· 开发研究与工程应用 ·　　文章编号：1000-3428(2018)11-0306-07　　文献标志码：A　　中图分类号：TP301.6

一种基于多服务器的 DASH 客户端优化算法

吴如亮，王子磊，奚宏生

（中国科学技术大学 自动化系，合肥 230027）

摘　要：在多服务器场景下，基于 HTTP 的动态自适应流媒体在传输过程中，由于用户请求的高度动态性及多服务器传输带宽的差异性，客户端获取的视频质量波动较频繁，影响到用户体验质量。针对上述问题，提出一种基于软件定义网络的视频分片请求调度算法。将多个分片组成一个视频块，利用分配策略把这多个分片分配至多个服务器，实现同一视频块内视频分片的码率同步，并使分片有序下载。同时，客户端算法采用 Q 学习方法，根据不同的环境状态决策出相应的码率。仿真实验结果表明，该算法可减少视频质量的波动，提高客户端平均视频质量。

关键词：基于 HTTP 的动态自适应流媒体；多服务器；体验质量；质量波动；软件定义网络；Q 学习

中文引用格式：吴如亮，王子磊，奚宏生．一种基于多服务器的 DASH 客户端优化算法[J]．计算机工程，2018，44(11)：306-312．

英文引用格式：WU Ruliang，WANG Zilei，XI Hongsheng. A DASH client optimization algorithm based on multiple servers[J]. Computer Engineering，2018，44(11)：306-312.

A DASH Client Optimization Algorithm Based on Multiple Servers

WU Ruliang，WANG Zilei，XI Hongsheng

（Department of Automation，University of Science and Technology of China，Hefei 230027，China）

【**Abstract**】In the transmission of Dynamic Adaptive Streaming over HTTP(DASH) for scenario，the client's video quality would fluctuate much frequently due to the user's high dynamic request and different transmission bandwidths of multiple servers. Such fluctuation would affect the Quality of Experience(QoE) seriously. To solve this problem，this paper proposes a video fragment request scheduling algorithm based on Software-Defined Network (SDN). By considering several video segments as one video block，these segments are assigned to a series of servers through an allocation strategy. This two-step method can not only realize the bit rate synchronization of the video segments in the same video block，but also enable the segments to be downloaded in an orderly manner. Furthermore，the client algorithm uses Q-learning method to deduce bit rate depending on different environmental state. Experimental results show that the proposed algorithm can reduce the fluctuation of video quality and improve the average video quality of the client.

【**Key words**】Dynamic Adaptive Streaming over HTTP(DASH)；multiple servers；Quality of Experience(QoE)；quality oscillation；Software-Defined Network(SDN)；Q-learning

DOI：10.19678/j.issn.1000-3428.0048773

0　概述

近年来，一种称之为 HAS(HTTP Adaptive Streaming)的流媒体传输标准出现，并且很快变成了流媒体传输的主要方式。据 Cisco[1] 估计，到 2019 年视频流量将占据互联网数据传输总流量的 80%，网络运营商也试图寻找新的方式以更好地利用网络容量。2011 年，MPEG 结合 3GPP 开发出一套标准化的解决方案——基于 HTTP 的动态自适应流媒体(Dynamic Adaptive Streaming over HTTP，DASH)[2]。在 DASH 标准中，原视频被分割成 2 s～10 s 的视频分片并以不同的码率等级进行编码，生成的分片存储在服务器上，客户端则根据网络状况及自身状态动态调整请求的视频质量。随着业务规模的扩大，DASH 服务模式开始从单服务器向多服务器转变。多服务器能够提供更大的带宽，传输路径更加多样化，服务质量也更加可靠。但是，由于服务器间的带宽差异，如何减少客户端视频码率的频繁波动，保证客户端体验质量(Quality of Experience，QoE)，仍是一个挑战。

相关算法[3]已被提出用来解决单服务器场景下的质量波动问题。现有多服务器场景下 DASH 客户端方法主要是在最大化带宽利用率的前提下，提高

基金项目：国家自然科学基金重点项目"三网融合业务接入系统的分析、建模与调控"(61233003)。
作者简介：吴如亮(1991—)，男，硕士研究生，主研方向为网络多媒体技术；王子磊，副教授；奚宏生，教授。
收稿日期：2017-09-25　　**修回日期**：2017-11-03　　**E-mail**：zlwang@ustc.edu.cn

