码: 采用降码率、降分辨率等方法来达到满足适应多终端、异构网络情况下的多媒体视频业务需求. 这种方法优点是实现起来比较简单, 市场应用性广; 缺点是视频的质量和视频的延时等问题难以解决. 例如文献[2]提出了一个新的控制转码码率 R-D 模型, 介绍了一种窗位码率控制算法 WRC, 最后提出了一个通用的实时速率控制的视频转码方案. 该方案的优点是很好的支撑了电视节目、VOD、在线直播等网络视频转码的实时性, 提高了流畅的视觉质量, 应用性很强; 文献[3]介绍了一种实时转码的系统, 主要是针对 IPTV 网络电视节目. 该方案主要是通过实时重量化降码率转码、实时降分辨率转码、延时稳定保证策略这三种转码技术来实现实时转码. 该方案的主要优点是解决了计算复杂性和转码质量均衡的关键技术问题,减少了视频转码质量下降的问题.

(3)基于云平台的转码: 采用云平台支撑转码系统的服务. 文献[5]中使用 Windows Azure 的云服务, 构建了支持和适配多终端、低带宽消耗的云转码系统, 并使用 CDN 技术对视频流进行分发. 该方案通过基于任务进度预测算法自适应潜在的需求和云端的资源消耗, 最后减少了视频转码适配的网络带宽拥塞问题, 并减少了模型的预测计算压力. 该方案的主要优点是减少了系统资源利用率, 适应高带宽情况下转码策略, 体现了云转码的优越性; 局限性是企业大型云平台的稳定性难以保证.

本文提出了一种基于 Hadoop 云计算平台的分布式转码解决方法. Hadoop 框架包含的两个核心设计是 MapReduce 编程模型和 HDFS(Hadoop Distributed File System)分布式文件系统. 该方法采用 MapReduce 海量数据处理的分布式计算能力, 使用 HDFS 存储需要转码的视频文件, 并结合 FFmpeg 开源项目中的视频转码功能, 共同进行分布式实时转码工作. 针对当前研究科室新媒体互动广播项目的单机转码视频业务, 本方案能够在一定程度上解决传统单机视频转码效率低、耗时高等问题.

1 基于Hadoop云计算平台转码的关键技术

在云平台上进行视频转码的技术主要包含两大模块, 一个是云计算技术, 一个是视频转码技术. 其中云计算技术采用 Hadoop 平台, 视频转码技术利用 FFmpeg 的音视频编解码、转换开源项目.

1.1 Hadoop 云计算平台

Hadoop 是由 Apache 基金会所开发的一款可靠的、稳定的开源分布式基础架构[6]. 该框架通过使用简单的编程模型来支持大规模数据集在集群中分布式计算. Hadoop 平台主要包括 HDFS 分布式文件系统和 MapReduce 计算模型.

1.1.1 HDFS 分布式文件系统

HDFS[7]以流式数据访问模式来存储超大文件, 运行于商用硬件集群上. 与其他文件系统的相比, 它具有高容错性, 支持部署在价格低廉的计算机硬件上, 提供高传输率来访问应用程序数据. HDFS 还提供了一个类似于 POSIX(可移植操作系统界面)文件系统接口, 如图 1 所示是 HDFS 体系结构. 一个 HDFS 文件系统集群包含了一个 NameNode, 主要是管理文件系统的命名空间和维护文件系统所有文件、目录. 集群中还包含多个 DataNode. DataNode 是文件系统的工作节点, 主要管理该结点上数据的存储. DataNode 还负责给文件系统客户端提供读写请求, 还可以从 NameNode 主节点创建、删除、并复制块. 在 HDFS 中, 文件也会被划分为块大小的多个分块, 作为一个独立的存储单元. 在文件读取的时候, NameNode 执行文件系统命名空间的操作, 如打开、关闭、重新命名文件和目录, 并决定了块在 DataNode 上的映射.

