学校代码：10004

密级：公开

北京交通大學

BEIJING JIAOTONG UNIVERSITY

硕士专业学位论文

分布式视频处理系统设计与实现

作者姓名 袁大山

工程领域 软件工程

指导教师 陈旭东 副教授

培养院系 软件学院

二零一五年六月

北京交通大学

硕士专业学位论文

分布式视频处理系统设计与实现

The Design and Implementation of Distributed Video Processing System

作者：袁大山

导师：陈旭东

北京交通大学

2015 年 5 月

学位论文版权使用授权书

本学位论文作者完全了解北京交通大学有关保留、使用学位论文的规定。特授权北京交通大学可以将学位论文的全部或部分内容编入有关数据库进行检索，提供阅览服务，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。同意学校向国家有关部门或机构送交论文的复印件和磁盘。学校可以为存在馆际合作关系的兄弟高校用户提供文献传递服务和交换服务。

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名：袁大山　　导师签名：陈之洋

签字日期：2015年6月5日　　签字日期：2015年6月5日

学校代码：10004 密级：公开

北京交通大学

硕士专业学位论文

分布式视频处理系统设计与实现

The Design and Implementation of Distributed Video Processing System

作者姓名：袁大山 学 号：13126162

导师姓名：陈旭东 职 称：副教授

工程硕士专业领域：软件工程 学位级别：硕士

北京交通大学

2015 年 5 月

致谢

光阴似箭，两年的研究生学习即将结束，在这里非常感谢我的导师、父母、同学们，让我研究生期间学到了知识，也学会了生活。

本论文的工作是在我的导师陈旭东教授的悉心指导下完成的，陈老师严谨的治学态度和科学的工作方法给了我极大的帮助和影响。在这里衷心的感谢陈老师的对我的栽培。

其次感谢我们父母、亲人，感谢他们理解我，并且一直在支持和鼓励我，让我更乐观的面对工作和学习。

同时，感谢在学习、生活上帮助过我的同学们，研究生生活有了你们才更加有意义！

最后，两年的校园生活让我受益匪浅，也感谢学校和学院，能够接纳我，并提供了一个很好地平台。

摘要

近年来，随着网络接入技术的进步和网络使用成本的降低，在线视频点播用户越来越多。据统计，2014 年视频数据已经占互联网总数据流量的一半以上，随着移动端、电视端设备的普及，在线视频需求量将会进一步增加，预计到 2019 年视频数据将超过总数据量的 80%。在这种情况下，传统视频业务面临着两方面挑战：一方面是如何在保障性能的情况下应对日益增多的视频点播量，另一方面是如何保障不同类型终端上视频的兼容性。视频点播量大，造成服务器转码任务越来越繁重，同时转码还需要兼顾不同播放平台对视频编码格式、帧率等参数多样化要求，这进一步加重了服务器的负担。在这种情况下，原有的集中式视频转码服务已经不能满足业务的需求，必须升级架构。在现有技术条件下，采用分布式集群进行并行转码是一个比较好的选择。

论文以网易公司网站部视频系统升级为背景，设计和实现了分布式视频处理系统，以此应对大规模、多需求的视频转码场景。分布式视频处理系统的核心任务是设计合理高效的转码系统，集中计算资源、提高服务器利用率，并对视频转码提供统一的服务接口，保障视频服务。论文完成的具体的工作内容包括：完成了视频业务的需求分析，确定了视频服务的具体需求和应用，并分析了现有的视频处理系统的流程和功能；设计了新开发的分布式视频处理系统业务逻辑和系统架构；开发并实现了相关功能模块，包括负载接入、视频预处理、视频任务分发、视频任务处理等模块，并为其他业务模块提供统一的视频处理接口；测试和优化了系统功能，保障系统可用性与高效性。

目前，论文完成的分布式视频处理系统最终已经正式上线，在处理大量视频文件时，能够根据业务需求进行自动转码并存储，根据视频格式的特点，可以实现高效的转码任务调度，提高了视频转码的效率，为视频用户提供了可靠的技术保障。

关键词：视频转码；分布式；FFMPEG

ABSTRACT

In recent years, with the development of network access technology and reduction of network cost, more and more people use the online video service. According to statistics, more than half of Internet data are video data in 2014.It predicts that the video data will make up over 80% of the total data in 2019. In this case, the traditional video business faces two challenges. The first one is how to guarantee the performance coping with the increasing amount of video on demand. The other one is how to ensure compatibility of video on different terminals. Transcoding server have more and more heavy task which also need to meet the demand of different parameters such as video coding format and frame rate on different platforms, due to the large quantity of video on demand. In the circumstances, it is time to upgrade the architecture because the primary service of centralized video transcoding already cannot satisfy the needs of the business. In this situation, it is a good choice to adopt the distributed cluster parallel transcoding using available technology.

The paper based on update of video system in NetEase Company, which researched on the design and implementation of the distributed architecture on video processing system, to deal with large-scale and various transcoding scenes. The task of the distributed video processing system is design a reasonable and efficient transcoding system, which centralizes computing resources and improves server utilization. It also provides a unified service interface for video transcoding to ensure the quality of service. The paper carved into several parts: The first part is analyzing the requirements of video, determine the specific requirements and application. Secondly, choose suited service logic and system architecture. Thirdly, develop system module and launch the system, then provide unified video processing interface to website developer. Fourthly, test and optimize the system to verify availability and efficiency of system.

Now, system for distributed video processing has officially launched. It could storage copes and automatic transcoding according to different transcode requirements. And it makes automatic schedule reality which raises efficiency and provided power support for user.

KEYWORDS: Video Transcode; Distributed System; FFMPEG

目录

1 引言

本项目来源于本人在网易互联网事业部的网站部实习期间的实习项目，是支撑整个部门视频处理的系统。本章将详细介绍项目的选题背景、研究内容和意义、建设目标和关键问题与技术难点，并在最后给出论文的组织结构。

1.1 论文背景

随着网速的提高，网络使用成本的降低，越来越多的用户接入了互联网，同时越来越多的互联网用户通过网络观看电影、电视等视频内容。视频作为一种直观的媒体形式，已经成为网民生活中非常重要的一部分。同时，随着手持终端的兴起，目前在线视频需要兼顾 PC 端和移动端，然而 PC 端和移动端对于在线流媒体视频的分辨率要求、带宽要求等不甚相同，传统单一的视频技术已经不能满足多样化终端的需求，因此，需要新的视频转码系统来完成这个任务。

总体看来，目前互联网视频的特点有：视频实时性要求较高；视频终端的类型多样化明显；带宽资源宝贵，终端带宽速度区别大；

这就要求对于不同的终端需要提供同量不同质的服务，即相同的内容，根据终端的特点传输不同的画质的视频数据，减少带宽占用。

网易公司用于视频业务的计算资源有限，但是视频业务确实网站中常用的业务，因此必须使尽量少的资源来发挥最大的作用，这就需要公司视频部门集中计算资源，使得计算资源的利用率更高，为公司创造更高的价值。

在这种情形下，一个网络提供者的网站视频处理系统的优劣将会直接决定着用户使用网络视频的体验。旧的视频处理系统架构由于在性能与架构上的设计，已经不能满足新的形势要求，迫切需要一个采用比较合理、高效的架构的新的视频处理系统，来提高处理效率，提升用户的浏览网络视频的体验。

目前国、内外专业的视频网站已实现的视频处理系统大多采用分布式的系统架构，比如 YouTube、爱奇艺、优酷等。而非专业的视频网站，随着视频点击量上升，服务器压力越来越大，也正在开发类似的分布式视频处理服务器系统来应对这种问题。

采用分布式架构的能够根据业务量灵活的改变配置，减少计算资源的浪费。同时，有比较好的扩展能力，方便以后架构升级、性能升级。

1.2 国内外分布式转码现状

目前来看，国内外一些公司已经应用了一些分布式转码技术。

国内来看，主要有迅雷转码系统、腾讯云转码系统。迅雷转码系统实现的功能是将视频文件直接转码，不需要用户下载后转码，通过指定源片地址，生成新的目标格式视频的地址。经了解，迅雷转码系统主要是依靠分布式框架，结合FFmpeg（Fast Forward Moving Picture Experts Group）开源视频处理技术，将视频整体发送给集群节点，通过单节点计算完成转码任务。腾讯云转码需要用户提供名称和格式，然后根据用户要求，将视频提交至 Hadoop 集群，返回处理结果。

国外厂商 Amazon 提供了 AET（Amazon Elastic Transcoder）系统，能够支持的转码类型较多，在系统设计上也是采用分布是框架，主要依靠亚马逊云平台构成逻辑上的分布式，将单个任务封装成一个 pipeline，随机分发给各节点转码，进而返回转码结果。

1.3 论文主要目标

网易公司的主营业务是门户网站，视频业务原本是公司的一个辅助业务。由于各部门对视频业务的需求不同，开发的具体细节也不相同，多个后台子系统提供了多个、功能部分重复视频处理的模块。由于视频处理所需的计算资源较多，并且对 IO 影响比较明显，因此，重复的模块就会明显降低系统资源的利用率。

针对这种情况，可以通过资源整合的形式来减少避免。本次任务最根本的目标就是更加有效地利用资源、整合资源，实现不同业务提供统一的服务接口。这样做的好处显而易见，不仅避免了当前系统计算资源的浪费，同时也减少了后续有视频方面需求的新项目中开发人员的工作量，加快开发周期，减少新项目开发、维护等成本。

本次任务的内容也是围绕着能够统一 API 的分布式视频转码系统的设计，实现内部各种服务角色的设计，同时完成分布式任务、优化转码质量的方案。

1.4 论文主要工作以及组织架构

本文完成的工作有以下几部分：首先，分析视频业务需求，确定视频服务的具体需求和应用，同时梳理现有的视频处理系统；第二步，确定即将开发的系统的业务逻辑、架构设计；第三步，分模块进行开发和测试，并最终业务上线，为各业务模块提供统一的视频处理接口。

本文各章节安排如下：

第一章是绪论。主要介绍了问题产生的背景、现状、国内外研究现状，以及通过本文希望实现的目标。

第二章是系统即将采用的技术介绍。包括负载均衡、任务分发、转码概念和技术以及 HDFS 等技术的基本内容。

第三章是需求分析以及方案框架设计。从多个角度总结了需求，进一步按照需求将系统框架大体设计出来。

第四章是方案的详细设计与实现。对于系统的设计原则、架构、详细设计方案等进行全方面的阐述。

第五章是系统的测试。包括功能与性能的测试、采用不同编码方式的优化方案。

第六章是总结全文。总结全文并对即将进行的工作展开讨论。

2 系统相关技术

本章主要介绍与系统有关的关键技术，介绍从基础分布式系统到视频处理功能实现所需要用到的关键技术点，总结起来包括 Gearman、FFmpeg 和 HDFS（Hadoop Distribute File System）等技术。

2.1 分布式系统

分布式系统是目前比较常用的集群处理架构，它能够将工作量比较好的分配给服务器集群中的单个服务器，使得系统负载稳定，能够提供比较大的并发计算量。目前分布式系统一般采用开源架构实现，其中 Gearman 就是其中一个性能较好的开源分布式架构，同时，分布式系统也涉及到的负载均衡技术、热备技术。

2.1.1 分布式系统介绍

分布式系统的一般概念是指基于网络的、由多个相互连接的计算机组成的计算系统，它们在系统的控制之下协作完成同一个任务[1]。一个典型的分布式系统架构如图 2-1 所示。

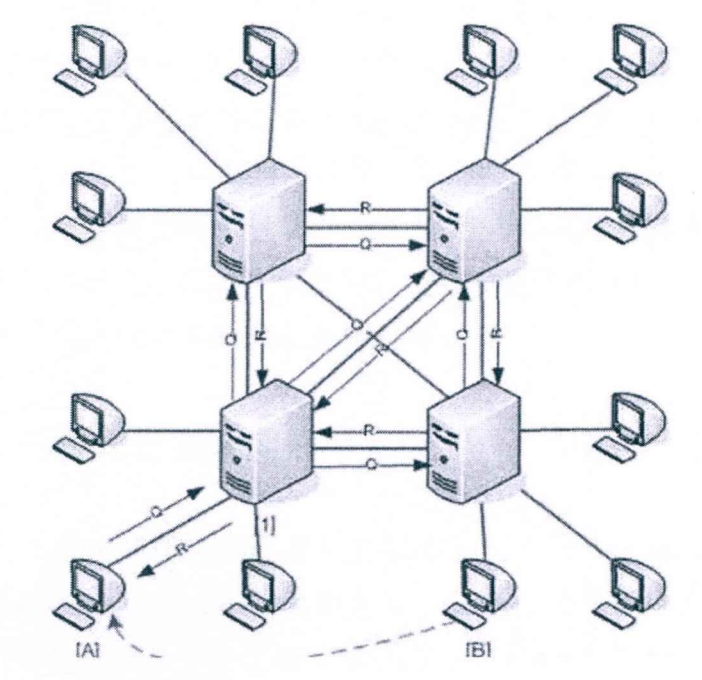

图 2-1 分布式系统架构图

Figure 2-1 Distributed System Architecture

分布式系统主要有两个单元组成：任务控制单元和任务处理单元。任务控制单元主要功能是控制系统的调度以及处理逻辑，将任务分配给任务处理单元，它一般是由一个具有较高性能和稳定性的服务器或服务器组构成的；任务处理单元是用来接收并进行数据处理的计算机或者计算机组，它能够处理控制单元分配到任务，并且返回一定的结果[2][3]。

分布式系统相对于单机系统的优势很明显。首先，它能够允许众多客户端协同工作，众多计算机的唯一目标就是以最快的速度完成一个任务，因此具有很高的计算速度；第二，分布式系统的可靠性较高，通过分发任务将负载分散到任务处理单元的集群中，集群可以并行处理，即使处理单元中的某台计算机宕机，分布式系统也可以通过任务重新派发的功能，使得系统正常工作，在这个过程中只是计算节点减少，损失了部分性能；第三，分布式系统是易于扩展的系统，能够通过轻松的以增加计算节点的方式，来处理日益增加的业务计算量，也可以通过减少计算节点的方式来应对业务量下滑的状况。因此，在集群计算中，经常采用分布式的计算框架，这有助于集群系统更好的应对未来不可知的任务[4]。

在分布式系统的设计中，难点就是如何设计效率更高的分布式通讯系统。一般来说，分布式系统的通讯模型有 RPC（Remote Procedure Call）、MOC（Message-Oriented Communication）、SOC（Stream-Oriented Communication）、MC（Multicast Communication）几种。这些通讯模型中，以 RPC 类应用最为广泛。RPC 即远程过程调用协议，是一种通过网络从远程计算机程序上请求服务，而不需要了解底层网络技术的协议。它基于既有的网络传输协议，一般分为应用层和传输层两层[5][6]。

在这个项目是以大规模视频转码的作为应用场景，视频转码工作对计算系统的性能要求很高，同时需要比较高的可靠性，因此，很难将系统完全架设到一个超高性能的计算机中，使用分布式计算系统的架构就可以满足视频转码系统业务量大、计算量高的应用特点。在本次任务中，将使用基于类 RPC 的一种分布式调度框架，这样能够大大的减少工作量，提高效率。

2.1.2 Gearman 分布式任务调度框架

Gearman 是一个分布式任务调度系统的框架。它支持多种编程语言的混合作业，使得应用程序将工作外包给其他机器去做；它允许进行工作并行执行，同时也支持事件负载均衡处理。是一种比较功能强大、效率较高的分布式调度框架[7]。

按照 Gearman 的框架，任务调度系统分为三种角色：请求发起者 Client，任务调度者 Job Server，任务接收并处理者 Worker。Gearman 官方文档的描述三者之间

的关系如图 2-2 所示。

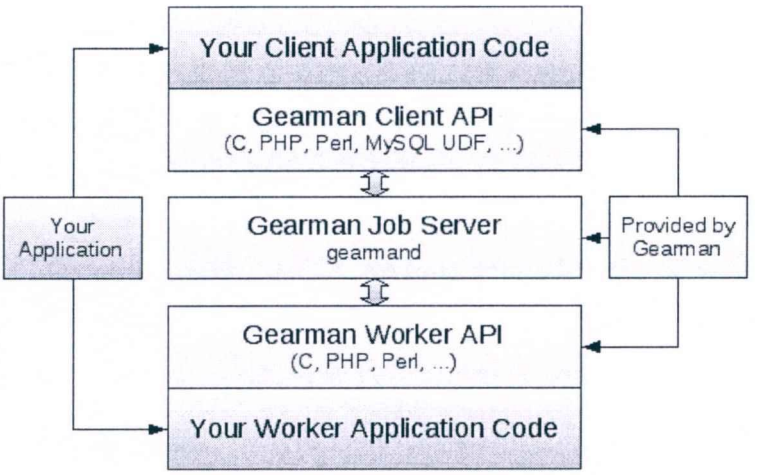

图 2-2 Gearman 角色关系图

Figure 2-2 Gearman Role Relationship

可以看到这三部分的作用[8][9]：

请求发起者 Client，提供对外部的服务 API（Application Process Interface），当外部使用者调用分布式系统的 API 时，Client 就会接收外部请求，创建任务调度的请求提交给 Job Server，再调用 Job Server 的任务调度 API；任务调度者 Job Server，提供分布式系统内部服务调度，当接受到了 Client 的调度请求，就会查询可调度的资源，并按照 worker 使用情况进行调度；任务接收并处理者 Worker，任务的最终处理者，接收 Job Server 的请求，完成处理。这三部分共同工作，直至完成任务。

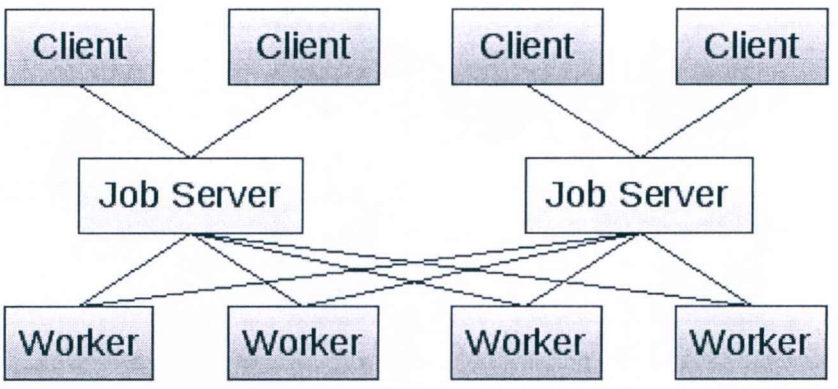

图 2-3 Gearman 工作流程图

Figure 2-3 Gearman Workflow

在系统开发过程中，开发者只需要调用 Client 和编写 Worker 处理程序，并不需要关心调度算法以及各机器的负载情况，Job Server 会自动判断并根据一定的分配算法进行高效分配，节省了大量的开发工作[10][11]。