客户端视频质量,但是往往忽视了视频质量频繁波动的问题。文献[4]为了优化多个服务器间的资源分配问题和提高客户端视频质量,提出了一种协同多尺度调度算法,该算法能够平衡服务器负载,优化整体视频质量。文献[5]针对多服务器场景下的公平、效率和稳定性问题,提出了一种新的协议 Presto,提高了用户体验质量。上述优化方法均是在传统网络下进行,由于视频下载的间歇性,客户端对带宽的估计往往不准确。近年来,软件定义网络(Software-Defined Network,SDN)出现,由于其集中控制机制及其在网络资源调度和网络状态监测等方面的智能性和灵活性,在流媒体视频领域越来越受欢迎[6]。文献[7]提出了一种 OpenFlow 协助下的 QoE 优化框架,减少了客户端和 SDN 控制器间的频繁通信,能够有效分配网络资源,但因其请求模式是单服务器,并没有解决多服务器场景下的质量波动问题。

针对上述问题,本文提出一种多服务器场景下基于 SDN 的视频分片请求调度算法。该算法将多个分片组成一个视频块,并把这多个分片分配至多个服务器,利用这种以视频块为单位的调度策略实现同一视频块内视频分片的码率同步,减少视频质量的波动。同时,在客户端通过使用 Q 学习训练码率决策矩阵,当新的分片请求到达时,通过查询该矩阵找到当前状态下 Q 值最大的动作作为最优码率。

1　基于 SDN 的 DASH 系统框架

1.1　问题描述

动态自适应码率切换算法是当前 DASH 技术研究的一个热点。但是目前很多算法[4-5,8]是纯客户端驱动,即仅仅依赖客户端及其获悉的网络状态来动态调整视频比特率,可能由于带宽估计不准确导致客户端频繁切换视频比特率,影响客户端 QoE。

在 DASH 多服务器场景下,客户端同时向多个服务器发起视频分片请求。但是由于客户端到各个服务器的带宽多样性,可能会出现分片失序到达的情况,此时若缓冲区耗尽,则视频播放会中断,从而引入不必要的时延。文献[9]为解决并行多服务器下视频块顺序下载的问题,提出一种概率分片调度方法,该方法根据时变带宽概率模型,估计在视频缓冲区耗尽之前视频分片成功下载的概率来优化视频传输。此外,由于多个服务器间服务带宽的差异性,往往会导致视频质量的频繁波动。文献[8]针对多个服务器服务能力的动态变化的情况,提出一种动态服务器选取策略来协助客户端选择服务能力最优的视频传输服务器,解决了传统 DASH 自适应策略仅选取视频码率的问题。这种方案在网络状况较好时每次都能请求最高的视频码率,但在网络环境较差的情况下,视频质量仍然会出现频繁波动的情况。

以上研究对 DASH 多服务器场景下的分片失序到达、视频质量波动等问题做出了一定的改善,但是由于传统网络的限制,不能从全局角度对系统进行优化,同时,带宽大多是基于预测手段获取,往往存在偏差,制约了优化效果。

本文研究内容可以描述为:在 SDN 网络中,客户端同时发起多个视频分片请求,分片调度算法将多个请求分配至多个服务器,以减少客户端视频质量的波动,同时客户端采用 Q 学习算法决策视频码率,为用户提供更佳的 QoE。

1.2　系统模型

OpenFlow[10]网络中的动态自适应视频播放系统模型如图 1 所示。整个系统由视频服务器、Web 代理服务器、视频客户端、OpenFlow 控制器及交换机组成。其中:视频服务器负责向视频客户端提供视频服务;Web 代理服务器负责将用户的视频分片请求以及将请求下发给 OpenFlow 控制器;视频客户端是视频请求的发起者;OpenFlow 控制器负责分片调度算法的决策,负责控制视频分片请求的下发;OpenFlow 交换机负责根据流表完成数据的转发。

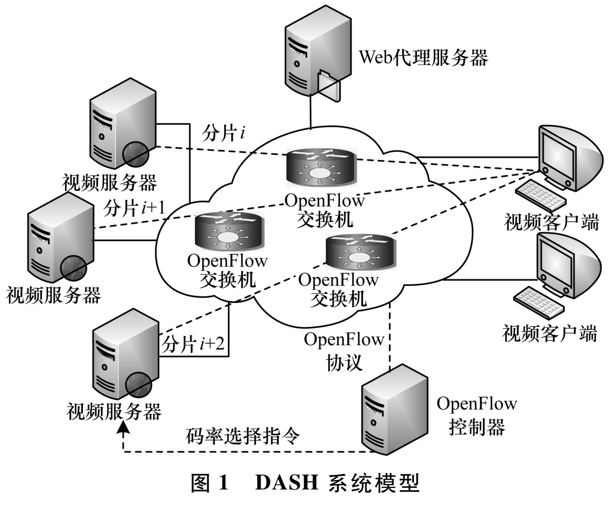

图 1　DASH 系统模型

本文的工作是设计视频分片请求调度算法和客户端码率决策算法。整个视频请求过程可以描述为:首先客户端向 Web 代理服务器发起视频请求,代理服务器将此请求转发至控制器,控制器根据收集的各个服务器带宽信息,执行分片调度算法,决策出需要请求的分片数及服务器数,并通过 OpenFlow 协议与交换机进行指令下发和信息收集;然后客户端根据 Q 学习算法决策出要请求的视频码率,根据控制器下发的指令开始向视频服务器发起视频传输请求。视频块请求分配示意图如图 2 所示。把多个视频分片组成一个视频块,每次对一块视频进行调度,下文将给出调度策略的具体描述。