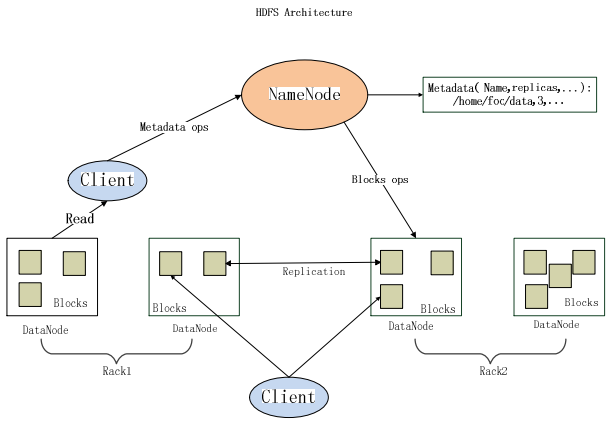

图 1 HDFS 体系结构

1.1.2 MapReduce 编程模型

MapReduce 是 Google 提出的一个软件编程框架, 用于大规模数据集(大于 1TB)的并行运算. MapReduce 的主要工作思想是对大规模数据集的操作, 分发给一个主节点管理下的各个分节点共同完成, 然后通过整合各个节点的中间结果, 得到最终结果[8]. 在 Hadoop 中, 用于执行 MapReduce 任务的机器角色有两个: 一个是 JobTracker; 另一个是 TaskTracker, JobTracker 是用于调度工作的, TaskTracker 是用于执行工作的. 一个 Hadoop 集群中只有一台 JobTracker.

MapReduce 框架负责解决了并行编程中分布式存储、工作调度、负载均衡、容错均衡、容错处理以及网络通信等复杂问题, 把处理过程高度抽象为两个函数: map 和 reduce, map 负责把任务分解成多个任务, reduce 负责把分解后多任务处理的结果汇总[9].

具体工作流程流程图如图 2 所示.

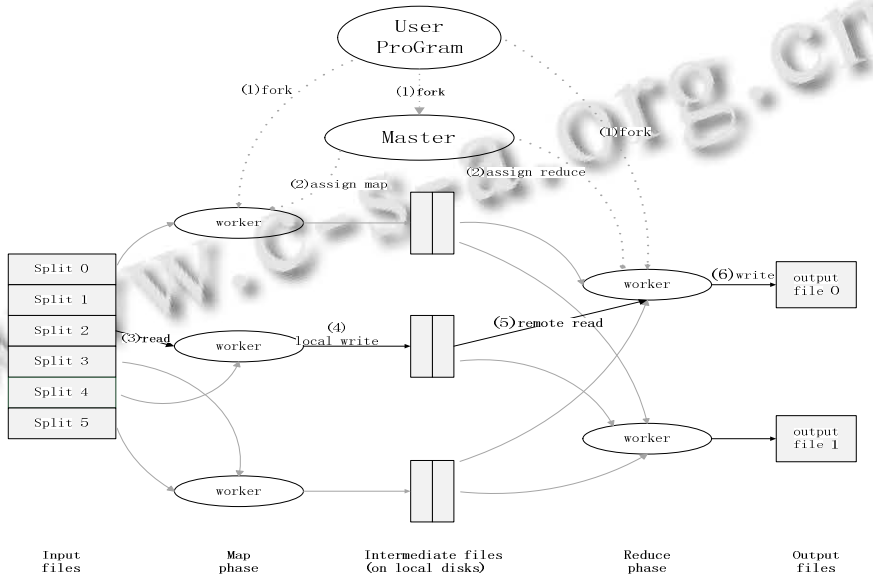

图 2　MapReduce 编程模型计算过程

1.2 **FFmpeg**

FFmpeg[10]是一个非常出色的多媒体框架, 能够记录、编码、解码、转码、复用, 并能将其转化为流的开源计算机项目. 它包含了非常先进的音视频编解码库 libavcoder, 支持 Linux 和 Windows 操作系统. 在多媒体视频方面, FFmpeg 强大的功能体现地淋淋尽致, 除了视频采集、视频格式转换, 还支持视频抓图、视频切割、视频合并、给视频添加水印等.

2　基于Hadoop云计算平台转码系统的设计

2.1 整体系统架构设计

整个云转码系统由 WebServer 和一个包含四个节点 Hadoop 集群组成. 其中 WebServer 主要是负责接收并处理用户请求的数据, 包括存取视频和视频转码的参数等. Hadoop 集群则分为两大块, 一块是由 NameNode 负责处理由 WebServer 转发自己接收到的用户请求, 另一块是 NameNode 调度集群中的 DataNodes, 并进行视频的存储和视频转码工作. 图 3 是该系统的架构图, 从 1-7 清楚地显示了整体架构中每一层之间的逻辑关系.