Gearman 服务创建简单，进程效率高，扩展方便，如果增加计算节点，只需要增加一台服务器到任务调度中心，注册成 worker 即可，这时 job server 会在请求到来的时候，将请求发送给空闲的 worker。

另外，Gearman 的 Client 和 Worker 支持多种编程语言进行开发，均支持 C/Python/PHP/Perl，而且 Client 和 Worker 可以是不同的语言，有助于不同开发者的协同工作。

2.1.3 浮动 IP 与负载均衡

浮动 IP 与负载均衡技术，是分布式系统中常用的技术手段。浮动 IP 的作用是使得某个固定的 IP 能够一直向外提供服务，负载均衡使系统能处理更大的并发量。

（1）浮动 IP 技术

一般来说，高可用的系统中，首先需要保证 IP 是浮动的，继而进行服务器的负载均衡工作。通俗的将，浮动 IP 是一个固定的 IP 可以同时或非同时映射到多台服务器的技术。利用浮动 IP 技术，可以实现对外提供统一的服务接口，防止单系统宕机造成服务不可用[12]。

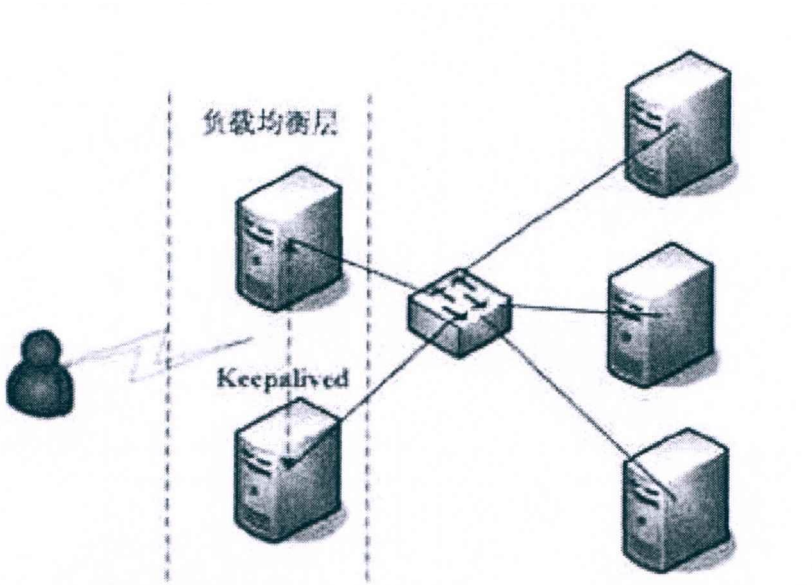

图 2-4 浮动 IP 架构图

Figure 2-4 Virtual IP Architecture

浮动 IP 技术核心是 IP 切换。为了实现顺利切换，需要进行一系列的数据包的同步和转发，架构如图 2-4。

在 live-standby 模式下，在主服务器工作的过程中，从服务器也会同时监听主服务器数据，通过心跳检测来确定是否需要接管主服务器的工作。

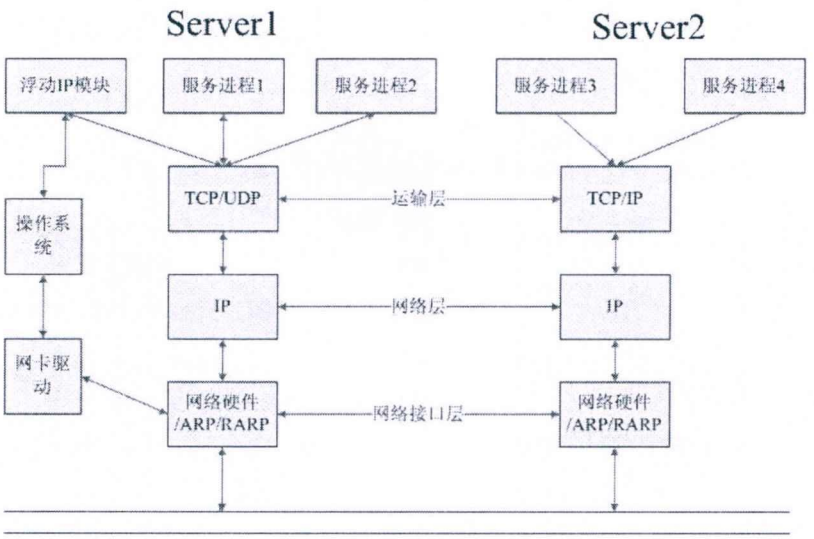

图 2-5 浮动 IP 工作原理图

Figure 2-5 Virtual IP Work Theory

在正常工作情况下，主服务器通过浮动 IP 模块保证数据同步与心跳检测，一旦主服务器宕机，从服务器将会按照主服务器的数据流向，以浮动 IP 所代表的身份返回信息给客户机。

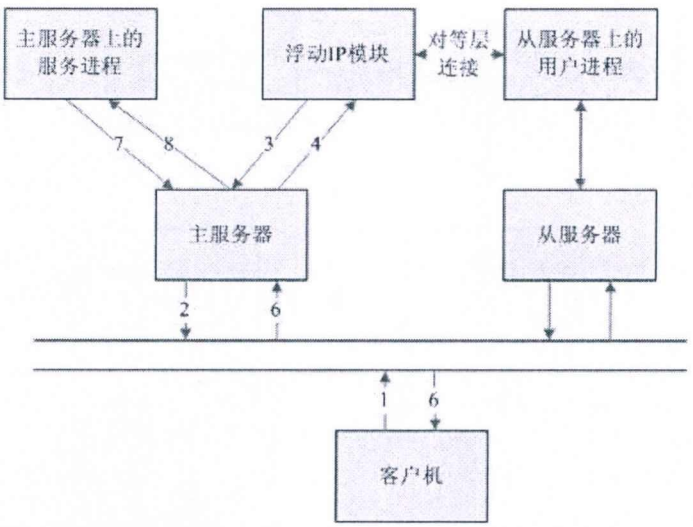

图 2-6 浮动 IP 工作过程中数据包的向量图

Figure 2-6 Virtual IP Data Workflow Vector Theory

（2）负载均衡

在 IP 浮动的基础上，可以设置 Nginx 的负载均衡服务器。Nginx 是一个高性能的 HTTP 服务器和反向代理服务器，同时也经常被用作负载均衡服务器。它占用内存少、并发能力强，越来越多的用户选择用 Nginx 作为网站服务器或者负载均衡器，具体数据 2-7 图，可以看到，Nginx 使用率为 23%以上，在 2012 年的时候使用率为 5%左右，用户增长迅猛[13]。

© W3Techs.com	usage	change since 1 February 2015
1. Apache	58.4%	-0.1%
2. Nginx	23.4%	+0.1%
3. Microsoft-IIS	13.2%	
4. LiteSpeed	2.2%	+0.1%
5. Google Servers	1.3%	

percentages of sites

图 2-7 Nginx 使用率

Figure 2-7 Nginx Usage Rate

Nginx 采用微内核设计，性能较好，抗压能力比较强，因此一般部署在网站或其他对外服务的最前端，用户直接可以与 Nginx 服务器交互，此时 Nginx 能够进行抗压、分流的作用，使得内部服务器集群的负载相对平均，不会造成任务延迟或者大规模阻塞。

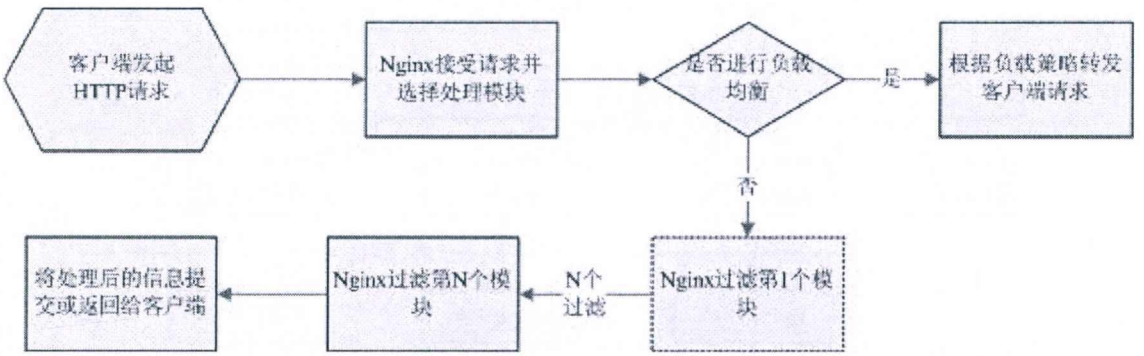

图 2-8 Nginx 请求处理逻辑

Figure 2-8 Nginx Request Process

在 Nginx 架构中，请求的处理逐级往下传递的，如图 2-8 所示，如有需要，可以直接启用某些模块来进行功能配置或者开启限制。通用的模块包括：处理模块、过滤模块、负载均衡模块。处理模块可以用来进行处理逻辑的处理和判断，完成

后返回程序正常的状态信息，若不正常则返回错误信息；过滤模块是相当于配置过滤准则用的，如果处理模块处理后未产生错误，则调用过滤模块，按照准则将不符合要求的信息丢弃；同时可以通过负载均衡模块的设置，指定负载均衡的原则和方法，此时 Nginx 成为双重服务器，对外与用户交互、对内与集群交互[14][15]。

因此，在指定 Nginx 规则时，只需要在配置文件中用 Lua 语言书写处理逻辑，当 HTTP（HyperText Transfer Protocol）服务请求到达时，Nginx 就会判断负载均衡配置，并进行请求转发。

2.2 视频处理技术

视频处理技术是一个广义的概念，主要实现的功能有视频的信息采集、拆分、合并，在本节中除了上述的概念，同时也包括本次设计中选择视频标准的依据和最终选择格式。

2.2.1 FFmpeg

FFmpeg 是一个开源项目，它可以记录、转换音频和视频，并可以将其解码为数据流的程序[16]。

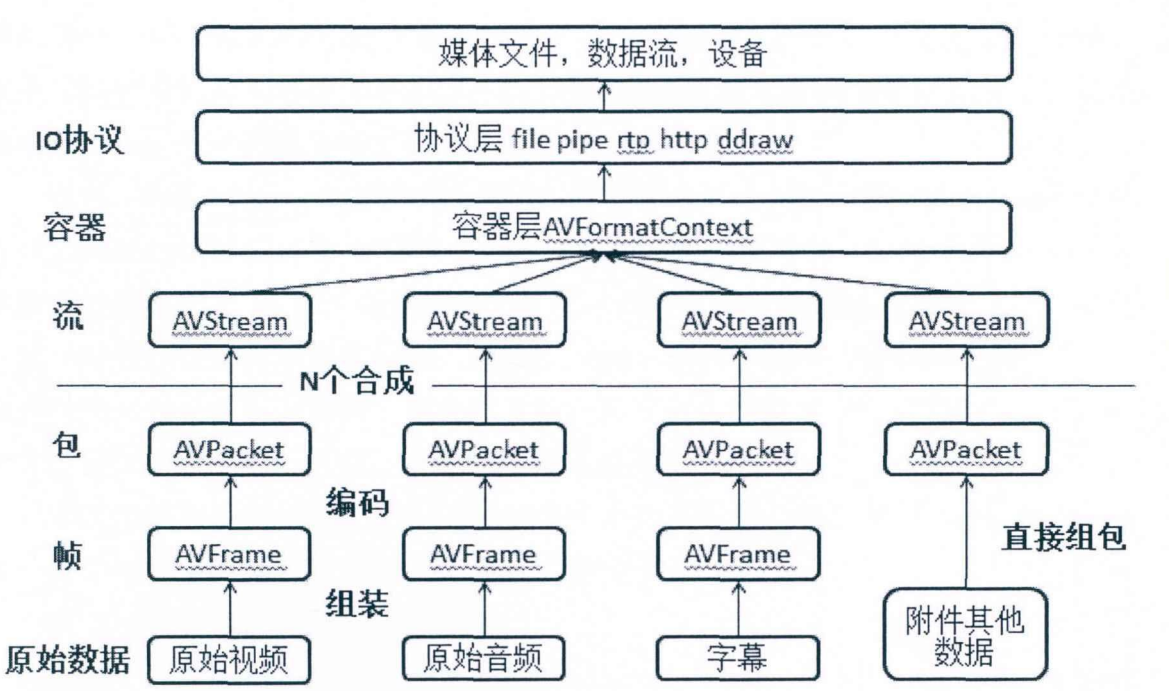

图 2-9 FFmpeg 处理视频内部过程

Figure 2-9 Processes of MMPEG

这个项目最早由 Fabrice Bellard 发起，现在由 Michael Niedermayer 维护。许多 FFmpeg 的开发人员都来自 MPlayer 项目，而且当前 FFmpeg 也是放在 MPlayer 项目组的服务器上。项目的名称来自 MPEG 视频编码标准，前面的"FF"代表"Fast Forward"[17][18]。

FFmpeg 包含许多优秀的库，比如 ibavformat 用于各种音视频封装格式的生成和解析，libavcodec 用于各种类型的音频、图像和视频的编解码[19]。它代码是在 Linux 下纯粹的 C 语言实现的，具有非常高的效率。

可以通过 FFmpeg 工具获取视频信息，进行视频转换，以及进行截图等等操作；同时也可以将视频拆分，进而分项处理。

2.2.2 视频分割与合并

为了让转码系统集群对一个视频同时进行处理，目前的方法是，将大视频分割成若干个小的视频文件，然后提交给不同的 worker 服务器进行处理，使得视频可以被并行处理，这样可以加快视频处理的速度[20]。

但是由于视频文件的特性，又不可以直接将视频文件切断成 N 份，否则可能将视频截断，造成某些视频的不可用。如图 2-10 显示的就是单个完整视频文件的内容。

FFmpeg 可以有效地对视频进行切分。FFmpeg 切分时，以时间为单位对视频进行切分，切分视频时，无须重新地对视频进行编码，因此切分速度相对较快，主要的耗时都耗费在对文件流的 IO 上，而不是耗费在大量的编解码计算上。这是使用 FFmpeg 进行视频文件分割、合并的第二个优势。

通常，采用 FFmpeg 对视频进行切分，有两种方法。一是采用 segment muxer 的方式，将视频切分成多个 ts 分片，同时生成 m3u8 播放列表；一是采用 ss 和 t 为参数将视频按照从 ss 到 t 切分成多个片段。前者，切分视频速度较快，缺点是有一定局限性，仅对有限格式(mp4，mpeg，mpg，mov，ts)系列的视频支持，同时，切分时间片时长固定。后者，切分较灵活，可以自由指定各个分片的时长，但是，切分点必须在 I 帧上，否则，会造成视频出现瑕疵。

本次工作的应用场景，主要是针对 MP4 作为视频源，因此，使用 segment muxer 是一种非常方便、高效的方法[21]。它的切割合并命令如下：

以 20 秒为单位将视频切分成多片：

ffmpeg -i test.mp4 -bsf h264_mp4toannexb -map 0 -c copy -f segment -segment_time 20 -segment_list m3u8 -segment_format mpegts part%5d.ts

合并命令：

ffmpeg -i concat：“part00000.ts|part00001.ts|part00002|part00003.ts” -c copy test_all.ts

可以看到，命令非常简洁，实测速度较快，因此暂时考虑使用 segment muxer 方式进行分割。

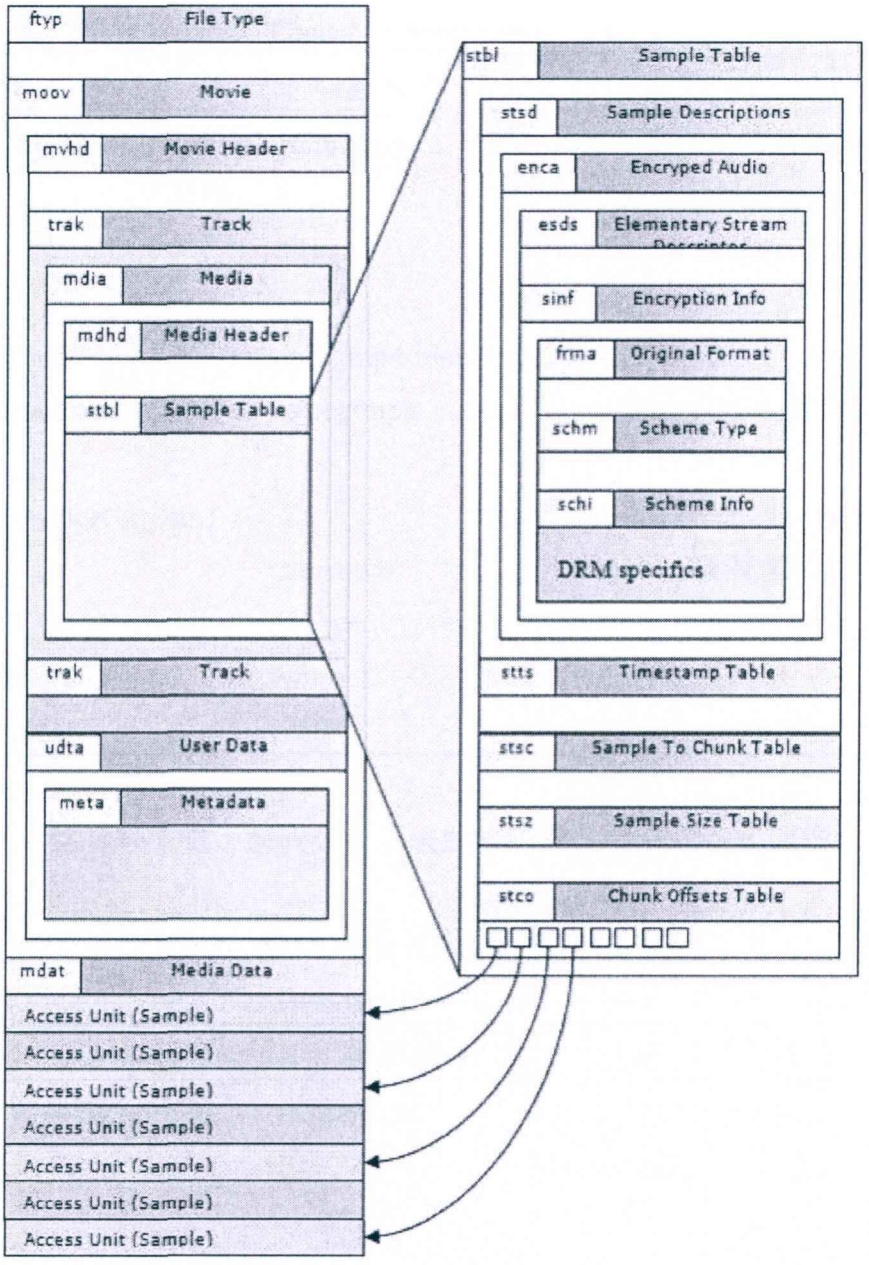

图 2-10 完整视频文件包含的内容（MP4 为例）

Figure 2-10 Video File Contains (MP4)