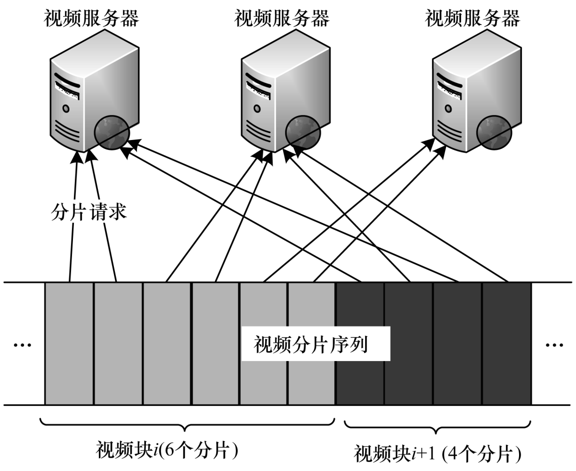

图2　视频块请求分配示意图

2　视频分片调度策略

视频分片调度策略首先动态选择视频块的长度(分片个数)及所请求的服务器;然后采用视频分片请求调度算法将各个分片请求分配至多个服务器。

2.1　视频块长度及服务器选取算法

多服务器请求模式为:一个客户端同时向多个服务器发起多个视频分片的请求,服务器分配的请求分片数与服务器可用带宽成正比,这样确保了多个服务器完成下载的时间几乎是同步的。但是随着时间的变化,服务带宽也在变化,同一视频块内的各个分片可能无法以预定同步的时间完成下载。因此,设置了视频块长度的上限阈值 $N_{\max}$。同时,将带宽最小的服务器分配一个分片,其他服务器按照此带宽成比例依次分配视频分片数。

假设存储客户端请求视频的服务器有 S 个,第 i 个服务器可用带宽为 c_i,降序排列使得 $c_1 \geqslant c_2 \geqslant \cdots \geqslant c_S$。令 S_k 表示请求第 k 块视频所用服务器数,显然 $S_k \leqslant S$。将第 k 块视频包含的分片数记为 N_k。下面将详细描述视频块长度 N_k 及服务器个数 S_k 的选取过程。

假设第 i 个服务器的请求分片数为 n_i,令第 S_k 个服务器请求的分片数 $n_{S_k}=1, n_i=\lfloor c_i/c_{S_k} \rfloor$是其他服务器请求的分片数,符号$\lfloor x \rfloor$表示对 x 向下取整,则请求的总分片数为 $N_k=\sum\limits_{i=1}^{S_k} n_i$。上述选取方法的缺点是一次请求的分片数 N_k 可能会很大,导致分片可能无法在预定时间内完成下载,因此,设置视频块长度的上限阈值 $N_{\max}$。当 $N_k > N_{\max}$时,舍弃可用带宽最小的服务器,重复上述过程,直至满足 $N_k < N_{\max}$。算法 1 给出了视频块长度 N_k 及所请求服务器的选取过程。

算法1　视频块长度及服务器选取算法

输入　视频服务器个数 S,各服务器可用带宽 $c=\{c_1, c_2, \cdots, c_S\}$

输出　视频块长度 N_k,服务器个数 S_k

```
1. Initialize:
2. S_k ← S
3. while TRUE do
4. n_{S_k} ← 1;
5. for i ← 1 to S_k - 1 do
6. n_i = ⌊c_i / c_{S_k}⌋;
7. end for
8. N_k ← ∑_i^{S_k} n_i
9. if N_k > N_max then
10. S_k ← S_k - 1
11. else
12. return N_k , S_k;
13. end if
14. end while
```

对算法 1 的解释如下:

算法第 1 行~第 2 行:算法开始时获取各个服务器可用带宽集合$\{c_1, c_2, \cdots, c_S\}$,并初始化本次请求所用服务器数目 S_k 为 S。

算法第 3 行~第 8 行:初始化第 k 个服务器分配的视频分片数目为 1,然后依次给其他服务器分配分片数为 $n_i=\lfloor c_i/c_{S_k} \rfloor$,并计算所有服务器分配的视频分片总和 $N_k=\sum\limits_{i=1}^{S_k} n_i$。