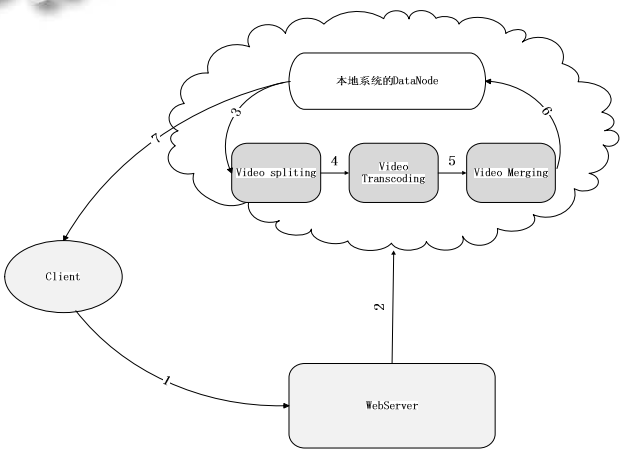

图 3　系统架构图

WebServer 和 Hadoop 集群在处理用户转码请求的时候主要执行步骤如下:

(1)用户发送请求: 获取视频. 用户从客户端(Client)等终端向 WebServer 发送请求, 该请求包含了用户终端设备所支持的视频规格.

(2)WebServer 处理用户请求: 设置转码所需参数. WebServer 根据步骤(1)里面 获取的用户设备支持的视频规格来设置视频转码的参数. 与此同时向 Hadoop 集群中的 NameNode 主节点发送转码任务.

(3)Hadoop 集群执行转码任务: 执行分布式实时转码. NameNode 调度集群中 DataNodes 进行分布式转码. 转码任务完成后, 由 NameNode 向 WebServer 返回转码后的视频文件所在位置.

(4)WebServer 返回用户请求结果: 返回转码后的视频文件位置给用户.

(5)用户接收并获取视频文件: 用户通过客户端(Client)等终端设备从 WebServer 返回的位置读取视频文件.

从用户发出视频请求到读取转码完成后的视频, 这一系列步骤详细地描述了整个系统架构的设计和架构每一模块的功能. 其中, Hadoop 集群执行转码任务的具体流程在 2.2 小节详细介绍.

2.2 基于 Hadoop 集群转码的实验流程

分布式转码的具体实验流程如图 4 所示, 该图主要是视频文件在集群中分割, 转码, 合并等过程.

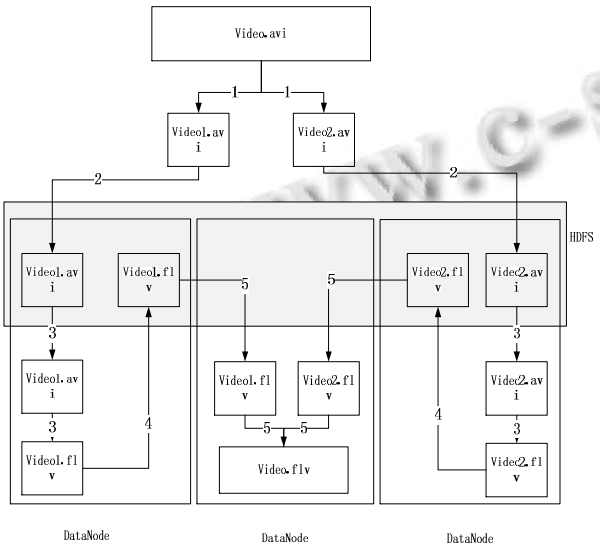

图 4 分布式转码具体流程

2.2.1 视频的分割

在分布式文件系统中, 文件是以块的形式存储. 一个文件一般由一个或者多个块组成, 其中块大小可以调整. 考虑到集群中负载均衡和整体转码效率问题, 视频的每一个分段都需要保证文件大小相同. 本文采用 Avidemux 视频编辑器来分割待转码的视频. Avidemux[11]是一款开源免费的视频编辑器,可以进行剪切、过滤和编码等任务. 它支持广泛的文件格式, 包括 AVI 文件的编辑, MPEG 文件, MP4 和 ASF, 并能将声音从文件中分解出来, 支持强大的队列任务处理和脚本功能. 下面是分割视频的一个命令:

avidemux2_cli inputvideo --autosplit 3 --save outvideo

视频分割在 Hadoop 集群里进行, 采用 Avidemux 开源视频分割工具将源视频文件分割成独立的几段视频, 不会改变视频 PTS 和 DTS 信息. 在视频分割成每一个独立的片段之后, 每一独立片段会分别上传并存储到集群 HDFS 中的 3 个 block 中.

2.2.2 视频的转码

每个分段存到 HDFS 后, 进入视频转码阶段, 采用 FFmpeg 自带的库和命令来进行转码. 以下是使用的一些命令, 本方案会同时对被分割的每个片段进行相应的转码.

ffmpeg -i InputVideo -f mp4 -acodec libflac -vcodec libx264 -b 512k -ab 320k outputvideo

2.2.3 视频的合并

每个独立的视频片段转码成功后, 需要进行合并生成一个新的视频文件. 本文采用的合并工具是 Mencoder, Mencoder[12]是一款命令行方式的视频处理开源软件. 下面是合并两个视频片段的一个命令:

mencoder -ovc copy -oac copy inputvideo1 inputvideo2 -o outvideo

视频合并是在 Hadoop 集群中进行的, 当转码成功后, 系统会根据分段 的视频文件名或者分段所在位置去找到视频文件, 进行合并操作.

2.2.4 Map 和 Reduce 的处理

视频转码工作是在 Map 内完成, Map 内包含有 map()函数. 在 2.2.1 里面, 视频被分割成片段, 这些视频分割的数据需要被格式化, 并传送给 Map, 这个操作将会用到一个抽象类 InputFormat. 通过 InputFormat 中的 RecordReader 从 InputSplit 获取一个个 key/value

对，并交给 map()函数处理，通过 Mapreduce 框架可以做到:

① 验证作业输入的正确性

② 将输入文件切割成逻辑分片(InputSplit)，一个 InputSplit 将会被分配给一个独立的 MapTask

③ 提供 RecordReader 实现，读取 InputSplit 中的“K-V 对”供 map()使用[13]

InputFormat 抽象类的源码如下:

```
Public abstract class InputFormat<K, V>
{
/*仅仅是逻辑分片，并没有物理分片，所以每一个分片类似于这样一个元组 <input-file-path, start, offset>*/
Public abstract List<InputSplit>getSplits(JobContextc ontext)throws IOException,InterruptedException;
Public abstract RecordReader<K,V> createRecordReader(InputSplit split,TaskAttemptContext context) throws IOException,InterruptedException;
}
```

map()函数的参数处理 key 和 value 之外，还包括 OutputCollector 和 Reporter 两个类型的参数，分别用于输出结果和修改 Counter 的值. map()函数内部执行的流程如下图 5 所示

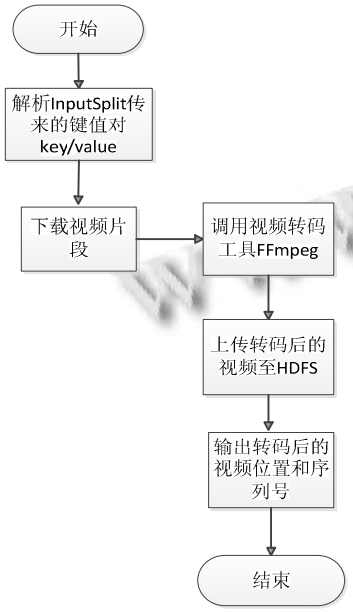

图 5　map 函数执行流程图

分割的视频片段数据被格式化后，并传送给 map()，map()函数从 InputSplit 获得 LongWritable 类型的 key 和 Text 类型的 value. 其中 value 值为文本每一行的内容，包括了视频片段的名字、在 HDFS 的存储路径及要转换的格式和分辨率, key 值为此行在文件中的偏移量. 由于大部分的视频转码工具都是基于本地操作系统的，所以先通过解析 value 中视频片段在 HDFS 存储位置，将视频片段下载到本地；然后使用 2.2.2 介绍的视频转码方法将视频片段转码成目标格式；最后将砖码好的视频片段上传至 HDFS 上，map()函数输出为<Tag,TranscodedVideoPath>，其中 Tag 代表了视频片段的序列号, Transcoded- VideoPath 为转码后的视频片段在 HDFS 上的存储位置.

