摘　要

最近几年，随着 P2P 技术的蓬勃发展，P2P 技术已经在流媒体服务中得到了广泛的应用，使得更多的用户可以享受到流式的视频服务（实时直播和点播）而无需下载整个视频文件。如今，云计算的发展为终端节点提供了一个成功的不同服务模式；通过云计算框架，共享的软硬件资源和信息可以按需提供给计算机和其他设备。

在本文中，我们提出结合 P2P 和 Hadoop 集群的系统架构，实现分布式云存储对等网络视频流服务。该系统使用 Hadoop 集群存储和管理大量视频资源文件，通过可挂载 HDFS 结合 Apache 的解决方案搭建基于 Hadoop 的视频流服务器。在终端节点中实现分布式 P2P 模式，使得终端节点间实现在不需要服务器参与的前提下互联互通，视频缓存资源共享。

通过结合 Hadoop 分布式计算框架和 P2P 技术，本文搭建的基于 Hadoop 的 P2P 视频流服务解决了传统视频流服务模式中存储容量不足、服务性能瓶颈、播放延迟性大等一系列问题。

关键词： Hadoop　　P2P　　视频流　　混合架构

ABSTRACT

Recent years, P2P technology is applied broadly in stream media with its development, which makes more users watch live or order programs by video stream service instead of downloading whole file. Nowadays, growth in cloud computing provides nodes with a different service mode, through the framework of cloud computing, shared resources of hardware and software could be offered to computers and devices according to their needs.

In this paper, a system architecture of combining P2P and Hadoop cloud is introduced, which implements P2P video service on distributed cloud storage. The system stores and manages a large amount of video files by Hadoop cloud. In addition, Hadoop-based video server is running on Apache solution to a portable HDFS. Achieving distributed P2P mode among nodes makes communication and sharing resource possible between nodes.

By combining Hadoop distributed computing framework and P2P technology. Hadoop-based P2P video stream service in the paper solves the problem like inadequate storage and performance in traditional video service, overlength playing latency.

Keywords: Hadoop P2P Video Streaming Hybrid architecture

目 录

第一章 绪论.....1

1.1 研究背景.....1

1.1.1 传统视频流服务.....1

1.1.2 P2P 视频流服务.....1

1.1.3 云存储服务.....2

1.2 基于云平台的视频流服务研究现状.....2

1.2.1 微软 Azure 云计算流式视频服务平台.....2

1.2.2 国内视频云平台.....2

1.3 研究目的.....2

1.4 章节结构.....3

第二章 相关技术介绍.....5

2.1 Hadoop 介绍.....5

2.1.1 HDFS 组件.....5

2.1.2 MapReduce 组件.....7

2.2 P2P 技术.....7

2.2.1 P2P 技术概念.....7

2.2.2 UPnP 协议.....8

2.2.3 P2P 资源定位方式.....9

2.3 视频流技术.....9

2.3.1 视频流概述.....9

2.3.2 传输协议.....10

2.3.3 流媒体新技术.....10

2.4 本章小结.....10

第三章 系统总体设计.....11

3.1 系统框架结构.....11

3.2 客户端设计.....12

3.3 Web 服务器设计 13
3.4 Hadoop 集群设计 14
3.5 本章小结 15
第四章 系统实现 17
4.1 基于 Hadoop 的视频流服务实现 17
4.1.1 概述 17
4.1.2 视频流服务方案实现 17
4.1.3 dfs-fuse 结合 Apache 方案的实现 18
4.2 Web 服务器的实现 20
4.2.1 页面展示实现 20
4.2.2 用户交互实现 21
4.3 P2P 客户端的实现 22
4.3.1 Web 展示层实现 23
4.3.2 视频播放层实现 24
4.3.3 P2P 对等节点共享层实现 26
4.4 本章小结 27
第五章 系统测试与分析 29
5.1 测试目标 29
5.2 测试环境 29
5.3 测试过程 30
5.3.1 定点访问 Hadoop 集群中的视频资源 30
5.3.2 客户节点间 P2P 功能模块的实现 33
5.3.3 视频播放性能 35
5.4 测试结果分析 35
5.5 本章小结 35
第六章 总结 37
6.1 本文总结 37
6.2 本文的不足之处及进一步的工作 37

致 谢.....39

参考文献.....41

第一章 绪论

1.1 研究背景

随着网络的普及和网络传输速率的提升，网络视频点播业务由于其便捷和个性化的服务特点受到众多青睐。用户希望可以更自由的选择视频内容和享受更流畅的视频流服务，但同时用户数量的动态随机变化，对服务器提出了新的要求。

1.1.1 传统视频流服务

视频流[1]（Video Streaming）是指视频数据以流的形式进行的传输。当前播放网上视频、音频等媒体信息的主要方案有：先下载，下载后再播放和用流媒体的方式边下载边收看。基于传统的下载方案播放在线视频资源，需要从服务器下载整个视频文件存储到本地计算机上，然后通过播放器播放本地视频文件。而流媒体的特点在于用户无需下载整个视频资源，而是由视频流服务器实时连续向用户提供流式数据，实现边下载边播放。

目前大部分是视频流技术应用主要是基于传统的 B/S 模式，普遍存在服务器性能不足、带宽不够、资源重复建设、资源存储空间不够等问题，导致视频流服务系统能力降低。

1.1.2 P2P 视频流服务

P2P 技术自出现以来一直受到广泛关注,最近几年,P2P 技术更是发展迅速。目前,在文件共享、分布式计算、网络安全、在线交流甚至是企业计算与电子商务等应用领域 P2P 都显露出很强的技术优势。P2P 网络用户节点即可作为客户端从其他节点下载数据，也可作为服务端为其他节点提供数据下载服务。

基于 P2P 的视频流技术[2]可以通过 P2P 网络中的 Peer 节点相互间共享缓存的视频资源和带宽资源，解决带宽不够的问题，能有效缓解视频流服务器的压力，提高分布系统的服务能力。但会过多的占用客户端资源、对学习终端性能提出了更高的要求，同时也没有解决服务器瓶颈及海量数据存储的问题。

1.1.3 云存储服务

云存储是在云计算概念上延伸和发展出来的一个新的概念：它是指通过集群应用、网格技术或分布式文件系统等功能，将网络中大量各种不同类型的存储设备通过应用软件集合起来协同工作，共同对外提供数据存储和业务访问功能的一个系统。

视频资源与云存储结合[3,4]，突破了传统的资源存储集中式管理模式，它将视频资源存储在云端的各个分布的数据节点上，解决了传统模式中存储容量不足、服务性能瓶颈等一系列问题。云存储与视频流技术相结合，可以有效解决传统视频流服务的不足。

1.2 基于云平台的视频流服务研究现状

1.2.1 微软 Azure 云计算流式视频服务平台

微软 Windows Azure Media Services 是作为微软继续扩大云计算产品线的一个新产品。企业可以利用 Windows Azure Media Services 提供流式视频服务，无需自己构建基础设施。Windows Azure Media Services 可以用于向员工发布培训视频，从网站播放视频，开发与 Hulu 或 Netflix 类似的流式视频服务。

与许多云计算服务平台产品相似，Windows Azure Media Services 旨在使客户能更方便地推出流式视频服务。Windows Azure Media Services 使客户能更简便地开发能对视频编码，向各种设备和客户端发布视频的媒体发布平台。

1.2.2 国内视频云平台

视频云平台，是由 CC 视频首次提出，基于云计算架构的 SaaS 产品，集发布、转码、存储、加速、加密、管理、播放器、统计八大基础核心功能于一身的网络视频应用平台。任何一家网站只需要通过管理后台或 API 接口即可获得从视频上传到转码分发，最后获取播放统计数据的全部功能，轻松实现高品质的视频播放体验。

1.3 研究目的

传统的流媒体技术的应用主要是以 B/S 或 P2P 模式来实现的，在实际应用中还

存在诸多不足。

为降低视频点播系统的建设成本，构建高效、低价的视频资源存储系统，本文将云存储技术引入到视频点播领域，应用开源云计算架构 Hadoop 的分布式文件系统组件 HDFS 建立文件集群，存储视频点播系统的海量视频数据。并且在客户端结合 P2P 技术可以用普通节点的缓存资源为其他节点提供服务，以实现更高效，更流畅的视频流播放体验。

1.4 章节结构

本章主要叙述了课题提出的背景以及一些与传统视频流服务、P2P 视频流服务和云存储的概念，并给出了本文主要研究的基于 Hadoop 的 P2P 视频流技术的目的。

第二章总结了 Hadoop 集群、P2P 技术和视频流技术分别进行了总结和描述，强调了 Hadoop 技术、P2P 技术和视频流技术结合使用能解决传统视频流服务中存储成本高、资源过分集中导致的系统瓶颈和网络阻塞等问题。

第三章根据对系统涉及的各项技术进行分析，系统结合流媒体技术、Hadoop 集群云存储技术和 P2P 对等网络技术进行设计，给出了系统的体系结构和功能设计。

第四章介绍了基于 Hadoop 的 P2P 视频流服务的具体实现，阐述了如何基于 Hadoop 搭建视频流服务和如何在客户端实现 P2P 功能共享视频缓存资源。

第五章主要描述了对第四章搭建的系统进行运行和测试，对运行结果和测试结果进行分析，寻找系统中存在的问题，对系统进行进一步的优化。

最后总结了全文，指出了课题研究及开发过程中的不足和进一步的研究方向。

第二章 相关技术介绍

本章将介绍本文的技术基础 Hadoop 集群、P2P 技术和流媒体技术。本文系统设计中使用 Hadoop 集群中的 HDFS 组件存储视频文件；在用户节点间实现 P2P 功能模块，达到视频缓存资源共享的效果；而流媒体技术是用于播放视频流数据。

2.1 Hadoop 介绍

Hadoop[5,6,7]是一个开发和运行处理大规模数据的软件平台，是 Apache 的一个用 java 语言实现的开源软件框架，实现在大量计算机组成的集群中对海量数据进行分布式计算。但是 Hadoop 是以一种可靠、高效、可伸缩的方式进行处理的。Hadoop 是可靠的，因为它假设计算元素和存储会失败，因此它维护多个工作数据副本，确保能够针对失败的节点重新分布处理。Hadoop 是高效的，因为它以并行的方式工作，通过并行处理加快处理速度。Hadoop 还是可伸缩的，能够处理 PB 级数据。此外，Hadoop 依赖于社区服务器，因此它的成本比较低，任何人都可以使用。

Hadoop 框架中最核心设计就是：HDFS 和 MapReduce。HDFS 提供了海量数据的存储，MapReduce 提供了对数据的计算。本文中会主要使用到 HDFS 组件，对海量视频资源进行存储，并对客户端节点提供资源的定点访问。