算法第 9 行~第 14 行:判断分片总数 N_k 是否大于视频块长度的上限阈值 $N_{\max}$。若 N_k 小于等于 $N_{\max}$,返回视频块长度 N_k 和服务器个数 S_k,程序结束;否则,若 N_k 大于 $N_{\max}$,舍弃带宽最小的服务器,更新可用服务器数目 S_k 为 S_k-1,程序跳转到第 3 行继续执行。

2.2　视频分片请求调度算法

确定视频块长度 N_k 和服务器个数 S_k 后,下一步的目标是将 N_k 个视频分片合理分配至 S_k 个服务器,使得分片下载时间最短。

假设 $\boldsymbol{A}$ 为服务分配矩阵,大小为 $N_k \times S_k$,元素 a_{ij}表示是否将第 i 个视频分片分配给第 j 服务器,$a_{ij} \in \{0,1\}$,$\boldsymbol{c}$ 为服务器可用带宽向量,大小为 $1 \times S_k$,元素 c_j 表示第 j 个服务器的可用带宽,b_k 为视频分片比特率,τ 为分片时长。算法优化目标是让分片下载时间最短,决策变量是服务器分配矩阵 $\boldsymbol{A}$,约束条件包括:1)每一个分片只从一个服务器下载,即 $\sum\limits_{j=1}^{S_k} a_{ij}=1$;2)服务器分配矩阵 $\boldsymbol{A}$ 是 0-1 矩阵,即 $a_{ij} \in \{0,1\}$;3)前一分片的下载完成时间早于后一分片的下载完成时间,即 $t(n) < t(n+1)$。

综上所述,以服务器分配矩阵 $\boldsymbol{A}$ 作为决策变量,视频分片请求的调度模型为:

$$\mathrm{Obj}: \min\left\{\max_{1 \leqslant j \leqslant S_k} \sum_{i=1}^{N_k} a_{ij} \frac{b_k \cdot \tau}{c_j}\right\}$$

Subject to:

$$\sum_{j=1}^{S_k} a_{ij}=1, i=1,2,\cdots,N_k$$

$$a_{ij} \in \{0,1\}, i=1,2,\cdots,N_k, j=1,2,\cdots,S_k$$

$$t(n) < t(n+1), n=1,2,\cdots,N_k-1$$

$$t(n)=\frac{\sum\limits_{j=1}^{S_k}\left(a_{nj} \cdot \sum\limits_{i=1}^{n} a_{ij}\right) \cdot b_k \tau}{\sum\limits_{j=1}^{S_k} a_{nj} c_j}$$

上述优化问题是一个 NP 难问题,无法直接使用凸优化理论进行求解,本文在此给出了启发式算法。求解时,借助于贪心算法的思想,对矩阵 $\boldsymbol{A}$ 逐行求解,将分片从前往后依次分配给各个服务器,分配策略如算法 2 所示。

算法 2　视频分片请求调度算法

输入　请求分片数 N_k,服务器数 S_k,各服务器可用带宽 $\boldsymbol{c}=\{c_1,c_2,\cdots,c_{S_k}\}$

输出　服务器分配矩阵 $\boldsymbol{A}$

1. Initialize:
2. A←0
3. for i←1 to N_k do
4. $j^* \leftarrow \underset{1\leqslant j\leqslant S_k}{\mathrm{argmin}}\left(\dfrac{b_k\times\tau}{c_j}+\sum_{t=1}^{i-1}a_{tj}\dfrac{b_k\times\tau}{c_j}\right)$;
5. $a_{ij^*}=1$;
6. end for
7. return A;

对算法 2 的解释如下:

算法开始时获取算法 1 的输出结果,即视频块长度 N_k、服务器数 S_k,将其作为本算法的输入,同时将服务器可用带宽 $\boldsymbol{c}=\{c_1,c_2,\cdots,c_{S_k}\}$ 作为算法的输入。

算法第 2 行:初始化分片分配矩阵 $\boldsymbol{A}=\boldsymbol{0}$,开始对 N_k 个视频分片进行分配。

算法第 3 行～第 6 行:对于任意一个视频分片 $i\in[1,N_k]$,依次计算该分片在各个服务器上的下载时间 $t=\{t_1,t_2,\cdots,t_{S_k}\}$,并计算前 $i-1$ 个已分配的视频分片在各服务器上下载时间 $\boldsymbol{t}'=\{t_1',t_2',\cdots,t_{S_k}'\}$,两部分时间求和,将分片 i 分配给下载完成总时间 $\boldsymbol{t}+\boldsymbol{t}'$ 最少的那个服务器。

