

基于缓存偏移量的 P2P 流媒体调度策略

王文波, 范训礼, 房鼎益

(西北大学信息科学与技术学院, 西安 710127)

摘 要: 现有 P2P 流媒体调度策略在确定数据块调度优先级时, 未能综合考虑数据块稀缺度和紧迫度因素, 导致了较差的播放连续度和启动延迟性能。提出的基于缓存偏移量的数据调度策略通过统计数据块在供求节点上的播放偏移量并对其进行加以处理, 可有效解决该问题, 使播放连续度、启动延迟性能得到明显提高。

关键词: P2P 流媒体; 数据调度; 稀缺度; 紧迫度; 缓存偏移量

Scheduling Strategy Based on Cache Offset for P2P Streaming Media

WANG Wen-bo, FAN Xun-li, FANG Ding-yi

(School of Information Science and Technology, Northwest University, Xi'an 710127)

【Abstract】 The rarity and urgency of data block cannot be concerned simultaneously when the popularity of data block is calculated in the existing P2P scheduling strategies, which leads to the inferior playback continuity and startup latency. This paper proposes a Scheduling Strategy based on Cache Offset(SSCO) strategy, the above-mentioned problem has been solved well by means of counting and processing the information about the Cache Offset(CO) of playback of requested data block and the two metrics have been improved noticeably.

【Key words】 P2P streaming media; data scheduling; rarity; urgency; Cache Offset(CO)

1 概述

目前, 基于 Gossip 协议的 P2P 流媒体覆盖网, 由于具有较好的可扩展性、可靠性以及构建简单等特点, 已成为构建 P2P 流媒体系统的主流方式。在此类系统中, 覆盖网拓扑呈现出无结构的特点, 系统性能主要依靠覆盖网的构造和数据调度算法。从用户角度来看, 系统的启动延迟、流媒体的播放连续度无疑是最重要的 2 个性能。而在一个具体的 P2P 覆盖网中, 这 2 项性能的表现则主要依靠数据调度算法的设计。

在基于 Gossip 协议的 P2P 流媒体模型中, 已有的数据调度策略主要包括以下几种: 最少优先策略(Rarest-First, RF)^[1], 该策略有利于加快数据块在覆盖网中的扩散, 提高系统的整体吞吐量, 获得较好的平均播放连续度, 但由于未考虑数据块的紧迫度因素, 因此产生了较大启动延迟; 权重循环鲁棒(Weighted Round-Robin, WRR)^[2] 策略以带宽的大小、数据块的层优先级为主要调度依据, 能够取得很好的负载平衡, 但对于提高数据共享几率方面考虑不足; Greedy 策略, 也称为最早截止期优先(nearest deadline first)策略, 直观上该策略应当能提供较好的启动延迟、播放连续度, 但经测试和分析^[3], 该策略表现较差, 原因在于 Greedy 策略会引发数据块的共享几率的逐步下降, 导致系统整体性能降低, 最终使每个节点无法实现期待的播放效果; 随机(random)策略^[4], 根据这种策略, 对于每个缺失的数据块, 节点从持有此数据块的伙伴节点中随机地选择一个进行请求。该策略性能不稳定, 尤其是在异构的网络环境中。

本文针对现有调度算法存在的问题, 提出了一种新的数据调度算法——基于缓存偏移量的数据调度(Scheduling Strategy based on Cache Offset, SSCO)算法, 该算法以数据块缓存偏移量的统计信息为调度依据, 在全面、准确掌握数据块供需状况基础上进行合理调度。实验结果表明, SSCO 算

法在播放连续度、启动延迟方面的性能指标比广泛使用的最少优先调度算法有所提高。

2 SSCO调度算法

2.1 算法设计思想

在基于 Gossip 协议的 P2P 流媒体系统中, 通常采用被请求数据块的稀缺度作为计算调度优先权的依据, 但是这一做法存在一些问题, 下面以典型的 RF 算法在数据分发过程中的表现加以说明, 如图 1 所示。

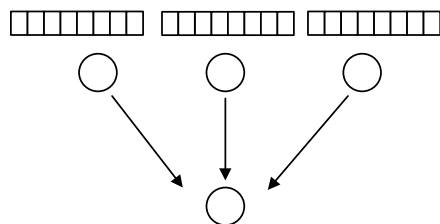


图 1 缺失数据块在提供节点缓存中的位置示例

在图 1 中, 请求节点 Pr 请求数据块 i 和 j 。拥有 j 的节点数量为 1; 拥有数据块 i 的节点数量为 3。若根据稀缺度进行调度, Pr 应当优先请求数据块 j 。然而, 根据数据块 i, j 在各个邻居节点缓冲区中的位置, 可以看到, 经过 3 个播放时间单位后, 所有发送节点上的数据块 i 将会被移出缓冲区, 而数据块 j 依然存在, 因此, 本次优先调度 i 更为合理。这说明, 为缺失数据块确定优先权时, 仅仅考虑其在系统中的拷贝数量是不够的, 还应当考虑每一份拷贝的生存期。此外,

