分布式系统 Hadoop 平台的视频转码①

杨　帆，沈奇威

(北京邮电大学 网络与交换技术国家重点实验室，北京 100876)
(东信北邮信息技术有限公司，北京 100191)

摘　要：在研究了目前主流的视频转码方案基础上，提出了一种分布式转码系统。系统采用 HDFS(Hadoop Distributed File System)进行视频存储，利用 MapReduce 思想和 FFMPEG 进行分布式转码。详细讨论了视频分布式存储时的分段策略，以及分段大小对存取时间的影响。同时，定义了视频存储和转换的元数据格式。提出了基于 MapReduce 编程框架的分布式转码方案，即 Mapper 端进行转码和 Reducer 端进行视频合并。实验数据显示了转码时间随视频分段大小和转码机器数量不同而变化的趋势。结果表明，相比单机转码，提出的系统在采用 8 台机器并行转码时，可以节约 80%左右的时间。

关键词：视频转码；分布式内容处理；Hadoop；FFMPEG

Distributed Video Transcoding on Hadoop

YANG Fan, SHEN Qi-Wei

(State Key Lab of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China)
(EB Information Technology Co. Ltd., Beijing 100191, China)

Abstract: Based on study of current video transcoding solutions, we proposed a distributed transcoding system. Video resources are stored in HDFS(Hadoop Distributed File System) and transcoded by MapReduce program using FFMPEG. In this paper, video segmentation strategy on distributed storage and how they affect accessing time are discussed. We also defined metadata of video formats and transcoding parameters. The distributed transcoding framework is proposed on basis of MapReduce programming model. Segmented source videos are transcoding in map tasks and merged into target video in reduce task. Experimental results show that transcoding time is dependent on segmentation size and trascoding cluster size. Compared with single PC, the proposed distributed video transcoding system implemented on 8 PCs can decrease about 80% of the transcoding time.

Key words: video transcoding; distributed processing; Hadoop; FFMPEG

1　引言

多媒体系统的发展给图像视频编码带来了很大的影响，随着视频分享网站（Youtube，Youku 等）的兴起，由于网络环境、终端类型、媒体内容格式的不同，网站经常需要进行大量的离线视频转码工作，以适应异构网络和多终端环境的需要。通常情况，视频转码系统与存储系统相分离，专门的视频存储服务器造价昂贵，并且转码系统一般采用单一服务器，负载较重且耗时较长；同时，在视频转码过程中由于视频在两个系统间的传输会带来大量的网络负载。本文采用 Hadoop 框架中 HDFS (Hadoop Distributed File System)存储备份机制对视频文件进行存储，使用 MapReduce 分布式计算思想，利用 FFMPEG 对视频进行分布式转码，将存储与转码集中，旨在达到对海量视频文件的廉价存储与快速转码。

① 基金项目:国家杰出青年科学基金(60525110);国家 973 计划(2007CB307100,2007CB307103);国家自然科学基金(61072057,60902051);中央高校基本科研业务费专项资金(BUPT2009RC0505);国家科技重大专项(2011ZX03002-001-01,2011ZX03002-002-01)
收稿时间:2011-03-20;收到修改稿时间:2011-04-24

1 现状

视频转换编码简称视频转码，是指视频从一种格式到另一种格式的转换，其中格式是由码率、帧率、空间分辨率和编码算法所表征[1]。随着高清电影的发展，影片的片源容量从几 G 增加到几十 G 不等，对此类片源转码所需的时间也急剧增加。另外，由于转码需要适配的用户终端类型增多，转码任务日趋繁多。对视频存储服务器和转码服务器都提出了严峻的考验[2]。

目前主流的转码模式有以下三种[3-5]：

(1) 单机：使用单一的转码服务器对视频进行转码工作。将视频传输至转码服务器，转码服务器转码，返回转码后的视频。这种方式优点是实现起来较简单；缺陷是转码服务器的性能限制转码时间，不能承受高并发的转码任务。

(2) 分布式：采用多台转码机器同时对视频文件进行转码。将视频在片源分段，将每段传输到对应的转码机器上，转码完成后将各段合并成一个视频文件，返回该视频[6]。这种方式优点是并行转码，时间成本低，可以应对并发转码任务；缺陷是实现复杂，需要考虑分段同步以及断后合并等问题。