2.1.1 HDFS 组件

HDFS[8]（Hadoop Distributed File System）具有高容错性，并且可以被部署在低价的硬件设备之上。HDFS 是典型的主从式文件分布系统，该文件系统效仿 Google 的 GFS 文件系统设计而成。HDFS 很适合那些有大数据集的应用，并且提供了对数据读写的高吞吐率。

对外部客户机而言，HDFS 就像一个传统的分级文件系统。可以创建、删除、移动或重命名文件，等等。如图 2.1 所示，HDFS 的架构是基于一组特定的节点构建的，这是由它自身的特点决定的。HDFS 集群是一个主从式（master/slave）的结构模型，就通常的部署来说，在 master 上只运行一个 NameNode，它在 HDFS 内部提供元数据服务；而在每一个 slave 上运行一个 DataNode，它为 HDFS 提供存储块。NameNode 是一个通常在 HDFS 实例中的单独机器上运行的软件，由于仅存在一个

NameNode，因此这是 HDFS 的一个缺点（单点失败）。

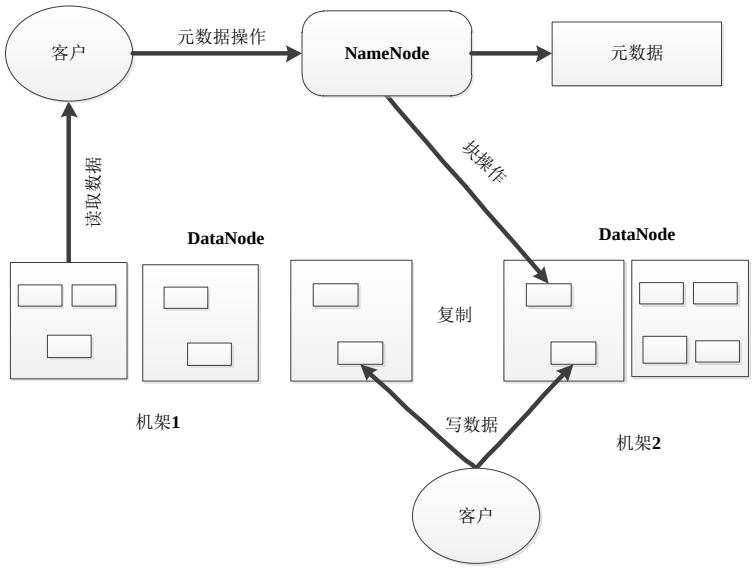

图 2.1 Hadoop 集群框架

HDFS 内部的所有通信都基于标准的 TCP/IP 协议。存储在 HDFS 中的文件被分成块，然后将这些块复制到多个计算机中（DataNode）。这与传统的 RAID 架构大不相同。块的大小（通常为 64MB）和复制的块数量在创建文件时由客户机决定，NameNode 可以控制所有文件操作。如果客户机想将文件写到 HDFS 上，首先需要将该文件缓存到本地的临时存储。如果缓存的数据大于所需的 HDFS 块大小，创建文件的请求将发送给 NameNode。NameNode 将以 DataNode 标识和目标块响应客户机。同时也通知将要保存文件块副本的 DataNode。当客户机开始将临时文件发送给第一个 DataNode 时，将立即通过管道方式将块内容转发给副本 DataNode。客户机也负责创建保存在相同 HDFS 名称空间中的校验和（checksum）文件。在最后的文件块发送之后，NameNode 将文件创建提交到它的持久化元数据存储（在 EditLog 和 FsImage 文件）。DataNode 响应来自 HDFS 客户机的读写请求。它们还响应创建、删除和复制来自 NameNode 的块的命令。需要进行文件操作时，客户端联系 NameNode 节点获取文件的元数据或修饰属性；真正的文件 I/O 操作是直接和 DataNode 节点进行交互实现的。

图 2.2 形象的描述了 HDFS 采取的副本策略，其目的是为了提高系统的可靠性，可用性。HDFS 的副本放置策略是三个副本，一个放在本节点上，一个放在同一机架中的另一个节点上，还有一个副本放在另一个不同的机架中的一个节点上。如图 2.2 中的文件 data1 被分成 3 块, 这 3 块以冗余镜像的方式分布在不同的机器中。

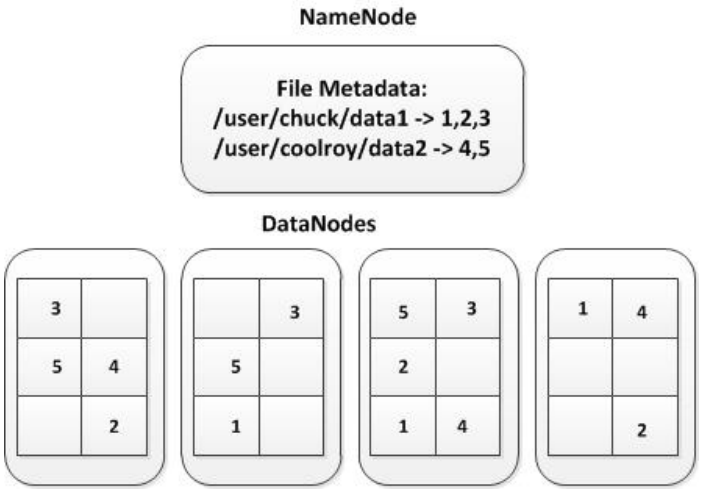

图 2.2 HDFS 数据存储副本策略

2.1.2 MapReduce 组件

Hadoop 的最常见用法之一是 Web 搜索。虽然它不是惟一的软件框架应用程序，但作为一个并行数据处理引擎，它的表现非常突出。MapReduce 是 Google 的一项重要技术，它是一个编程模型，用以进行大数据量的计算。对于大数据量的计算，通常采用的处理手法就是并行计算。最简单的 MapReduce 应用程序至少包含 3 个部分：一个 Map 函数、一个 Reduce 函数和一个 main 函数。main 函数将作业控制和文件输入/输出结合起来。在这点上，Hadoop 提供了大量的接口和抽象类，从而为 Hadoop 应用程序开发人员提供许多工具，可用于调试和性能度量等。

MapReduce 本身就是用于并行处理大数据集的软件框架。MapReduce 的根源是 Functional Programming（函数式编程）中的 map 和 reduce 函数。它由两个可能包含有许多实例（许多 Map 和 Reduce）的操作组成。Map 函数是把一组数据一对一的映射为另外的一组数据，其映射的规则由一个函数来指定，比如对[1, 2, 3, 4]进行乘 2 的映射就变成了[2, 4, 6, 8]。Reduce 函数是对一组数据进行归约，这个归约的规则由一个函数指定，比如对[1, 2, 3, 4]进行求和的归约得到结果是 10，而对它进行求积的归约结果是 24。

2.2 P2P 技术

2.2.1 P2P 技术概念

随着 P2P 技术（peer-to-peer 技术，计算机对等网络技术）的研究的不断深入

和应用，P2P 技术被广泛应用于网络视频、文件共享、网络电话等领域。P2P 是一种分布式网络，网络的参与者共享他们所拥有的一部分硬件资源（处理能力、存储能力、网络连接能力、打印机等），这些共享资源需要由网络提供服务和内容，能被其它对等节点(Peer)直接访问而无需经过中间实体。在此网络中的参与者既是资源（服务和内容）提供者（Server），又是资源（服务和内容）获取者（Client）。

P2P 与传统的 C／S 模式的最大区别是网络中的每个对等节点的地位都是对等的，每个节点既是资源的拥有者，也是资源的访问者。节点在为其它节点提供共享资源的同时也享用任何其它节点所提供的资源。以分布式资源共享和并行传输的特点，为用户提供了更多的资源、更高的可用带宽以及更好的服务质量。

2.2.2 UPnP 协议

UPnP[9]是通用即插即用（Universal Plug and Play）的缩写，它主要用于实现设备的智能互联互通。使用 UPnP 协议不需要设备驱动程序，因此使用 UPnP 建立的网络是介质无关的,它可以运行在几乎所有的操作系统平台之上,可以使用 C,C++,JAVA 和 VB 等开发语言，使得在办公室、家庭和其他公共场所方便地构建设备相互联通的网络环境。

UPnP 是实现智能设备端到端网络连接的结构。它也是一种架构在 TCP/IP 和 HTTP 技术之上的，分布式、开放的网络结构，以使得在联网的设备间传递控制和数据。UPnP 技术实现了控制点、设备和服务之间通讯的支持，并且设备和相关服务也使用了 XML 定义并公布出来。使用 UPnP，设备可以动态加入网络，自动获得一个 IP 地址，向其他设备公布它的能力或者获知其他设备的存在和服务，所有这些过程都是自动完成的，此后设备能够彼此直接通讯。

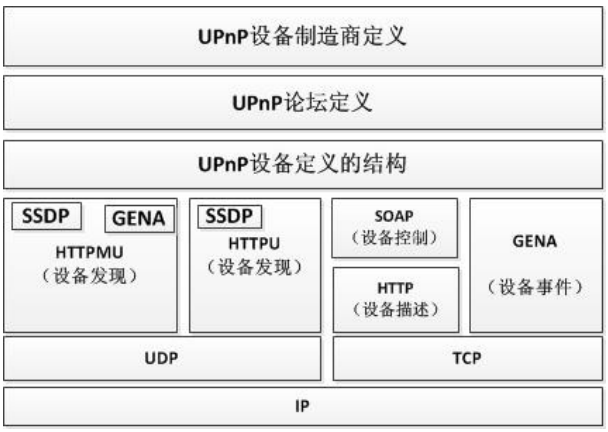

图 2.5 UPnP 设备协议栈

如图 2.5UPnP 设备协议栈所示，完整的 UPnP 由设备寻址、设备发现、设备描述、设备控制、事件通知和基于 Html 的描述界面几部分构成。其中，本文主要通过实现了 UPnP 协议栈中设备发现、设备描述和事件通知三个部分的功能，使得客户端节点实现 P2P 模式，节点间互联互通，缓存资源共享。

2.2.3 P2P 资源定位方式

P2P 网络中进行资源定位是首先要解决问题。一般采用三种方式：

1. 集中方式索引

每一个节点将自身能够提供共享的内容注册到一个或几个集中式的目录服务器中。查找资源时首先通过服务器定位，然后两个节点之间再直接通讯。例如早期的 Napster。这类网络实现简单，但往往需要大的目录服务器的支持，并且系统的健壮性不好。

2. 广播方式

没有任何索引信息，内容提交与内容查找都通过相邻接节点直接广播传递。例如 Gnutella。一般情况下，采取这种方式的 P2P 网络对参与节点的带宽要求比较高。

3. 动态哈希表的方式

上述两种定位方式可以依据不同的 P2P 应用环境进行选择，但是人们普遍看好 DHT(Distributed Hash Table，分散式杂凑表)方式。基于 DHT 的 P2P 网络在一定程度上可以直接实现内容的定位。目前大多数 DHT 方式的 P2P 网络对节点所提供共享内容的表示都很简单，一般仅仅为文件名。