基金项目: 博士后科学基金资助项目(20070410381); 陕西省教育厅基金资助项目(08JK447)

作者简介: 王文波(1974—), 男, 硕士研究生, 主研方向: 网络与分布式系统; 范训礼, 副教授; 房鼎益, 教授、博士生导师

收稿日期: 2009-07-09 **E-mail:** bowenwang2008@tom.com

从上述过程还可以看出：稀缺度高的数据块正是那些数据源最新分发的数据块，于是，序列号较大的数据块得到优先调度，这在一定程度上降低了 deadline 即将达到的数据块下载几率，对播放连续度造成影响。

另一方面，由图 1 可以看出，被请求数据块在其各自缓冲区中的位置具有重要的意义：(1)对于发送节点，被请求数据块在缓冲区的位置代表了其在此节点上的生存期，距离当前播放数据块越远，生存期就越长，从该节点成功下载此数据块的几率就越高，数据块的调度优先权应当相应减少。这样，可以采用数据块在多个发送节点上的生存期作为计算调度优先权的因素。(2)对于请求节点，被请求数据块在缓冲区中的位置代表了请求节点对该数据块渴求程度，即数据块的紧迫度。若距离请求节点当前播放数据块越远，则请求节点对这一数据块的渴求程度越低。可以使用请求数据块到请求节点当前播放数据块的距离作为计算数据块调度优先权的另一个因素。因此，节点缓存中被请求数据块到当前播放数据块的距离充分反映了数据块的稀缺度和紧迫度。

基于上述考虑，本文提出的 SSCO 其设计思想为：以数据块缓存偏移量(Cache Offset, CO)作为计算调度优先权的基础，再根据带宽信息选择恰当的提供者节点。其中，CO 指节点缓存中被请求数据块到当前播放数据块的距离。SSCO 内容涉及数据块调度优先权的计算、调度算法、伙伴节点的发送速率的确定。

2.2 数据块调度优先权的计算

相关参数及假设：设任意节点上已播放过的数据块不再保存于缓存中；系统中所有节点缓存大小为 B ，表示能够保存 B 个数据块；每个数据块包含等长的流媒体数据；设 n 为请求节点的邻居节点数； $offset_i^k$ 表示数据块 A_i 在第 k 个邻居节点上的缓存偏移量； $offset_i^R$ 表示被请求数据块在请求节点缓冲区中的缓存偏移量； urg_i 表示被请求数据块 A_i 的紧迫度； $rarity_i$ 表示被请求数据块 A_i 的稀缺度； $priority_i$ 表示被请求数据块 A_i 的优先权。其中，

$$rarity_i = \prod_{k=1}^n (1 - \frac{offset_i^k}{B}), \quad 0 \leq offset_i^k < B \quad (1)$$

$$urg_i = 1 - \frac{offset_i^R}{B} \quad (2)$$

$$priority_i = rarity_i \times urg_i \quad (3)$$

在式(1)中，如果数据块 A_i 在第 k 个邻居节点上的缓存偏移量 $offset_i^k$ 为 0，表示该节点未拥有此数据块，相应地 $1 - \frac{offset_i^k}{B} = 1$ ，若全部 n 个邻居节点均未拥有此数据块，则 $rarity_i = 1$ ，即达到最高的稀缺度；而对于一个因式 $1 - \frac{offset_i^k}{B}$ 来说， $offset_i^k$ 越大，表示在邻居节点 k 上该数据块生存期就越长，因式 $1 - \frac{offset_i^k}{B}$ 的值就越小，从而使 $rarity_i$ 值降低；在式(2)中，偏移量 $offset_i^R$ 越大，表示数据块 A_i 的紧迫度就越低， urg_i 的值就越小。式(3)显示了如何利用数据块 A_i 的 $rarity_i$ 值、 urg_i 来计算其调度优先权。

2.3 调度算法

计算出每个数据块的调度优先权后，将需要下载的数据块按优先权降序排列。对于数据块 A_i ，可能会有多个邻居节点拥有它，通常选择能最快传输 A_i 的节点作为 A_i 的发送者。但是，这种选择方式可能会导致下载任务的冲突，例如，为

2 个数据块选择了同一个发送节点时，其中一个数据块必须等待或需要重新选择发送节点。因此，在分配传输任务时，要考虑发送节点的负载均衡。另一方面，为数据块选择合适的发送节点被归结为并行机调度问题的一个变种^[1]，属于 NP 难问题，很难找到一个最优解。因此，本文采用一种以尽早获得调度优先权高的数据块为目标、兼顾发送节点的负载均衡、不刻意追求整体最优效果的贪婪算法。表 1 给出该算法相关参数。