2.2.3 视频格式标准选择

目前视频的压缩格式标准很多[22]，但是日常生活中经常使用的种类却不是很多。本次设计面向的对象主要有 PC 机、iOS 设备、Android 设备，这些设备支持的主要格式如表 2-1。

表 2-1 常见设备支持格式

Table 1 Common support video format

格式名称	扩展名	说明
Apple HTTP Live Streaming	m3u8/m3u	iOS 支持的视频格式
AVI	Avi	常见视频格式
Flash video	Flv	Macromedia Flash 格式
Matroska	mkv	—
MOV/QuickTime/MP4	mov/mp4/m4a/3gp	—
MPEG-PS (program stream)	mpeg/mpg	也就是 VOB SVCD DVD 格式
MPEG-TS (transport stream)	Ts	transform stream 传输流
RealMedia	rm/rmvb	real media 视频格式
Webm	webm	webm 视频格式
Widows Media Video	wmv	微软视频格式
DVD video object	vob	dvd object 视频格式

在考虑到网络视频带宽、解码效率要求的特殊性，将目标格式调整为：

表 2-2 调整后支持格式

Table 2-2 Supported video format after alter

格式名称	扩展名	说明
Apple HTTP Live Streaming	m3u8/m3u	iOS 支持的视频格式
Flash video	flv	Macromedia Flash 默认的视频格式
MPEG-TS (transport stream)	ts	transform stream，即传输流
MOV/QuickTime/MP4	mp4	—

在上述这几种格式下，我们可以清晰地知道，需要支持的编码格式包括以下几类：

表 2-3 支持的视频编码格式

Table 2-3 Supported the video encoding format

编码格式	说明
h261、h263、h264	ITU-T 和 MPEG 联合推出的编码格式，目前最新版本为 h264，h264 是 mpeg-4 标准 part10 的一部分
divx	mpeg4 压缩格式
xvid	有 divX 发展而来，但是开源
flv1	flv 视频格式
rv10、rm20	real video 编码格式
mpeg1video	mpeg1，常用于 vcd
mpeg2video	mpeg2，常用于 dvd
mpeg4	mpeg4 常用于网络流
wmv1/2	windows media video，微软视频编码

支持的目标音频编码如表 2-4 所示。

表 2-4 支持的音频编码格式

Table 2-4 Supported the voice encoding format

编码格式	说明
aac	高级音频编码器（advanced audio coding），推荐使用，具有比 mp3 更为优质的品质和压缩比，音质接近 cd
mp2、mp3	较为常用的编码格式，绝大多数设备支持，mp2、mp3 都是有损压缩格式
vorbis	有损压缩格式，开源，通常以 ogg 为扩展名，支持多声道
ac3、eac3	杜比数码环绕声，立体音效果良好，支持多声道，有损压缩
flac	自由无损音频压缩方法，通常用于高保真音乐
wmav1、wmav2	微软音频格式，音频未经压缩，不会有失真，但是，通常，音频文件较大

支持的视频容器和音频编码及视频编码组合表的组合很多，具体组合如下表所示[23][24]。

这些格式编码均可以很好地被 FFMPEG 支持，因此使用 FFMPEG 是非常合适的[25]。

表 2-5 系统最终支持的编码格式

Table 2-5 System normally support coding format

容器格式	视频编码	音频编码	说明
mp4	h26X	aac	ok
mp4	h26x	mp3	ok
mp4	h26x	vorbis	ok
mp4	h26x	ac3	ok
ts	h26x	aac	ok
ts	h26x	mp3	ok
ts	h26x	ac3	ok
flv	h26x	aac	ok
flv	h26x	mp3	ok

2.2.4 相关名词解释

（1） 分辨率

视频所成图像的大小或尺寸，单位为 dpi。在成像的两组数字中，前者为图片长度，后者为图片的宽度，两者相乘得出的是图片的像素，长宽比一般为 4：3 和 16：9。

（2） 比特率（码率）

码率就是数据传输时单位时间传送的数据位数，一般用的单位是 kbps 即(kbit/s)千位每秒。基本的算法是：文件体积=时间 X 码率/8 （1B=8bit），在网络上传输的视频想要流畅播放，下载速度也必须大于码率。

（3） 视频帧率

帧率是指，每秒有多少帧。帧率越大，视频就越流畅，画面的连续性越高，但清晰度降低。因为人眼看不出 15 的帧率和 30 的帧率在画面流动差异的。一般美国和日本的为 29.97fps，中国和欧洲为 25fps。

2.3 分布式存储 HDFS

HDFS（Hadoop Distribute File System）起源于 GFS（Google File System），是 Hadoop 架构下的分布式文件系统。它具有较高的吞吐量、容错性、可靠性等优秀的存储性能。用户在使用的时候也比较方便，提供了对外统一且功能强大 API，

使得用户不必关心 HDFS 底层的数据，不用花精力管理存储细节[26]。

HDFS 有以下特点：

（1） 数据存储量大。HDFS 集群设计上能够承受 PB 级别的数据，数据文件具有较大的存储量，因此非常适合 HDFS。

（2） 高效的访问模式。按照“一次写入，多次读取”的高效访问模式，能够具有较好的性能。

（3） 低成本高可靠。HDFS 不同于一般磁盘阵列，它部署在低端服务器集群上，包含了一定数量的冗余节点，HDFS 较好的容错机制，使得系统更加可靠。

（4） HDFS 具有较好的扩展性。可以方便增加或者减少节点，同时允许服务不停止。

HDFS 的以上优秀特性，使得它成为存储大量视频文件非常合适的架构，能够兼顾性能与成本[27]。

HDFS 的系统结构很清晰，如图 2-11 和图 2-12 所示。HDFS 属于 Master/Slave 结构，由 NameNode（NN）和 DataNode（DN）组成，一般一个 NN 对应多个 DN，还可增加 SecondaryNameNode（SNN），保证系统可靠性。

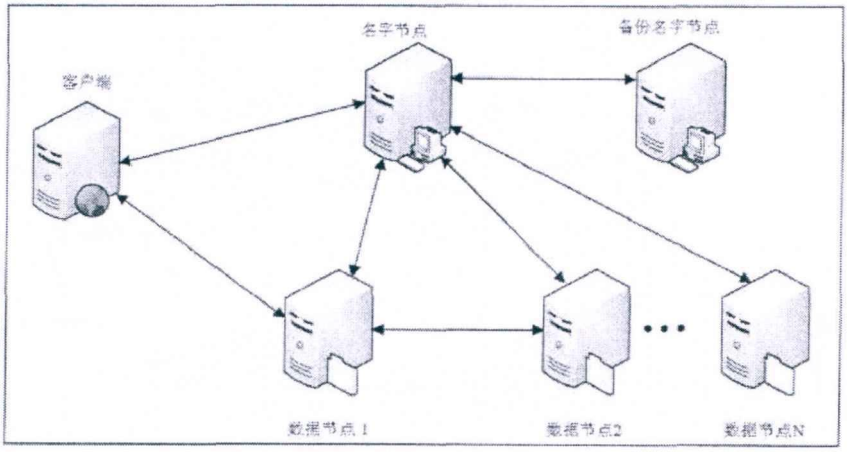

图 2-11 HDFS 物理连接图

Figure 2-11 HDFS Physical Connection

在这个架构中[28]，NameNode 是集群中的核心节点，它的主要作用就是存放 Hadoop 集群中存储元数据，同时完成对整个系统的控制；Secondary NameNode 作为 NameNode 的备份节点而存在，主要作用就是备份 NameNode 上的所有数据；DataNode 是真正实现数据存储的节点群，在 NameNode 节点的控制下进行数据的读写和差错控制处理；DFSClient 的作用就是允许用户读写操作 HDFS 上存储的数

据文件。

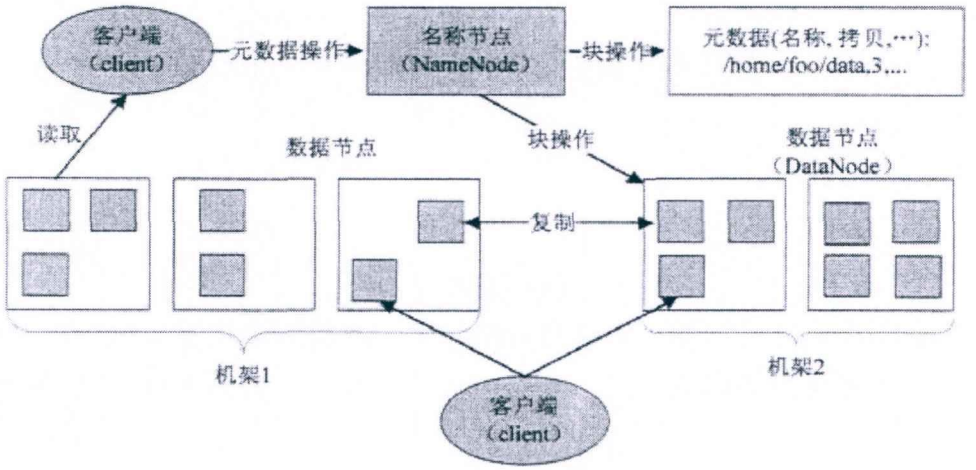

图 2-12 HDFS 体系架构图

Figure 2-12 HDFS Architecture

HDFS 中，默认的 Block 为 64MB，一个文件所对应的 Block 全部按照预定的策略存于 DN 中，当客户端进行块读取操作时，则可以从 NN 上请求数据，NN 将数据块位置返回给客户端，客户端可以通过这个块信息对数据进行操作，简化交互过程。

在本项目中，最常用的方式是“一次上传，多次读取”，因此 HDFS 优秀的读取性能，非常适合本项目的设计。

2.4 本章小结

本章研究了分布式系统相关知识、负载均衡的相关技术、视频技术的相关知识以及 HDFS 的相关内容。

第一部分介绍了分布式架构的基本原理和 Gearman 分布式框架。分布式处理时本次设计的重点内容，接下来会在 Gearman 分布式框架基础上，制定出比较好的处理逻辑，为视频转码提供较好的基础架构第二部分涉及到浮动 IP 和负载均衡技术。一般来说，负载均衡技术的需要搭配浮动 IP 技术，使得系统可以对外提供统一的 API。第三部分介绍了部分视频的知识和视频处理的工具。视频本身比较复杂，借助工具，就能够将视频处理简单化。第四部分介绍了 HDFS 的相关内容，分布式视频转码的系统是基于 HDFS 的，提供了分布式处理的可能性。

3 分布式视频处理系统方案设计

本章主要内容是对目前具有的需求进行汇总和分析，得到详细的需求内容，将得到的需求与现有技术框架相结合，从而得到系统整体的进一步设计，为详细设计打下基础。。

3.1 需求分析

本次设计的主要工作是提供一个视频处理的统一接口，但是由于各工作组的需求不同，因此需要彻底分析各组的具体需求，然后整理各组需求，为系统设计提供依据。

3.1.1 分布式视频处理系统总体分析

在比较成熟的网站体系中，各个功能模块是相互独立的。开发也是相对独立的，因此当前情况是，前端开发人员需要制定特定的视频处理服务器，同时要求编辑必须上传制定格式的视频文件，这就造成了很大的不方便；同时，每个项目组都会有不同的视频需求，因此，开发成本巨大。

一般来说，视频由编辑上传，经过转码处理，生成若干个匹配不同终端的链接，用户直接可以通过链接访问视频。从这个角度来说，其实编辑和前端开发人员没有必要关注视频处理细节，只需要的是能够视频转码请求，返回视频链接，将链接呈献给最终用户。那么，这个过程简化为：编辑提交视频文件和格式需求到视频处理系统，视频处理系统返回 URL，供用户访问。

图 3-1 系统工作宏观过程

Figure 3-1 System Work Process

当精确到某个节点时，可以进行如下描述：

（1） “产品运营人员”提交视频到“转码调度”模块

（2） “转码调度”把转码任务分配给各个“视频转换”服务

（3） “视频转换”模块按指定的“视频编码参数”对视频进行转码

（4） “视频转换”模块完成转码后，把视频存储到“视频存储”模块

（5） “视频存储”模块完成视频的存储后，生成“视频播放链接”

（6） “产品运营人员”获取“视频播放链接”(如果转换或者存储失败，则获取到失败信息)

（7） “最终用户”通过“视频播放链接”访问“视频播放”模块，点播视频

在这个过程中，视频上传者也可以认为是用户（非最终用户，指产品运营人员），用户角色的用例图如图 3-2 所示。

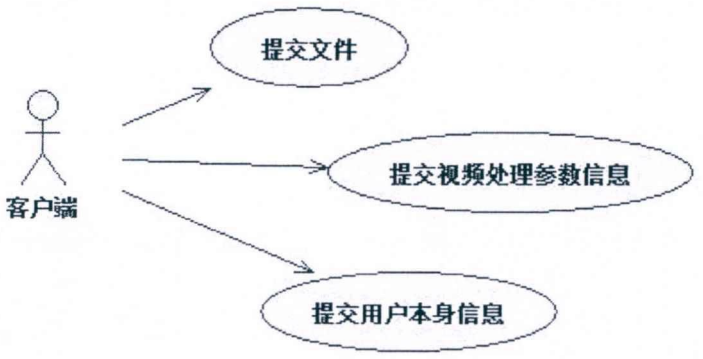

图 3-2 客户端用例图

Figure 3-2 Client Use Case

视频处理系统，作为服务端，它的用例图为

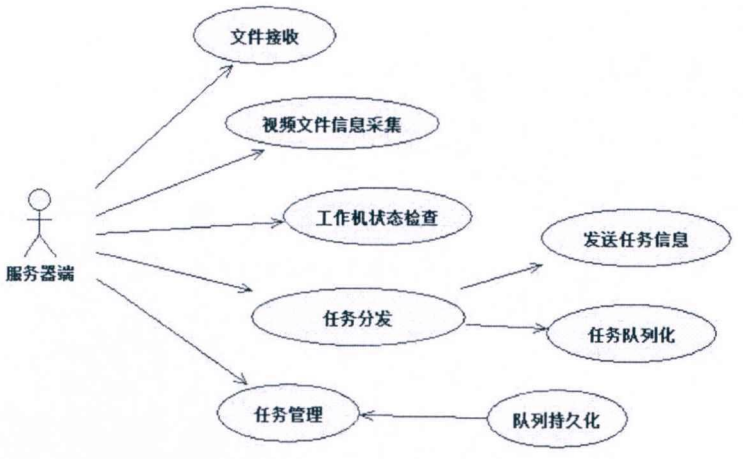

图 3-3 服务器端用例图

Figure 3-3 Server Use Case

在这个系统中，用户作为上传数据的一方，需要提供的资源有：视频源文件和含转码参数的 xml（或 json）任务信息。用户需要上传视频源文件，并提供上传后文件的位置信息，同时按照一定方式配置生成 xml 文件，上传至转码系统后，系统通过 xml 中标签获取转码所需的参数，当所需参数获取完毕时，提交转码操作请求。

在服务端，系统经过解码，提交转码操作后，在内部就会开始对视频进行预处理分析，继而根据不同的业务类型进行后续操作：判断是否需要分片（将视频分割成小片）、转码是否有特殊要求、是否有空闲服务器、转码队列状况判断，最后提交转码任务，经过任务分发，将任务分发至处理服务器，等待直至完成转码任务并返回标示信息，完成转码任务。

整个系统可以分为若干功能点，包括上传视频、截图、转码等，具体如图 3-4 所示。

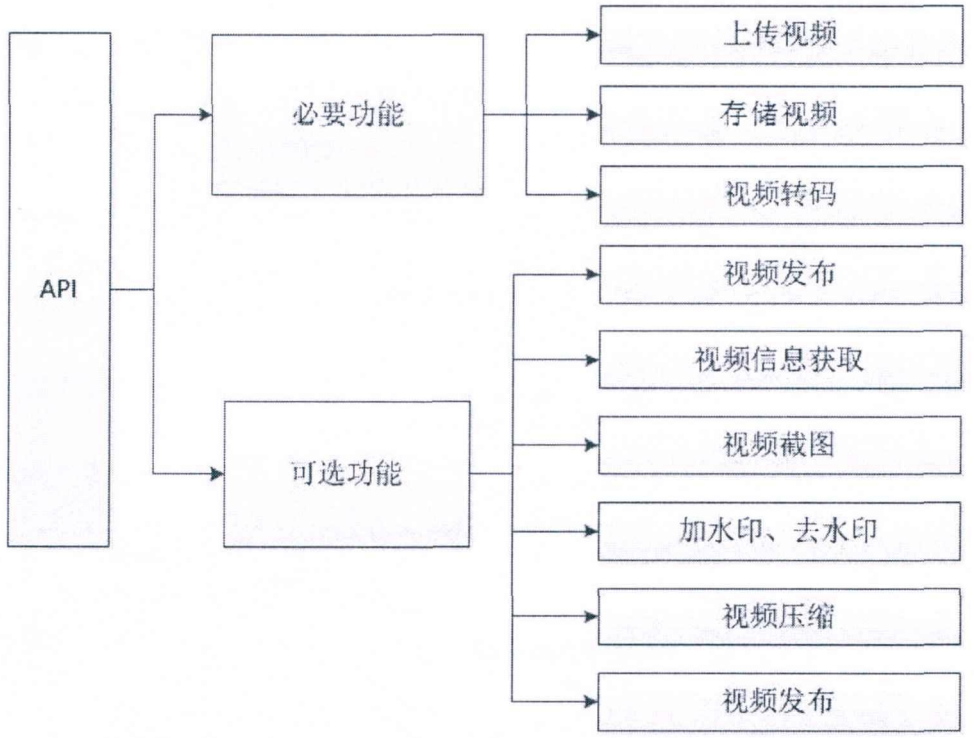

图 3-4 平台整体功能图

Figure 3-4 Platform Entirety Function

3.1.2 转码系统需求分析

视频转码需求较多，抽象成系统功能，可以概括为如下几点：

（1） 获取视频元信息；

（2） 获取视频缩略图，并返回给视频上传者；

（3） 视频转码操作，提供三种方式：采用流式转码，视频内容边转码边返回给用户；非流式转码，视频内容在服务端转码完成后返回给用户，适用于客户端不支持 http chunked encoding 或视频转码后需要后处理的情况；视频转码内容输出到 HDFS 目标桶。

（4） 对于转码操作，以 ECU（Elastic Compute Unit）为单位提供资源隔离机制，同时保障产品间资源隔离和产品内部 pipeline 资源隔离；

（5） 对于转码操作，提供超售机制增加系统整体资源利用率；

（6） 转码操作，保存历史信息，方便用户查询；不提供暂停操作；

（7） 用户可指定 timeout 时间，超过时间仍未完成转码则直接返回失败；

在具体实现上，可以设计如下主要模块：

设计的转码系统服务器端的包括预处理模块（Razer）、任务分发模块（Job-server）、工作模块（Worker）。这三个模块提供所有类型的转码服务。这三部分的用例如下。

