第34卷 第20期 计 算 机 工 程 2008年10月
Vol.34 No.20 Computer Engineering October 2008
·软件技术与数据库· 文章编号：1000—3428(2008)20—0082—04 文献标识码：A 中图分类号：TP391

基于文件分片的P2P视频点播系统片断选择

朱 骏，吕智慧，刘毕升，钟亦平
(复旦大学计算机与信息技术系网络与信息工程中心，上海 200433)

摘 要：在基于BT技术的文件分片传输片段选择方案基础上，提出基于KAD网和文件分片的P2P视频点播系统的设计和相应的片断选择方案，主要描述系统模型和新型的算法，该算法包括节点分类、节点服务速度的定义和估算方式、任务分配调度机制和紧急处理原则。介绍系统的实现方案，并与中心VOD方式进行对比。对比实验表明，该方案是高效和可行的。

关键词：P2P视频点播；片断选择；Kademlia网络

Segment Selection Scheme of P2P VoD System Based on File Segment

ZHU Jun, LV Zhi-hui, LIU Bi-sheng, ZHONG Yi-ping
(Networking and Information Engineering Center, Dept. of Computing and Information Technology, Fudan University, Shanghai 200433)

【Abstract】 With P2P VOD segment selection solution based on BT, this paper proposes a novel segment selection solution on P2P VOD system using KAD network and file segments. This paper describes the system model and main algorithms, including peer classification, definition and estimating of peer's service speed, principles of how to assign peers' tasks and dealing mechanism of emergency situations. And then it introduces the implementation scheme of the system and makes a contrast experiment with centric VOD method. Experimental result proves high efficiency and feasibility of the solution.

【Key words】 P2P VoD; segment selection; Kademlia network

1 背景

近年来，随着互联网技术的快速发展，视频点播服务成为互联网的主流业务。传统的视频点播服务基于Client/Server模式，由于视频数据量大，对于服务器带宽要求很高，因此随着客户数目的增多，服务器带宽资源会很快耗尽，系统扩展性受到极大限制。为此，出现IP组播技术及CDN内容分发网络，但其仍然存在缺陷。

随着互联网的发展，相继出现了如BT, eMule[1]等多种形式的P2P文件共享形式。在这些系统中，不同用户下载相同的文件，文件被分成许多个片段，并在不同节点间直接进行传输。虽然与视频点播中多用户观看相同的视频信息有相似的地方，P2P文件共享系统却不能保证文件块数据的实时性和顺序性，这就很难保证流畅、完整地播放视频文件。

最近也出现了许多基于P2P的流媒体系统及其架构[2-4]，如采用应用层组播的P2P流媒体传输技术以及基于P2P单播的P2P流媒体传输技术，并论证了采用P2P文件共享方式的文件分片传输媒体数据的可行性。

本文在Kademlia式P2P网络上实现了文件分片的P2P视频点播系统，主要阐述了如何动态选择多个节点下载文件片段实现实时播放媒体的机制。

2 相关介绍

Kademlia[5]网络是在eMule中采用的一种分布式哈希网络，通过将各节点发布的信息存放在其他节点上来实现对资源的快速搜索。

文献[6]提出了基于BT技术的文件分片传输片段选择方案，本文将节点分类，提出用UDP响应时间及TCP传输速度加权后定义节点服务速度的方式、节点丢弃和紧急原则，并对节点中不存在需要数据的情况进行了处理，对整个系统进行了优化扩展和实现。

3 基于文件分片的P2P点播系统片断选择方案

3.1 需要解决的问题

在P2P网络中，不同节点的上下行带宽和速度都可能不同，服务负载能力也因此而不同，如何在具有不同负载能力的节点中选择适当节点获取需要的数据，是分片式P2P点播系统中的一个关键问题。