算法第 7 行:当 N_k 个视频分片分配完成之后,返回分片分配矩阵 $\boldsymbol{A}$。

3　基于 Q 学习的客户端算法

当 DASH 客户端请求视频时,由于视频码率的多样性,往往需要一个客户端算法对视频码率进行决策。本文采用 Q 学习算法作为客户端码率决策算法[11-12]。

3.1　基于 Q 学习的决策

Q 学习是强化学习[13]的一个分支,它需要解决如下问题:一个能够感知环境的自治智能体,根据当前的环境状态,通过学习选择一个最优动作,使得智能体获取的回报值最大。

基于 Q 学习,客户端码率选取过程为:根据环境状态(带宽、客户端缓冲区和视频码率等),采取某一动作(选择视频码率),使得回报值(QoE)最大。式(1)给出了智能体在状态 s 下,采取动作 a 获取回报值 r 时 Q 值的更新方式。其中,(s,a) 是状态-动作对,$\alpha\in[0,1]$ 是学习速率,表示新获取信息覆盖旧信息的程度,α 越大保留之前训练的效果就越少;$\gamma\in[0,1]$ 是折扣因子,决定了对未来回报的重视程度。

$$Q(s,a)=(1-\alpha)Q(s,a)+\alpha[r+\gamma Q(s',a')] \tag{1}$$

基于学习到的 Q 值可选择要请求的码率。在 DASH 系统中,需要为每个动作 a 分配一个回报值 r 用来更新学习 Q,其中,回报值 r 是通过回报函数来确定的,下面将给出回报函数的定义。

3.2　回报函数

客户端算法的优化目标是最大化 QoE,因此,本文定义的回报函数是 QoE 的一种度量,主要包括 3 个方面[14]:1)视频质量等级;2)视频质量波动;3)缓冲区下溢。下面分别介绍这三方面对应的回报函数。

3.2.1　视频质量等级

文献[15]表明客观 PSNR 值和主观平均意见得分(MOS)之间存在类似线性的关系。因此,本文建立的视频质量等级对应的回报函数 R_{quality} 为一个线性函数,见式(2)。其中,视频有 L 个质量等级,QL_i 表示请求第 i 个分片的质量等级,且 $QL_i<QL_{i+1}$。

$$R_{\text{quality}}=-(QL_L-QL_i) \tag{2}$$

3.2.2　视频质量波动

在 DASH 系统中,由于带宽的波动,请求的视频质量等级会随之变化,导致视频质量发生波动,影响用户体验质量。本文建立了质量波动对应的回报函数 $R_{\text{oscillation}}$ 为波动值的线性函数,见式(3)。当无波动发生时,回报为 0;当有波动时,波动越大回报越小。

$$R_{\text{oscillation}}=-|QL_{i+1}-QL_i| \tag{3}$$

3.2.3　缓冲区下溢

缓冲区下溢导致视频播放中断是影响 QoE 的第 3 个方面,因此,缓冲区下溢也成为回报函数的一个组成部分。当客户端缓冲区耗尽时,视频播放中断,重缓冲事件发生。定义回报函数时,本文考虑根据缓冲区长度大小,将缓冲区下溢对应的回报函数分段定义。当缓冲区长度 B_i 低于缓冲区总长度 $B_{\max}$ 的 10% 时,给予 −100 的惩罚;否则,将回报函数定义为一个线性函数,见式(4)。

$$R_{\text{buffer}}=\begin{cases}-100, & B_i\leqslant 0.1\cdot B_{\max}\\ -(B_{\max}-B_i), & B_i>0.1\cdot B_{\max}\end{cases} \tag{4}$$

3.2.4　总回报函数

最终的回报函数可以归结为以上三者的线性叠

加之和,见式(5),其中 C_1、C_2、C_3 为权重因子。

$$R = C_1 R_{\text{quality}} + C_2 R_{\text{oscillation}} + C_3 R_{\text{buffer}} \tag{5}$$

3.3 环境状态

在构建环境状态时,结合回报函数的定义,本文采用视频质量等级和缓冲区长度作为环境状态。此外,因为带宽也是码率选择中的一个重要因素,所以将带宽也加入到环境状态中。表 1 给出了环境状态的定义。

表 1　环境状态定义

环境状态	范围	数目
质量等级	$[1, L]$	L
缓冲区长度	$[0, B_{\max}]$	$B_{\max}+1$
带宽	$[0, BW_{\max}]$	$L+1$

Q 学习的算法描述详见算法 3,算法维护一个状态空间和行动集构成的矩阵 $\boldsymbol{S}\times\boldsymbol{A}$ 来存储 Q 值,下面对整个算法进行详细描述。