预处理模块（Razer）用例图：

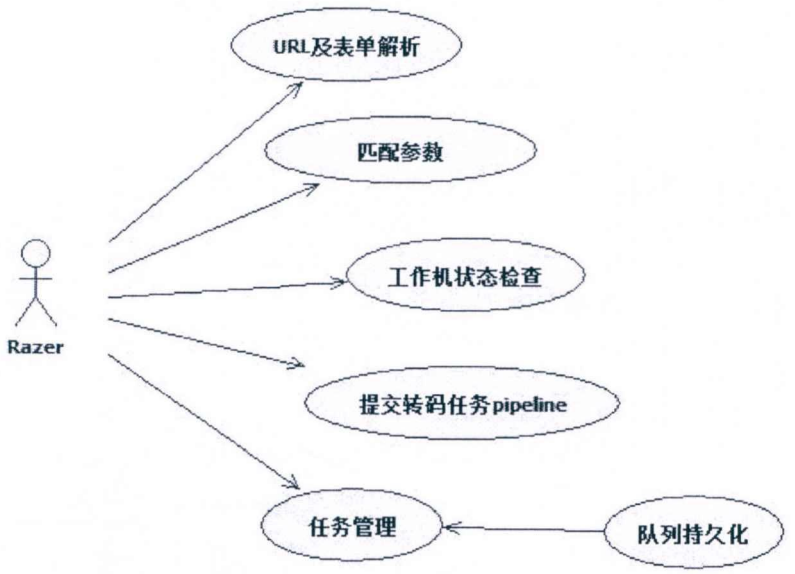

图 3-5 预处理模块模块用例图

Figure 3-5 Razer Module use case

任务分发模块（Job-server）用例图：

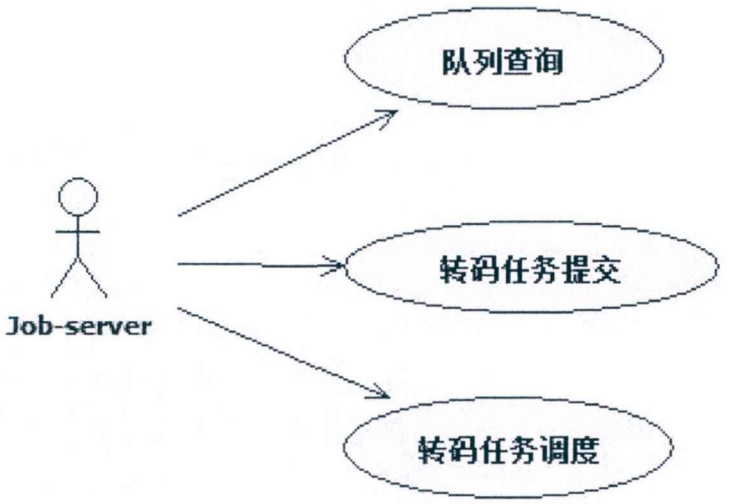

图 3-6 任务分发模块用例图

Figure 3-6 Job-server Module use case

工作模块（Worker）用例图：

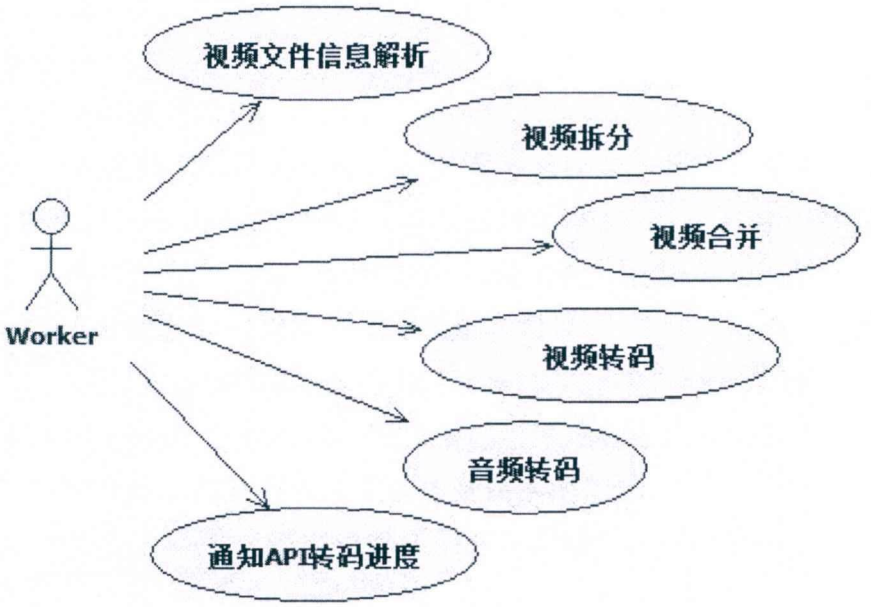

图 3-7 工作模块模块用例图

Figure 3-7 Worker Module use case

3.1.3 视频处理系统开发所需要解决的问题

视频转码在提供转码功能的同时，需要考虑到后续开发调用以及上线运维、调度流程，综合分析，问题分解为以下 6 点：

（1） 系统通信级别上，转码系统涉及后台 web 开发与编码系统的接口，也涉及到集群调度服务，因此，需要设计接口形式。

（2） 系统转码具体工作上，设计到可靠的解码、编码、转码，需要确定通过何种方式。

（3） 编码涉及到的存储服务，由于服务器处于不同机房，各机房之间距离较远，需要考虑 HDFS 性能的基础上设计和搭建。并且涉及到 CDN（内容缓存节点）溯源问题，要充分考虑 HDFS 的连通方式，否则 CDN 溯源较慢，即使覆盖广也会对用户体验有直接的影响

（4） 兼容性，要能处理各种各样的源视频，不同格式的片源，不同码率的片源如何输出对应清晰度版本的片子，需要经验和规则；标清高清超清的编码用怎样的指令去优化得到最佳清晰度，同时兼顾编码的时间成本。

（5） 编码层面是否需要调优：有实时性要求的，要考虑单机处理大视频的能力是否满足业务需求；有高清要求的，要考虑转码系统是否可以切更多的分片同时转码。需要进行集群性能调优。

（6） 具有加水印、去水印、去黑边、去台标、合成字幕、多音轨优化等功能，这是线上常用的。

与之对应的解决思路有四点：

（1） 第一个是系统层面上的问题，也是业务合作上的问题，考虑到交互过程中，以网站开发人员居多，因此对外可以采用 HTTP REST 的方式对外提供接口；内部来说，集群的调度的通信与分布式系统有具体关系，从底层开始设计成本较高，可以考虑以现有开源分布式架构为基础来进行设计。

（2） 第二个问题关于转码系统实现细节，考虑到系统开发周期短、性能要求高的情况，考虑用 python 进行连接，可以很好地适用分布式系统和已有转码技术，可以以直接调用开源工具的方式或者调用开源库的形式。

（3） 关于兼容性，要看上线后业务复杂度，编码工具可以提供的功能全可以实验性的增加。

（4） 其他问题，如果能够通过编码直接满足可以直接满足，否则考虑粘合多种开源工具，完成系统的搭建。

3.2 方案设计

本章根据已经汇总的需求分析，对系统进行整体的设计，需要考虑的是系统

的功能、性能，以及如何利用分布式架构将系统设计的更加合理，从而设计更合适的架构。

3.2.1 分布式视频处理系统设计原则

在系统开发中，需要指定必要的原则，保证系统的稳定、及可持续，并有比较好的前瞻性。因此，设计的原则如下：

（1） 集群负载稳定，任务均匀分发，能充分利用计算资源

转码系统是由不同配置的服务器组成，有比较新、性能比较强的服务器，也需要整合当前正在运行的服务器，同时希望能够将性能较弱的服务器也添加进去，成为计算池的一部分，尽可能最大量的扩充计算资源池。同时，根据不同服务器性能分发资源，使得每台服务器充分利用，又不至于有较长队列产生。使得集群负载较为合理，不至于造成性能故障。

（2） 具有较好的容错机制，转码任务不受个别服务器宕机影响，能够快速自动重转。

由于系统将会有很多服务器组成，也充分利用旧服务器，因此系统中单台机器发生故障的可能性比较大。必须使得系统具有一定的容错机制，使得某节点故障造成的影响最小，不至于影响整个系统的稳定服务。

（3） 对外部用户来说，服务是透明的，不可见处理细节，简化系统

尽可能的简化 API，调用的时候，只需要提供文件和响应的参数，输出则为指定的结果。使得用户调用时候能够快速返回，不需要过多的了解细节。

优化转码系统网络流量，支持边转边提交，边转边播放。

集群系统需要优化网络的流量，使得内部流量均匀，外部访问时速度稳定，同时支持优先转码视频前部分，支持一边转一边播放。

（4） 保证转码后是视频的品质。

使用工具转码时，要保证视频的品质不会下降的非常明显，同时保证视频音频同步。

（5） 多终端支持。

由于不同系统支持视频格式不同，因此，转码后的视频需要考虑支持各种终端播放，包括 x86 平台，android，iOS 等。

（6） 逻辑简洁、界面友好。

实现逻辑简单，系统结构、代码实现等方面简洁，且具有好的可维护性。同时保持 API 友好，用户使用方便，运维方便。

3.2.2 系统架构设计

本系统的架构分为五个角色：负载接入模块（API），视频预处理模块（Razer）、视频任务分发模块（Job-server）、视频处理模块（Worker），另外还有数据库（RDS）、HDFS 以及流媒体服务器。

在这个 5 层的平台中，API 负责抓取任务，分配服务器（Razer）负责资源隔离与任务创建，Job-server 主要负责工作分发，worker 负责具体的转码等视频处理工作，RDS 负责转码过程中的持久化存储。整个系统的文件存储和交互是基于 HDFS。具体架构如图 3-8 所示。

第一层，最前面的是 API，在 API 服务器上部署 Nginx 服务，达到复杂均衡的作用，API 服务器通过双千兆以太网连接至冗余的 razer，保证其高可用。一半 API 服务器的将会接收收到的文件，对收到的文件进行信息提取，然后推送至 HDFS 上，然后发起转码请求。转码请求的参数为 XML 或者 JSON 格式。

第二层，是 razer 服务器，对信息进行预处理，进行资源整合、闲置资源判断等，进而根据服务器性能和负载进行创建任务资源。这个角色是集群中唯一的，这样设计可以保证视频处理任务的唯一性，保证各个 pipeline 不会发生冲突或者混乱，保证转码安全[29]。

第三层是 Job-server，接收到任务之后，分解为若干个作业，打散分配给不同的 worker 服务器。Job-server 的角色是承担 pipeline 的载体，每个 pipeline 只会交送给一个 job-server，此时，pipeline 中包含的任务就会在某个较小的服务器范围内进行处理，防止 pipeline 延迟过大、环节错误以及数据丢失。

第四层是 worker，启动后自动向 Job-server 注册，接收到转码任务，进行转码，分为视频转码任务、音频转码任务等。Worker 承担所有的具体工作，因此数量上应为最多，同时性能应为最高，这样才能保证有足够的 worker 来进行工作。同时并不需要 worker 具有很高的可靠性。在部署的时候应该注意 worker 与 job-server 之间的通信开销较小，不应有较大的延迟，这样能保证处理速度。

最后返回给 razer，razer 调用 merge 模块，将视频合并。期间 RDS 作为提供队列持久化存储的设备，HDFS 作为文件共享的基础，贯穿整个系统中。

用户上传的视频通过 API 服务器中 nginx 的 upload 模块保存到临时文件夹，然后调用 web.py 处理上传的视频（截图，读取信息等）并存入 HDFS，之后给 razer 发送信息，razer 根据不同任务来判断发送给 Job-server，之后通过 Job-server 的 gearman 下发转码任务到编码机器（worker）。编码完成之后会通过 rsync 推送到源站。转码的历史记录会保存在 RDS 中，转码信息存在 video 数据库中，按照视频 id 查找即可。

根据架构实现的硬件拓扑如图 3-9 所示。

视频经过提交，经过 Razer 转发，再经过 Job-server 的分配，最终进行音频、视频编码格式的转换。在系统设计中，需要考虑到分布式任务分发，也同时兼顾 worker 的处理逻辑。