要想从参差不齐的节点中高效地获取媒体数据，最简单的方式是根据节点的服务速度来选择最快的节点下载，这需要考虑评估节点服务速度的方法。

P2P点播系统与P2P直播系统最大的不同在于，直播系统中所有节点观看的内容几乎完全一样，每个节点上存在的数据相差不大，也不需要很大的数据缓存，然而在点播系统中，每个节点播放的内容可能相差数十分钟，这导致节点上需要较大的数据缓存，并且每个节点上拥有的同一个媒体文件的数据片断都可能不同。在这种情况下，就需要根据不同

基金项目：上海市科委2005年度信息技术领域重点科技攻关基金资助项目“面向IPv6环境的可管理运营的P2P流媒体内容分发服务系统与关键技术研究”(055115009)

作者简介：朱 骏(1983－)，男，硕士研究生，主研方向：P2P技术；吕智慧、刘毕升，博士；钟亦平，教授

收稿日期：2007-12-26 **E-mail：**052021184@fudan.edu.cn

节点目前拥有的文件数据来选择服务节点。

3.2 系统模型

基于文件分片的 P2P 点播系统模型见图 1。

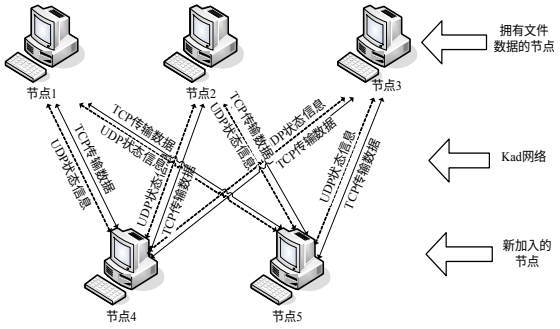

图 1 系统模型

为了解决上述问题，本文采取如下策略：

(1)在 P2P 网络中,每个节点在开始播放一个媒体文件时，通过 Kad 网络发布自己拥有这个文件数据的消息。

(2)每个节点维护自己当前拥有的数据及服务状态以供其他节点查询。

(3)需要下载数据的节点通过 Kad 网络搜索拥有某个文件数据的节点,并定时地通过 UDP 方式询问部分节点拥有的数据及服务状态，记录每次询问时对方的 UDP 响应时间，以作为参考。

(4)节点“速度”信息为节点 UDP 响应时间和 TCP 传输速度的加权和。UDP 响应时间通过 UDP 获取节点状态信息得到。TCP 通过传输文件数据得到。

(5)节点根据“速度”信息分为 3 种：1)正在服务节点，拥有 UDP 响应时间和 TCP 传输时间；2)准备服务节点，拥有 UDP 响应时间；3)备选服务节点，暂无速度信息。

(6)定期从备选服务节点中选取部分节点以获取 UDP 响应时间，并与准备服务节点中其他节点进行比较，替换慢的节点。

(7)定期从准备服务节点中选取部分节点尝试传输数据以得到 TCP 传输速度，并与正在服务节点中其他节点比较速度，替换慢的节点。

(8)定期维护正在服务节点和准备服务节点中的节点状态信息。

(9)正在服务节点根据速度信息，将其分为高速节点和低速节点，分配任务时优先让高速节点下载更靠前、更多的数据，低速节点下载靠后、较少的数据。

(10)当节点中缓存的数量低于某个阈值并且下载速度低于当前媒体的播放速度时，通过在 TCP 消息加入紧急标志来提高自己的紧急度。

(11)在每个服务节点上都有一个上传优先队列，根据不同请求节点的紧急情况来优先处理某些节点的数据请求。

3.3 算法设计

将媒体文件分成 200 KB 大小的数据块，为了节省状态信息占用的网络带宽，节点的状态信息用 0,1 的数组来表示节点拥有的数据块，由于媒体文件数据往往比较连续，因此状态信息数组还通过游程编码进行压缩。下载时，一次分配 10 块共 2 MB 的连续数据任务。