算法 3　基于 Q 学习的客户端码率选择算法

```
1. Initialize:
2. Q←0
3. for each episode//video start play
4. s←randSelectState();
5. for each step of episode//request segment
6. a = selectAction(s);//use ε-greedy
7. calculate R by formula (5)
8. calculate Q by formula (1)
9. s←s′
10. end for
11. end for//video end
12. return Q;
```

对算法 3 的解释如下:

首先将 Q 值矩阵初始化为全 0 元素矩阵,episode 为智能体学习的次数,算法第 3 行开始对智能体进行训练,第 4 行是从所有码率中随机选择一个码率作为初始状态。

算法第 5 行~第 10 行:每一次的码率请求对应一步,对于每一次的码率请求,智能体使用 ε-greedy 方法选择一个码率作为下一时刻要请求的视频码率,根据式(5)计算该码率对应的回报值 R,根据式(1)更新 Q 值矩阵,同时,将当前的状态 s 进行更新,然后进行下一步的循环过程。

当设定的 episode 循环结束之后,Q 学习过程结束。将最终训练得到的 Q 值矩阵作为客户端的码率决策依据。当客户端请求到来时,根据客户端不同的状态,查询 Q 值矩阵找到当前状态下效用值最大的动作为最优码率。

4 仿真与分析

4.1 仿真参数说明与设定

仿真实验不考虑链路的连接关系,使用的拓扑包含 4 个服务器,各个服务器的带宽数据基于中国科学技术大学校园网络内某一服务器的真实网络带宽情况。选取 1 个客户端发起 DASH 视频请求。视频选择开源视频源"Big Buck Bunny",该视频由 2 s 长的视频分片组成,并且每一个视频分片被编码成 7 个质量等级:300,400,600,800,1 200,1 500,1 800,单位为 Kb/s。客户端播放器的缓冲区上限阈值设置为 30 s(15 个分片)。通过调整实验参数进行多次实验,获得一组比较好的设置参数。本文最终回报函数中参数 C_1、C_2、C_3 均设置为 1。式(1)中的 α 设置为 0.1,γ 设置为 0.5。

实验的流程共分为 3 个阶段:训练学习获取码率决策矩阵,即 Q 值矩阵,根据分片调度算法获取分片分配矩阵 $\boldsymbol{A}$,根据环境状态及 Q 值矩阵选择码率并按分配矩阵 $\boldsymbol{A}$ 中分配方式发起视频请求。

1)客户端学习算法

本算法是在 Ubuntu14.04 系统下通过编写 python 脚本实现的,主要是调用 python 下的工具包 numpy,同时在实验中 episode 设置为 200 000,step 设置为 100 000,数值较大,运算时间较长,而 python 脚本执行速度较快。因此,本文选择使用 python 执行 Q 学习算法。

2)分片调度算法

本算法是 DASH 流媒体视频传输系统中的重要一步。在实验中,采用真实的服务器带宽进行实验,另外,由于 Matlab 具有强大的矩阵处理能力,因此本文选择在 Matlab 上执行分片调度算法。在执行分片调度算法的过程中,通过改变服务器带宽进行多次实验,观察客户端质量变化及缓冲区变化情况。

3)客户端码率决策

客户端码率决策是实验的最后一步,决策过程是在 Matlab 上执行的。首先根据客户端当前环境状态,查询 Q 值矩阵中对应的状态行,找到该状态下效用值最大码率作为本次要请求的视频码率,然后根据分片分配矩阵 $\boldsymbol{A}$ 向服务器发起视频分片请求。客户端维护着请求的视频质量列表,作为客户端性能评估的依据。

在仿真实验过程中,具体的测试环境及软硬件配置信息如表 2 所示。

表 2　实验配置

项目	参数
实验主机	Lenovo 启天 M4380 台式电脑
操作系统	Windows7 专业版
分片调度算法环境	Windows7 下 Matlab R2015b
Q 学习算法环境	虚拟机 Ubuntu14.04LTS(64-bits)
Q 学习算法语言	Python2.7

4.2　仿真结果分析

使用真实服务器带宽进行仿真实验时,考虑到真实服务器带宽和本文给定的视频码率不匹配问题,对真实数据进行了线性放大处理。在放大实际数据量的同时,保留了原始数据良好的周期性及波动性。本文实验拓扑结构如图 3 所示,设置了 4 个 DASH 服务器,选取 1 个客户端发起视频请求。