(3) 基于云：使用云的存储和计算能力对视频进行转码。例如 Grio[7]使用 Amazon 的 S3（Simple Storage Service）服务器对视频文件进行存储，利用 EC2（Elastic Compute Cloud）对视频文件进行转码。其中，EC2 的一个实例（Instance）负责一个视频的转码任务。

本文提出一种基于 Hadoop 平台的分布式视频转码方案，使用 HDFS 存储视频文件和利用 MapReduce 编程框架进行分布式转码，转码工具采用 FFMPEG，将视频存储和转码集成一体，达到减少转码时所需网络带宽和耗时的目的。

2 HADOOP与FFMPEG

Hadoop[8]是当前热门的云计算存储平台之一，是 Apache 组织的一个开源的分布式计算框架，在如 Amazon，Facebook，Yahoo！等很多大型企业中得到了广泛应用。Hadoop 框架中最核心的设计是 HDFS (Hadoop Distributed File System)和 MapReduce 软件编程框架。

HDFS 是分布式文件系统，参照 GFS[9]（Google File System）实现，拥有多机备份、扩展性强且经济廉价等特点，可以运行在普通 PC 机上；同时存储采用 Key-Value 形式，适合视频和音频此类非结构数据存储[10]。HDFS 采用 master/slave 架构。一个 HDFS 集群是由一个 NameNode 和多个 DataNodes 组成。其中 NameNode 是中心服务器，负责管理文件系统的命名服务和客户端对文件的访问。DataNode 是存储节点，一般情况下一台机器拥有一个存储节点，负责管理数据存储。HDFS 对用户透明，在 HDFS 内部，一个文件被分割为一个或多个数据块（block），这些数据块被存储在一组 Datanodes 上[11]。图 1 是 HDFS 的体系架构。

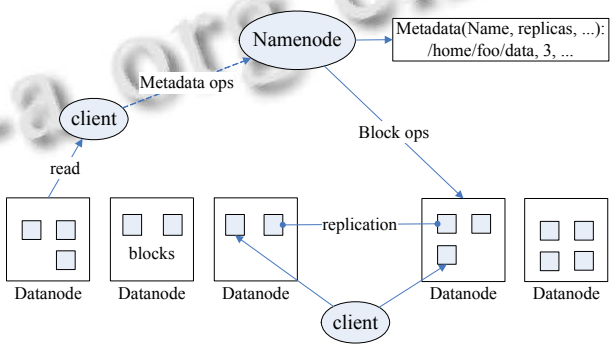

图 1 HDFS 体系架构

MapReduce[12]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。一个 MapReduce 作业会被拆分为 Map 和 Reduce 两个阶段，Map 阶段对数据进行分布式处理，Reduce 阶段将 Map 输出的结果合并，得到最终结果。通常，MapReduce 框架和 HDFS 运行在一组相同的节点上，即计算节点和存储节点通常在一起。这种配置允许 MapReduce 框架在那些已经存储好数据的节点上高效地调度任务，充分利用整个集群中的网络带宽。

FFMPEG[13]是一个集录制、转换和音视频编解码功能为一体的开源解决方案，可以运行在 Linux 和 Windows 操作系统下。FFMPEG 支持 MPEG、MPEG4、FLV、Div 等 40 多种编码，AVi、MPEG、Matroska 等 90 多种解码。它能快速实现音视频格式转换、切割、合并、加水印等多种媒体处理功能。

3 系统架构

如图 2 所示，整个系统由一个 WebServer 和一个 Hadoop 集群组成。WebServer 负责处理用户请求，包括视频的存取和转码。Hadoop 集群中的 NameNode 负责接收 WebServer 转发的用户请求，调度集群中的