2.3 视频流技术

2.3.1 视频流概述

所谓视频流是指视频数据的传输，例如，它能够被作为一个稳定的和连续的流通过网络处理。流媒体应用使得用户不需要花费很长时间将视频数据全部下载到本地后才能播放，而仅需将起始几秒的数据先下载到本地的缓冲区中就可以开始播放，后面收到的数据会源源不断输入到该缓冲区，从而维持播放的连续性。流媒体系统要比下载播放系统复杂得多，所以需要将视频的编解码和传输技术很好地结合在一起，才能确保用户在复杂的网络环境下也能得到较稳定的播放质量。

2.3.2 传输协议

流媒体在实时应用中，根据当前的网络状况和用户的终端参数，多媒体数据是一边被编码一边被流媒体服务器传输给用户。生成的码流还需要进一步打包成为特定网络传输协议的数据包用于网络传输，由于现在许多网络并不能保证传输的数据能够及时并完全正确地被用户收到，传输的数据包可能需要加前向纠错编码（FEC）来保护，经过这些处理后多媒体数据就可以通过网络传输给用户，目前常用的传输协议有 HTTP 渐进下载、RTP/RTSP 和 HTTP Live Streaming 协议[10,11]。

2.3.3 流媒体新技术

1. 高效的编码技术

流媒体系统中的多媒体数据要通过网络来传输给用户，高效的编码技术可以极大地降低流媒体系统对网络带宽的要求。目前标准化和商业化的视频编码技术都是基于运动补偿和 DCT 变换的。

2. 可伸缩性编码技术

在流媒体应用中需要解决的一个基本问题是网络带宽的波动。目前在流媒体系统中所用的编码技术都是生成固定码率的码流，它们很难适应如此复杂的网络带宽的波动。

3. 对等网络技术（P2P）

P2P 技术应用到流媒体[12]，每个流媒体用户也是一个 P2P 中的一个节点，在目前的流媒体系统中用户之间是没有任何联系的，但是 P2P 技术让用户可以根据他们的网络状态和设备能力与一个或几个用户建立连接来分享数据，这种连接能减少服务器的负担和提高每个用户的视频质量。

2.4 本章小结

本章介绍了 Hadoop 集群中 HDFS 及 MapReduce 组件，P2P 技术和视频流、流媒体技术的相关知识，并阐述了 Hadoop 技术、P2P 技术与视频流技术结合的优势。这些都为本文设计的基于 Hadoop 的 P2P 视频流技术的系统框架架构、系统功能实现方案的设计提供了理论基础。

第三章 系统总体设计

系统框架的设计对任何一个系统的成功实现都是非常重要的，通过对基础理论知识的学习与研究以及对用户的需求进行详细分析，我们可以得到用户的终端模型以及系统功能模块图。

3.1 系统框架结构

本文根据分析传统视频流服务中存在的问题，P2P 技术应用及云计算框架 Hadoop 的海量存储和超级计算的优势[13,14]，设计基于 Hadoop 的 P2P 视频流技术视频流服务方案，搭建具有 P2P 功能的基于 Hadoop 的视频流服务系统。

本系统的主要功能是实现基于 Hadoop 存储视频流资源[15]，视频流服务器搭建在虚拟集群上提供视频点播服务；网络节点间实现 P2P 技术，对等节点视频缓存资源共享。视频资源与 Hadoop 云存储的结合，可以有效解决传统视频流服务的不足。由于系统采用集中目录结构式网络结构，因此，本系统分为基于 Hadoop 的视频流服务器和对等节点两部份。系统结构如图 3.1 所示：

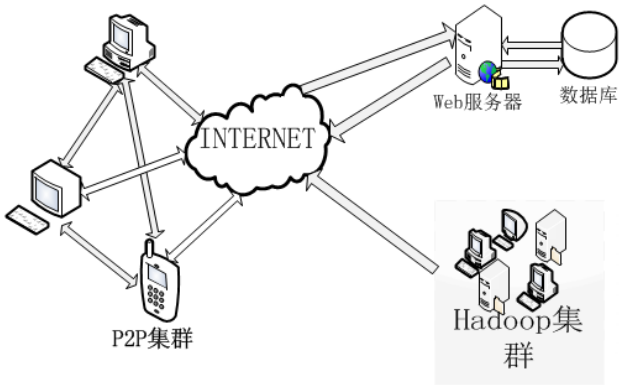

图 3.1 系统架构图

图 3.1 是本文基于 Hadoop 的 P2P 视频流服务系统的系统架构图。在传统的视频流服务中，存在海量视频资源或大容量视频文件对存储空间的需求导致服务端的存储容量不足和存储成本昂贵等问题。Hadoop 集群通过将大量廉价计算机的存储能力整合起来实现 HDFS 集群能够提供廉价且大量的存储空间。因此在本文的系统架构设计中，采用搭建 Hadoop 集群存储和管理海量视频资源。在视频流服务中，客户端视频播放的流畅性和稳定性一直是良好的视频流服务标准，传统的视频流服务中常常发生视频音像马赛克现象或视频延迟过长等性能问题。而 P2P 模式主要是

利用客户端中部分的存储能力和网络带宽，在对等节点间共享缓存资源信息以实现更高效的数据传输和更好的服务质量。因此在系统的客户端设计中，融入完全分布式 P2P 设计模式，在客户端中实现 P2P 功能以实现节点间互联互通，缓存资源共享，达到更高效的获取视频资源信息提高视频播放质量。

在图 3.1 展示的系统架构中，主要由 Hadoop 集群、Web 服务器和客户端组成，其中每个主要部分的功能描述如下：

1. Web 服务器

通过 Web 页面呈现视频流服务提供的视频节目信息，视频节目的信息保存在 Web 服务器的数据库中。主要是响应客户端的播放请求，返回视频节目在 Hadoop 集群中的网络地址。

2. 客户端

负责显示视频流服务 Web 网页，视频信息的检索，播放请求，播放及播放界面的控制。客户端响应对等网络中其他节点的缓存资源访问，及传输。

3. Hadoop 集群

Hadoop 集群主要是使用其分布式文件存储系统（HDFS）存储视频资源，接受来自客户端的资源请求。

在本文所设计的系统架构中，完成一次简单的视频服务流程如下：客户端用户选择某一视频节目 V1 后，通过 Http 协议向 Web 服务器请求视频节目的详细信息；Web 服务器对用户的请求进行反馈，将视频节目的详细信息从数据库中检索出来，反馈给客户；客户端运行流媒体播放软件，并通过 URL 直接向 Hadoop 集群各个存储节点申请访问视频资源。同时，客户端节点访问其他在线节点缓存视频资源，过滤得到视频节目 V1 的缓存；客户端将接收到的视频流数据进行播放，及响应客户的其他播放操作；客户端随时响应对等网络节点的缓存资源访问和传输。

3.2 客户端设计

本文设计的客户端是基于 android 系统的机顶盒设备，选择 android 系统的机顶盒是由于移动终端设备的普及性，价廉且配置不低。而且 android 系统的开放性，使得 android 设备和 android 应用有着广大的消费者和厂商。因视频流服务系统采用 B/S 模式，本文设计的客户端主要可以展示服务端提供的 web 服务，同时可以播放

服务端上提供的视频；客户端对等节点之间缓存资源的搜索和传输。

如图 3.2 所示，在客户端的架构设计中，客户端主要实现视频流服务的视频资源浏览及获取，并实现完全分布式 P2P 中的节点发现模块和缓存资源描述及获取模块，通过结合来自服务端和对等节点中的视频流实现更高效、更好的视频播放质量。

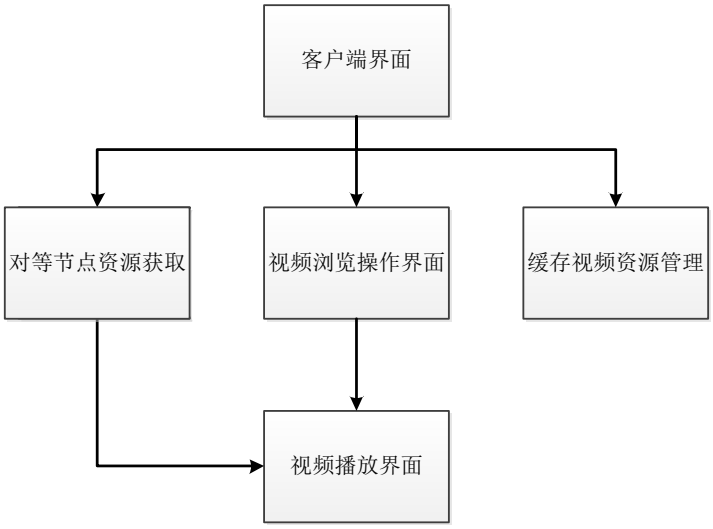

图 3.2 客户端架构

其中客户端设计中主要实现的功能模块有：

1) Web 页面展示功能，为用户提供，视频搜索和视频详细信息展示。

2) 视频播放功能，当用户在 web 页面点击播放按钮之后，应用程序需要响应播放事件来播放视频。这时候就要使用客户端的视频播放功能，将服务端提供的视频流展现出来。

3) 对等网络节点资源读取功能，当用户在选择播放某一个视频资源后，客户端会在当前对等网络列表中，搜索和过滤视频缓存资源，获取当前视频的缓存信息，并通过 Http 协议获得缓存资源到播放缓冲区中。

4) 缓存资源的描述。

5) 响应其他对等网络节点访问本地缓存资源。

3.3 Web 服务器设计

本文主要研究 Hadoop 集群技术，P2P 对等网络技术与视频流结合的研究，因此本文设计的 Web 服务器主要功能是对视频信息管理服务，具体负责向用户发布视频节目信息，同时对 Hadoop 集群视频文件的管理。

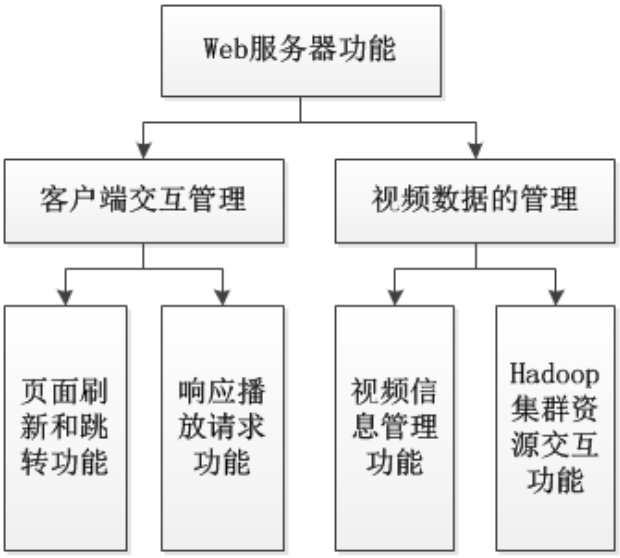

图 3.3 Web 服务器功能结构

图 3.3 是本系统中 Web 服务器模块的功能结构图，通过图 3.3 可以看出在本系统中，对 Web 服务器功能的需求进行了弱化。Web 服务器主要负责向客户端展示视频资源信息，通过 JavaScript 语言与客户端中的 WebView 控件进行交互，完成视频点播服务的基本流程；Web 服务器可以实现对存储在 Hadoop 集群中的视频文件的存储位置信息进行管理；Web 服务器可以响应用户播放视频的请求，并向用户反馈视频资源在 Hadoop 集群中的访问 URL。

