

基于客户-服务器双端去重的 Web 预取新方法

姚 瑶

(郑州工程技术学院 信息工程学院, 河南 郑州 450044)

摘 要:网络数据量的不断增长和网络内容的不断多样化,使得提高网络访问效率、有效降低网络访问延迟成为当前网络存储领域面临的严峻挑战。Web 缓存技术和预取技术是目前解决该难题的有效方法。针对缓存技术受限于命中率、预取技术受限于带宽等诸多问题,在 Web 预取系统引入数据去重技术,提出一种客户-服务器端双端数据去重的 Web 预取系统改进策略。首先改进传统预取系统框架,分别引入代理服务器端数据去重模块 SDM 和客户端数据去重模块 CDM 以及相关模块;然后采用 LBFS 算法实现双端数据去重;最后,将该方法应用于预取系统,评价其预取性能。实验结果表明,相对于标准传输,该系统一方面可以减少大约 34% 的字节传输量,另一方面降低大约 8% 的用户预期的访问延迟,从而优化预取系统。

关键词:数据去重;SDM;CDM;预取

中图分类号:TP393

文献标识码:A

文章编号:1673-629X(2019)04-0181-06

doi:10.3969/j.issn.1673-629X.2019.04.036

A New Web Prefetching Method Based on Client-server Double-ended Deduplication

YAO Yao

(School of Information Engineering, Zhengzhou Institute of Technology, Zhengzhou 450044, China)

Abstract:The increasing amount of network data and the diversification of network contents make it a serious challenge to improve network access efficiency and reduce network access delay. Web caching technology and prefetching technology are currently effective ways to solve this problem. For the problem that the cache technology is limited by the hit rate and the prefetching technology is limited by the bandwidth, we propose a Web prefetching method based on the client-server double-end data deduplication technology. Firstly, the traditional Web prefetching system is improved. The proxy server data deduplication module (SDM) and the client data deduplication module (CDM) are introduced to implement the client-server dual-end data deduplication. Then, LBFS algorithm is used to achieve double-end data deduplication. Finally, this method is applied to the prefetching system to evaluate its prefetching performance. The experiment shows that compared to standard transmission, on the one hand this system can reduce the amount of bytes transferred by about 34%, and on the other hand, in order to optimize the prefetching system, it can reduce the expected access delay by about 8% of users.

Key words:data deduplication;SDM;CDM;prefetching

0 引言

随着网络用户和服务内容的爆炸式增长,用户对网络服务质量的要求越来越高,尤其是用户访问延迟正面临着严峻的考验。为了解决上述问题,学者们虽然采取了较为有效的方法,比如基于时间局部性原理的 Web 缓存机制^[1]、基于空间局部性原理的 Web 预取机制^[2],以及基于地域性的 CDN 系统,但是对网络延迟的影响仍然受限于内存、命中率和带宽等因素。

Web 预取技术试图在用户提出请求之前主动提取资源,在一定程度上提高了命中率,降低了访问延迟。最近客户端访问历史常被用来预测在不久最可能被访问的资源。但是存在投机性,有时还会额外增加带宽。因此,该方法需要谨慎使用。也正是因为潜在的带宽受限而使得 Web 预取技术在商业领域的应用不是很好。Mozilla Firefox 浏览器和 Google 搜索引擎相继采用 Web 预取技术减少延迟。

收稿日期:2018-05-07

修回日期:2018-09-12

网络出版时间:2018-12-20

基金项目:河南省高等学校青年骨干教师培养计划(2016GGJS-201);河南省科技攻关项目(182102310982);河南省高等学校重点科研项目(18B520038)