DataNodes 进行视频存储或者转码。

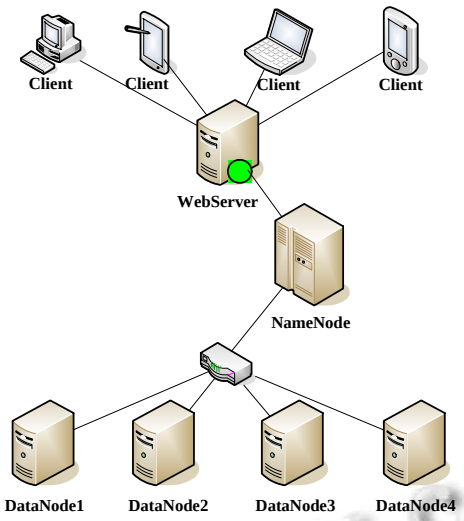

图 2　系统架构图

系统处理一个普通的用户转码请求可分为以下几个步骤：

① 视频请求：用户从远端向 WebServer 发送获取视频的请求，包括视频名称与用户设备规格。

② 设定转码参数：WebServer 根据用户设备参数设定转码参数，向 NameNode 发出转码任务的命令。

③ 分布式转码：NameNode 调度集群进行分布式转码。

④ 转码完成：分布式转码完成后，NameNode 向 WebServer 返回转码完成后的视频文件所在位置 DataNodeX。

⑤ 返回视频所在位置：WebServer 将位置 DataNodeX 返回给用户。

⑥ 读取视频：用户从 DataNodeX 机器上读取转码后的视频文件。

第四节将详细介绍视频分布式存储与分布式转码的实现方案。

4　视频分布式存储

4.1 视频文件分段

HDFS 中文件都以数据块形式存储，一个文件由一个或多个数据块组成，其中数据块的大小可以调整，故在存储视频文件时需要分段（对视频分段即分块，本文中统一采用分段的说法）。由于视频是非结构化数据，压缩编码方式会导致视频中帧与帧之间产生关联性。如图 3 所示，以 MPEG-2 为例，MPEG-2 中有两种 GOP（Group Of Pictures），分别为 Open-GOP 和 Closed-GOP[6]，GOP(k-1)是 Closed-GOP，GOP(k)是 Open-GOP，在解码 GOP(k)的帧 B14 和帧 B15 时，需要 GOP(k-1)的帧 P13，如果视频分段后，某个段的第一个 GOP 是 Open-GOP，则在解码这个 Open-GOP 的时候需要前一个段的某一帧，从而造成段与段之间的关联性。

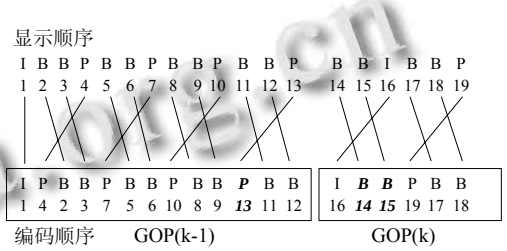

图 3　GOP 的分段

如果不对视频做任何处理，直接存储到 HDFS 中，会造成上述段与段的关联性。这样在分布式转码时会与 MapReduce 软件架构适合处理解耦合数据的思想冲突，故考虑如下三种存储策略：

① 一个视频文件一个段：视频没有被切割，可以保证单一段的完整性。但是如果视频文件过大（10GB 以上，蓝光影片可以达到 50GB），在转码时如果采用单点本地转码会造成单点转码负载过大。

② 一个视频文件多个段：如果将视频文件分为几部分，分发到其他机器上进行转码，会涉及到视频完整性问题，且在分发时会造成单点网络负载过大。

③ 一个视频文件多个独立段：采用 FFMPEG 独立分割视频的方法，将一个视频文件分割成 n 段独立的视频文件，每一段都可以单独进行播放。将视频文件以固定大小 m（例如 m=64MB）的分段切割成独立的 n 段，前 n-1 段大小为 m，最后一段小于或等于 m。由于 n 段互相独立，则不会产生段与段之间的关联性问题。将分割后的 n 段视频分别存入 HDFS 中的 n 个 block 中，block 大小为 m。在 8 台 DataNodes 的 Hadoop 集群中，对文件大小为 1.3GB 的视频存取时间随 m 取值的变化如图 4 和图 5 所示。

从实验结果中能看出分段的大小对文件读取和写入效率没有太大影响。但是在对大视频进行转码时，分段太小会造成分段过多，进而造成转码任务的个数增多，消耗在启动转码任务上的时间也会随之增加，