3.4 Hadoop 集群设计

在本章的系统架构中，Hadoop 集群的模块主要是为了解决视频流服务中服务端视频资源存储容量不足、存储成本昂贵等问题。Hadoop 集群非常适合用于一次写入，多次读取的场合，这适好与视频流服务的应用场合相符合。通过搭建 Hadoop 集群实现将大量廉价计算机的存储能力整合起来实现 HDFS 集群，大量廉价计算机组成的分布式文件系统存储海量的视频文件能够有效的解决存储空间和存储成本的问题。为了实现更好的视频流服务，本系统中 Hadoop 集群直接为客户端提供视频流服务，客户端可以直接访问存储在集群中的视频资源并进行定点访问。

在设计中，Hadoop 集群需要跟 Web 服务器、客户端节点均有直接的交互过程。Web 服务器端可以管理 Hadoop 集群中的视频资源，而客户端可以直接访问存储在 Hadoop 集群中的视频资源数据并进行定点访问，实现流式传输视频资源。图 3.4 主要描述本文设计的系统中 Web 服务器、客户节点和 Hadoop 集群的相互关系：

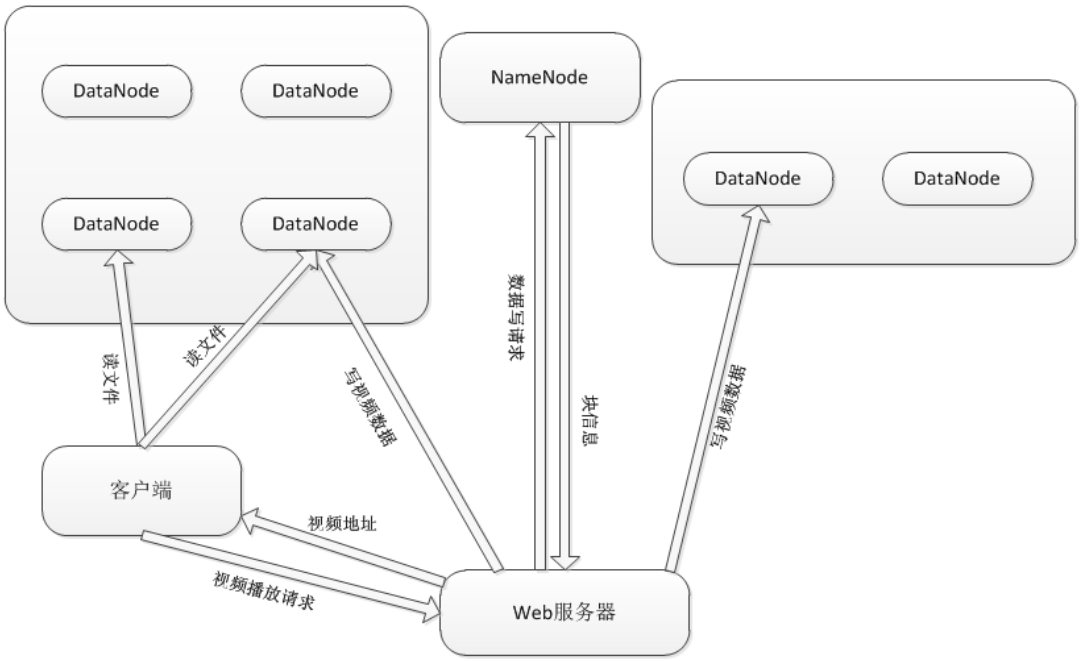

图 3.4 Web 服务器、客户端与 Hadoop 集群的关系图

如图 3.4 所示，Web 服务器、客户端与 Hadoop 集群的关系，对于 Hadoop 集群来说，Web 服务器和用户节点都是它的客户，但两个客户的操作非常不同：Web 服务器可以将视频文件上传到 hadoop 集群，或将不需要的视频文件从 Hadoop 集群中删除，对于 Hadoop 集群，Web 服务器是一个从事写操作的客户；用户节点访问 Hadoop 集群，并从 Hadoop 集群读取视频文件，进行播放。并且在播放过程中，如果用户拖动进度条或点击快进按钮，客户端可以对 Hadoop 集群中的视频资源进行定点访问，达到良好的播放体验。对于 Hadoop 集群用户节点是一个从事读操作的客户。

3.5 本章小结

本文系统的设计结合了视频点播技术、Hadoop 集群云存储结束以及 P2P 技术，具有技术成熟、高可靠性、扩展性强、投资经济、播放流畅等特点。

系统结合流媒体技术、Hadoop 集群云存储技术和 P2P 对等网络技术进行设计，流媒体技术、P2P 对等网络技术成熟，而 Hadoop 技术也通过了上千个节点的运行测试，并且成功地应用于多个知名大型系统中，成熟可靠的技术是本系统可靠性的强大保证。视频文件在 Hadoop 集群中以多副本的形式保存，提高文件存储的可靠性。P2P 对等网络节点间的视频缓存资源共享，增强了视频播放的流畅。

Hadoop 强大的存储可扩展能力，可以根据需要随时扩展 Hadoop 集群的规模，

增加系统的视频文件存储能力。因此可以充分利用现有的廉价硬件资源，节省系统建设成本。

第四章 系统实现

通过对第三章中系统框架进行进一步的分析与理解，本系统的主要功能是实现具有 P2P 功能的基于 Hadoop 的视频流服务[16]。系统利用 Hadoop 集群存储、管理视频流资源，视频流服务器搭建在虚拟集群上提供视频点播服务；并实现网络节点间视频缓存资源共享。视频资源与 Hadoop 云存储的结合，可以有效解决传统视频流服务的不足。由于系统采用集中目录结构式网络结构，因此，本系统重点实现基于 Hadoop 的视频流服务器和对等节点两部分。

4.1 基于 Hadoop 的视频流服务实现

4.1.1 概述

本文设计的基于 Hadoop 的 P2P 视频流技术解决方案，主要应用 Hadoop 集群中的 HDFS 分布式文件系统来存储视频流资源，并对客户端节点提供视频资源的访问。使用 HDFS 来存储视频流资源，而 HDFS 原则上设计只能通过 Hadoop 的用户进行访问，并且无法实现定点访问。

4.1.2 视频流服务方案实现

为了实现标准的视频流服务，对存储在 HDFS 中的视频文件进行定点访问，本文对两种可能的解决方案进行了研究分析：Hoop（Hadoop 服务器）和 Apache/FUSE[17]。

Hoop 是一种 HTTP-HDFS-Connector。它允许 HDFS 文件系统通过 HTTP 协议去访问，以及开发了一种非在线服务的本地原型，使用 JW 播放器和一个大容量视频文件。这样，就能实现流媒体应用了，但在视频播放过程中，跳转到一个没有缓存的节点时播放会停止，这对视频流服务的体验具有严重的限制。这也说明了 Hoop API 不支持在文件中跳转，所以本文系统实现中不采用这个方法。

第二种解决方案是基于 HDFS/FUSE。FUSE（用户空间文件系统）是一系列 API 捕获文件系统操作并实现，通过特殊的功能运行在用户的活动空间。FUSE 在 Hadoop 中以组件的形式存在，命名为“Mountable HDFS”，允许标准文件系统用户或程序把 HDFS 命名空间作为一个本地挂载的目录。所有的文件系统操作，包括目

录浏览，打开文件和内容访问，都可以通过 FUSE 接口访问 HDFS 内容。并且测试显示基于 HDFS/FUSE 的解决方案很好的适用于视频流服务中。因此本文使用 dfs-fuse 结合 Apache 实现对 HDFS 文件中的定点访问，搭建视频流服务器。

由于 Hadoop 集群是一个非在线服务分布式架构，Hadoop 集群外的节点无法直接访问存储在集群中的资源，也无法对集群中的资源进行定点访问。因此通过实现 dfs-fuse 结合 Apache 方案，使得外部节点可以通过访问 Apache 目录对存储在 HDFS 的文件直接进行访问。因此在本系统中 Web 服务器响应客户端的视频播放请求，所反馈的视频资源地址也就是 Apache 下面的 dfs 目录地址，实现客户端直接访问 Hadoop 集群中存储的视频资源。

4.1.3 dfs-fuse 结合 Apache 方案的实现

1. 搭建 fuse

在系统搭建中，本文使用的是 fuse-2.8.5，其中并不是 fuse 的版本越高越好，在搭建过程中我们尝试过使用高版本 fuse，但实现结果存在很多不足的地方，如资源定点访问出现错误等问题。

2. 设置环境变量

为了实现对 Hadoop 集群的定点访问，集群中需要使用 ant，并需要对其环境变量进行修改且不同的操作系统，设置并不完全相同。编辑文件/etc/profile（或其它环境变量文件），在其底部添加当前 Hadoop 虚拟机所使用的操作系统位数，并设定 Java 路径变量和 Hadoop 路径变量，可以通过 Hadoop 路径访问编译环境 build 中 libhdfs 的内容：

```
export OS_ARCH=i386   //如果你的系统是 64 位的，应该写成 am64
export OS_BIT=32              // 如果系统是 64 位的，这里应该写 64
export LD_LIBRARY_PATH= $JAVA_HOME/jre/lib/ $OS_ARCH/server:
${HADOOP_HOME}/build/c++/Linux-$OS_ARCH-$OS_BIT/lib:/usr/local/lib:/u
sr/lib: ${HADOOP_HOME}/build/libhdfs
```

3. libhdfs 的制作

在本文的系统实现中，使用的操作系统是 32 位，java 版本也是 64 位，所以编译之前修改$HADOOP/src/c++/libhdfs/Makefile,in，向其中添加 OS_ARCH=i386，并修改下面两项为：

```
LDFLAGS = -L$(JAVA_HOME)/jre/lib/$(OS_ARCH)/server -ljvm -shared -m32 -Wl,-x
CPPFLAGS = -m32 -I$(JAVA_HOME)/ include -I$(JAVA_HOME)/ include/ $(PLATFORM)
```

我们需要重新编译 libhdfs，在 libhdfs 编译完成后，我们能够在$HADOOP_HOME/c++/Linux-$OS_ARCH-OS_$BIT/lib/目录下找到 libdhfs.so 等一系列文件，将这些文件复制到$HADOOP_HOME/build/libhdfs 下。