作者简介:姚 瑶(1982-),女,硕士,副教授,CCF 会员(49307M),研究方向为 Web 高性能计算。

网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20181220.1035.034.html

利用数据去重技术分析数据的冗余度,从而适当减少网络数据传输量,由此释放被占用的带宽资源^[3]。该技术可以利用数据对象之间的信息冗余,获得远高于传统压缩方法及增量备份方法的空间利用率。数据去重技术早期应用在存储系统和备份系统中,目前是云存储应用中一种重要的存储优化技术^[4-7]。基于存储系统内在的数据相似性和数据重复访问的特点,数据去重技术还被应用于内存性能优化和延长 SSD 使用寿命等方面。随着人们不断利用其数据缩减优势来改进和优化现有存储和网络系统的不足,数据去重技术的应用将会越来越广泛。文献[8]提出基于网页正文结构和特征串的相似网页去重算法,虽然取得了较高的去重率,但是应用于预取系统时导致延迟较高,并且降低了预测准确率。在云计算环境下,文献[9]提出了一种基于流水线的数据读取模型,但是随着冗余率增加,系统速度降低,影响了预取性能。文献[10]在客户端实现重复数据删除技术,通过对文件进行分块和在备份过程中去除重复数据块,减少了客户端与

服务期间需要传输的数据量,使得文件备份的速度获得较大提高,也降低了网络带宽要求,显著提高了网络备份系统的性能,但却没有考虑预取。

Web 预取技术受限于带宽,而数据去重技术由于减少转移数据的体积从而释放部分占用带宽。所以,将预取技术与数据去重技术结合起来在减少网络延迟方面有重大意义。文中首先改进了 Web 预取系统,其次通过移植数据去重模块实现了预取系统的双端去重,最后通过实验所提方法进行验证。

1 基于数据去重的 Web 预取系统

对传统的 Web 预取系统框架进行改进,引入了客户端重复数据去重模块(CDM)和代理服务器端重复数据去重模块(SDM),实现了双端数据去重。

1.1 Web 预取系统架构

1.1.1 代理服务器端

改进后的代理服务器端增加了响应重组模块、SDM 模块和响应拦截模块等,如图 1 所示。

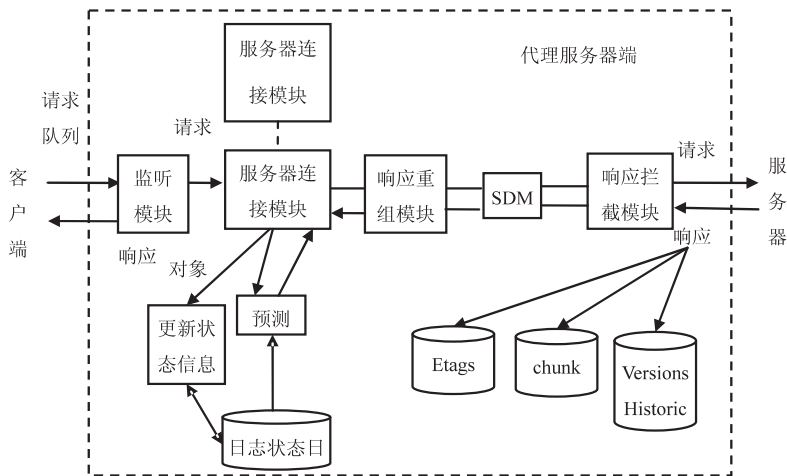


图 1 改进的代理服务器端体系结构

服务器连接模块:主要负责直接连接客户端和服务端,尤其是在处理响应的时候,该模块第一时间接收最原始 HTTP 响应消息报头,并传递给客户端。然后把从服务器端接收的所有资源数据原封不动地传递给客户端。文中所提方法对 HTTP 响应消息报头做了进一步的改进。报头信息不仅仅包括现有保证连接的附加标题信息,而且涉及修改了标题值。例如,需要修改 Content-Length。此外,由于该系统需要对消息实体做拆分处理,所以需要第一时间从服务器端接收完整的消息实体。当该过程终止时,再把修正过的实体内数据回传给客户端。

响应拦截模块:由于改变了代理服务器端 HTTP 响应报文的处理方式,故新设一个响应拦截模块,拦截来自服务器端的最原始的 HTTP 响应报文,存储在临时缓冲区,并丢弃数据,判断传输方式是标准数据还