从而影响整体转码时长。

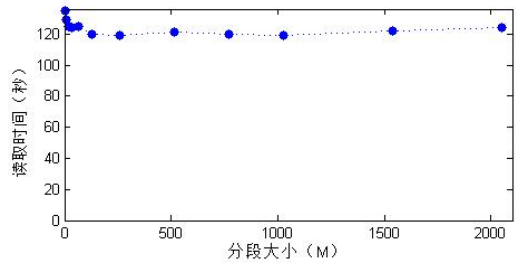

图 4 读取时间

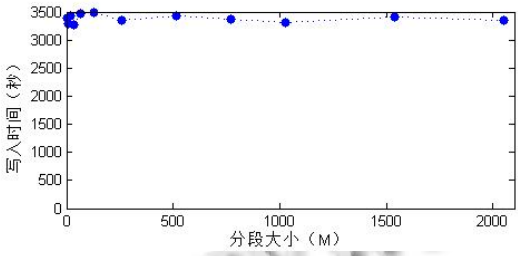

图 5 写入时间

本系统将视频分段后存储在 HDFS 中后，在 NameNode 上运行 Hadoop balancer 命令，尽量保证各段均匀分布在 Datanode 上，这样是为了在 MapReduce 转码过程时尽量达到数据本地化，减少网络传输。

4.2 元数据存储

国际图联 IFLA 对元数据的定义是：“元数据就是数据的数据，此术语指任何用于帮助网络电子资源的识别、描述和定位的数据”[14]。元数据用来描述数据，分布式视频存储中的元数据包括两类：

① 转换元数据：由于用户终端设备类型的不同，转码参数也不相同。为了对终端类型进行统一管理，使用 UAProfile[15]存储用户设备参数，也就是目标视频转码参数。转换元数据的格式图 6 所示：

```
<device>
  <name>iPhone3GS</name>
  <format>mp4</format>
  <bitRate>304</bitRate>
  <width>320</width>
  <height>480</height>
  <resolution>2:3</resolution>
</device>
```

图 6 转换元数据格式

② 视频元数据：由于将一个完整的视频分割成 n 段独立的视频，需要记录 n 段独立视频的顺序，以便从 HDFS 中还原出原始视频，故视频元数据需要保存一系列文件名、原视频的相关参数（例如比特率、分辨率等）以及 n 个独立视频的相关参数。视频元数据的格式如图 7 所示：

```
<video>
  <videoInfo>
    <name>Clash.of.the.Titans</name>
    <size>1363081129</size>
    <numOfPiece>12</numOfPiece>
    <duration>1:52:46</duration>
    <format>m4v</format>
    <bitRate>1601</bitRate>
    <width>640</width>
    <height>360</height>
    <resolution>4:3</resolution>
  </videoInfo>
  <piece>
    <part id="1">
      <name>part1</name>
      <size>123803490</size>
      <duration>00:10:00</duration>
    </part>
    <part id="2">
      <name>part2</name>
      <size>115774994</size>
      <duration>00:10:00</duration>
    </part>
  </piece>
</video>
```

图 7 视频元数据格式

5 视频分布式转码的实现

5.1 视频转码

MapReduce 是一种软件编程框架，分为 Map 和 Reduce 两个过程。本文提出的分布式转码算法为 Map 转码和 Reduce 合并，转码流程如图 8 所示。在 Map 端做转码，需要对一个视频文件的多个段进行转码，应该尽量保证视频文件的多个段分布在不同的 Datanode 上，从而达到数据本地化和并行化；Reduce 端采用单一 Reducer，负责将来自多个 Mapper 的输出文件进行合并，形成转码后完整的视频文件。

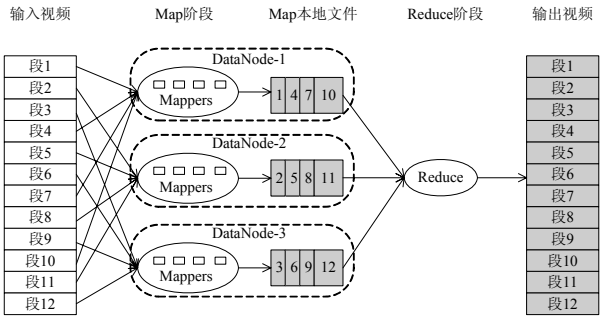

图 8 转码流程图

Mapper

在 DataNode 上，每一个视频段都会相应启动一个 Mapper，也就是说 Mapper 数量与视频段数量相同，

Mapper 负责读取视频元数据以及转码元数据，通过启动 FFMPEG 设定转码参数，对一个视频段进行转码。同时 Mapper 可以监控 FFMPEG 转码过程，记录转码日志。