4. fuse 连接 dfs

本文搭建 Hadoop 集群所使用的 hadoop 版本号为 0.20.203.0，在实现过程中该版本中 fuse 存在一个 bug，需要先修改掉才能继续编译完成后续的工作。打开$HADOOP_HOME/src/contrib/fuse-dfs/src/fuse_connect.c，需要找到 hdfsFS fs = hdfsConnectAsUser(hostname, port, user, (const char **)groups,numgroups)，并将其修改为 hdfsFS fs = hdfsConnectAsUser(hostname, port, user)，然后保存退出。执行下面命令，可以将 fuse 和 dfs 连接起来。

```
xidian@master:~$ cd $HADOOP_HOME
xidian@master:~/hadoop-0.20.203.0$ sudo ant compile-contrib -Dlibhdfs=1 -Dfusedfs=1
```

5. 配置 fuse

重新部署 Hadoop 集群，将 master 节点上的 Hadoop 发送到各个 slave 节点，启动 Hadoop 集群。再修改$HADOOP_HOME/build/contrib/fuse-dfs/fuse_dfs_wrapper.sh 脚本，修改内容放置在脚本内容的最上边。

```
export JAVA_HOME=java 的根目录，根据系统来设置
export HADOOP_HOME= hadoop 的根目录，根据系统来设置
export OS_ARCH=i386 // 如果系统是 64 位的，这里应该写 amd64
export OS_BIT=32 // 如果系统是 64 位的，这里应该写 64
```

同时我们需要授予$HADOOP_HOME/build/contrib/fuse-dfs/fuse_dfs_wrapper.sh 可执行权限 chmod +x $HADOOP_HOME/build/contrib/fuse-dfs/fuse_dfs_wrapper.sh。

6. 配置 apache 服务器

本系统实现中，需要 apache 的一个扩展工具 apxs2，并需要下载 H264 的源码进行安装。在安装必要的插件和工具后，需要编辑 apache 的配置文件 (in

/etc/apache/httpd.conf)，向其中加入 H264 的模块信息，以实现直接通过 apache 提供视频流服务。

```
LoadModule h264_ streaming_ module /usr/ lib/ apache2/ modules/ mod_ h264_ streaming.so
AddHandler h264-streaming.extensions .mp4
```

7. 挂载 fuse-dfs

在 apache 下面新建目录 dfs，并将 HDFS 挂载到该目录下。

```
Sudo mkdir dfs
Cd ~/hadoop-0.20.203.0/build/contrib/fuse-dfs/
./ fuse_dfs_wrapper.sh dfs://master:9000 /var/www/dfs
```

方案实现成功后，我们可以通过访问 apache 下面的 dfs 目录，就相当于访问 HDFS 中的文件了。本系统中 Web 服务器响应客户端的视频播放请求，所反馈的视频资源地址也就是 Apache 下面的 dfs 目录地址，实现客户端直接访问 Hadoop 集群中存储的视频资源。

4.2 Web 服务器的实现

本文研究的基于 Hadoop 的 P2P 视频流服务系统，侧重研究使用 Hadoop 集群存储海量视频资源并直接向客户端节点提供视频流访问服务。Hadoop 集群中的 HDFS 分布式文件系统的廉价性、易扩展性能够有效的解决视频流服务中大量视频资源存储问题及网络阻塞问题；在客户端实现 P2P 功能，以实现客户端节点间视频缓存资源共享，增强播放流畅度。根据系统结构分析，Web 服务器主要作为客户端与 Hadoop 集群之间的一个交互界面，客户端可以通过访问 Web 页面获取 Hadoop 集群所存储的视频资源的基本信息。因此在本系统中简单化 Web 服务器的功能，仅实现页面展示及用户交互功能。页面展示主要是向客户端提供一个可视化的资源访问路径，而用户交互功能则是主要响应用户的点播操作返回视频资源的访问地址。

4.2.1 页面展示实现

页面展示功能的实现主要是为了将视频流服务系统中所提供的视频资源通过图文的形式显示给用户，提供用户浏览和选择的功能。本文主要通过 JSP 语言实现

页面显示，部分页面信息如图 4.1 所示：

幽闭末日 世界尽头

图 4.1 Web 页面视频信息展示

图 4.1 是通过 Web 页面向用户展示了当前服务端所提供的视频资源服务，包括视频海报和视频名称，用户可以根据点击 Web 页面中的不同视频选项进行点播服务。图片链接响应地址为 Apache 目录下的显示的 Hadoop 集群中的资源，这也是通过实现 dfs-fuse 结合 Apache 方案实现的功能，这样使得外部节点可以直接访问 Hadoop 集群中存储的视频资源。

本文所设计的客户端是基于 Android 平台的机顶盒，客户端中直接嵌入一个 web 页面。这样做的好处：一个是功能更新方便，维护起来容易，只需要维护服务器的页面即可，不需要更新客户端；另一个是功能通用，不仅 android 可以用，ios 也可以用，symbian 也可以直接用。

4.2.2 用户交互实现

客户端用户可以在显示的视频资源中，选择适合的资源进行播放，则 Web 页面需要对用户的操作进行交互反馈。在实现中，Web 页面与用户交互过程中反馈点播操作的消息为该视频资源的访问路径。又因客户端使用 WebView 控件内嵌 Web 服务器的页面，所以主要实现 WebView 与 JavaScript 的交互调用。

在视频单项中，Web 服务端主要的工作就是 Web 页面与用户界面进行交互响应，响应用户的视频点播操作，返回视频资源在 Hadoop 集群中的存储位置。通过 <img onClick="call_playvideo('http://192.168.3.61/tmp/videos/3040.mkv')">实现监听元素被点击的操作，调用 JavaScript 函数 call_playvideo(url)，通过 JavaScript 函数将该视频在 Hadoop 集群中的存储地址发送到客户端。

```
<script language="javascript">
    function call_playvideo(url){
        window.callplayer.playvideo(url);
    }
</script>
```

通过在 JavaScript 函数 call_playvideo(url)中调用 android 客户端程序中的 Java 函数 void playvideo(String url)，将视频资源地址传递给客户端并实现视频资源的播放。

4.3 P2P 客户端的实现

本文设计的基于 Hadoop 的 P2P 视频流服务解决方案中，客户端的主要功能是展示服务端提供的 web 服务，同时可以播放服务端上提供的视频。其中，本文设计的客户端结合了 P2P 技术思想，客户端作为对等网络节点，为其他对等网络节点共享本节点的视频缓存资源，以增强视频播放流畅性。

鉴于开源系统 Android 上的开放性，在客户端流媒体播放器实现中主要实现了视频播放开源框架 Vitamio。网络视频播放开源框架 Vitamio 与 Android 默认的 MediaPlayer 工作方式相似，但包含更加强大的功能，并且它是完全免费的，支持的格式有 Divx/Xvid, flv, rmvb, avi, mkv, wmv, mp4，提供专业级的高清支持，完美支持在线流媒体以及本地视频播放。因此，客户端的视频播放内核采用 Vitamio 内核，而在节点视频缓存资源管理和共享方面参考并实现 UPnP 协议栈，实现具有对等节点互联互通，资源共享的效果。客户端的架构设计如图 4.2 所示：

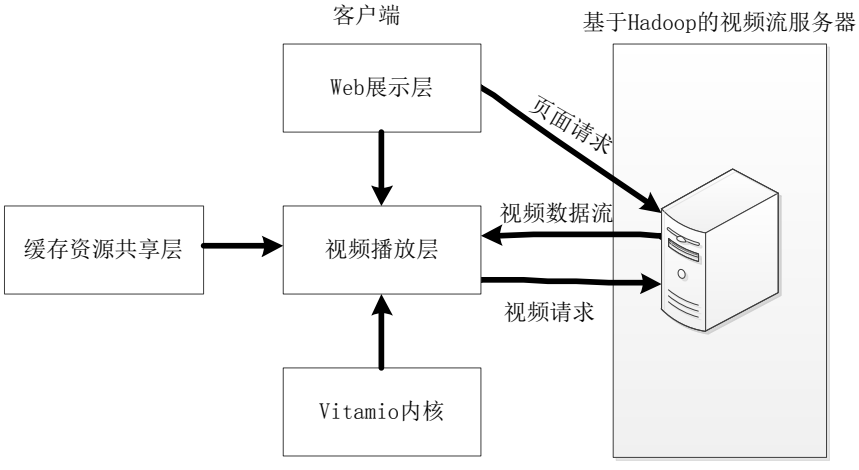

图 4.2 客户端架构图

图 4.2 中描述了客户端实现四个层次的开发，其中 Web 展示层主要负责将服务器提供的 Web 服务显示出来，并响应用户交互，视频点播请求等。而视频播放层的主要是在用户点击 web 页面中的播放按钮时响应播放事件，从视频服务器获取视频流并在解码后播放出来。Vitamio 内核是属于视频播放层的，它是视频播放层实现视频解码的核心，本文实现 Vitamio 内核中的 MediaPlayer 类和 MediaController 接口，实现具有良好用户体验的网络视频播放器。

针对完全分布式 P2P 脱离服务端参与的特点，在客户端的开发中缓存共享层作为研究和实现的重点。缓存共享层主要实现 UPnP 协议栈中的三个功能模块，使得客户端可以发现对等网络中其他在线的节点，同步对等节点的缓存资源信息，并在播放过程中，从对等节点获取缓存资源，增强播放的流畅度。而在本地，缓存层还会对本地缓存资源进行统一管理，随时响应对等网络节点的访问和资源申请。

4.3.1 Web 展示层实现

客户端设计中会先从首先项的 url 中载入 web 页面，如果载入失败的话，会弹出输入框要求输入 url，然后从新的 url 中载入 web 页面。Web 页面载入成功之后，用户开始与 web 页面交互，当用户点击播放视频之后，客户端进入视频播放层，播放完成之后，返回 web 展示层。如果用户选择退出，则客户端结束程序。

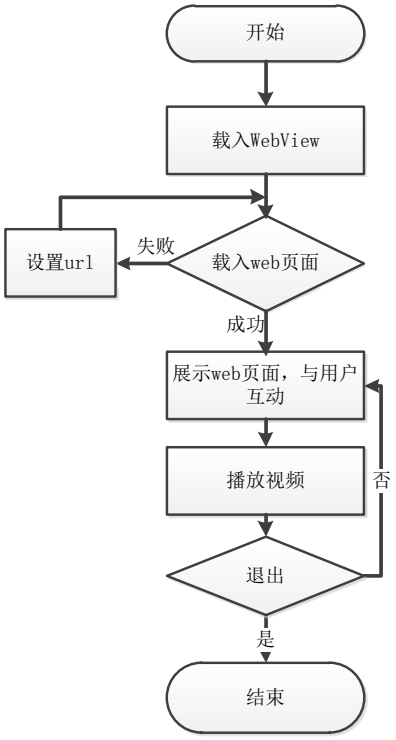

图 4.3 web 展示层工作流程图

如图 4.3 所示，Web 展示层的主要工作流程是加载 Web 服务器页面，用户根据页面提供的点播选项进行点播。Web 层会响应用户的点播操作，返回用户选择视频对应在 Hadoop 集群中的访问 IP 地址。