是分块方式。该模块的具体工作:第一,检查是否存在 ETag(被请求变量实体的值),如果存在取出其值;第二,检查 Content-Length 和 Transfer-Encoding 字段,如果是前者存在并有效,则 HTTP 服务器响应的报文必须和消息内容的传输长度完全一致。对于动态的内容或者在发送数据前不能判定长度的情况下,可以使用分块的方法传送编码。该模块检索客户端缓存关联资源的自定义报头的、以数组形式存储的资源标识符。然后把装有存储的响应头、实体消息数据和资源标识符数组的缓冲区传送给代理服务器数据去重模块 SDM。

SDM 模块:对响应拦截模块转交过来的消息实体的数据执行拆分。分两个步骤完成:块划分和数据去重。该模块存储的主要字段是 ETag、Chunk 和 Versions Historic。ETag 存储唯一 ETag 资源标识符,由资

源版本号索引。Chunk 存储块元信息,SDM 模块处理的所有资源版本号的所有块,由块哈希索引。Versions Historic 存储了所有资源版本号的 Chunk 的链接队列,由资源版本号索引。

SDM 模块首先检查 ETag 的状态,如果是新资源,则生成一个新的资源标识符存储于 ETag 中,使用分块算法为资源分块,生成块链表,存储在 Versions Historic 中。如果 ETag 已经存在,意味着该资源先前已经被 SDM 处理过,不需要重新分块,仅检索 Versions Historic 中存储的资源版本号的块链表。接下来,SDM 执行数据去重算法,使用两个缓冲区分别存储元数据和非冗余数据。SDM 模块遍历块链表,试图找到一个匹配的散列块内的客户端参考源。如果不匹配,SDM 模块在已经被处理过的当前资源版本块中再做一次查找(例如:自冗余数据案例)。如果找到一个匹配成功的冗余块,SDM 模块添加块的元信息到对应缓冲区。对于非冗余块,实体消息数据需要添加到非冗余缓冲区。SDM 模块建立了元数据和非冗余内容的最终组合,通知重组响应模块数据拆分过程已经完成。

响应重组模块:准备和发送回传的响应报文。响应报头信息进行重组,更新/创建 Content-Length 附加上新的实体消息数据长度;添加两个新的报头信息 X-vrs(资源版本号标识符)和 X-mtd(元数据长度)。响应重组模块通过附加非冗余内容块到元数据块之后重组新的实体内容数据块(entity content data block),最终重组后的回传响应报文包含报头和消息实体内容。

组合过的响应报文被多功能的服务器连接模块拦截,背负式(piggybacks the hintlist)报头信息并丢弃未改变的实体消息块,最终重组过的 HTTP 响应报文传递给客户。

预测引擎模块:主要任务是在每一次资源被请求的时候预测即将可能访问的页面。该模块依据预测算法将产生一系列最近被访问频率最高的资源的 URL,并将结果放入决策数据库中。

1.1.2 客户端

改进后的客户端引入了 Resource Versions 数据文件模块、CDM 模块,体系结构如图 2 所示。

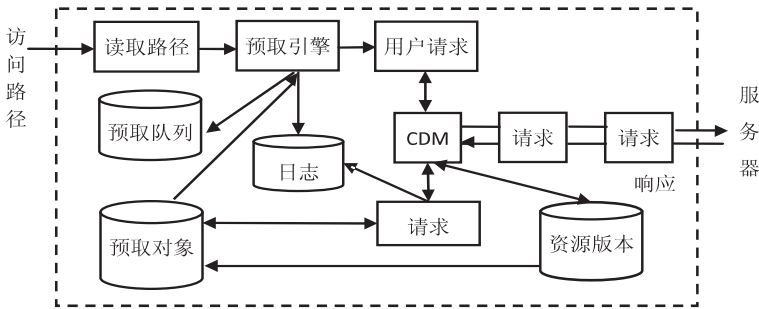


图 2 改进的客户端体系结构

Resource Versions 数据文件:客户端新增的模块,用来存储重组的消息实体数据,由资源标识符索引。实质上担任了客户端浏览器的角色。数据存储文件中的资源信息与 Fetched Objects 中的信息是同步的,存储的是资源关联而不是具体的消息实体。资源版本号(Resource Versions)由客户端数据去重模块 CDM 保存。