① 如果 FFMPEG 转码失败，重启转码任务。在失败次数超过阀值（这里设为 3）时，报告任务失败。重新在拥有此段备份的另一个 datanode 上执行转码任务，如果 3 个备份转码都失败，则报告此段转码任务失败。

② 如果 FFMPEG 转码成功，采用 Gzip 压缩格式将转码后的视频段输出为中间结果。如表 1 所示，将文件大小为 1.3GB 的视频文件分割为 12 段，对比使用和未使用 Gzip 压缩的中间结果大小，能看出使用 Gzip 压缩后，Map 输出结果缩小了 19404KB，减少 8.5%；Reduce 读取结果缩小了 19404KB，减少 8.5%。所以，对 Mapper 输出结果采用 Gzip 压缩从一定程度上减少了网络通信量与磁盘 IO。

表 1　使用和未使用 Gzip 压缩的中间结果大小

	Map Read(KB)	Map Write(KB)	Reduce Read(KB)	Reduce Write(KB)
未压缩	1355440	228130	228129	228129
Gzip 压缩	1355440	208726	208725	208725

由于需要将转码后的视频返回给 NameNode，故使用一个 Reducer 将各个已完成转码的视频段合并成一个完整的视频文件。由于各个 Mapper 所在的 datanode 机器性能各不相同，每个 Mapper 对视频段的转码时间也不尽相同，而且 Reducer 对视频的合并需要按照视频段的先后顺序进行合并，故采用如下策略：

Recuder 等待 Mapper 将 n 个视频段转码完毕后，通过网络读取所有视频段，再做顺序合并。合并完成后将生成本地文件，将文件路径发送给客户端，即文件路径的传递顺序为 Reducer->NameNode->WebServer->客户端。客户端读取本地文件，转码任务完成。

5　实验结果分析

实验环境

WebServer：16 核 CPU，4G 内存的刀片服务器。

Hadoop 集群：9 台普通 PC 机，每台机器 CPU 为 PD2.8GHz，内存 1G，硬盘 160G。其中一台机器作为 NameNode，其他 8 台机器作为 DataNodes，并安装 FFMPEG，负责视频转码任务。

输入条件

以图 6 中大小为 1.3GB 的视频文件为例，将文件分别按 4M、8M、16M、32M、64M、128M 和 256M 分割，分割后的视频段数如表 2 所示。

表 2　分段大小与分段个数

段大小	4M	8M	16M	32M	64M	256M	512M
段个数	339	139	92	46	23	11	6

将分段后的视频分别在 4 台、5 台、6 台、7 台和 8 台 DataNodes 的转码集群中进行分布式转码，转码参数如表 3 所示。

表 3　视频转码参数

	编码格式	比特率	分辨率
输入	MPEG-4	1601Kbps	640*360
输出	MPEG-2	304Kbps	320*480

实验结果

图 9 为分布式转码所消耗的时间。从图中可以看出，转码时间随着分段大小的增加，呈现先下降后上升的变化。在分段大小为 32M 时，转码时间最少，速度最快。同时，随着转码节点的增加，整体转码时间呈下降趋势。

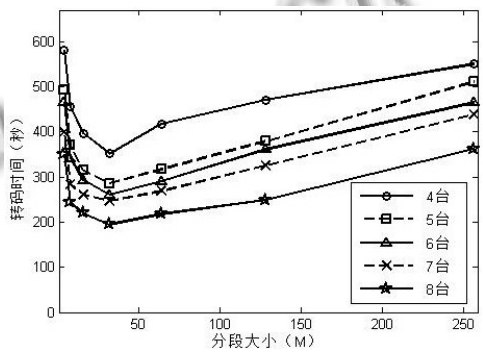

图 9　分布式转码消耗时间对比

以 32M 为分段大小，图 10 显示了转码时间随着转码节点从 1 台变化到 8 台的变化趋势。从图中能看出使用单机时转码时间需要 994 秒。而采用本文中的分布式转码系统，在使用 8 台机器作为集群的情况下，仅需要 195 秒，缩短了约 80%的时间，很大程度上提高了转码的效率。