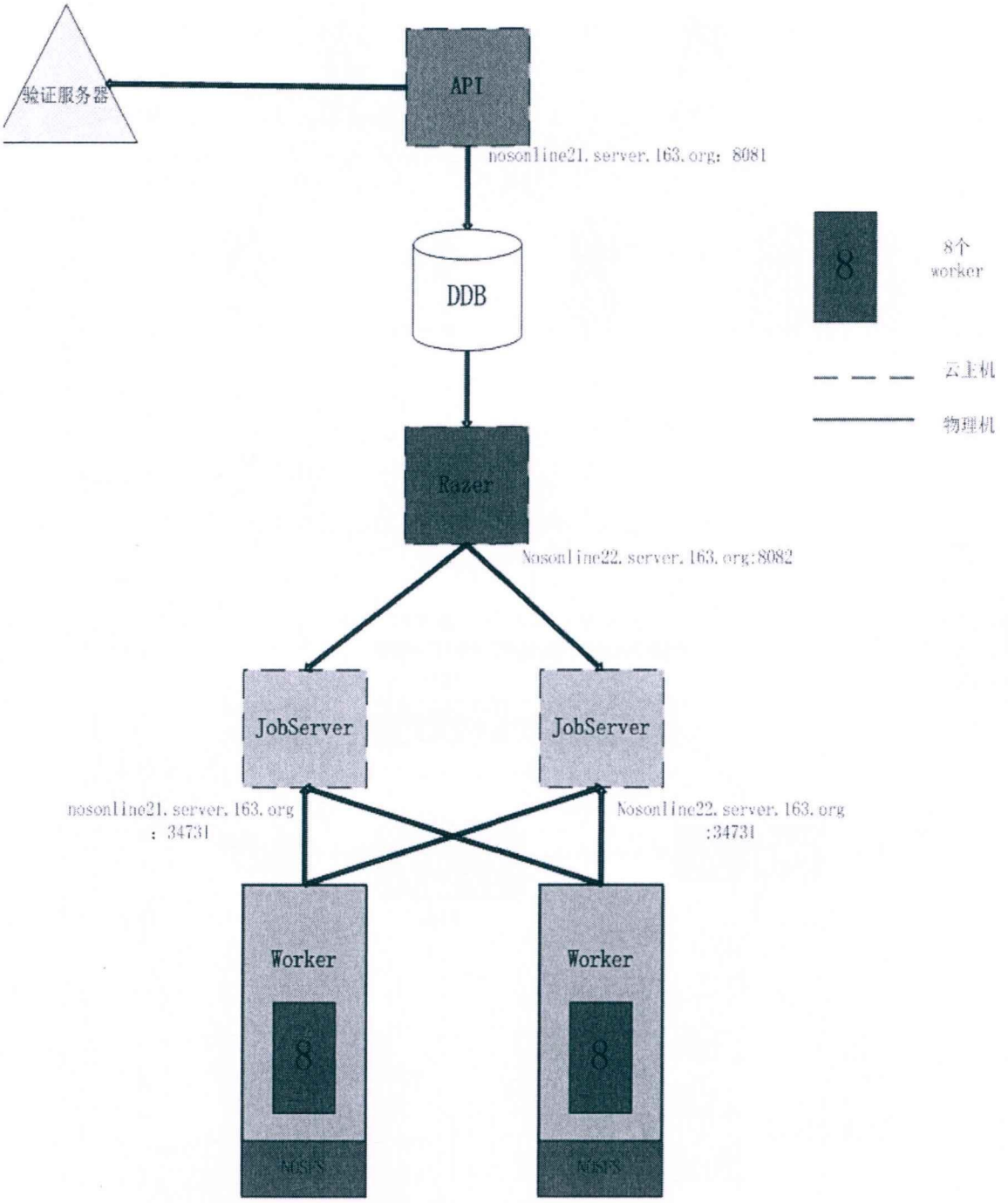

图 3-8 系统架构图

Figure 3-8 System Architecture

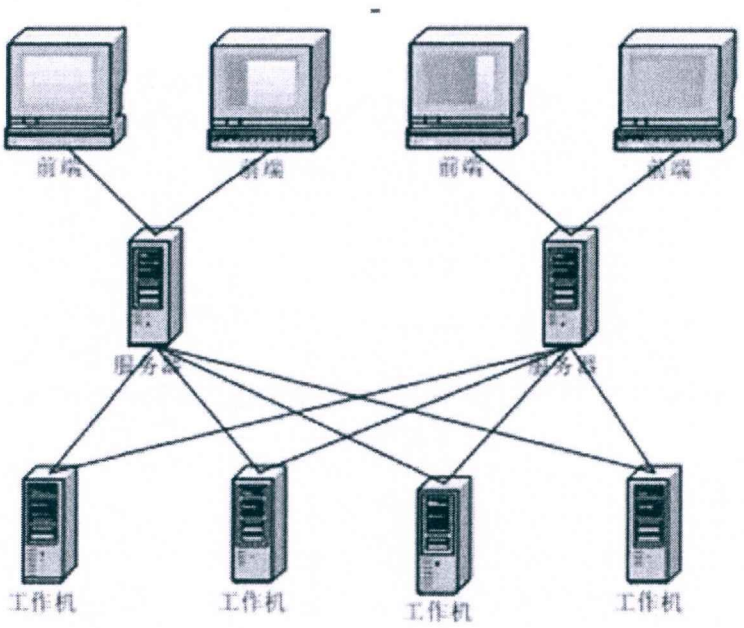

图 3-9 硬件组网示意图

Figure 3-9 The Hardware Network

从用户原片的角度来看，对视频文件是经过如图 3-10 处理：

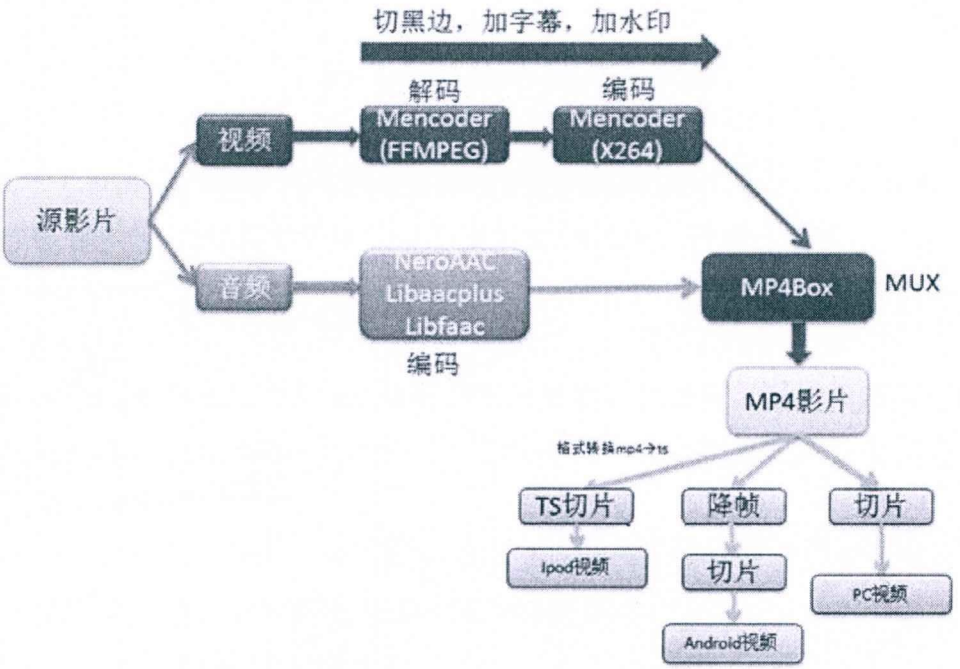

图 3-10 视频文件处理过程

Figure 3-10 Video File Process

3.3 系统类图

平台整体类图如 3-11 所示。

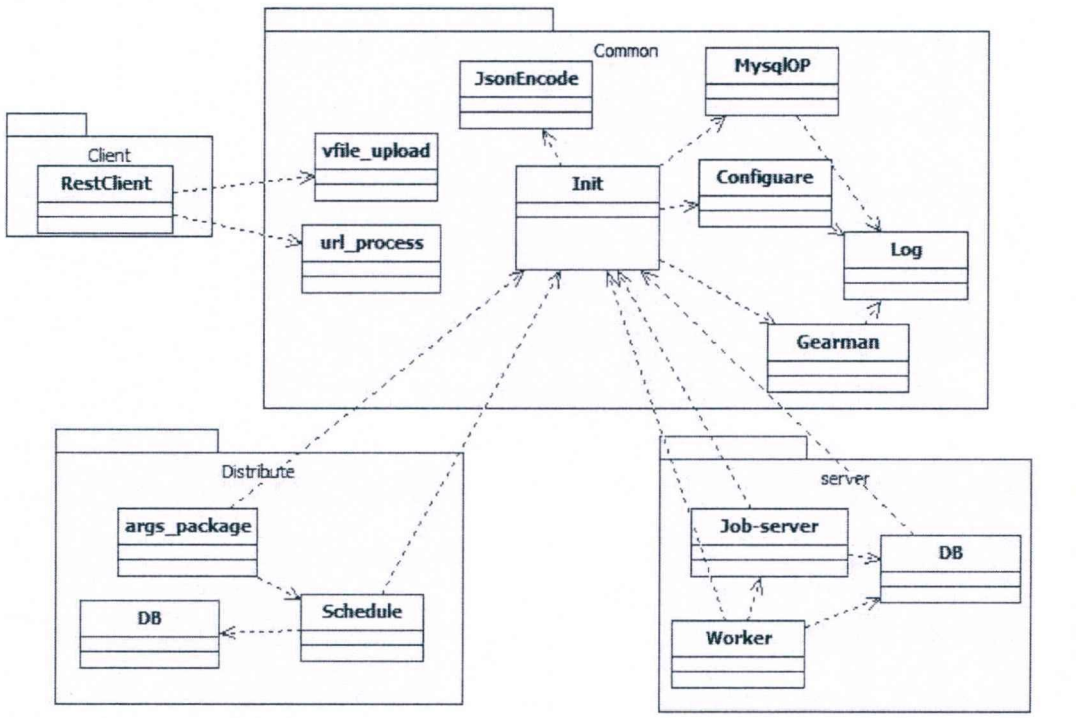

图 3-11 系统的类交互图

Figure 3-11 System Class Interaction

其中 Client 包为客户端代码，主要用于生成 rest 请求，提交给服务器。Common 包为服务器端包，主要用于接收客户端提交的任务，并根据策略，转发给相应的 Distribute Server，它也提供了一些工具类，包括文件上传接收、数据库操作、JSON 编解码等。

Distribute 主要是针对从 url 解析出来的参数进行处理，根据指定的策略，将文件和参数打包成 pipeline，提交给后续处理模块，这个部分主要是根据需求制定的定制化操作和要求。

Server 包括 Job-server、worker 和 DB，Job-server 接收请求后按照 pipeline 按照策略转发给 worker。同时都会在 DB 进行操作。

3.4 本章小结

本章主要内容是系统的需求分析以及整体设计。主要包括汇总用户视频需求、

转码需求，在这个基础上，设计系统的整体架构，设计视频文件的处理过程，同时设计系统整体的类图，定义类的名称，同时定义类之间的交互，从而为视频处理系统的下一步工作提供了思路。

4 系统详细设计与实现

本章的内容是对系统整体理解上进行详细设计，首先需要对方案进行总体的梳理，进而设计和实现模块，设计和实现主要包括各个部分的模块的设计、模块间功能交互等。

4.1 方案

本系统已经包含的功能 3 章已经确定，具体的功能流程如图 4-1。

首先，系统等待用户提交数据，发现有数据提交，则 API 获取提交的文件和参数，然后对文件完整性、参数有效性进行校验，如果校验不通过，那么则向客户端返回消息，提示检查提交的内容，然后重新提交。

如果校验通过，那么则进行下一步，将接收到的文件上传至 HDFS，添加相关的数据库记录，然后创建请求，提交给 Razer 服务器。

Razer 服务器接收到请求信息后，提交一个获取文件信息的请求，得到返回到文件信息后，Razer 根据返回值以及制定的策略，来决定下一步的做法：如果文件比较小，转码需求简单，那么直接查询对应的转码策略，提交转码任务给 Job-server；如果文件比较大，那么首先对文件进行切分处理，将文件切分成若干个小段，然后验证小段视频的有效性，保证视频有效的情况下，将每一小段视频作为源视频文件，创建 pipeline 提交给选定的 Job-server。

某个 Job-server 收到请求后，查询 worker 情况，如果有空闲 worker，则将请求提交至 worker；如果 worker 不为空，那么把任务或者 pipeline 插入到队列尾部，等待队列顺序执行。

当 worker 将提交的转码工作做完，就会向数据库写入状态，同时 API 发送消息，API 向客户端发送消息，提示客户端完成。此时客户端只需要按照 worker 返回的信息直接读取文件即可。

整个系统流程图如图 4-1 所示。

4.2 负载接入模块（API）

本节主要是对负载接入模块，即 API 进行详细的功能描述以及逻辑设计，同时进行 API 模块其他细节设计，API 模块是系统的第一关，需要保证具有较高的网络性能和 IO 性能。

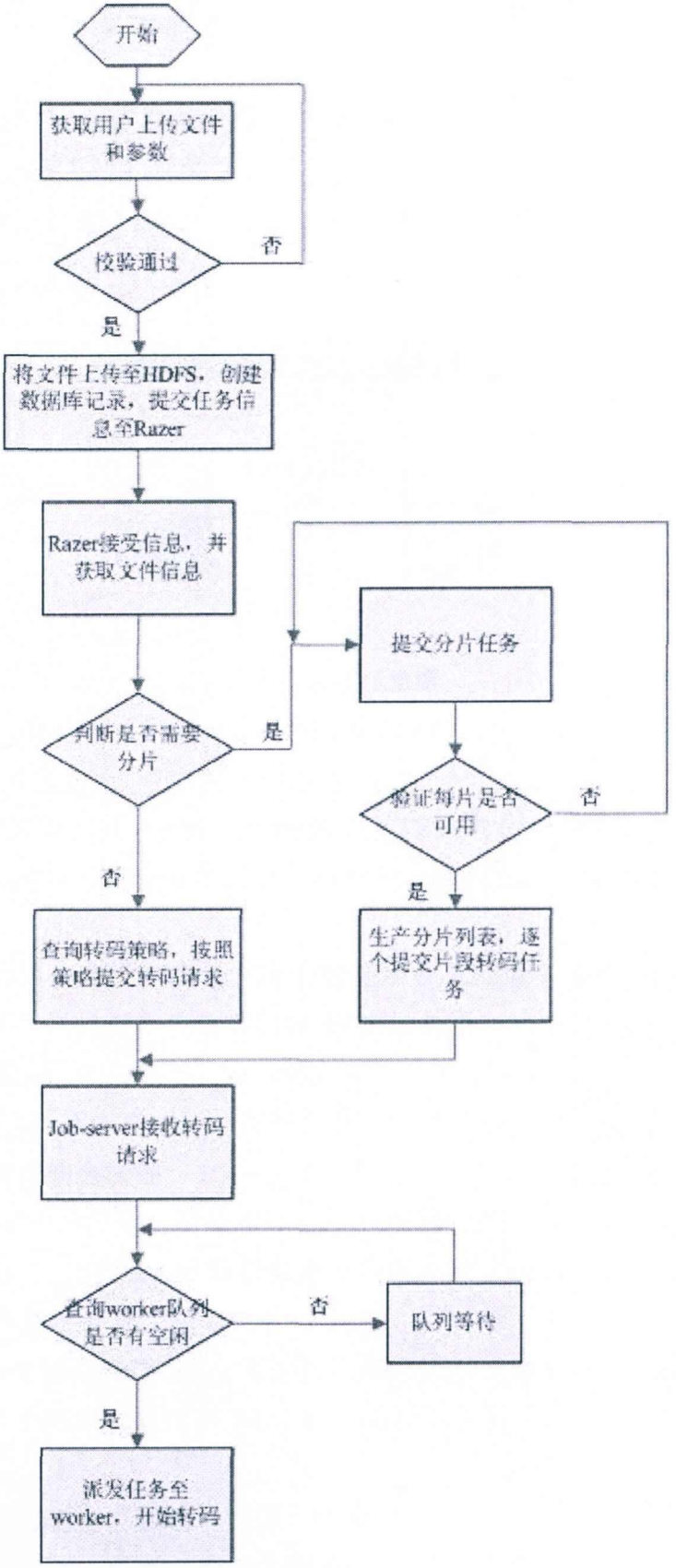

图 4-1 系统流程图

Figure 4-1 System workflow

4.2.1 具负载接入模块功能设计

接入模块（API）主要功能是实时判断服务器负载的软件，结合负载均衡（Nginx）模块，验证上传的视频文件并上传至 HDFS。由于负载接入模块比较重要，不应该存在单点故障，因此使用 keepalive 将 API 模块设计为冗余架构，保证其高可用。

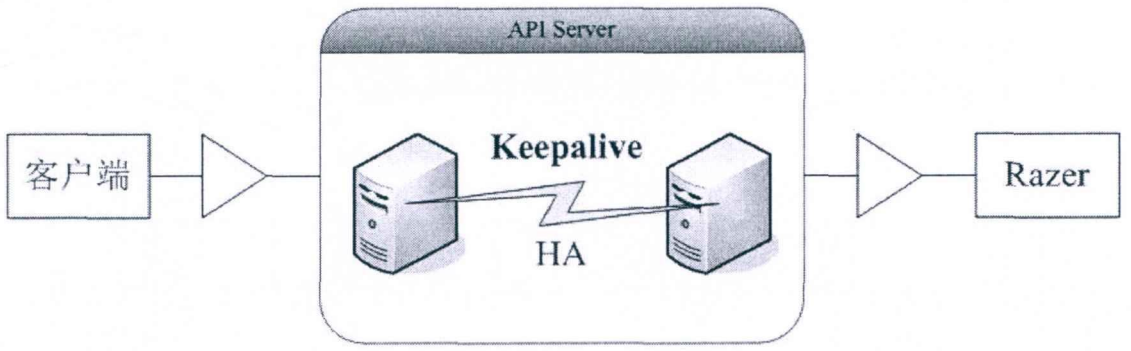

图 4-2 冗余 API 示意图

Figure 4-2 Redundance API

API 是掌控着接入系统，API 进程提供了 NTS 对外服务接口，包括 Task 的创建、删除、更新、查询。API 进程提供了两种用户接入方式：白名单和用户签名。具体是：

（1）API 服务器会维护一个 ip 列表，包含所有 API 服务器信任的客户端的 IP 地址，白名单认证就是检查用户请求的源 ip 地址是否在这个信任列表中。如果列表中不包含请求源 ip，则白名单认证失败；

（2）用户签名认证依托于云计算平台统一的认证服务器，采用对称密钥加密的方式检查用户身份的合法性

外部用户同 API 的交互主要是通过 http 连接进行的，建立的都是短连接。连接的超时时间由用户自定义。API 服务器通过白名单和认证服务器对外部用户访问 NTS 系统的合法性进行检查。

此时，API 服务器与 RDS 的交互由客户端的访问行为决定，一般而言，客户端的每个请求，都会转换为 API 和 RDS 的一到若干次交互，这些交互主要包括以下行为：

（1）pipeline 的创建、删除、更新、查询；

（2）job 表的创建、更新、查询、删除

（3）output 表的创建、更新、查询、删除

（4）user 表的创建、更新、查询、删除

4.2.2 API 工作过程

首先，客户端向 API 提交编码请求。此时，客户端是以 POST 的方式提交表单和上传文件，目标是 API 的浮动 IP。提交的表单内容包括适配类型（默认所有设备）、画质、优先级、校验、用户签名等信息。

API 接收到上传的请求后，浮动 IP 选定 API 服务器组中的某台服务器，将接收到的内容接入至 Nginx 服务器，首先通过 nginx 的鉴权规则脚本，来鉴定是否允许上传。接下来，确定上传目录。上传文件完毕后，将文件的网络地址同时提交给 API，同时创建 RDS 中 VideoFile 的记录，将环境与文件本身属性添加进去，创建 VideoTask 记录，将文件转码属性添加进去。

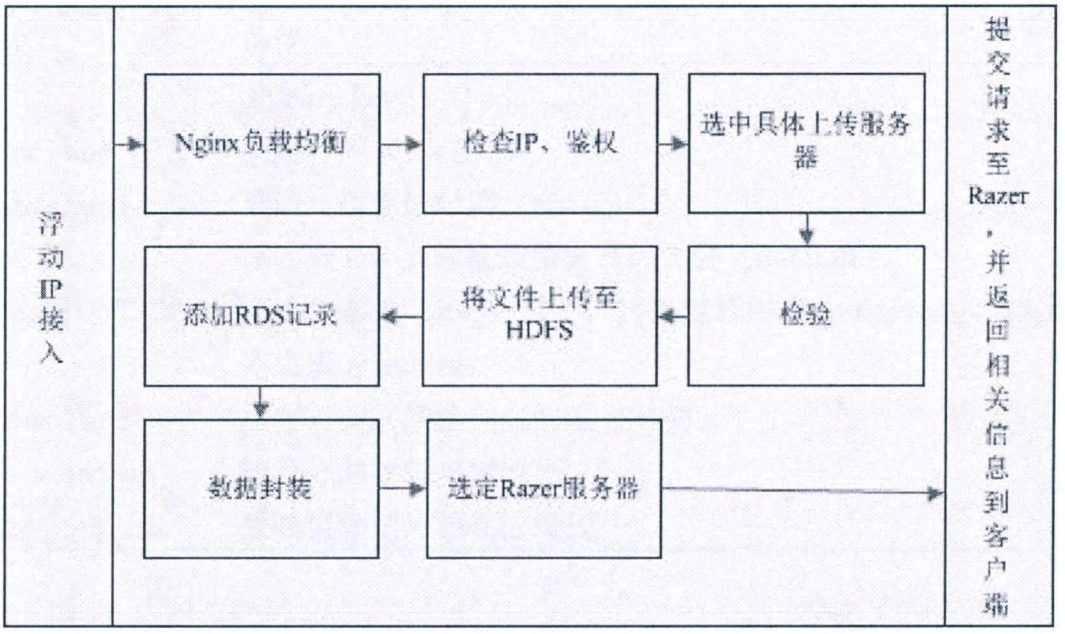

图 4-3 API 工作流程图

Figure 4-3 API Work Process

下一步是，Nginx 通过 Upload 模块获取数据上传数据，上传完毕后更新数据库内容。

当文件完全上传后，系统调用校验函数，快速校验文件完整性，通过后调用 HDFS 的上传命令，将文件推送至 HDFS 的 video_source 目录，同时向 VideoTask 更新转码信息。Nginx 的鉴权通过 Lua 脚本实现，具体定义在 nginx 的配置文件中，它利用 ngx_http_access_module 限制 ip 对 nginx 的访问，第二步通过 python lexer 生成 token，如果发生鉴权请求，则调用这个模块。

4.2.3 API 与 RDS 交互

首先，需要建两张相关的表来辅助转码任务：

（1）视频文件表（VideoFile），记录需要转码的文件的基本信息。

（2）视频任务表（VideoTask），记录跟踪一个文件的转码流程，主要是被 Razer 使用。

对于视频文件表，记录的是最基本的信息，用于 API 的输入和返回值，因此在 API 工作过程中需要用到的信息全部都要记录在内，设计如下：

表 4-1 视频文件表

Table 4-1 Video file table

字段	备注
id	primary key
createTime	创建时间（not null）
modifyTime	最后一次修改时间（not null）
file	保存到 nos 上的视频原格式的文件（not null）
status	初始状态为 init（必须），转换过程中为 processing，转换完成后为 publish
publishTime	发布时间（转换完成的发布时间）
dest_video_url	转换好的视频存放路径
video_img_url	截取视频图片的存放路径

视频任务表，主要是用在 API 提交转码请求之后，后续模块使用，可以理解为队列的持久化操作，防止意外使得集群停止工作时内存中数据丢失。因此，视频任务表的设计应该与转码任务的状态等相关，故设计如表 4-2。

在系统完成鉴权时，视频文件表和视频任务表新增两条记录，完成上传校验后，更新这两条记录内容，以便后续使用。

表 4-2 视频任务表

Table4-2 Video task table

字段	备注
id	primary key
createTime	创建时间（not null）
modifyTime	最后一次修改时间（not null）

表 4-2（续表）

Table4-2 (Cont)

字段	备注
file	保存到 dfs 上的视频原格式的文件
local_orig_video_path	转换过程中，本地待转换的视频文件路径
local_dest_video_path	转换后的视频文件路径
local_dest_big_img_path	截取视频的大缩略图文件路径
local_dest_small_img_path	截取视频的小缩略图文件路径
taskStatus	工作状态：init（初始 必须）->wait（加入任务队列）->processing(正在转换)->success（操作成功）/fail(操作失败)
stage	download（初始状态，必须）->convert->capture_big->upload->finish
failCount	总共失败的次数，目前设置为超过 12 次便不再重试，不同的阶段重试权重不同，download 为 3，capture*为 2

4.3 视频预处理模块（Razer）

本章主要是对 Razer 模块进行设计，Razer 模块是本系统的关键模块，系统的用户信息处理逻辑主要是依靠 Razer 模块，Razer 在系统中唯一，保证 pipeline 的自然隔离。

4.3.1 功能逻辑描述

视频处理模块是处于 API 与 Job-server 之间的一个模块，主要的功能是：

（1）资源隔离，根据配置的 pipeline 资源配额，结合资源分配算法，为每个 pipeine 分配资源；

（2）调度转码 job，根据每个 pipeline 分配到的资源数目，从数据库中取出相应数目的 job 分发给 job server，并且接受 worker 的完成心跳，根据 worker 返回的心跳状态决定任务重试还是通知给客户端；