CDM 模块:拦截客户端 UserRequests 或者是 Prefetch Requests 模块发送的 HTTP 请求报文。具体工作:查询 Resource Versions 数据存储文件,获得客户端缓存中的所有资源的资源标识符。CDM 通知 Communications Interceptor(通信拦截)模块对报文做继续处理。CDM 分配一个缓冲区,采用块划分算法重组来自服务器的原始资源。CDM 检查元数据,每一 chunk 由一个元组引用,复制 Resource Versions 数据存储中的消息实体数据到重组缓冲区。当数组和 chunk 之间不匹配时,CDM 复制非冗余数据块,维持原 chunk 的顺序。通过这种方法,原资源版本被重组。CDM 在

Resource Version 数据存储中存储最新的资源版本号,如果发现相同资源较早的版本出现,则替换出去,保证客户端总能保证资源最新的版本。

预取管理模块(Prefetch Manager Block):预取引擎位于客户端则会有较好的预取效果。故在客户端添加预取管理模块。通过读取记录模块的访问队列,检查预取对象池(Fetched Objects),确认资源是否已经预取过。若没有预取过,则把请求发送给服务器,预取管理模块创建多个 User Request 线程并等待一个新的请求。当从服务器接收到响应资源时,预取管理模块检查其 URL 是否在预取队列中,若在则移除 URL 并插入到预取对象池。预取模块同时检查请求队列是否为空,若为空,当一个新用户请求入队时直接被送至服务器。当一个新的客户端请求入队时,预取管理模块暗示删除已经预取过的资源并清空 Hint List 数据存储。

User Requests 模块接收来自预取管理模块的请求,再传递给 Request 模块。为了避免后来资源的预

取,当从服务器接收到一个响应资源时,User Requests 把响应报头的 hint 插入到预取队列 (Prefetch List) 数据存储中,把资源的 URL 插入到预取对象池中。

1.2 数据去重模块基本原理

文中采用的数据去重系统来源于 dedupHTTP 系统。其中,CDM 和 SDM 分别工作在客户端和代理服务器端。双端数据去重系统架构如图 3 所示。

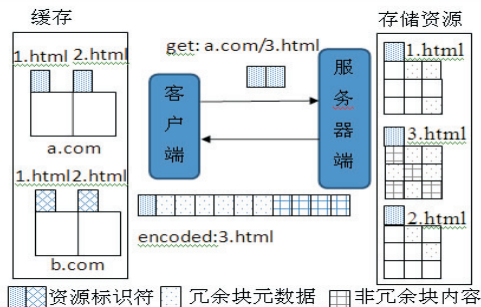


图 3 双端数据去重架构

具体工作流程如下：

当 SDM 接收到一个给定的资源请求,检索由 CDM 发送的参考源标识符的一个自定义的请求报头。然后 SDM 从服务器中取出该资源。接收过充分响应的报头和数据后,SDM 给资源分配一个新的标识符。将资源数据划分为块,块的元信息存储在数据存储文件中。在这个数据存储库中 SDM 保证了块的哈希索引的元信息资源的所有版本的所有块。

SDM 遍历资源所有块。对每个资源块,将在参考源具有相同散列块。如果有一块没有匹配的参考源,SDM 会在当前响应资源块中搜索。因此,冗余检测不仅在资源在 CDM 的高速缓存执行,而且在资源被送交 CDM 时也执行。SDM 组合最终响应资源发给 CDM。响应由元数据段开始。元数据的大小(字节)存储在一个自定义 HTTP 响应报头中。元数据的内容从响应的资源标识符开始。在每个参考源中包含了四元组,每个元组包含的信息是 CDM 能够在缓存中找到每一个冗余块的必需信息。具体包含的字段是当前响应的偏移量 Offset,资源标识符 Resource identifier,资源内部的偏移量 InOffset 和块的长度。