整个 web 展示层都是在一个 WebViewActivity 类中实现的，主要通过一个 WebView 控件将 Web 服务器中的网页嵌入客户端。这个类中的函数及内部类关系如图 4.4 所示：

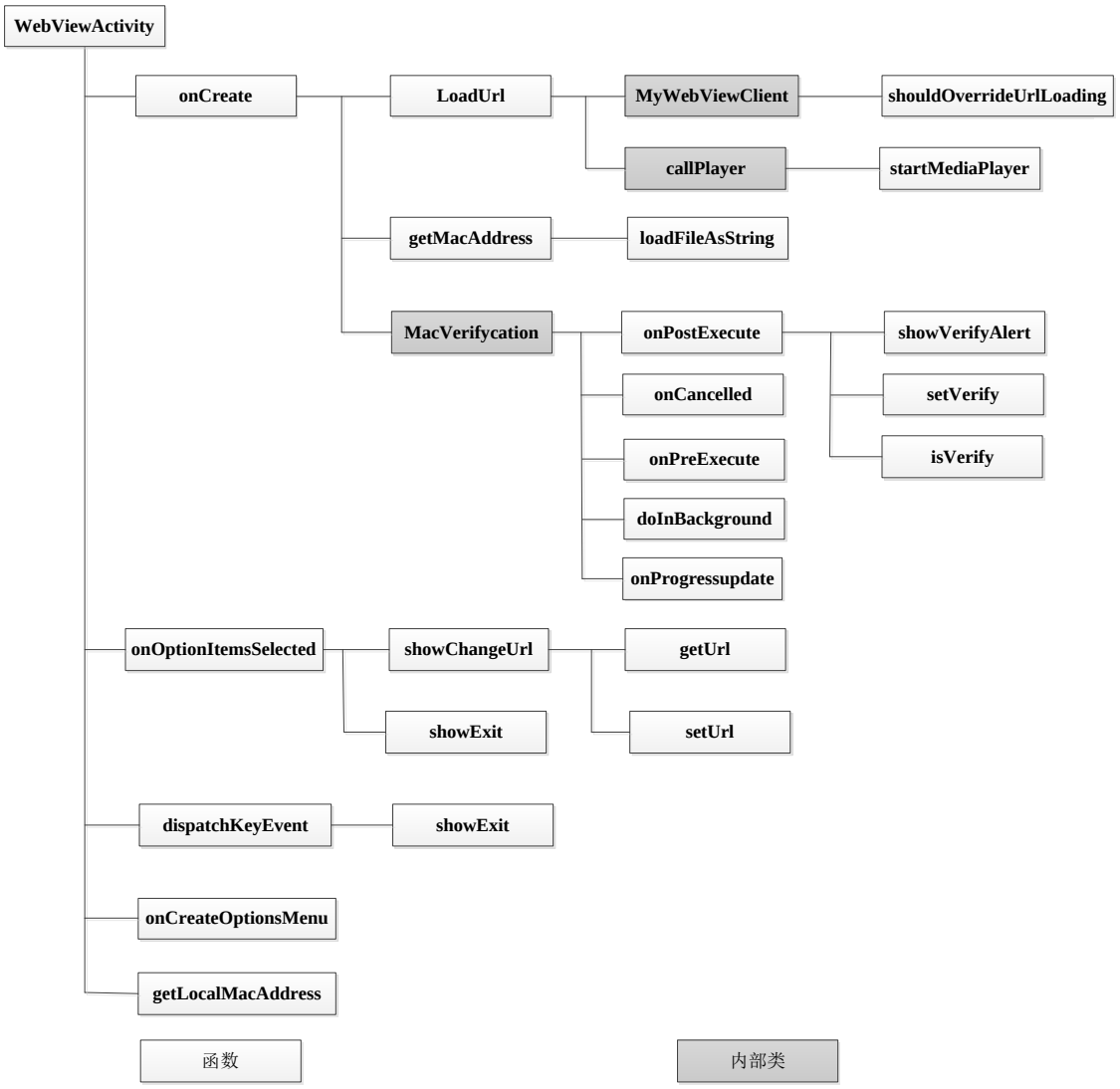

图 4.4 WebViewActivity 类内部函数关系

4.3.2 视频播放层实现

Web 展示层中，当用户选择视频资源之后，客户端将 Web 服务器反馈的视频资源地址 URL 传递给视频播放层，开始播放视频。其中 Web 服务器返回的 URL 是通过 Apache 目录下访问到视频资源在 Hadoop 集群中的存储地址，客户端可以直

接从 Hadoop 集群中以流的方式对视频资源进行访问，并在播放过程中整合在对等节点集群中获取的视频缓存资源，以达到更流畅的播放效果。视频播放层的工作流程如图 4.5 所示：

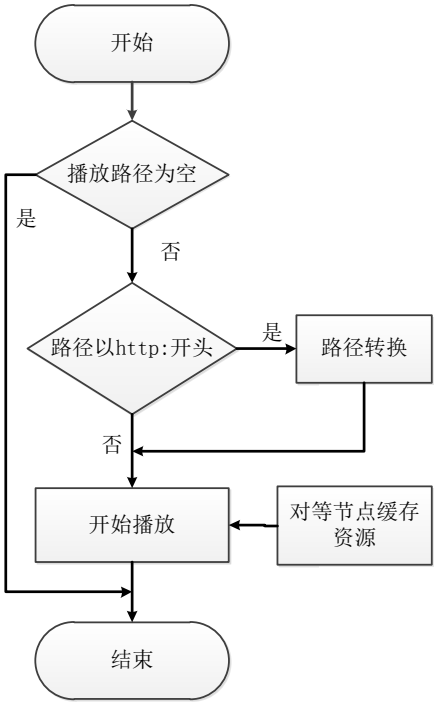

图 4.5 视频播放层流程图

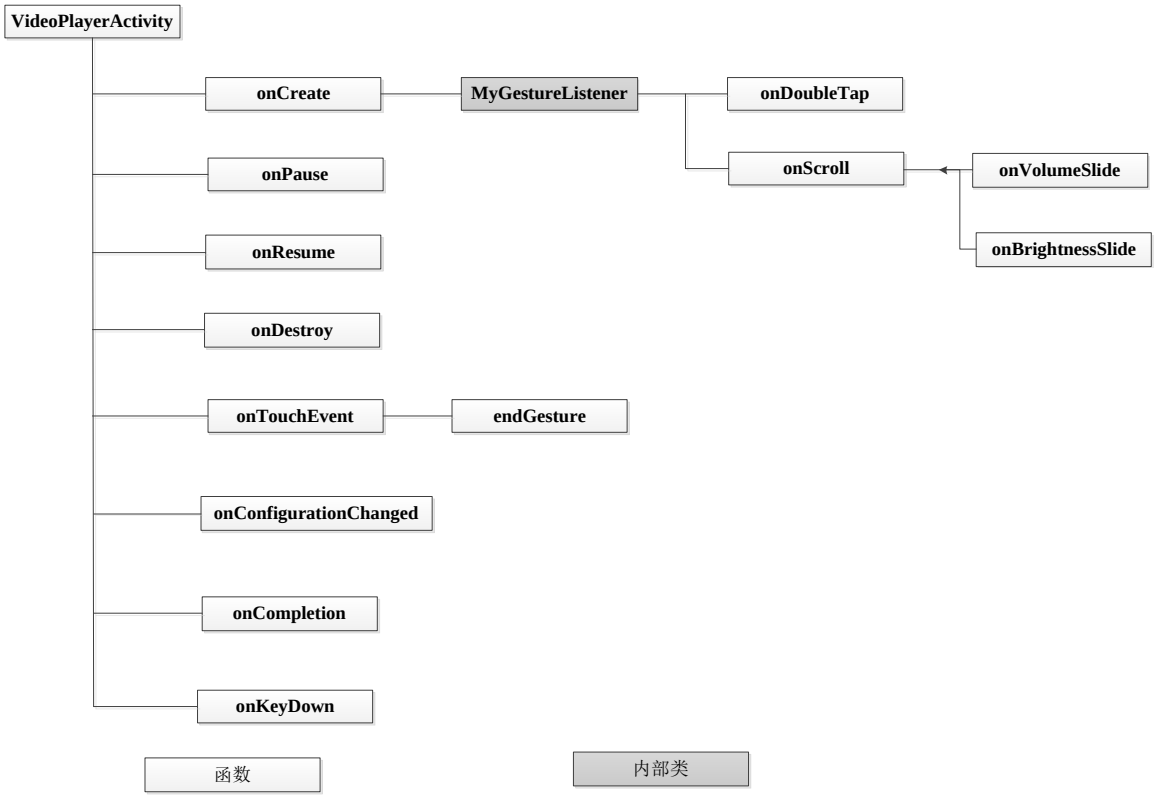

图 4.6 VideoPlayerView 类内部函数关系

图 4.6 描述了 VideoPlayerActivity 类中的主要方法及其之间的相互关系。视频播放层的功能是在一个 VideoPlayerActivity 类中实现的，其中该类调用了 Vitamio 开源库，继承了 VideoView 类实现视频的播放，同时继承 MediaController 类以实现播放过程中对视频快进快退，音量等控制操作。因为需要将 Hadoop 集群中的资源和 P2P 中的缓存资源结合，该类实现了 OnCompleteListener，OnBufferUpdateListener 等接口，在实现的接口中整合视频资源，实现播放。

4.3.3 P2P 对等节点共享层实现

P2P 技术的融入，主要是为了弱化服务端的功能，同时提供了更高效更流畅的视频流服务。在本文设计的终端客户端设计中使用 Intel 提供的 UPnP 开发包实现终端节点的互联互通，视频缓存资源同步，以及通过 HTTP 协议实现对等资源传输。其中图 4.6 描述了客户端中 P2P 功能实现的流程：