它与 RDS 的交互主要体现在以下过程中：

（1）razer 启动时，加载所有的 pipeline 到内存中，并且加载每个 pipeline 的

运行状态的 job 到内存；

（2）razer 定期扫描 pipeline 的状态，检测到 pipeline 的配额修改，则更新内存中 pipeline 的配额，检测到 pipeline 被删除，则删除内存中的 pipeline，扫描周期有配置文件中的 pipelineScanInterval 项控制，线上环境配置值为 5s；

（3）razer 检测到有空闲的 worker 时（参考 razer 与 job server 交互一节），会从 rds 数据库中取得一定量的 job，并修改 pipeline 的计数；取得 job 数量由系统的空闲 worker 数目、等待的 job 数目中的较小值决定；

（4）razer 接受到 worker 的转码结果后，更新 RDS 中的 job 状态和 pipeline 计数；

（5）预控制，使得集群负载不至于很大。

4.3.2 资源隔离与 Pipeline 控制

总体过程是通过提交方式，每一个资源生成一个 pipeline，然后提交给某个 jobserver，同时 razer 上保存一个 Pipeline 副本，直至 job-server 返回 pipeline 完成的信号。

具体将 pipeline 提交给那个 Job-server，需要 razer 预估上传的任务与哪个 Job-server 的小组开销最短，比如文件传到 1 号机房，同时 1 号机房有 job-server 空闲或者队列较短，总体开销比使用 2 号机房代价小，则通过 pipeline 将任务打包给 1 号机房的 server，实现单个资源的整体隔离。

Pipeline 到达 Job-server 之后，将经历提交、转码、返回状态的过程。

具体如图 4-4 所示。

4.3.3 转码策略

视频转码的相关策略应该部署在 Razer 服务器上。在 API 对系统内部提交请求后，请求直接到达 Razer 服务器，Razer 服务器将根据视频转码的需求、与此需求相关的转码最优方案、当前服务器可用情况等指定转码最优参数，进一步提交给 Gearman 的 Job-server。

目前限于带宽和 CPU 处理能力的限制，大多数的视频分辨率和码率并不是很高。因此，结合不同终端的需求，指定三种转码标准：

（1） sd（标清）：分辨率的长：640px，视频比特率：256kbps

（2） hd（高清）：分辨率的长：720px，视频比特率：448kbps

（3） shd（超高清）：分辨率的长：1080px，视频比特率：832kbps

由于公司要求自制的视频质量要求较高，因此特指定自制视频统一规格为：分辨率：1280x720，视频比特率：1500kbps，帧率：15fps，音频比特率：64kbps，视频编码：h264，音频编码：aac，封装器：mp4。

在以上标准的基础上，在转换前先用 ffmpeg 和 mplayer 获取视频的基本信息，然后根据视频信息决定转换策略。

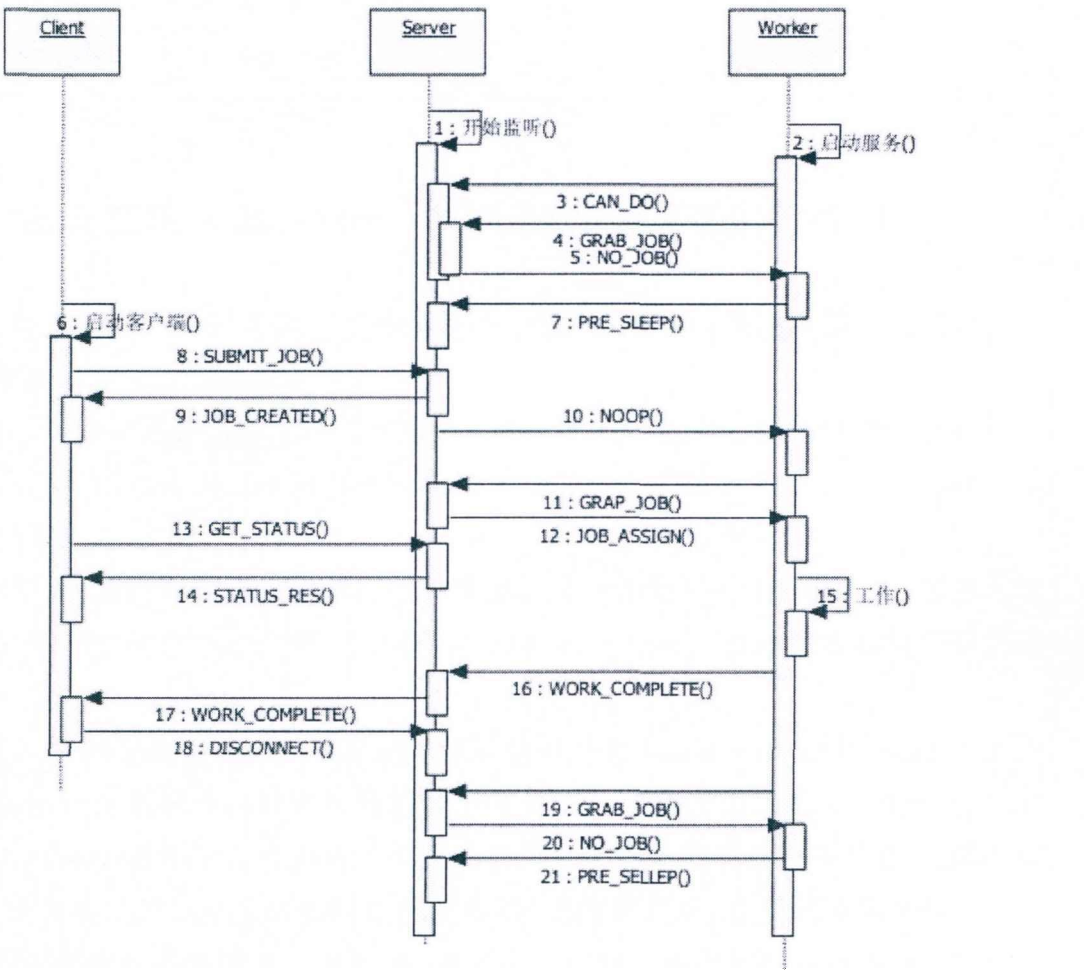

图 4-4 消息交互时序图

Figure 4-4 Message Interactive

具体的策略为：

（1） 视频编码为 H264，音频编码为 AAC，总比特率小于 1600Kbps。以 copy 模式进行一次检测拷贝

（2） 视频编码为 H264，音频编码不为 AAC，总比特率小于 1600Kbps。视频以 copy 模式，音频以 aac 编码的转换模式，进行转换

（3） 视频编码为 H264，总比特率超过 1600Kbps。以 hd 的转换标准进行转

换

（4） 视频编码不为 H264。以 hd 的转换标准进行转换，如果分辨率小于 hd 标准，按原分辨率转。如果比特率小于 hd 标准，按原比特率转。

（5） 转换完后统一用 qt-faststart 处理 metadata 信息，将本来放在视频结尾的 metadata 信息提到视频头部，这样可以保证视频边下载边播放。

经过转码的最终文件应该包括以下几类：

（1） FLV，针对 PC 端，包括网页以及客户端

（2） Mp4，针对 Android 用户

（3） M3u8，针对 iOS 用户，分段生成策略

4.3.4 Razer 获取 Worker 状态，并制定针对服务器的策略

Razer 的调度首先应该查找可用服务器。这个过程可以使用负载监控工具，同时结合数据库中关于服务器队列的信息。

首先，Razer 通过监控工具查询服务器本身状态。负载监控系统使用已有的系统，服务运行完进行客户端向服务端注册过程即可，不用再进行二次开发，节省时间和成本。

再次，通过数据库记录来查询任务状态。各个角色进行转码的时，都会与数据库有交互，将目前的状态持久化保存，以便 razer 查阅，按照服务器状态制定策略。

最后，根据规则选择服务器将任务派发至相关的 Job-server。这是 razer 中比较复杂的部分。在流程上，首先查询集群整体负载，尽量选择开销较小的链路，比如北京用户自动选择北京机房的上传服务器，如果本地服务器状态忙碌、同时负载较高，此时将选择本地负载相对低的服务器，上传完之后，立刻同步到公共 HDFS 上；然后再根据服务器情况，来选择 job-server 小组，并将参数以 pipeline 的形式提交至对应 job-server 上，完成任务分发。

4.3.5 Razer 与 RDS 交互

Razer 在工作时，接收到任务之后，首先获取任务的信息，按照制定策略，制定出转码任务；第二步，判断服务器可用状态，其中涉及到从服务器信息表获取数据，获取合适的服务器。进而将任务派发至此 Job-server 中。

同时，定期清理转码任务表中已经完成的任务。

这个过程中，涉及到服务器记录的数据表，此表包含服务器地址，Job-server

队列状态和完成度等信息，故将“服务器信息表”设计表 4-3。

表 4-3 服务器信息表

Table 4-3 Server information table

字段	备注
ID	编号
Server_ip	当前服务器地址
Job_server	对应的 job-server 地址
Worker_count	开启 worker 服务的数量
In_process_count	正在转码中的 worker 的数量
Time	时间表示

如果视频文件过大，需要进行分片，那么需要再用一个临时的表，作用有两方面，第一是判断方便判断是否完成了所有分片的转码，第二是一旦出现转码异常，可以经过查询数据库的形式，来再一次自动提交转码任务。

表 4-4 服务器状态表

Table 4-4 Server status table

字段	备注
ID	编号
Job-server	Gearman 服务器地址 IP
ip_addr	服务器 IP
Current_process_count	总分片数
In_process_id	当前正在转码插入的是第几片片
Is_finished_flag	完成标识
Time	时间标示

4.4 视频任务分发模块（Job-Server）

本节主要是对 Job-server 进行描述，其中包括 Job-server 的逻辑功能、服务器端实现等。Job-server 是 pipeline 的承担者，并将 pipeline 拆解分发给各个 worker，使得各 worker 并行运行。在拆分的时候，由 Gearman 调用调度算法进行拆分，调度算法会均衡队列长度、机器性能等数据，将任务拆开并形成队列。

4.4.1 功能逻辑描述

Job Server 的“上级”是 Razer，它的“下级”们是若干台负责转码的 Worker。Job server 功能主要是将 razer 分发的任务调度给 worker，转码服务器 worker 都是基于 gearman 的 worker 服务器框架开发。

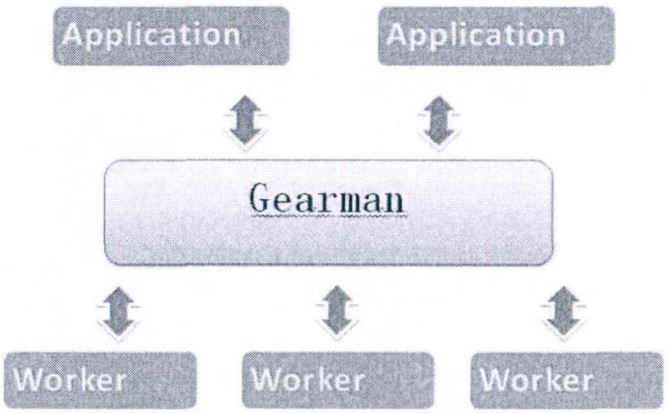

图 4-5 消息交互时序图

Figure 4-5 Message Interactive

Job server 采用 Acitive-Acitive 的服务模式：1) gearman-jobserver 两个节点只要有一个存活，就能正常工作。所有 worker 都会在两个节点注册，gearman 客户端会尝试往两个 jobserver 提交任务；2)不同于 nos 的使用方式，在 nts 中 gearman-jobserver 队列的内容无需持久化到 mysql；3) gearman-worker 任务做到一半挂掉，jobserver 负责分配到其他的 worker；4）job server 的队列长度采用默认配置，即为无限，依赖于 razer 的调度策略，job server 上缓存的 job 不会太多。

4.4.2 Gearman 服务器端逻辑与实现

Gearman 服务端是是 Job-server 的核心。

Gearman 架构的系统工作过程是：首先，是在服务端启动 Gearman；然后配置 Worker，指定 Gearman 的 Job-server 配置，启动 Worker，此时 Worker 将会自动注册至 Job-server 上；接下来，Client 请求注册内容，并传入 1 数据，Job-server 就会将 Client 派送给 Worker；最后，Worker 处理接收的数据，并进行处理，完成后将结果返回给 Client，进一步返回给最终用户。

在本系统中，Razer 根据作为 Client 角色，预判服务器情况，将任务注册到不

同的 Job Server 中，进一步平衡了负载，提高了效率。

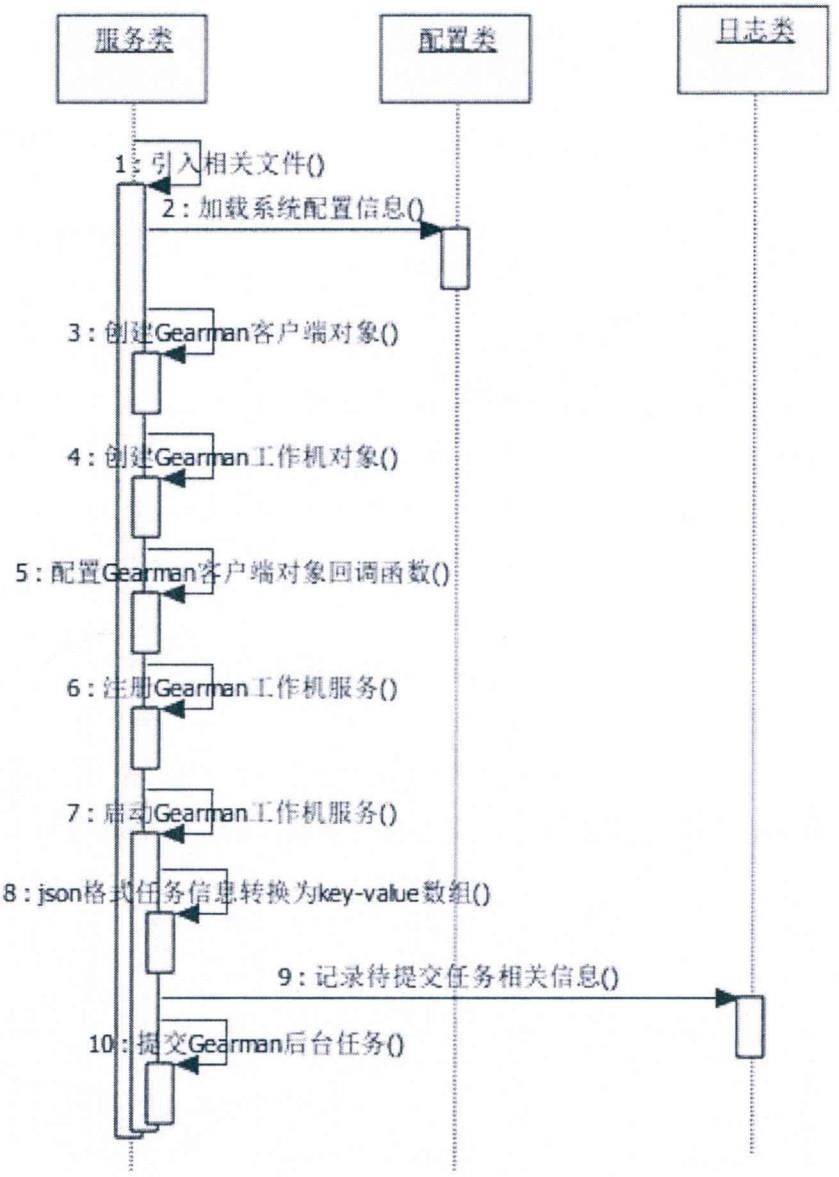

图 4-6 Gearman 工作示意图

Figure 4-6 Gearman work

4.5 视频任务处理模块（Worker）

本节主要是对 Worker 进行描述，其中包括 Worker 的逻辑功能描述，逻辑功能的实现，包括截图、视频、音频转码等。Worker 承担的任务最多，因此 worker 为并发执行，计算能力最强，但是设计实现较简单。

4.5.1 功能逻辑描述

Worker 是最终任务处理的节点，一般来说，worker 工作时负载较高，因此要求性能较强。Worker 需要做的任务主要有以下几方面：

（1） 获取视频信息

（2） 截图

（3） 视频的拆分与合并

（4） 音频转码

（5） 视频转码

（6） 打水印/去水印等

同时，根据每台服务器的性能配置不同，需要开启不同数量的 Worker 服务，目前来说，每个 worker 同一时间只能进行一个转码操作，并且每个 worker 无状态，任意的一个 worker 挂掉，job server 或者 razer 会将挂掉 worker 上运行的 job 重新调度到其它的 worker 上执行。Worker 之间具有不相关性。

4.5.2 Worker 获取信息以及截图功能实现

每当任务初次提交，首先返回文件信息以及截图。

Razer 提交获取信息和截图请求时，首先定义 json 格式的数据，将这些数据从数据库中或者直接从文件读取到：

```
gm_data = {
'json_data':  { 'vid': vid,
                'vfname': vfname,
                'srcpath': uploaded_f,
                'spdestpath':  snapshot_destpath,
                'destpath':  destpath,
                'reportformat': 'flv',
                }
}
```

注册任务并获取返回状态的代码是

```
completed_job_request =
        gm_client.submit_job('video_gatherinfo', gm_data)
```

此时，如果完成截图任务，那么返回的状态中将包括返回多个截图文件的链接地址，供上传用户选择。

4.5.3 视频的拆分与合并

如果视频过大，Razer 将会在转码前发起视频拆分请求，当转码结束，视情况发起视频合并请求。

当接收到视频拆分请求时，razer 将会根据服务器负载情况、当前文件大小等综合考虑，将文件分为若干子块，并提交拆分任务；此时 job server 将会首先向 worker 提交拆分任务，worker 调用 ffmpeg 来讲文件拆解，并返回拆解信息，razer 此时就会将拆解信息添加到 pipeline，再次提交给 job server。

当接收到视频合并请求时，此时分片转码已经提交至 HDFS 上，razer 将会抽取具有任意空闲 worker 的 job 小组，直接在 HDFS 上进行合并操作同时将结果写入数据库，看策略决定是否推送至 CDN，最后信息返回给客户端。

4.5.4 Worker 转码功能的实现

实际进行转码时，目前会利用开源的 ffmpeg 来进行转码，计划先支持 3 种容器格式： flv，mp4，m3u8。

在提交不同的任务时，worker 使用 ffmpeg 结合传过来的参数，启动视频转码处理[30][31]，另外值得注意的是这里的 m3u8 采用 ts 列表的形式。

下面以 IPHONE4S 为例，描述转码程序参数设置[32]：

（1） 视频流转成 H.264，音频流转成 aac (并封装成 mp4 格式)

ffmpeg -i MVI_0087.MOV -acodec aac -ac 2 -strict experimental -b：a 160k -s 960x640 -vcodec libx264 -preset slow -profile：v baseline -level 30 -maxrate 10000000 -bufsize 10000000 -b：v 1200k -f mp4 -threads 0 MVI_0087.iphone.mp4