接着是 Reduce 实现, Reduce 类的体系结构和 Map 非常相似. Reduce 类包含了 reduce()函数，Mapreduce 框架将从 Map 输出的 key 及相同的 key 合并 value 后的 Iterator 交给 reduce()函数来处理. reduce()函数和 map()函数类似，都有 OutputCollector 和 Reporter 两个类型的参数，另外两个参数是 key 和包含 value 的容器的 Iterator<value>. reduce()函数内部执行的流程如下:

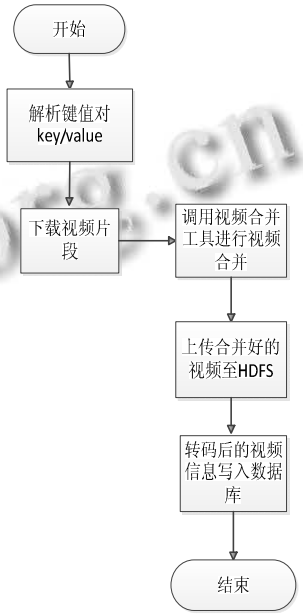

图 6　reduce 函数执行流程图

reduce()函数首先遍历 Iterator<text>，解析出转码后的视频片段的位置信息，并将视频片段从 DataNode 上拷贝到本地系统中. 等所有的视频片段都下载到本地系统后，调用 2.2.3 中介绍的视频合并工具合并所有

视频片断，再合并好的视频文件上传到 HDFS 并将转码后的视频信息存入数据库中，reduce()函数不用输出任何键值对.

3 分布式转码实验结果和分析

3.1 Hadoop 平台转码环境介绍

在 Hadoop 集群是由四台普通 PC 组成，相同的配置. 电脑处理器为 Intel(R) Core(TM) i3-4150，最高主频是 3.5GHZ，内存 4G，硬盘大小是 500G. 集群组成情况是一台作为 NameNode，其他三台是 DataNode.

3.2 实验数据和实验结果

本实验对四个大小不同的视频文件进行实验，四个的视频文件分别是 200M、500M、800M、1.2G，再对五种分段大小情况下所采集的实验数据进行分析，获得四个视频文件对应不同分段大小转码所需的时间趋势如图 7 所示.

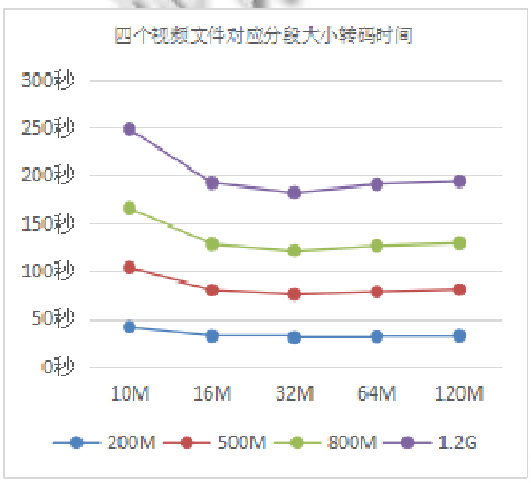

图 7 四个视频文件对应不同分段大小的转码时间趋势

从图 7 可以看出，随分段大小 10M、16M、32M、32M、64M、120M 依次递增，分布式转码的时间先减小后增大的趋势. 同时可以发现在五种分段大小中，分段大小为 32M 时，所需转码时间最小. 为了比较分布式转码相对于传统单机转码的效率，本文对上述的四个视频文件进行了单机转码，最后对单机转码时间和在分布式条件下分段大小为 32M 的时候转码时间进行对比，对比结果如图 8 所示.