(1)节点分类

通过 Kad 网络找到拥有某个文件数据的节点分为 3 类：

1)正在服务节点：

$OnServeList$={与其建立连接且已获得其状态信息的节点}

2)准备服务节点：

$ReadyToServeList$={仅通过 UDP 方式获得了状态信息的节点}

3)候选节点：

$WaitingList$={包含还没有获得状态信息的节点和无法服务节点}

(2)速度定义

在媒体文件开始播放前，实际的播放速度B_{play}未知，为了在分配节点任务时作参考,可设定播放速度为一个猜测值。

未开始播放速度为 300 Kb/s：开始播放的速度为L_{file}/t_{file}。其中，L_{file}为文件的总长度；t_{file}为文件的总播放时间。

设正在服务的节点共有 m 个：

$OnServeList = \{p_1, p_2, \cdots, p_m\}$

每个正在服务 Peer 的下载速度：

b_i = TCP速度+α/UDP响应时间

在节点没有 TCP 速度时，可以通过 UDP 的响应时间长短来比较节点可能的速度快慢，而当节点已经有了 TCP 速度之后，UDP 响应时间对速度的影响应当变得很小，所以，这里 α/UDP 响应时间应为一个相当小的值，如 TCP 速度单位为 B/s，UDP 响应时间单位为 ms 时，可以取 α 为 10 000。这样，即使 UDP 响应时间短到 1 ms，α/UDP 响应时间的值也只有 10 KB/s，相对于 TCP 速度来说，影响不大。

正在服务 Peer 的平均下载速度为

$$A=\sum_{i=1}^{m} b_i/m$$

高速节点为

p_{High}={速度$b_i \geq A$的正在服务节点}

低速节点为

p_{Low}={速度$b_i<A$的正在服务节点}

高速节点速度总和为

$B_{High}=\Sigma b_i(p_i \in p_{High})$

低速速节点速度总和为

$B_{Low}=\Sigma b_i(p_i \in p_{Low})$

(3)任务分配原则

在当前 2 MB中，需要下载的块的总数为N_{all}。

因为可能存在 10 块数据块中部分已经被下载的情况，所以N_{all}为没有下载也没有分配任务的块的总数。

分配任务时，高速节点速度较快，所以，可以让其分配更多的任务，并且优先分配靠前的数据块。分配的原则是，根据B_{high}在所有OnServeList节点速度总合中占的比重来分配。由此可得高速节点要分配的靠前的块数为