（2）转换成 ts 格式 (IOS 设备可用)

ffmpeg -i MVI_0087.iphone.mp4 -f mpegts -vcodec copy -acodec copy -vbsf h264_mp4toannexb -y MVI_0087.ts

（3）切割 ts 文件，并生成 m3u8 元信息

ffmpeg -i MVI_0087.ts -codec copy -map 0 –f segment -segment_list MVI_0087.m3u8 -segment_time 10 MVI_0087-%03d.ts

转码程序：

```
set DISK=D:
set SCRIPT_DIR=d：\transfer
set WORK_DIR=%SCRIPT_DIR%\working
set path=%SCRIPT_DIR%\bin；%path%
```

```
%DISK%
cd %WORK_DIR%
rm *.ts
rm *.m3u8
set SRC=%1
set DES_MP4=MVI_0087.iphone.mp4
ffmpeg.exe -i %SRC%  -s 480x320 -b 1000k -y %DES_MP4%
ffmpeg.exe -i %DES_MP4% -vcodec copy -acodec copy -vbsf h264_mp4toannexb -map 0 -f segment -segment_time 10 -segment_list iphone.m3u8 -segment_format mpegts iphone-%%05d.ts
```

4.6 流媒体服务器

视频提交至 HDFS 上后，将会向流媒体服务器发送消息，此时视频文件将由接管，对外在线播放功能。

目前流媒体的点播技术主要有两种：

（1）HTTP + mp4 (H.264 编码) 方式

这种方式主传输上走 HTTP 协议，封装格式以 mp4 为标准，视频编码一般都会转成 H.264。

目前，视频的在线播放终端有一些不同，pc 端 flash player 播放，android 设备点播播放，由于技术限制，IOS 设备播放时间超过 10 分钟或者五分钟内数据量达到 5M 以上必须使用 HLS。

（2） HTTP Live Streaming (HLS)

使用 HLS，可以在 HTTP 之上获得接近 RTSP/RTP 效果的播放效果，支持直播和点播，同时，它支持媒体加密和用户认证等版权保护手段。

HLS 目前只支持 H.264 视频和 AAC 音频，解决方案是生成 H.264 视频和 AAC 音频，然后再封装成 ts 格式，继而再对其进行分割，并建立索引文件 (.m3u8)。直播的时候，对编码好的 stream 进行实时分割，当分割出新的 ts 片段时，更新.m3u8 索引文件实现直播。

此时需要考虑到的问题有：1.web cache 的配置里面必须进行配置，保证客户端可以及时获取每次更新后的 m3u8 文件；2.支持同一个视频编码成多个码率，这样客户端可以根据用户网络状况自动切换到合适的码率进行播放，可以通过在索引文件上做配置完成这个功能。

综合这两种方式，最终的流媒体服务器方案是可以利用 nginx 的 H264

Streaming Module 来搭建，支持的文件包 mp4、flv、ts 格式的视频文件，此时，视频编码统一用 h.264，音频编码统一用 AAC。

同时，利用 nginx 来播放视频文件时，必须在 nginx 配置中指定视频文件所在的目录。而在方案中，视频文件存储在云计算的 NOS 服务中，这样就需要一个机制把存储在 NOS 服务中的文件映射到本地目录，才能通过 nginx 播放。这里使用 NOS 提供的 NOSFS 程序来做这个映射(基于 fuse 的 fuse-nos)。利用 fuse-nos 可以把 NOS 挂载到本地文件系统，这样 nginx 可以像访问本地文件一样，直接访问存储在 NOS 中的视频文件。

4.7 本章小结

本章是本文阐述详细设计和实现的章节，描述的是整个系统各个模块的详细设计与实现。

第一部分内容是根据需求确定了系统的逻辑，进一步设计了系统详细流程，并将各部分封装成模块，进行进一步的设计与实现。

第二部分是各个模块的设计与实现。负载接入模块利用 keepalive 和 nginx 保证系统的可靠以及负载均衡；视频预处理模块利用 python 将 ffmpeg 封装，同时根据定制的逻辑策略，对视频系统进行预处理，将视频处理任务形成队列；视频分发模块利用 python 模块调用 gearman 开源框架提供的功能，实现对视频队列进行分发；视频任务处理模块，即 gearman 的 work，是视频处理任务最终承担者，使用 json 文件将转码信息导入系统，并利用 ffmpeg 等开源工具转码。

5 系统测试

本章主要内容是系统的测试，包括测试方案目标、测试方案设计等，最终进行系统测试，在系统测试过程中监控主机性能各项指标，最终提出系统的优化方案。

5.1 测试方案设计

本章主要内容是对目前具有的需求进行汇总和分析，得到详细的需求内容，从需求内容点出发，判定系统的性能指标要求，在系统设计过程中，保证指标为基本点，进一步对系统整体进行更好的设计。

5.1.1 测试目标

功能性上，保证系统可用，并具有冗余性；性能上，可以支持满足需求的数量的请求，丢包率少，得出分布式系统配置与处理能力的关系。

5.1.2 性能指标预估

（1） 服务可用性要求

NTS 服务可用性承诺以整个集群作为衡量目标，SLA 承诺为 99%，即每月停服 7 小时。尽量实现整体系统的高可用方案，包括 API、Razer、Job-server/worker，第一阶段要求 Job-server/worker 的高可用必须实现。

（2） 数据可靠性要求

任务数据持久化依赖于数据库，可靠性为：99.99999%。为保证可靠性，数据库使用 2 个 SAS 盘做 RAID1，并配置定时备份，确保数据可靠性。

（3） 用户请求响应时间指标

PIPELINE/JOB/PRESET 资源的增删改查请求，99%响应时间在 4 秒内返回结果。

统计方式：通过统计平台收集 API 服务器的日志，日志中包含用户每条请求的实际响应时间。

（4） 转码吞吐量

对于一个标清视频转码任务，1 个 ECU 计算资源在一分钟内，输出的视频长

度保证在 1.5 秒以上。

统计方式：通过统计平台收集日志进行统计。

预估值的理由：一台 R620 服务器，每分钟输出 4 分钟的标清视频。每台 R620 服务器目前在云主机中换算成 112 个 ECU。

（5） 系统资源利用率

视频处理任务提交后，NTS 目前采用异步模式处理请求。当产品内有任务正在排队时，用户申请的资源平均 CPU 利用率不低于 70%；

统计方式：目前尚无法统计，以后需在统计平台提供针对此场景的专门统计项。

系统平均 CPU 利用率不低于 70%，是一个预估值，基于下述理由：系统支持资源超售机制，当一个 PIPELINE 任务堆积而其他 PIPELINE 空闲时，会自动把配额暂时借用到任务排队队列；在实际测试中发现 ffmpeg 并不能把 CPU 长期跑满在 100%，并且为了系统响应和稳定性，在实际部署时，会预留几个空闲内核给 worker 进程。因此 70%CPU 利用率应该是一个比较合理的值。

（6） 系统可伸缩性指标

统计程序监测 PIPELINE 内队列长度，大于阀值时在 10 分钟内发出报警；API 服务器、转码服务器的服务器扩容时间，在资源已到位的情况下，能在 2 小时内完成扩容操作；

5.2 测试环境搭建

本章主要内容是对目前具有的需求进行汇总和分析，得到详细的需求内容，进一步对系统整体进行设计。

（1） 实现环境

作为实验环境，并不需要很高的性能。但是需要服务器的数量比较多。具体如下：

1）服务器 A 和服务器 B，作为 API server，两台机器进行互备，对外提供统一的 API 接口。

2）服务器 C，作为 Razer 服务器，负责任务策略性分配。

3）服务器 D 和服务器 E 和 F，其中 D 作为 Job-server，E、F 作为 worker。

4）服务器 G 作为 HDFS 服务器（NameNode，SecondaryNameNode，均兼做 DataNode）。

5）服务器 H 作为数据库服务器

操作系统：debian version 7.6 (3.2.0-4-amd64)

（2） 搭建流程

首先，搭建 HDFS，映射到各个服务器上，第二步，API 服务器上，安装双机软件 keepalive，配置浮动 IP，第三步，API 服务器上，安装 nginx，第四步，API 服务器上，配置上传逻辑，第五步，Raser 服务器上，安装资源监控，第六步，Raser 服务器上，安装预处理逻辑，解决方案，第七步，Job-server 服务器上，安装 gearman-server，配置处理逻辑，第八步，Worker 服务器上，安装 worker，安装 ffmpeg。

5.2.1 搭建 HDFS，映射

搭建 HDFS 使用的软件有：hadoop1.2.1，jdk1.7 等，地址为 NN 的 IP 为 10.0.0.100

（1） 首先将源码拷贝到主机上，解压：

#tar -xzvf /data/installation/jdk-7u40-linux-x64.tar.gz -C /data/software/java/

#tar -xzvf /data/installation/hadoop-1.2.1-bin.tar.gz -C /data/software/hadoop/

（2） 接下来配置变量，在/etc/profile 中追加以下内容，并使用 source 命令使其立即生效。

HADOOP_INSTALL=/data/software/hadoop/hadoop-1.2.1/

JAVA_HOME=/data/software/java/jdk1.7.0_40

PATH=$JAVA_HOME/bin：$HADOOP_INSTALL/bin：$PATH

CLASSPATH=$JAVA_HOME/lib

export JAVA_HOME PATH CLASSPATH HADOOP_INSTALL

然后配置 ssh 无密码登陆。

由于实验环境，直接生成密钥，并添加至本机的 authorized_keys 列表。

#ssh-keygen -t dsa -P "" -f ~/.ssh/id_dsa

#cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys

（3） 配置 hadoop 环境，进入/data/software/hadoop/hadoop-1.2.1/conf 目录。配置 haddoop-env.sh，core-site.xml，mapred-site.xml，hdfs-site.xml 四个文件。分别配置如下：

1) haddoop-env.sh，更改 JAVA_HOME，修改为实际地址，即

#export JAVA_HOME=/data/software/java/jdk1.7.0_40

2) core-site.xml，添加地址和配置目录内容。

3) mapred-site.xml，添加数据目录。

4) hdfs-site.xml，添加监听地址。

（4） 最后一步，格式化并启动 hadoop。

#cd /data/software/hadoop/hadoop-1.2.1/bin && ./hadoop namenode –format

#cd /data/software/hadoop/hadoop-1.2.1/bin && ./start-all.sh

5.2.2 安装 keepalive，配置浮动 IP 实现

本次系统使用 keepalive 作为浮动 IP 技术的实现工具，使用 keepalive 的好处是能够更容易的与 nginx 等工具结合，增强系统可用性。

首先定义虚拟网卡，在 keepalive 的配置文件中定义：

```
vrrp_instance VI_1 {
    state BACKUP  #主从设置 MASTER
    interface eth2  #网卡名
    virtual_router_id 51
    mcast_src_ip 10.0.1.133 #本机 ip
    priority 50  #从机小于主机
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass chtopnet
    }
 virtual_ipaddress {
        10.0.1.2  #VIP 的 IP
    }}
```

再定义虚拟网卡所绑定的物理网卡：

```
virtual_server 10.0.1.2 80 {
    delay_loop 6
    lb_algo rr
    lb_kind DR
    persistence_timeout 50
    protocol TCP
 real_server 10.0.1.132 80 {
        weight 3
        TCP_CHECK {
            connect_timeout 10
            nb_get_retry 3
            delay_before_retry 3
```

```
            connect_port 80
        }}
    real_server 10.0.1.133 80 {
        weight 3
        TCP_CHECK {
            connect_timeout 10
            nb_get_retry 3
            delay_before_retry 3
            connect_port 80
        }}
    }
```

经过这样配置，即可将两个物理网卡的 Read IP 绑定到 Vitual IP 上，实现 IP 浮动在两个主机端口上。

5.2.3 安装和配置 Nginx，配置

（1） 选定源码目录

可以是任何目录，本文选定的是/home/nginx/src

（2） 安装 PCRE 库

PCRE(Perl Compatible Regular Expressions)是一个 Perl 库，包括 perl 兼容的正则表达式库。可以去 ftp：//ftp.csx.cam.ac.uk/pub/software/programming/pcre/ 下载最新的 PCRE 源码包，使用下面命令下载编译和安装 PCRE 包：

```
#cd /home/nginx/src
#wget ftp：//ftp.csx.cam.ac.uk/pub/software/programming/pcre/pcre-8.34.tar.gz
#tar -zxvf pcre-8.34.tar.gz
#cd pcre-8.34
#./configure
#make
#make install
```

（3） 安装 zlib 库

zlib 是提供数据压缩用的函式库。它的下载位置是 http：//zlib.net/zlib-1.2.8.tar.gz，使用下面命令下载编译和安装 zlib 包：

```
#cd /home/nginx/src
```

#wget http：//zlib.net/zlib-1.2.8.tar.gz

#tar -zxvf zlib-1.2.8.tar.gz

#cd zlib-1.2.8 && ./configure

#Make& make install

（4） 安装 ssl（某些 vps 默认没装 ssl)

#cd /home/nginx/src

#wget http：//www.openssl.org/source/openssl-1.0.1c.tar.gz

#tar -zxvf openssl-1.0.1c.tar.gz

（5） 安装 nginx

Nginx 一般有两个版本，分别是稳定版和开发版，您可以根据您的目的来选择这两个版本的其中一个，下面是把 Nginx 安装到 /home/nginx/nginx 目录下的详细步骤：

#cd /home/nginx/src

#wget http：//nginx.org/download/nginx-1.4.2.tar.gz

#tar -zxvf nginx-1.4.2.tar.gz #cd nginx-1.4.2

#./configure

\--sbin-path=/home/nginx/nginx/nginx \--conf-path=/home/nginx/nginx/nginx.conf

\--pid-path=/home/nginx/nginx/nginx.pid

\--with-http_ssl_module

\--with-pcre=/home/nginx/src/pcre-8.34

\--with-zlib=/home/nginx/src/zlib-1.2.8

\--with-openssl=/home/nginx/src/openssl-1.0.1c

#make &&make install

其中：--with-pcre=/usr/src/pcre-8.34 指的是 pcre-8.34 的源码路径。

--with-zlib=/usr/src/zlib-1.2.7 指的是 zlib-1.2.7 的源码路径。

（6） 启动

确保系统的 80 端口没被其他程序占用，运行/home/nginx/nginx/nginx 命令来启动 Nginx，

#netstat -ano|grep 80

如果查不到结果后执行，有结果则忽略此步骤（ubuntu 下必须用 sudo 启动，不然只能在前台运行）

#sudo /home/nginx/nginx/nginx

5.2.4 安装 Gearman

（1）Server 端：

下载源码直接编辑安装

#tar xvzf gearmand-0.20.tar.gz && cd gearmand-0.20

#./configure && make

然后启动 gearman 服务器

./gearmand -uroot -d -L gearman_server_ip -p 4730

（2）Client 端：

从 pip 下载 gearman-2.0.1.tar.gz，解压并安装

#tar xvzf gearman-2.0.1.tar.gz

#cd gearman-2.0.1 && python setup.py install

5.2.5 安装 FFmpeg

首先添加新的源列表至/etc/apt/sources.list，即加入以下内容：

deb http：//www.deb-multimedia.org squeeze main non-free

deb http：//www.deb-multimedia.org squeeze-backports main

然后 update 源，直接通过 apt-get 的管理工具，安装 faac、libopencore-amrnb-dev、libopencore-amrwb-dev、libsdl1.2-dev、libtheora-dev、libtheora-dev、libx11-dev、libxfixes-dev、libxvidcore-dev、yamdi、libass-dev 等用到的库。

接下来安装 ffmpeg，方式如下：

#git clone --depth 1 git：//source.ffmpeg.org/ffmpeg

#./configure --disable-yasm --enable-shared --enable-gpl --enable-nonfree --enable-pthreads --enable-libfaac --enable-libmp3lame --enable-libx264

#make && make install

5.3 功能测试

系统最基本的要求是保证系统的可用性，即完成功能测试。功能测试比较简单，即以模拟客户端的方式提交请求，查看返回。在本项测试中，需要明确服务接口。

5.3.1 服务接口设计

REST（Representational State Transfer）是一种分布式应用架构风格，它的特

点是，通过 URL 来进行一系列的调用，包括：指定所需资源，对资源进行获取、创建、修改、删除的操作，与 HTTP 协议的 GET、POST、PUT、DELETE 等方法对应，对应操作非常灵活[33][34]。

本系统采用 Restful 的服务形式，在服务端搭建了一套基于 HTTP 协议和 REST 架构策略的服务系统。本次接口提供三种方式，包括 GET、POST、DELETE，提交类接口以 POST 为主，查询类接口仅支持 GET，并且 API 参数传入均只接受 UTF-8 编码，返回的信息是 JSON 类型。

REST API 的 URL 是：http：//trans.v.xxx.com/rest/，主要接口如表 5-1：

表 5-1 主要接口信息表

Table 5-1 Interface information

请求接口	请求方式	期望响应信息
/service/getserinfo.php	GET	转码服务器状态信息
/service/getvideoinfo.php	GET	根据 vid 查询视频文件转码进度等信息
/service/upload.php?{query_string}	POST	上传转码视频文件
/service/transcode.php?{query_string}	POST	执行转码任务
/service/autotranscode.php?{query_string}	POST	上传文件并转码
/service/auth.php?{query_string}	POST	鉴权
/service/delete.php?{query_string}	DELETE	删除文件

5.3.2 功能测试

功能测试项目较多，此处取 4 项内容，作为代表。

（1） 查询服务器状态

1) 请求信息：

GET

/service/getSerInfo.php?lan=zh-cn&format=json&cmd=checkstatus&token=XXX

2) 期望返回信息：

Json 字符串，包含转码 worker 进程数量，当前工作中 worker 数量，当前队列长度

3) 返回信息：

```
HTTP/1.1 200 OK
Content-Type:application/json
{
  "id":"NULL",
  "err_no":0,
  "err_msg":"success",
  "result":[
   {
    "worker_count":"6",
    "worker_in_process":"6",
    "queue":"2",
   }
   ]
}
```

（2） 上传视频文件

1) 请求信息：

<form method="post"
action="http：//trans.v.xxx.com/rest/service/upload.php"enctype=”multipart/form-data”>

<input id="txtUserName" name="txtUserName" type="text" />

<inputtype="mp4" name="test">

</form>

2) 期望返回信息：

Json 字符串，包含视频文件 vid，视频校验信息，各种格式的地址

3) 返回信息：

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "vid":"gF9HJ7A6Va4",
  "err_no":0,
  "err_msg":"success",
  "result":[{
"mp4h":"http://trans.v.x.com/rest/video_source/view_mp4h_online/gF9HJ7A6Va4",
"mp4m":"http://trans.v.x.com/rest/video_source/view_mp4m_online/gF9HJ7A6Va4",
"mp4l":"http://trans.v.x.com/rest/video_source/view_mp4l_online/gF9HJ7A6Va4",
"avih":"http://trans.v.x.com/rest/video_source/view_mp4h_online/gF9HJ7A6Va4",
"avim":"http://trans.v.x.com/rest/video_source/view_mp4m_online/gF9HJ7A6Va4",
"avil":"http://trans.v.x.com/rest/video_source/view_mp4l_online/gF9HJ7A6Va4",
```