与传统单机转码相比较，分布式转码的时间明显更小，更有效率. 本实验选取分段大小最佳的 32M 为基准，计算出分布式转码时间比传统单机转码提升了 60.3% ，这样的提升显著提高了转码的效率，提高了视频业务的体验效果，适应了当前时代新媒体快速发展的潮流.

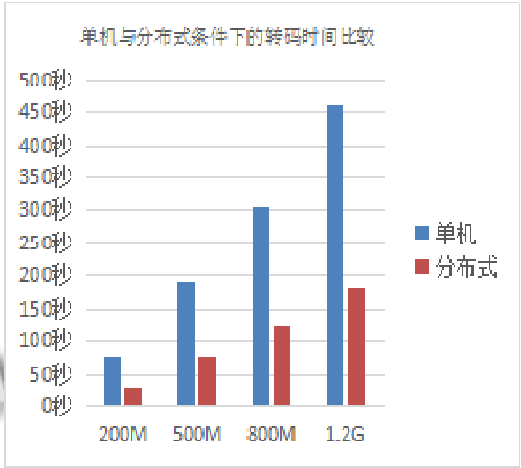

图 8 四个视频文件对应分段大小转码时间

4 结语

本文提出了一种基于 Hadoop 云计算平台的分布式转码方案，采用 HDFS 分布式文件系统和 MapReduce 计算模型分别对视频文件进行备份存储和转码工作. 这种系统架构的设计不仅仅大大提高了转码效率，节省了转码时间. 由于 HDFS 的高容错性和备份存储机制，使得如果有节点发生了崩溃或者意外，还能保证转码工作继续进行，从而还提升了转码成功率. 实验数据表明，与传统单机转码相比，以 32MB 分段大小为例，分布式转码时间提升了 60.3%. 此外，这种基于Hadoop云平台的分布式转码方案，还便于扩展集群的大小. 在今后的研究工作中，优化视频的分段时间、寻找最佳分段大小对视频转码的影响是需要进行大量地努力实践. 另外，Hadoop 2.x 的升级对分布式转码平台可扩展性、易部署性、高效性的影响也是今后的研究重点.

参考文献

1 杨帆,沈奇威.分布式系统 Hadoop 平台的视频转码.计算机系统应用,2011,20(11):80–85.

2 Xu L, Kwong S, Wang HL, Zhang Y, Zhao DB, Gao W. A Universal Rate Control Scheme for Video Transcoding. Dan Schonfeld, ed. IEEE Trans. on Circuits and Systems for Video Technology. New Jersey: IEEE Circuits and Systems Society, 2012: 489–501.

3 王海蓉,邢卫,鲁东明.面向移动网络的实时视频转码系统.计算机工程,2009,35(2):245–247.

4 Li ZH, Huang Y, Liu G, Wang FC, Zhang ZL, Dai YF. Cloud transcoder: Bridging the format and resolution gap between internet videos and mobile devices. Proc. of the 22nd International Workshop on Network and Operating System Support for Digital Audio and Video. ACM. 2012.7. 33–38.

5 Cheng R, Wu WJ, Chen YQ, Lou YH. A cloud-based transcoding framework for real-time mobile video conferencing system. 2014 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering. Oxford. IEEE. 2014. 236–245.

6 李乔,郑啸.云计算研究现状综述.计算机科学,2011,38(4):30–37.

7 Shvachko K, Kuang HR, Radia S, Chansler R. The hadoop distributed file system. 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies. Nevada. 2010. 1–10.

8 Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. Communications of the ACM, 2008, 51(1):107–113.

9 包盛,段保通,邵锋军.三网融合下基于云计算的实时转码技术研究和应用.电信科学,2011(3):12–16.

10 Bellard FM. Niedermayer. FFmpeg. http://www. ffmpeg.org/.

11 Avidemux. Avidemux. http://www.avidemux.org/.

12 Mencoder. Mencoder. http://www.mplayerhq.hu/DOCS/HTML/en/mencoder.html.

13 Song CW, Shen WF, Sun LQ, Lei Z, Xu WM. Distributed Video Transcoding Based on MapReduce. 2014 IEEE/ACIS 13th International Conference on Computer and Information Science (ICIS). Taiyuan. 2014. 309–314.