表 1 数据调度算法的相关参数

参数	说明
τ	数据调度周期
I	请求节点接收带宽，以数据块数/秒来表示
S_i^j	数据块 A_i 的第 j 个拥有节点
S_i	数据块 A_i 的拥有节点集合， $S_i = \{ S_i^1, S_i^2, \dots, S_i^k \}$
$Length$	数据块的长度

数据调度算法描述：

算法 1 SSCO 调度算法

输入

- (1)根据调度优先权降序排列的数据块序列 A_1, A_2, \dots, A_m ;
- (2)相应数据块拥有节点集合序列 S_1, S_2, \dots, S_m ;
- (3)全部邻居节点发送速率 $R(1), R(2), \dots, R(n)$;
- (4)当前调度周期内各个邻居节点预计发送等待时间 $Wait(1), Wait(2), \dots, Wait(n)$ ，初始值均为 0。

输出 每个数据块 A_i 的发送节点 $sender_i$

算法：

- (1) $get_{max} \leftarrow \min(m, \tau \times I)$; //当前调度周期内最多能接收的数据块数量
- (2)for $i=1$ to get_{max} do
- (3) $t_{min} = \infty$; //初始化 A_i 最早获取时间
- (4) for $j=1$ to k do
- (5) $t_{deliver} = \frac{Length}{R(S_i^j)}$; //从 S_i^j 处获得 A_i 预计时间
- (6) if $t_{deliver} + Wait(S_i^j) < t_{min}$ and $t_{deliver} + Wait(S_i^j) < \tau$
- (7) $t_{min} \leftarrow t_{deliver} + Wait(S_i^j)$; $sender_i \leftarrow S_i^j$;
- (8) end for j ;
- (9) if $sender_i \neq null$
- (10) $Wait(sender_i) \leftarrow t_{min}$;
- (11)end for i ;

算法 1 的时间复杂度为 $O(n \cdot \min(m, \tau \cdot I))$ 。其中， n 表示邻居节点数； m 表示请求的数据块数量，其取值范围为 $[0, B]$ (B 为缓冲区的大小)，显然， m 与数据块的分布状态有关； $\tau \cdot I$ 为下载带宽。通常，系统中的 n, τ 为常数，所以，影响算法执行时间的主要因素是下载带宽与数据分布状态。算法 1 充分考虑了带宽因素，能够根据邻居节点提供的带宽分配传输任务。另外，由于稀缺度、紧迫度高的数据块得到优先调度，所以，各个节点算法执行一段时间后，覆盖网中数据块的分布会趋于均衡。而从整体上看，节点不仅可在起始阶段获得较低的启动延迟，而且能以较为平稳的速率下载数据，实现较好的播放连续度。

2.4 发送速率的确定

采用历史信息估算法确定邻居节点的发送速率。具体地，在执行一个调度之前，利用先前 M 个周期内从一个邻居节点获得的数据量来估算当前该邻居节点能够提供的数据发送速率。设 $q_k^{(m)}$ 为：在第 m 个周期内请求节点从其邻居 k 那里收

到的全部数据块数。则第 $m+1$ 个周期内邻居节点 k 的发送速率为

$$R^{(m+1)}(k) = \alpha \cdot \left(\sum_{l=m-M+1}^m q_k^{(l)} \right) / M\tau$$

其中, 常量 α 称为扩张系数。每次进入一个调度周期时, 检查上一次周期内对每个邻居节点请求数据的返回率。全部成功返回则说明这个邻居节点至少能以此发送速率提供上传服务。当累计成功返回数据的次数达到一定值后, 可以认定: 以当前发送速率作为对此邻居节点分配传输任务的依据过于保守, 应当尝试增大发送速率的估计值, 此时增加 α 的值; 同理, 若根据估计的发送速率请求数据没有获得成功, 并累计达到一定次数, 则认定该邻居节点以此发送速率不能提供很好的服务, 因而在下一调度周期内将其值降低, 以适应此节点的服务能力的变化, 于是减小 α 的值。

3 仿真试验与结果分析

3.1 仿真环境及参数设置

采用 NS2 作为仿真器, 并使用 GT-ITM 拓扑生成器^[5]生成 transit-stub 模式的底层网络。拓扑包含 30 个连接域(transit domain), 每个连接域有 6 个 transit 节点, 每个 transit 节点上附 3 个端域(stub domain), 每个端域有 3 个 stub 节点。这是一种 ts-large 型的网络拓扑^[6], 用户节点可以定位在拓扑中的任意 stub 节点上, 资源服务器置于一个 transit 节点上。随机地选择 100 个~1 500 个 stub 节点作为对等节点, 并将用户节点的输出带宽为[100, 500] Kb/s, 接入带宽为[400, 1 000] Kb/s, 服务器输出带宽为 100 Mb/s。流媒体文件播放时长为 30 min。用户节点以泊松分布加入网络。流媒体文件的播放速率为 400 Kb/s。