$$N_{front} = N_{all} \times B_{High} /(Am)$$

需要分配给低速节点的靠后的块数为

$$N_{end} = N_{all} - N_{front}$$

每个高速节点p_i可以连续下载的块数为

$$N_i = N_{all} \times b_i /(Am)$$

由上式可知，N_{front}实际上就是所有p_{High}里的节点分配的任务之和。

当高速节点p_i有分配的N_i块数据时，则这N_i块数据全部由节点p_i下载，如果不全，则将N_i中p_i有的块中靠前的连续几块分配给p_i下载。

如：若N_i为从第 20 块开始的 5 块，而p_i仅有 20,21,22 这 3 块数据，则将 20,21,22 分配给p_i下载。

低速节点分配任务公式：

$$j = \lceil B_{play} / b_i \rceil$$

分配的块为当前 2 MB 数据块的起始块号+j，如果超出了 10 块的范围，则分配 2 MB 中最靠近尾部的没有分配任务的数据块。

当低速节点的速度过慢时，需要对其进行丢弃，不再从该节点下载数据。

低速节点丢弃原则为 $1/b_i > N_{all}/(Am)$。如果其他节点都下载完了，该节点都还没有下载完一块，此时应丢弃该节点，让其他节点代替其进行下载。

紧急原则如下：

(1)$Am < B_{play}$且缓存量低于某个值a时，需要在下载的时候通知服务节点紧急。

(2)$Am \geq B_{play}$且缓存量高于某个值b时不再紧急($a<b$)。

通过采取紧急原则，可以在下载速度可以保证流畅播放的前提下，将富余的带宽资源让给其他节点。

3.4 总体结构

P2P 点播系统中的片断选择下载模块结构见图 2。

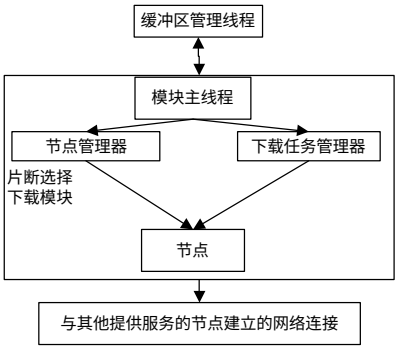

图 2 片断选择下载模块结构

3.5 类设计与关系

与图 2 对应，本文主要实现了 4 个不同类：(1)CPeer 类，对应节点；(2)CPeerManager 类，对应节点管理器；(3)CDownloadManager 类，对应下载任务管理器；(4)CDownloadThread 类，对应于整个片断选择下载模块线程。

CDowndloadThread 类与上层缓冲管理线程进行消息通信，以处理缓冲管理线程的下载请求，并将请求转交 CDownloadManager 分配任务。CPeerManager 负责根据上层请求的文件信息搜索 P2P 网络中已发布该文件的节点，并对这些节点进行状态信息维护、分类。CDownloadManager 负责根据片断选择算法计算高低速节点，并分配响应的任务，在任务完成或失败时，向上层返回成功或失败的消息。

3.6 典型流程分析

节点管理器工作流程、任务管理器工作流程见图 3、图 4。

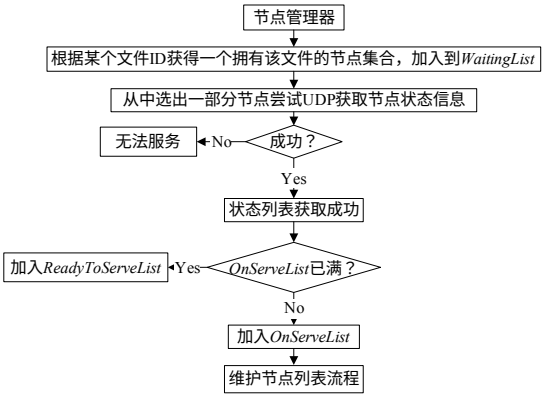

图 3 节点管理器工作流程

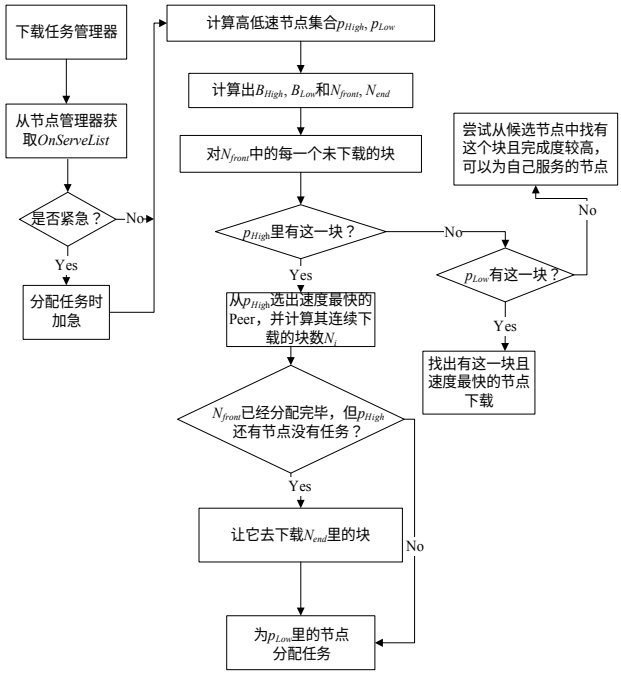

图 4 任务管理器工作流程

每当 DownloadManager 收到来自下载主线程转发的下载请求时，首先会从 PeerManager 获取当前的 *OnServeList*，然后根据 *OnServeList* 的平均速度与播放速度的大小关系决定是否需要标识紧急状态。计算出 *OnServeList* 中的高速节点和低速节点集合，并分别分配任务。在给高速节点分配任务块时，如果高速节点中没有这一块，从低速节点里找一个节点来下载，如果依然没有，就尝试从 *RSList* 中找到一个节点来下载这个块，再没有，就暂时放弃这一块，跳到下一个任务块进行分配。