```
"m3u8":"http://trans.v.x.com/rest/video_source/view_m3u8_online/gF9HJ7A6Va4",
"flv":"http://trans.v.x.com/rest/video_source/view_flv_online/gF9HJ7A6Va4",
    } ]
}
```

（3） 错误请求的例子

1) 请求信息：

GET

/service/getSerInfo.php?lan=zh-cn&format=json&cmd=checkstatus&token=abc

2) 期望返回信息：

返回错误码，并返回内部自定义的纠错码，显示出错原因

3) 返回信息：

```
 HTTP/1.1 403 Forbidden
Content-Type:application/json
{
   "vid":27,
   "err_no":1000,
   "err_msg":"error token",
   "result":[]
}
```

5.4 性能测试

本次性能测试包括两部分，包括压力测试和系统负载测试。系统前端由 API 服务器对外提供接口，API 以 Nginx 为基础，提供了比较好的性能和负载均衡。因此，抗压能力非常好，在做测试的时候，丢包率较低，由于系统是以低配虚拟机作为压力测试的试验服务器，因此，此时压力测试不具有参考意义。

系统负载测试，可以通过持续性发送转码请求，通过监控系统了解具体情况。本次负载测试将发送 100 个 1G 的视频文件，给服务器足够的压力，观察负载情况。通过监控系统，截取 3 个小时的数据，颗粒度是分钟，这里，可以看到图 5-1 结果。

由图可以看到，当提交转码任务的时候，API 负载迅速上升，同时 API 分片上传通道，当上传完毕，将任务提交给 Razer，可以看到 Razer 负载上升，检查信息，查询转码逻辑，封装转码 pipeline，完成后交给 job-server，job-server 进行一些分片信息获取、逻辑处理、任务分配、队列生成等等，然后经过 razer 的策略分配，将任务分派给 Worker。

可以看到，系统再分配任务的时候，分布比较平均，worker 的负载相对均衡。系统负载符合预期，但是系统利用率并没有达到预期，需要进行优化操作。

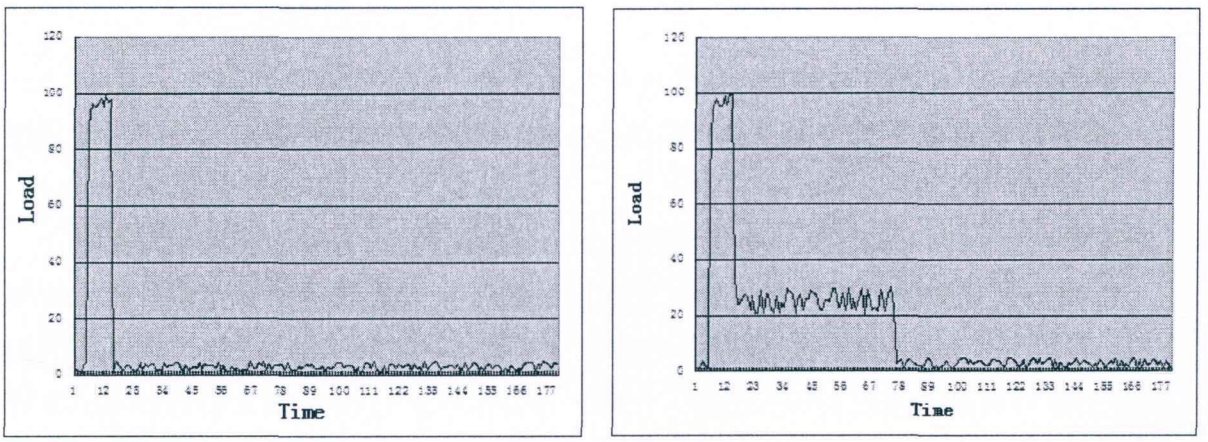

图 5-1 API 与 Razer 负载监视图

Figure 5-1 API and Razer load monitoring

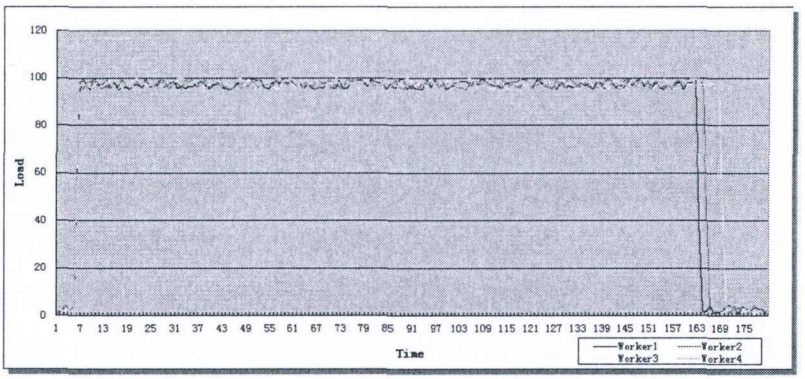

图 5-2 Worker 负载监视图

Figure 5-2 Worker load monitoring

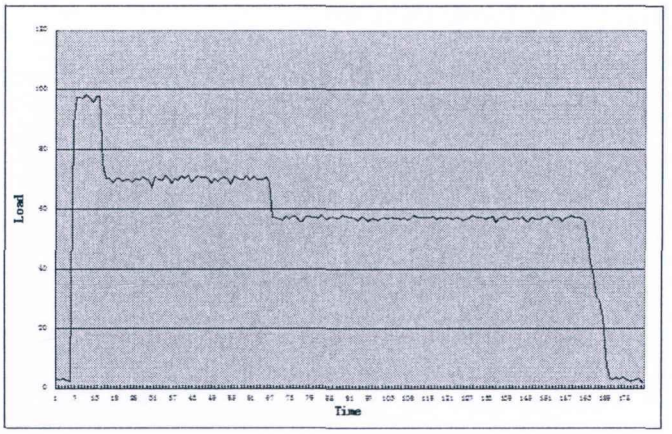

图 5-3 系统整体负载监视图

Figure 5-3 System load monitoring

5.5 系统优化

依据测试结果，可以对系统进行相应的优化，主要从两方面着手：

（1） FFmpeg 转码效率优化

Ffmpeg 转码效率高低取决于两方面，一方面是视频文件类型，另一方面是参数设定。因此，需要进行一些参数的设定，如可以降低帧数量来提高速率等等。

（2） 系统负载优化

从测试结果可以看出，API 和 Razer 的系统负载变化比较大，有一些浪费系统资源，但是这两台服务器设计上必须保证稳定，保证具有较好的突发处理能力，因此并不可以替代。针对负载变化巨大、负载平均值较低的情况，可以采用在 API 和 Razer 上启动少量 Worker 进程，如果一旦进入任务，可以迅速切换到原本任务。

5.6 小结

本章主要是对系统进行搭建与测试。首先是制定测试标准，标准制定的要求是能满足当前各类系统的转码需求。

第二部分是系统搭建，系统搭建安装从低到高的层次，即先进行系统类软件的安装，再进行转码系统的部署。这里主要安装一些开源的工具如 nginx、gearman、FFmpeg 等，然后进行简单的测试，保证工具的可用性。

接下来是进行测试，主要包括功能测试和性能测试。功能测试的方式是直接调用 API 接口，以 REST 请求的方式进行；性能测试主要查看性能指标，以压力测试为主，查看系统负载和网络负载，保证系统能够有较好的性能，满足要求。

6 结论与展望

网易视频处理系统作为网站部的数据支撑平台，其目标是为整个门户网站的所有视频业务提供转码支持。因此，随着的需求在不断的改变，系统也有很大的改进空间。

6.1 本章总结

本文对分布式视频处理系统进行了研究，从分析国内外视频转码系统的架构入手，对比研究各种分布式的架构的基础上，设计出了本次所需的视频转码系统，并加以实现。

具体来说，论文完成的主要的工作如下：

（1） 需求调研与分析

对视频转码系统使用需求的确认，并根据不同的需求，确认转码后的输出的细节，包括视频编码方式、码率、格式等，综合考虑使得系统以比较低的成本兼容了更多的用户平台，可以通过识别设备来提供不同的服务。

（2） 对 HDFS、分布式架构以及转码技术的研究。

针对具体的需求，查询可能的转码方案，调研优秀的转码技术，做到在保证可用性的基础上，使得转码系统性能更强、转码效率更高。

（3） 设计分布式转码系统的架构。

结合目前比较先进的分布式技术以及转码技术，设计基于分布式平台、共享分布式存储的并行转码的系统架构，并将有关集群、分布式的新技术应用其中。

（4） 完成系统设计。

包括具体的角色定义、角色设计，以及转码系统的细节设计，完成高可用、容错等功能的设计，增强系统的健壮性；同时完善系统逻辑，保证各模块间的高效配合。

（5） 对系统的设计进行实现，对各个子项目进行编码工作。

首先是对系统平台的搭建，然后进行系统的子项目进行编码，以 python、lua 等语言为基础，将系统有机的结合在一起。

（6） 系统集成并测试。

完成各个模块和角色的开发后，将系统统一进行部署，并进行整合测试，在保障功能完整的基础上，进行参数的优化，进一步提升性能。

本人在本次项目中主要参与的过程是设计分布式转码系统的架构、对系统的

设计进行实现以及系统集成测试，在这个过程中对于分布式系统、分布式程序编写、以及开源视频处理工具有了比较深的认识。

6.2 未来展望

分布式转码系统主要是面向前台技术人员，由于具有面向的用户数量多、部门多的特点，将来对转码系统的功能、性能的要求将会不断提升。因此，需要进行一系列的措施，进一步提升视频转码系统平台的可用性。具体可以包括：

（1） 向云平台迁移，便于快速部署

由于目前系统是服务器组，并没有以云计算的技术作为基础，因此，每次增加或者减少设备的时候，需要的一般步骤是：安装新机器的操作系统、安装系统必备的工具软件、安装系统监控软件、部署转码系统、验证并测试转码系统。这一系列的步骤将浪费大量的时间。如果将系统整体迁移至云平台中，那么可以快速部署可用的虚拟机，并制定参数，使用集群软件部署工具统一部署新机器，使得云主机从创建到可用花费非常少的时间。当机器性能不够时，可以通过云计算平台通过点击鼠标来创建可用的 worker，能够快速进行平台的横向扩展；当系统资源过多时，也可以快速的删除主机，将计算资源给让给其他服务。这样就会使得分布式转码系统的灵活性进一步提升。

（2） 采用更好地底层技术，提高转码效率

目前采用的转码技术，主要是依靠 ffmpeg 的框架，进行 CPU 转码。因此，这提供了两个提高转码效率的思路，第一就是寻找更好地转码工具，使用同样的资源，进一步提升转码效率；第二就是引入硬件加速技术[35]，众所周知，硬解码的效率非常高，非常有利于系统整体性能的提升。并通过不断尝试新技术，使得转码技术与时俱进，使得系统具有更高的性能，更好地服务用户。

参考文献

[1] Bernstein P A. Middleware：a model for distributed system services[J]. Communications of the ACM，1996，39(2)： 86-98.
[2] Garcia-Molina H，Barbara D. How to assign votes in a distributed system[J]. Journal of the ACM (JACM)，1985，32(4)： 841-860.
[3] Josuttis N M. SOA in practice： the art of distributed system design[M]. " O'Reilly Media，Inc."，2007.
[4] 李西宁. 分布式系统[M]. 科学出版社，2006.
[5] Coulouris G，康乐瑞斯，Dollimore J，et al. 分布式系统概念与设计[M]. 机械工业出版社，2004.
[6] 胡华平，金士尧. 分布式系统可靠性模型[J]. 计算机工程与应用，1999，35(8)： 1-3.
[7] Ewart J. Instant Parallel Processing with Gearman[M]. Packt Publishing Ltd，2013.
[8] Ewert R H. Gears and gear manufacture： the fundamentals[M]. Kluwer Academic Pub，1997.
[9] Haen C，Bonaccorsi E，Neufeld N. Distributed monitoring system based on Icinga[J]. WEPMU035 ICALEPCS，2011.
[10] 宋益泉. 基于 Gearman/MongoDB 的非金融交易监控报警系统的研究和实现[D]. 电子科技大学，2013.
[11] 苗龙. 面向分布式环境的自动化集成测试平台的设计与实现[D]. 哈尔滨工业大学，2013.
[12] 钱景辉，廖锂. 基于 Keepalived 的动态浮动 IP 集群实现[J]. 化工自动化及仪表，2012 (7).
[13] Nedelcu C. Nginx HTTP Server：Adopt Nginx for Your Web Applications to Make the Most of Your Infrastructure and Serve Pages Faster Than Ever[M]. Packt Publishing Ltd，2010.
[14] Reese W. Nginx： the high-performance web server and reverse proxy[J]. Linux Journal，2008，2008(173)： 2.
[15] 刘思尧. 基于 Linux 平台的高可用集群管理系统的研究与实现 [D][D]. 西北大学，2012.
[16] 林园. 分布式视频转码系统设计[D]. 华中科技大学，2012.
[17] 褚晶辉，俞斯乐，鲁照华. 视频转换编码及其实现技术的研究[J]. 电子学报，2004，32(10)：1678-1683.
[18] 钟代笛，张琦. 视频转码技术概论[J]. 世界广播电视，2004，17(12)： 59-60.
[19] 肖友能，薛向阳. 视频转码技术回顾[J]. 通信学报，2002，23(8)： 72-80.
[20] 杜辉. 数字媒体视频转码技术研究[J]. 新闻界，2008 (5)： 74-77.
[21] 刘昱，李桂苓. MPEG-2 视频码流的拼接[J]. 有线电视技术，2002，9(10)： 5-6.
[22] 2006 G B T. 中华人民共和国国家标准. 信息技术先进音视频编码第二部分： 视频 [S][D]. ，2006.
[23] 陈涛. 视频转码器的研究与实现[D]. 北京邮电大学，2008.
[24] 王海蓉，邢卫，鲁东明. 面向移动网络的实时视频转码系统[J]. 计算机工程，2009，35(3)：245-247.
[25] Tomar S. Converting video formats with FFmpeg[J]. Linux Journal，2006，2006(146)： 10.
[26] 陈俊，陈孝威. 基于 Hadoop 建立云计算系统[J]. 贵州大学学报： 自然科学版，2011，

28(3)： 91-93.
[27] 朱珠. 基于 Hadoop 的海量数据处理模型研究和应用 [D][J]. 北京邮电大学，2008，37(5)：47-49.
[28] Porter G. Decoupling storage and computation in hadoop with superdatanodes[J]. ACM SIGOPS Operating Systems Review，2010，44(2)： 41-46.
[29] Lämmel R. Google's MapReduce programming model—Revisited[J]. Science of computer programming，2008，70(1)： 1-30.
[30] 胡旭迈，任金昌. 一种基于 GOP 的 MPEG-2 媒体流切割与合并方法[J]. 微型电脑应用，2005，21(7)： 51-53.
[31] Tomar S. Converting video formats with FFmpeg[J]. Linux Journal，2006，2006(146)： 10.
[32] Cheng Y，Liu Q，Zhao C，et al. Design and implementation of mediaplayer based on FFmpeg[M]//Software Engineering and Knowledge Engineering： Theory and Practice. Springer Berlin Heidelberg，2012： 867-874.
[33] Masse M. REST API design rulebook[M]. " O'Reilly Media，Inc."，2011.
[34] Li L，Chou W. Design and describe REST API without violating REST： A Petri net based approach[C]//Web Services (ICWS)，2011 IEEE International Conference on. IEEE，2011：508-515.
[35] 李岩，王显山. 实时操作系统任务调度算法的硬件实现[J]. Computer Engineering and Applications，2010，46(35).

作 者 简 历

袁大山，男，2008 年 9 月至 2012 年 7 月就读于山东大学机电与信息工程学院通信工程专业，2012 年 7 月-2013 年 6 月任职于浪潮电子信息产业股份有限公司任存储产品部技术支持工程师，2013 年 7 月进入北京交通大学软件学院攻读软件工程专业硕士学位，研究方向为嵌入式。

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作和取得的研究成果，除了文中特别加以标注和致谢之处外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京交通大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文作者签名：  签字日期：2015年 6月5日

学位论文数据集

表 1.1： 数据集页

<table>
<tr><td>关键词*</td><td>密级*</td><td>中图分类号</td><td>UDC</td><td>论文资助</td></tr>
<tr><td>视频转码；分布式；FFMPEG</td><td>公开</td><td></td><td></td><td></td></tr>
<tr><td colspan="2">学位授予单位名称*</td><td>学位授予单位代码*</td><td>学位类别*</td><td>学位级别*</td></tr>
<tr><td colspan="2">北京交通大学</td><td>10004</td><td>工程</td><td>硕士</td></tr>
<tr><td colspan="2">论文题名*</td><td colspan="2">并列题名</td><td>论文语种*</td></tr>
<tr><td colspan="2">分布式视频处理系统设计与实现</td><td colspan="2"></td><td>中文</td></tr>
<tr><td>作者姓名*</td><td colspan="2">袁大山</td><td>学号*</td><td>13126162</td></tr>
<tr><td colspan="2">培养单位名称*</td><td>培养单位代码*</td><td>培养单位地址</td><td>邮编</td></tr>
<tr><td colspan="2">北京交通大学</td><td>10004</td><td>北京市海淀区西直门外上园村 3 号</td><td>100044</td></tr>
<tr><td colspan="2">工程领域*</td><td>研究方向*</td><td>学制*</td><td>学位授予年*</td></tr>
<tr><td colspan="2">软件工程</td><td>嵌入式</td><td>两年</td><td>2015</td></tr>
<tr><td>论文提交日期*</td><td colspan="4">2015 年 5 月</td></tr>
<tr><td>导师姓名*</td><td colspan="2">陈旭东</td><td>职称*</td><td>副教授</td></tr>
<tr><td>评阅人</td><td colspan="2">答辩委员会主席*</td><td colspan="2">答辩委员会成员</td></tr>
<tr><td></td><td colspan="2">刘磊</td><td colspan="2"></td></tr>
<tr><td colspan="5">电子版论文提交格式 文本（√） 图像（ ） 视频（ ） 音频（ ） 多媒体（ ） 其他（ ）
推荐格式：application/msword；application/pdf</td></tr>
<tr><td colspan="2">电子版论文出版（发布）者</td><td colspan="2">电子版论文出版（发布）地</td><td>权限声明</td></tr>
<tr><td colspan="2"></td><td colspan="2"></td><td></td></tr>
<tr><td>论文总页数*</td><td colspan="4">59</td></tr>
<tr><td colspan="5">共 33 项，其中带*为必填数据，为 21 项。</td></tr>
</table>