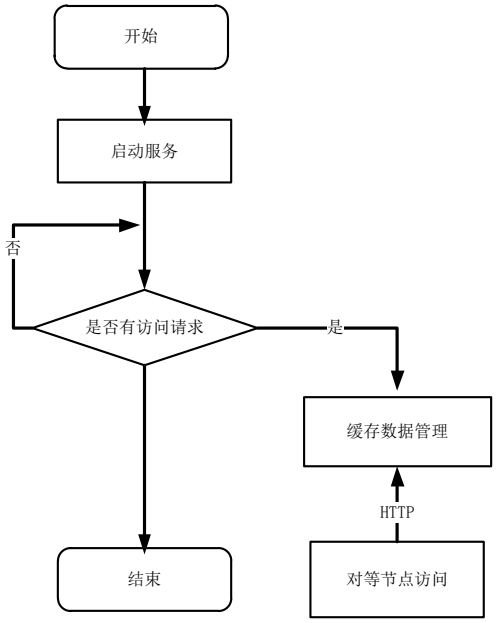

图 4.7 P2P 实现流程图

在本文设计的客户端中，使用 Cling 开源库来构建基本的 UPnP 协议架构。其中，协议处理模块、信息传输模块、XML 解析模块由 Cling-core 生成，具体实现节点发现模块、资源描述模块和资源访问模块以实现系统客户端之间对等网络视频缓存资源共享的功能：

1. 节点发现

一旦有客户端节点访问视频流服务系统（启动软件）并且分配了地址，就要进行发现的操作。节点发现是客户端利用 UPnP 网络实现 P2P 功能的第一步。节点发

现是由简单发现协议 SSDP（Simple Service Discovery Protocol）来定义的。在节点发现操作之后，节点间可以相互访问缓存资源信息，读取缓存资源等操作。在一个新节点加入网络时，那么它将传送一发现消息向对等网络中其他节点公开本地的视频缓存资源信息，并允许其他节点对缓存资源进行读取。

2. 资源描述

利用 UPnP 协议实现客户端 P2P 功能的第二步就是对节点存储的视频缓存资源进行描述。在节点间相互发现的时候，彼此对其他节点的了解甚少。可能仅仅知道节点或服务的 UPnP 类型，节点的 UUID 和节点描述的 URL 地址。为了让其他节点能找到本地缓存的视频缓存文件，其他节点会发送一个描述资源的请求。本节点在接收到请求后，会以特定的方式对视频缓存文件进行描述，如资源 ID，资源块大小，资源块对等网络中的地址等，并把结果反馈给其他节点。其他节点则可根据这描述对缓存资源进行过滤和读取。

3. 资源访问

由通过上面节点发现和节点中资源的描述，将得到对等网络中所有在线节点视频缓存资源文件对应的网络地址 URL。需要对某一特定的视频资源获取其缓存资源时，使用 HTTP 协议即可访问获取到该资源。

4.4 本章小结

本章是论文的核心，重点阐述了系统的实现原理。以第二章介绍的 Hadoop 集群技术、P2P 技术及流媒体技术相关知识为基础，以第三章介绍的系统框架设计为背景，本章探究如何结合 Hadoop 集群中 HDFS 组件提供的大量文件存储的优点实现视频流服务的解决方案，并详细介绍了解决方案的搭建过程；接着在 Web 服务实现中，简化了功能模块达到用户只需要进行简单的操作便能实现需求的效果；最后，主要阐述客户端节点实现 P2P 功能的原理及通过 P2P 进行缓存资源获取的过程。

第五章 系统测试与分析

通过第三章的分析与设计与第四章的系统搭建与实现，系统开发的功能已符合简单的视频流服务需求。本章将在局域网内对第四章搭建的基于 Hadoop 的 P2P 视频流服务系统进行测试，第一节介绍本章测试目标，对系统功能的可用性和系统性能的稳定性进行测试；第二节将详细描述系统的测试过程及测试结果。最后，对测试结果进行分析及寻找系统存在的问题，纠正系统存在的错误，提高系统的可靠性。

5.1 测试目标

本文主要工作是搭建基于 Hadoop 的 P2P 视频流服务系统，以实现将海量视频资源存储在 Hadoop 集群中，并提供客户节点直接访问 Hadoop 集群中的数据以提供视频流服务；在终端节点中实现 P2P 各个功能模块，实现节点间缓存视频资源的共享。在测试过程中，主要为了测试系统的功能模块的实现，以及基本的视频点播效果体现。测试目标输出有如下几点：

1) 能通过 Apache 目录访问存储在 Hadoop 集群中的视频资源文件。

2) 客户节点间各个 P2P 功能模块的实现，包括节点的发现，资源的描述和资源的访问。

3) 客户节点播放视频的流畅度，在快进快退的状态下能够在最短的时间内完成缓冲，并且播放过程中没有马赛克现象，或视频卡住现象出现。

5.2 测试环境

本文实现的基于 Hadoop 的 P2P 视频流服务系统主要的开发和测试环境均在局域网内完成。系统中的 Hadoop 集群的搭建是在 HP PC 上启动 3 个虚拟机，集群环境所处的宿主机软硬件配置为 Intel Xeon E5504 2.0GHz CPU，4GB 内存，64 位的 Windows 7 操作系统和 VirtualBox 4.1.4 的虚拟机工具。组成 Hadoop 集群的虚拟机配置均为 512M 的内存，8GB 的虚拟存储空间。由于测试主要是在局域网内完成，所以每个虚拟机或终端设备分配的 IP 地址均在同一个子网段上。其中 Master 节点

的 IP 地址为 192.168.3.61，两个 Slave 节点的 IP 地址分别为 192.168.3.62、192.168.3.63。

本系统的客户端主要针对 Android 系统的移动终端进行开发，在系统开发和测试的过程中使用到的终端设备如表 5.1 所示：

表 5.1 终端设备配置情况

设备型号	Android 版本号	IP 地址
三星 Galaxy Nexus	4.1.1	192.168.3.123
小米 1S	4.0.4	192.168.3.111
中兴 V880	2.2.2	192.168.3.125
华为 U880	2.2.2	192.168.3.136
天敏机顶盒	4.0.1	192.168.3.138

其中，系统主要在小米 1S 和天敏机顶盒上进行开发和测试，并且因为系统主要在局域网内进行测试，测试设备的 IP 地址与 Hadoop 集群的 IP 地址属于同一网段才能正常访问。

5.3 测试过程

5.3.1 定点访问 Hadoop 集群中的视频资源

在本文第四章的系统实现中，主要采用 dfs-fuse + Apache 实现对 HDFS 文件中的定点访问,搭建视频流服务器，因此需要对搭建的系统的可用性进行测试。

通过调用 hadoop 中 dfs 的 put 命令，将虚拟机中/home/xidian/videos/3064.mp4 视频资源文件上传到 hadoop 集群中:

```
xidian@master:~$ cd hadoop-0.20.203.0/
xidian@master:~/hadoop-0.20.203.0$ bin/hadoop dfs -put
/home/xidian/videos/3064.mp4 /videos/
xidian@master:~/hadoop-0.20.203.0$
```

其中 put 命令主要是将资源文件上传到 Hadoop 中的文件系统中。

通过浏览器访问 Hadoop 集群中的 master 节点 192.168.3.61: 50070 查看文件系统，可以找到上传到 Hadoop 集群中的视频资源文件 3064.mp4，如图 5.1 所示:

Contents of directory /videos

Goto : /videos go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
3040.mkv	file	1.02 GB	3	64 MB	2013-04-08 15:56	rw-r--r--	root	supergroup
3041.mp4	file	117.2 MB	3	64 MB	2013-04-08 16:22	rw-r--r--	root	supergroup
3042.mp4	file	165.32 MB	3	64 MB	2013-04-08 16:32	rw-r--r--	root	supergroup
3043.mkv	file	235.01 MB	3	64 MB	2013-04-08 16:41	rw-r--r--	root	supergroup
3044.mp4	file	145.68 MB	3	64 MB	2013-04-08 16:46	rw-r--r--	root	supergroup
3045.mkv	file	315.4 MB	3	64 MB	2013-04-08 16:51	rw-r--r--	root	supergroup
3046.mkv	file	278.18 MB	3	64 MB	2013-04-08 16:55	rw-r--r--	root	supergroup
3047.mkv	file	185.74 MB	3	64 MB	2013-04-08 17:00	rw-r--r--	root	supergroup
3050.mkv	file	225.86 MB	3	64 MB	2013-04-08 18:03	rw-r--r--	root	supergroup
3051.mp4	file	117.13 MB	3	64 MB	2013-04-08 18:12	rw-r--r--	root	supergroup
3052.mp4	file	247.25 MB	3	64 MB	2013-04-08 18:18	rw-r--r--	root	supergroup
3053.mkv	file	43.63 MB	3	64 MB	2013-04-08 18:39	rw-r--r--	root	supergroup
3054.mkv	file	3.21 MB	3	64 MB	2013-04-08 18:44	rw-r--r--	root	supergroup
3064.mp4	file	61.13 MB	2	64 MB	2013-05-22 23:10	rw-r--r--	xidian	supergroup

图 5.1 Hadoop 集群中文件系统在 videos 目录下的视频资源情况

本文中在 master 节点上我们搭建了 Apache 服务器，可以通过在 Apache 目录下查看访问到存储在 Hadoop 集群中的视频资源，因此在于 master 在同一个子网段的任何机器上，打开浏览器，访问 192.168.3.61 的 Apache 目录 tmp/videos，可以查看到存储在 Hadoop 集群中所有的视频资源文件。其中 Apache 访问到 Hadoop 集群中的视频资源情况如图 5.2 所示：

http://192.168.3.61/tmp/videos/

Index of /tmp/videos

Name	Last modified	Size	Description
Parent Directory		-	
3040.mkv	08-Apr-2013 15:56	1.0G	
3041.mp4	08-Apr-2013 16:22	117M	
3042.mp4	08-Apr-2013 16:32	165M	
3043.mkv	08-Apr-2013 16:41	235M	
3044.mp4	08-Apr-2013 16:46	146M	
3045.mkv	08-Apr-2013 16:51	315M	
3046.mkv	08-Apr-2013 16:55	278M	
3047.mkv	08-Apr-2013 17:00	186M	
3050.mkv	08-Apr-2013 18:03	226M	
3051.mp4	08-Apr-2013 18:12	117M	
3052.mp4	08-Apr-2013 18:18	247M	
3053.mkv	08-Apr-2013 18:39	44M	
3054.mkv	08-Apr-2013 18:44	3.2M	
3064.mp4	22-May-2013 23:10	61M	

Apache/2.2.14 (Ubuntu) Server at 192.168.3.61 Port 80

图 5.2 Apache 目录下访问 Hadoop 文件系统的视频资源情况

在本文搭建的系统中 Web 服务器作为系统与用户之间的交互界面，当添加或删除一个或多个视频资源服务后，修改 Web 服务器中显示的视频资源信息，以供用

户选择点播。添加 3064.mp4 视频资源后，在 Web 服务器中添加相应的显示界面。

```
<a href="#"
onClick="call_playvideo('http://192.168.3.61/tmp/videos/3064.mp4')">
<img src="http://192.168.3.111/20130519145459625962.jpg" ></a>
```

这里主要是在 Web 页面中添加一个图片链接，如图 5.3 所示，用户在新加载的页面中可以找到该链接并通过点击会调用 JS 函数 call_playvideo(url)跟 android 中的 WebView 进行交互，实现视频播放。