4 方案实现及实验结果

对比实验分为中心服务器架构点播压力实验和 P2P 架构点播压力实验两组。

中心服务器架构实验中的中心服务器和 P2P 架构实验中的强节点配置均为 CPU: Pentium D 2.80 GHz，内存为 1 GB，网络带宽 100 Mb/s。

普通节点的配置均为 CPU: Pentium4 1.7 GHz，内存为 512 MB，网络带宽 100 Mb/s。

两组实验从 10 台~200 台普通节点的情况下测试多节点点播大小为 733 MB 的 AVI 文件，文件码率大致为 970 Kb/s，实验数据为所有普通节点播放文件时的平均下载速度 v Kb/s 及等待数据时间在总播放时间中的百分比 p。

实验得出的结果见图 5、图 6。

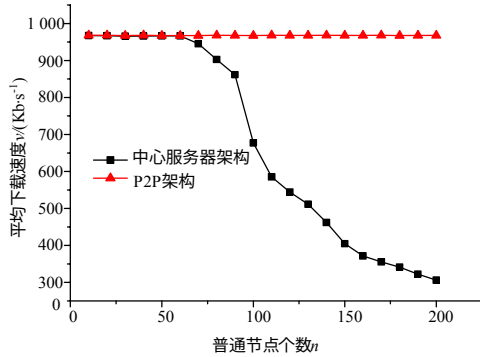

图 5 普通节点数 n 对 v 的影响

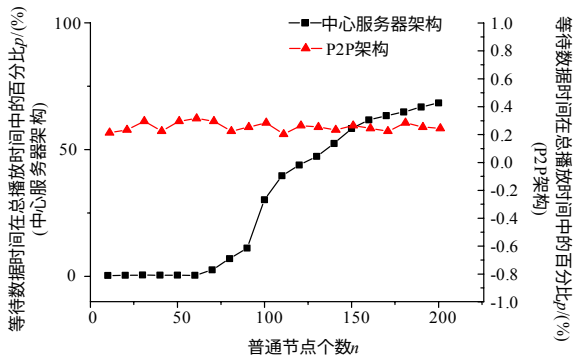

图 6　普通节点数 *n* 对 *p* 的影响

从图 5 和图 6 的方块图可以看到，在中心架构中只有 10~60 个普通节点时，系统可以很好地工作，普通节点数增加到 70 以上时，由于中心服务器资源与带宽逐渐耗尽，节点平均下载速度开始下降，出现了比较明显的等待时间，当超过 100 个普通节点后，由于等待时间在总播放时间中占的比例太高，因此中心服务器的服务性能大大降低。实验表明，新型设计方案的结果(三角形图标显示)有了较大改善，普通节点的平均下载速度并没有因为节点数的增加而有明显的变化，等待时间也仅仅占了总播放时间中极小的一部分，图 6 中的等待时间主要是系统开始启动做缓冲的时间。

5　结束语

本文在 Kademlia 式 P2P 网络上实现了基于文件分片的 P2P 视频点播系统，并阐述了片段选择的算法。通过与中心服务架构的点播系统的对比实验，可以看到，基于此片段选择方案的 P2P 架构点播系统能够很好地解决服务器高负载的问题，保证媒体数据播放的实时性和顺序性。

P2P 架构点播系统相较于中心服务架构的点播系统有诸多优势。在未来的工作中，如何对 P2P 点播系统中的节点进行有效管理，并尝试将其与 CDN 系统搭配使用以分担 CDN 服务器的负载都是主要研究方向。

参考文献

[1] What is eMule[EB/OL]. (2002-10-02). http://www.emule-project.net/home/perl/general.cgi?l=1.