为了更好地对比分析 SSCO 算法的性能, 在仿真实验中也实现了相同环境下的 RF 策略以进行比较, 仿真结果如图 2、图 3 所示。

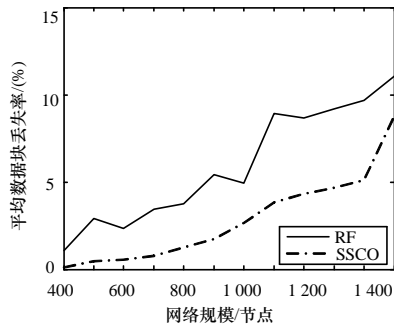


图2 平均数据块丢失率

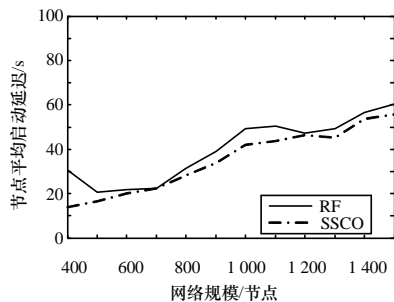


图3 平均启动延迟

3.2 仿真结果分析

多次实验结果表明, 当网络规模小于 100 个节点时, SSCO 算法的播放连续度性能低于 RF 算法; 而网络规模大于 400 个节点时, SSCO 算法具有较好表现。其原因在于: 计算数据块优先权时, 加入紧迫度因素会降低稀缺度对优先权的影响, 导致数据块扩散速度一定程度的下降, 在小规模网络环境下, 这种情况使得 SSCO 算法在播放连续度性能方面不如 RF 算法。

3.2.1 数据块丢失率

在 P2P 流媒体应用系统中, 数据块的丢失率是一个重要指标, 它体现了用户播放连续度。在图 2 中, 可以看出随着系统规模的增加, 2 种策略的平均数据块丢失率均在增加, 这是由于对等节点上传带宽有限, 导致了部分数据块无法在播放期限到达前传输到请求节点。此外, 相比于 RF 策略, SSCO 算法的指标曲线增长的较为缓慢、平滑, 这说明在相同的带宽约束条件下, 稀缺度、紧迫度高的数据块得到优先调度能够有效降低数据块丢失率, 提高系统整体的播放连续度性能。

3.2.2 启动延迟

P2P 流媒体系统性能的另一个重要指标是节点的启动延迟, 图 3 给出了 RF 策略和 SSCO 算法指标曲线。由于在计算调度优先级时引入了数据块的紧迫度因素, 因此使用 SSCO 算法能够在较短的时间内缓冲到足够的数据块。图 3 所示的试验结果验证了 2.3 节中算法 1 的分析结论。

4 结束语

数据调度策略是 P2P 流媒体系统的重要组成部分, 对系统整体性能影响较大。本文在分析了现有调度策略不足的基础上, 提出了一种综合了数据块稀缺度、紧迫度的调度策略——SSCO 算法, 并给出了调度优先权的计算方法以及调度算法描述。实验结果表明, 相比于传统调度策略, SSCO 算法在播放连续度和启动延迟方面的性能都有所提高。

参考文献

- [1] Zhang Xinyan, Liu Jiangchuan. Coolstreaming/DONet: A Data-Driven Overlay Network for Peer-to-Peer Live Media Streaming[C]//Proceedings of IEEE INFOCOM'05. Miami, USA: IEEE Press, 2005: 2012-2111.
- [2] Agarwal V, Rejaie R. Adaptive Multi-source Streaming in Heterogeneous Peer-to-Peer Networks[C]//Proceedings of the 12th Annual Multimedia Computing and Networking. [S. l.]: ACM Press, 2005: 102-109.
- [3] Zhou Yipeng, Chiu Dah, Lui Ming. A Simple Model for Analyzing P2P Streaming Protocols[C]//Proceedings of IEEE ICNP'07. [S. l.]: IEEE Press, 2007: 226-235.
- [4] Pai V, Kumar K. Chainsaw: Eliminating Trees from Overlay Multicast[C]//Proceedings of the 4th International Workshop on Peer-to-Peer Systems. [S. l.]: IEEE Press, 2005: 127-140.
- [5] Zegura E, Calvert K, Bhattacharjee S. How to Model an Internetwork[C]//Proceedings of the IEEE INFOCOM'96. New York, USA: IEEE Press, 1996: 594-602.
- [6] 邱彤庆, 陈贵海. 一种令 P2P 覆盖网络拓扑相关的通用方法[J]. 软件学报, 2007, 18(2): 381-390.

编辑 索书志