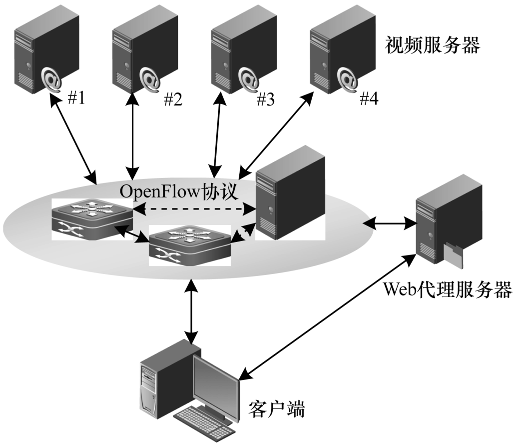

图 3　实验拓扑

首先,本文运行客户端上的 Q 学习算法训练学习客户端码率选择过程,训练参数设置已在 4.1 节描述,最终得到训练的 Q 值,作为本文后续实验的码率选择依据;然后,客户端发起视频分片请求,运行控制器上的分片调度算法,算法决策出应该向哪几个服务器发起视频分片请求,接着,客户端根据控制器收集的带宽信息和客户端自身的状态信息,查询 Q 值矩阵中对应的状态行,找到该状态下效用值最大码率作为本次要请求的视频码率,开始视频传输服务。

实验将客户端 QoE 作为性能指标,分别测量了客户端平均视频质量等级、平均缓冲区大小和码率切换次数。此外,为了更好地反映本文策略的效果,仿真实验中加入了一组对照实验:协同多尺度调度算法[4],该算法也是针对 DASH 多服务器场景,能够平衡服务器负载,自适应各个服务器带宽,优化视频服务质量。实验将码率等级从低到高依次编号,从 1 开始,得到 7 个质量等级,实验结果如表 3 ~ 表 5 所示。表中第 1 行给出了服务器带宽的放大倍数 x,下面 2 行分别对应不同策略下的实验结果。

表 3　平均质量等级对比

算法	$x=4.85$	$x=4.90$	$x=4.95$	$x=5.00$	$x=5.05$
本文算法	6.24	6.28	6.34	6.41	6.45
协同多尺度调度算法	5.79	5.88	5.91	5.97	6.02

表 4　不同策略平均缓冲区大小对比　s

算法	$x=4.85$	$x=4.90$	$x=4.95$	$x=5.00$	$x=5.05$
本文算法	22.46	23.09	25.87	25.76	26.35
协同多尺度调度算法	22.04	22.91	23.63	25.85	25.46

表 5　不同策略码率切换次数对比

算法	$x=4.85$	$x=4.90$	$x=4.95$	$x=5.00$	$x=5.05$
本文算法	35	34	30	22	19
协同多尺度调度算法	44	38	30	26	23

实验结果表明,本文提出的分片调度算法能够在 SDN 控制器指导下有效改善客户端 QoE。在不同的带宽放大倍数下,本文策略在平均视频质量、码率切换次数和平均缓冲区大小等方面均有所优化,其中分片调度算法保证了连续几个分片的码率一致性,使得码率切换频率明显降低。

另外,为了更直观地反映本文策略的实验效果,本文在服务器带宽倍数设置为 4.90 时,对客户端码率等级和缓冲区波动情况进行了图形化展示。图 4 对比了 2 种不同方案下的质量等级随时间波动情况。图 5对比了 2 种不同方案下的客户端缓冲区的变化情况。

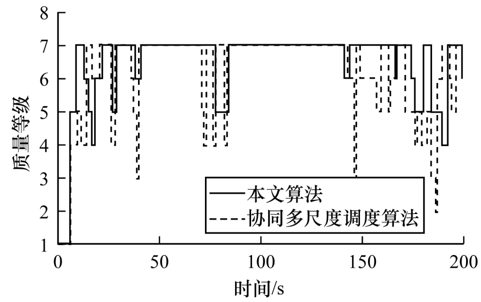

图 4　质量等级波动图

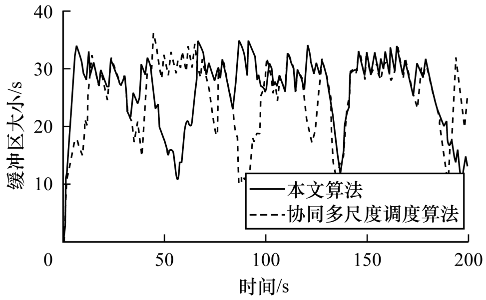

图 5　客户端缓冲区波动图

图 4 表明,在视频播放过程中,本文算法下视频质量波动次数明显减少,并且在波动幅度上相对于协同调度算法也略有减少,进而提高了用户体验质量。图 5 表明,协同多尺度调度算法下客户端缓冲区的波动更加剧烈,而本文算法下客户端缓冲区波动稍缓慢,这主要得益于本文中客户端 Q 学习算法将缓冲区状态引入了环境状态中,当缓冲区状态很低时,会给系统一个较大的惩罚。

5 结束语

本文提出了一种基于多服务器的 DASH 客户端优化算法,借助于 SDN,实现了对服务器可用带宽的准确测量。通过将多个视频分片组成一个视频块,并对这个视频块中的码率进行统一化处理,一方面减少了客户端视频码率的频繁切换,另一方面提高了总体视频质量。同时,本文采用 Q 学习算法,通过自我学习训练 Q 值矩阵,为每一个视频块选择合适的码率,大幅提高了码率选择的效率。下一步将考虑客户端和服务器之间的路径规划问题,使客户端能够自动切换到网络状况较好的传输路径上。

参考文献

[1] Cisco. The Zettabyte era:trends and analysis[EB/OL]. [2017-09-10]. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html.