[2] Hefeeda M. On Peer-to-Peer Media Streaming[C]//Proc. of ICDCS'02. [S. l.]: IEEE Press, 2002.

[3] Do T, Hua K, Tantaoui M. P2VoD: Providing Fault Tolerant Video-on-demand Streaming in Peer-to-Peer Environment[C]//Proc. of IEEE International Conference on Communications. Paris, France: [s. n.], 2004.

[4] Padmanabhan V. Distributing Streaming Media Content Using Cooperative Networking[C]//Proc. of NOSSDAV'02. Miami Beach, Florida, USA: [s. n.], 2002.

[5] Maymounkov P. A Peer-to-Peer Information System Based on the XOR Metric[C]//Proc. of the 1st International Workshop on Peer-to-Peer Systems. [S. l.]: Springer-Verlag, 2002: 53-65.

[6] Chen Tianding. The Feasibility of Using BT Technology to Test VOD System[J]. China Digital Cable TV, 2005, (24): 2392-2397.

(上接第 81 页)

金融系统专用ETL中存在多个相关联的、执行频度和时机差异很大并且常有启、停或增、删的任务。本文的ETL程序将所有任务分成多个工作流，工作流又由多个子任务组成，采用流间并行、流内串行的调度策略。为了达到较高的并行性，提高任务的执行效率，流分配的原则是尽量使流间的相关系数大，而流内的相关系数小[5]。

将 Spring 框架的 IoC 技术与 Timer 相结合可以灵活地处理 ETL 的任务调度问题。任务调度的元信息(比如任务流的分配、各个粒度的启/停控制、任务的运行频度及运行时机)都被配置在 XML 文件中，由框架在运行时读取，在适当的时机启动要执行的任务。调度模块的核心是 Timer 类，在应用程序代码中，Timer 只需要按配置的周期触发 TimerTask，任务调度策略对它是透明的。

利用 Spring 对复杂数据结构的多层注入方式可以将 ETL 任务调度策略配置在 XML 文件中，本文 ETL 程序采用三级粒度的调度模式。运行时，由框架读取 XML 中的调度策略，控制 ETL 核心处理模块工作。任务流中的每个任务对应一组转换规则，指导完成从源模式到目标模式的逻辑映射。

4　结束语

本文实现了一个基于 Spring 框架的金融数据仓库专用 ETL 程序。该程序利用 Spring 框架的 IoC 机制，将 ETL 的元数据对象配置到 XML 文件中，满足了 ETL 程序实现的复杂性与灵活性；对数据库的访问采用了 DAO 中间件与 Spring JDBC 相结合的方式，提高了运行效率；程序中的任务调度利用了 Java 内建的 Timer，并与 Spring IoC 机制结合，取得了较高的并行性和灵活性。该程序已经投入实际生产，能很好地满足金融系统数据仓库构建的需要。

对于该程序，进一步需要研究完善的问题主要有：(1)任务队列的持久化，以增强程序的鲁棒性。(2)增强脏数据检测机制，使程序能比较准确地定位脏数据，从数据源上解决问题。(3)提供友好灵活的定制界面，使工具更易于使用。

参考文献

[1] Vassiliadis P, Simitsis A. Conceptual Modeling for ETL Processes[C]//Proc. of DOLAP'02. McLean, Virginia, USA: [s. n.], 2002.

[2] Walls C, Breidenbach R. Spring in Action[M]. [S. l.]: Manning Publication Co., 2005.

[3] Simitsis A. Mapping Conceptual to Logical Models for ETL Processes[C]//Proc. of DOLAP'05. Bremen, Germany: [s. n.], 2005.

[4] Johnson R. Professional Java Development with the Spring Framework[M]. [S. l.]: Wiley Publishing Co., 2005.

[5] Simitsis A, Vassiliadis P. State-space Optimization of ETL Workflows[J]. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(10): 1404-1419.