图 5.3 为新添加的视频在 Web 页面中增加视频链接

在客户端中启动 App，访问系统的 Web 服务器，选择该视频点播服务选项后，可以顺利播放视频。其中对新上传的 3064.mp4 视频资源进行点播，客户端的播放界面如图 5.4 所示：

图 5.4 客户端视频播放界面

由上述测试过程可以看出，通过搭建可挂载的 dfs-fuse 实现通过 Apache 直接访问存储在 Hadoop 集群中的视频资源文件的架构已经实现，并且在客户端节点能够顺利的访问到 Hadoop 集群中的视频资源。通过拖动进度条选择当前播放进度，

在缓冲较短的时间后，视频也可流畅播放，由此可以得知，对存储在 Hadoop 集群中的视频资源可以进行定点访问和以流的形式进行传输和播放。

5.3.2 客户节点间 P2P 功能模块的实现

在系统实现中，本文所主要实现的是完全分布式的 P2P。在没有服务端的参与下，用户节点之间实现节点设备的发现，视频缓存资源的发现和对其他节点视频缓存资源的访问。

1. 客户节点设备的发现和缓存资源发现

完全分布式 P2P 模式的特点，在没有服务端的参与下，节点之间互联互通，数据共享。这里我们主要为了测试节点启动 APP 后，节点是否能自动发现其他节点，并同步到其他节点上的视频缓存资源。

图 5.5 节点监听到其他节点的加入，获取该节点的缓存资源情况

如图 5.5 所示，根据测试过程的客户端显示，在节点启动 APP 后，当有新的节点启动了 APP，节点能发现新加入的节点，并获取到该节点的视频缓存资源。而新加入的节点在较短的时间内也能发现其他已经存在的节点及获取缓存视频资源。

2. 客户节点缓存资源的访问

节点中缓存的视频资源片段是视频资源的某一块，而某一块又作为一段小视频进行存储。这里测试节点间是否能相互访问到彼此的视频缓存资源，测试的理想结果是能对获取到的其他节点中缓存的资源片段进行播放。

图 5.6 在线用户节点 ZTE-T U800 上的视频缓冲资源块播放效果

在节点的缓存资源描述中，其他节点所得到的资源描述主要有缓存资源在节点组中的 IP 地址，因此其他节点可以通过 HTTP 协议访问该缓存资源信息。如图 5.5 所示，每个节点中可能存储有多个缓存资源片段，而每个缓存块作为一小段视频资源，可以播放，其中播放效果如图 5.6。

从上述测试中，可以说明在客户节点中实现的 P2P 各个模块功能均可使用。但在某个节点加入或离开在线节点组后，其他节点对该节点的缓存信息进行读取或删除的发现行为会有部份设备上出现延误现象。因此在 APP 设计和实现中，添加了一个搜索在线设备的菜单选项，通过选择该选项能够重新发现其他在线节点设备。

图 5.7 通过搜索选项重新加载对等节点信息

如图 5.7 所示，用户可以通过点击 menu 控件，然后选择 Search LAN 选项重新发现 P2P 集群中其他的在线节点，并重新加载所有节点中的缓存资源信息。

5.3.3 视频播放性能

在视频流服务中，视频的播放质量是制约视频流服务的关键因素，常见的问题有视频播放过程出现马赛克现象，因为视频流顺序出错出现视频抖动现象，或传输速率过慢导致缓冲时间过长。通过多节点同时访问系统，或同时访问同一个视频资源的方法对系统的性能和视频播放性能进行测试。测试结果显示，客户端在播放视频的时候，播放流畅，在快进快退的状态下，视频缓冲时间平均低于 7 秒。并且在 10 个节点同时访问一个视频资源的时候，播放正常，并且缓冲速度不受影响。

5.4 测试结果分析

通过测试，系统运行正常，Hadoop 集群提供了良好的，廉价的存储性能，能够存储和管理海量的视频资源。并且通过搭建的 dfs-fuse，能够很好的支持对 Hadoop 集群中的文件系统进行定点访问，从而提供良好的视频流服务。

客户端节点间的 P2P 各个功能模块运行正常，能够彼此发现在线设备，并能互联互通，共享视频缓存资源。但也发现了问题：在节点设备配置一般的情况下，当部分节点脱离在线组后，该节点无法识别该广播。针对这个问题，本系统在 APP 设计和实现中，添加了一个搜索在线设备的菜单选项，通过选择该选项能够重新发现其他在线节点设备。

5.5 本章小结

在本章中，主要通过运行系统，利用多个终端设备同时响应视频点播服务，对搭建的基于Hadoop集群存储视频资源并提供视频流服务方案的可用性和稳定性进行测试。所测试得到的结果良好，有力的说明了本文采用的方案的有效。接着通过可视化界面展示 APP 后台各个 P2P 功能模块的实现进行测试，得到较好的测试结果并且发现了存在的不足。对测试过程发现的问题，本文也针对该问题作出了一定有效的解决方案，以保证系统的可用性。但文章内容仍有些不足，系统的测试主要在局域网环境中完成，客户端在没有 P2P 的参与下和有 P2P 的参与下视频播放效果基本不变，在快进快退的情况下，视频缓冲时间较短。

第六章 总结

6.1 本文总结

随着网络技术的不断发展，在线视频点播技术正成为目前产业界和科研机构密切关注的焦点。它涉及到数据库管理和页面显示等学科知识，通过网络将视频内容发送到千家万户。同时它还可以作为一个平台来搭载一些商业性质的娱乐功能等，有着广阔的发展空间和极富潜力的商业前景。

云计算是一个虚拟化的计算机资源池，一种新的 IT 资源提供方案。通过这种方式，共享的软硬件资源和信息可以按需提供给计算机和其他设备。云计算可以根据不同用户的需求提供不同的服务。云存储是在云计算概念上延伸和发展出来的一个新的概念：它是指通过集群应用、网格技术或分布式文件系统等功能，将网络中大量各种不同类型的存储设备通过应用软件集合起来协同工作，共同对外提供数据存储和业务访问功能的一个系统。

而 P2P 技术使得每个用户节点在作为资源的申请者同时，成为缓存资源的提供者，在不占用客户端大量的网络和存储资源的同时，有效的利用了节点的上下行通道进行数据的共享。

本文重点将视频流技术、P2P 技术与云存储结合，突破了传统的资源存储集中话管理模式，它将视频资源存储在云端的各个分布的数据节点上，解决了传统模式中存储容量不足、服务性能瓶颈等一系列问题。同时客户端可以从对等网络中发现需要的视频缓存资源，充分的利用了节点的网络性能和存储空间，更高效的获得更多的视频资源提高播放的流畅度和稳定性。

本文实现的基于 Hadoop 的 P2P 视频流服务系统，能够用 Hadoop 集群有效的管理大量视频资源，并为客户端提供较流畅的视频流点播服务。并且系统具有稳定性、可扩展性强、投资经济等特点。

6.2 本文的不足之处及进一步的工作

本文的不足主要体现在没有在公网环境中测试系统视频流播放的效果和系统的健壮性。客户端中的 P2P 功能尚未稳定，从对等网络中获取过来的资源与当前播

放进度资源如何混合使用，以减少服务端数据传输的需求尚待更好的去设计和实现。本文设计的系统中并没有提供 QoS 函数保证的 SaaS，以便于实现视频流服务系统的商业化。

进一步的工作则是更好的将 Hadoop 和 P2P 的优势更好的结合起来实现更流畅，更高效的视频流服务，为系统设计一个 QoS 函数为提供给客户的视频流服务定义价格包。并保证视频流服务系统的稳定性和高效性是我们进一步的研究工作。

致 谢

在论文完成之际，谨向指导我毕业设计和毕业论文的沈沛意教授，董洛兵博士和徐虎讲师致以崇高的敬意和衷心的感谢，在老师们的严格要求、认真指导下我的毕业设计和毕业论文才得以按时按质按量的完成。从毕设题目的确定到论文章节的安排进而到文字的表达，都凝聚着导师的心血。

同时，感谢近四年来培养我的西安电子科技大学和所有的老师、同学。近四年来的大学生活，各位西电科大的老师，同窗们对我的照顾和关怀，才得以让我能顺利的完成本科学业，在学习中成长。在此，特向西电科大的所有教职工、软件学院的同学致以最诚挚的谢意！特别感谢毕业设计过程中项目组成员董洛兵老师，张文学长，孙庚泽，陈璟和崔致远，感谢你们对我的帮助和支持。

感谢所有帮助过，关心过我的人！

最后，特别感谢我的父母在我外出求学给予的极大支持和理解！

参考文献

[1] 李太君,吴泽晖.流媒体传输协议及其应用开发.计算机工程与应用,2004,40(3):138-141

[2] L. Gong. \JXTA: A Network Programming Environment".IEEE Internet Computing, 5(3), 88-95, 2001

[3] 庞瑞娟, 夏又新. 一种分布式 VOD 系统的方案设计. 电脑知识与技术, 2010, 6(7):1637-1638

[4] 鄢仁祥, 高远. 一种分布式视频点播系统模型. 小型微型计算机系统, 2002, 23(8):1010-1013

[5] 陆嘉恒,Hadoop 实战,北京：机械工业出版社，2011

[6] 刘鹏，实战 Hadoop-开启通向云计算的快捷，北京，电子工业出版社，2011

[7] SHVACHKO K, KUANG H, RADIA S, et al. The Hadoop distributed file system; proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies, MSST2010, May 6, 2010 - May 7, 2010, Lake Tahoe, NV, United states, F,2010 . IEEE Computer Society

[8] GHEMAWAT S, GOBIOFF H, LEUNG S-T. The Google File System. Operatingsystems review, 2003, 37(5): 29-43

[9] http://www.upnp.org/

[10] 霍龙社，甘震，流动流媒体协议综述，中国联通研究院，2010 年 4 月

[11] ISO/IEC 13818-1：Information Technology-Generic Coding of Moving Pictures and Associated Audio Part 1:Systems,2007

[12] 吴杰. P2P 流媒体内容分发与服务关键技术研究，博士学位论文，上海:复旦大学, 200

[13] 徐非，杨文广，基于 Peer-to-Peer 的分布式存储系统的设计，清华大学 计算机科学与技术系，软件学报

[14] 向锋，一种基于 P2P 的分布式文件共享系统的设计与实现，电子科技大学 2010 届硕士论文，2010 年 5 月

[15] 孙志佳，基于 Hadoop 的在线购物原型系统的设计与实现，东北大学 2010 届硕士论文，2010 年 6 月

[16] IrenaTrajkovska，Technical University of Madrid DIT, ETSIT, UPM, Spain，A Novel P2P and Cloud Computing Hybrid Architecture for Multimedia Streaming with QoS Cost Functions . In Proceedings of IEEE INFOCOM, 2012

[17] http://h264.code-shop.com/trac/wiki/Mod-H264-Streaming-Apache-Version2