[2] STOCKHAMMER T. Dynamic adaptive streaming over HTTP--: standards and design principles [C]// Proceedings of the 2nd Annual ACM Conference on Multimedia Systems. New York, USA: ACM Press, 2011:133-144.

[3] JIANG J, SEKAR V, ZHANG H. Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with festive [C]//Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies. New York, USA: ACM Press, 2014:97-108.

[4] PU W, ZOU Z, CHEN C W. Dynamic adaptive streaming over HTTP from multiple content distribution servers[C]//Proceedings of Global Telecommunications Conference. Washington D. C., USA:IEEE Press,2011: 1-5.

[5] ZHANG S,LI B,LI B. Presto:towards fair and efficient HTTP adaptive streaming from multiple servers[C]// Proceedings of IEEE International Conference on Communications. Washington D. C., USA:IEEE Press, 2015:6849-6854.

[6] McKEOWN N, ANDERSON T, BALAKRISHNAN H, et al. OpenFlow:enabling innovation in campus networks[J]. ACM SIGCOMM Computer Communication Review,2008,38(2): 69-74.

[7] 曹荣珍. 基于 OpenFlow 的视频用户 QoE 优化研究[D]. 北京:北京交通大学,2016.

[8] BOUTEN N, CLAEYS M, VAN POECKE B, et al. Dynamic server selection strategy for multi-server HTTP adaptive streaming services [C]//Proceedings of the 12th International Conference on Network and Service Management. Hong Kong,China:[s. n.],2016:82-90.

[9] LIU L, ZHOU C, ZHANG X, et al. Probabilistic chunk scheduling approach in parallel multiple-server DASH[C]// Proceedings of Visual Communications and Image Processing Conference. Washington D. C., USA:IEEE Press,2014:5-8.

[10] 左青云,陈 鸣,赵广松,等. 基于 OpenFlow 的 SDN 技术研究[J]. 软件学报,2013,24(5):1078-1097.

[11] WATKINS C J, DAYAN P. Q-learning [J]. Machine Learning,1992,8(3/4):279-292.

[12] CLAEYS M,LATRÉ S,FAMAEY J,et al. Design of a Q-learning-based client quality selection algorithm for HTTP adaptive video streaming [C]//Proceedings of Adaptive and Learning Agents Workshop. Saint Paul, USA:[s. n.],2013:30-37.

[13] KAELBLING L P, LITTMAN M L, MOORE A W. Reinforcement learning:a survey[J]. Journal of artificial intelligence research,1996,4:237-285.

[14] MOK R K,CHAN E W,CHANG R K. Measuring the quality of experience of HTTP video streaming[C]// Proceedings of IFIP/IEEE International Symposium on Integrated Network Management. Washington D. C., USA:IEEE Press,2011:485-492.

[15] NEMETHOVA O, RIES M, ZAVODSKY M, et al. PSNR-based estimation of subjective time-variant video quality for mobiles[EB/OL]. [2017-09-10]. https://pdfs.semanticscholar.org/a496/411a16b59c265c53aa07de11a9c32ec90f65.pdf.

编辑 金胡考

(上接第 305 页)

[17] SUN J T,ZENG H J,LIU H,et al. CubeSVD:a novel approach to personalized Web search[C]//Proceedings of the 14th International Conference on World Wide Web. New York,USA:ACM Press,2005:382-390.

[18] 刘振娇,王新华. 基于上下文学习和张量分解的个性化推荐[J]. 山东师范大学学报(自然科学版),2017, 32(1):36-42.

[19] 陈梅梅,薛康杰. 基于改进张量分解模型的个性化推荐算法研究[J]. 数据分析与知识发现,2017(3): 38-45.

[20] KOREN Y,BELL R,VOLINSKY C. Matrix factorization techniques for recommender systems[J]. Computer,2009, 42(8):30-37.

[21] RENDLE S,MARINHO L B,NANOPOULOS A,et al. Learning optimal ranking with tensor factorization for tag recommendation [C]//Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, USA: ACM Press, 2009:727-736.

编辑 司淼淼