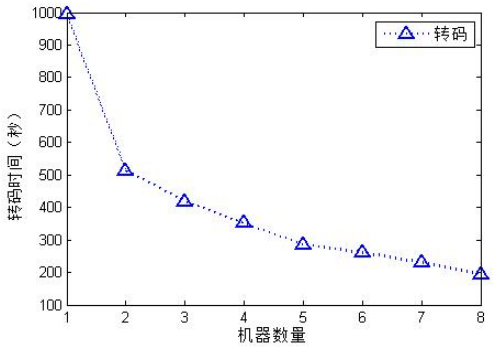

图 10 分段大小为 32M 时转码时间对比

7 总结

本文提出了一种基于 Hadoop 分布式平台的视频转码系统，采用 32M 为视频分段大小，在 HDFS 上多机备份存储；使用 MapReduce 编程框架，在 Mapper 端进行转码，Reducer 端进行视频段合并，从而完成分布式转码工作。与普通分布式转码系统[3-5]有所区别，本系统将视频存储与转码结合，在开始转码之前，视频已经分布式存储于各个转码节点，转码时无需向转码节点分发源视频文件，从而减少网络流量。实验数据表明，相比单机转码，采用 8 台机器并行转码能减少大约 80%的转码时间，转码时间与转码机器的数量呈反比。本系统不足之处在于将视频存入 HDFS 之前，需要使用 FFMPEG 进行视频独立分段，会耗费一定时间。后续工作是修改 HDFS 本身的分段机制，在视频存入 HDFS 的同时进行 FFMPEG 独立分段。另外，增加对视频的多任务转码也是今后即将研究的重点。

参考文献

1 Ahmad I, Wei XH, Sun Y, Zhang YQ. Video Transcoding: An Overview of Various Techniques and Research Issues. IEEE Trans. on Multimedia, 2005,7(5).

2 杨戈,廖建新,朱晓民.流媒体分发系统关键技术综述.电子学报,2009,37(1):137−145.

3 Barlas G. A Taxonomy and DLT-based analysis of Cluster-based Video Trans/Encoding. 14th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP 2006,388−395.

4 Cardellini V, Colajanni M, Lancellotti R, Yu PS. A distributed architecture of edge proxy servers for cooperative transcoding. The 3rd IEEE Workshop on Internet Applications, 2003,66−70.

5 Guo JN, Bhuyan L. Load Sharing in a Transcoding Cluster. Distributed Computing, 2003,835.

6 Sambe Y, Watanabe S, Yu D, Nakamura T. Distributed video transcoding and its application to grid delivery. Proc. ITC-CSCC2003, 2003, 921−924.

7 Grio African American Breaking News and Opinion, http://www.thegrio.com/.

8 The Apache Hadoop project. http://Hadoop. apache. org/.

9 Ghemawat S, Gobioff H, Leung ST. The Google File System. 19th Symposium on Operating Systems Principles. Lake George, New York, 2003,29,43.

10 Hadoop Makes Sense of Lots of Data. http://www.enterprisestorageforum.com/article.php/3890191/Hadoop-Makes-Sense-of-Lots-of-Data.htm

11 Borthakur D. The Hadoop Distributed File System: Architecture and Design. http://hadoop.apache.org/core/docs/current/hdfs_design.html, 2007

12 Dean J, Ghemawat S. MapReduce: Simplied Data Processing on Large Clusters. Proc. of OSDI'04: 6th Symposium on Operating System Design and Implemention, San Francisco, CA, Dec. 2004.

13 Bellard F, FFMPEG multimedia system. http://FFMPEG.sourceforge.net/index.php

14 Mathes A. Folksonomies-cooperative classification and communication through shared metadata. Computer Mediated Communication, LIS590CMC (Doctoral Seminar). Graduate School of Library and Information Science, University of Illinois Urbana-Champaign, December 2004.

15 Matsuyama K, Kraus M, Kitagawa K, Saito N. A Path-Based RDF Query Language for CC/PP and UAProf. Proc. of the 2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW'04), 2004.