数组按照在原始响应中相应块的顺序排列。在元数据块的结尾,附加上非冗余数据内容,仍然保持原响应中的顺序。由于对该元组和非冗余数据的顺序维护,CDM 仅需数组的第一偏移量就可知道如何在非冗余数据附加上冗余数据。

当 CDM 收到响应后,为所有的数据重建原有的资源,包括从本地缓存资源中复制块引用信息和复制接收到响应的非冗余数据内容。CDM 并不存储任何一块元信息,也不需要发送或存储哈希。

通信拦截模块:负责对 CDM 和请求端信息做语

法处理。当 Communications Interceptor 接收到 CDM 的通知时,该模块为信息添加自定义报头“X-vrs”,即客户端参考源的 X-vrs 报头就包含了资源标识符。Communications Interceptor 把报头信息附加在 HTTP 请求报头上,给 Requests 模块,最后传给服务器。

当从代理服务器端接收到 HTTP 响应报文,Communications Interceptor 模块对报头进行语法分析,分别从 X-vrs 和 X-mtd 报头提取资源标识符和元数据的长度。为了进一步处理拆分数据,分别进入三个不同的缓冲区,一个缓冲区存放报头信息,一个存放元数据,一个存放非冗余数据。最后又将所有数据传给 CDM。

1.3 基于滑动窗口的数据块划分

基于滑动窗口的数据块划分方法主要依据块的内容划分,该方法的查重率高、分块更为合理有效^[11-12]。SDM 模块采用 LBFS 算法将服务器响应给客户端的每一个资源版本号划分为索引数据 chunk。

第一步:当客户端发出一个请求时,服务器将资源划分为若干索引 chunk。判断出该资源的哪些部分客户端已经存在,哪些部分是新的、需要发送的。

令 c_i 为第 i 个资源流的字节, k 为 Karp-Rabin 块的长度, b 为进制的基数。Karp-Rabin 块^[13]的哈希如下:

$$H(c_i \cdots c_{i+k}) = c_i \times b^{k-1} + c_{i+1} \times b^{k-2} + \cdots + c_{i+k-1} \times b + c_{i+k}$$

其中, b 是一个常数。

函数的应激性允许计算下一个字节的哈希,如下:

$$H(c_{i+1} \cdots c_{i+1+k}) = \{ [H(c_i \cdots c_{i+k}) - c_i \times b^k] + c_{i+1} \} \times b$$

第二步:选择代表资源的哈希。为了节省内存空间,选择合适的哈希作为较大数据块的边界。选择窗口机制,能够提供更好的冗余检测。分别确定最小和最大的 chunk 尺寸,对比期望 chunk 大小,基于文中方法的内容可能会造成过大的 chunk 或者过小的 chunk。

块哈希表的查找复杂度是 $O(p/k)$,其中 p 是参考源中块的数量, k 是一个常数。最坏情况下,参考源中不存在块信息。SDM 的哈希查找算法的复杂度是 $O(n * m * p)$,其中 n 是当前处理的块数量, m 是参考源的数量。当 m 相当小,例如最终用户并且缓存中只剩下很少的资源数量,大文件小容量块,将得到 $n * p \gg m$,可见算法复杂度的决定性影响因子是块的大小。

2 实验与结果分析

2.1 实验参数

系统的评价标准分别是字节节省率和延迟。前者

是指每次请求所节省的字节传输作为冗余块的总和,即没有从代理发送客户端而是从客户机获得引用资源的数据块的字节数的总和。后者指对于一个给定资源,从客户端 GET 请求直到客户端响应该资源的时间间隔。

客户端机器配置: Intel Pentium 4, 2.16 GHz, 4 GB 内存, 操作系统 Linux Mint 14-32b。服务器配置: Intel Pentium 4, 3.20 GHz, 6 GB 内存, 操作系统 Linux Mint 14-32b。实验运行在带宽为 54 MB/s 的局域网。限制带宽下测试, 模拟蓝牙有效带宽 2.1 Mb/s。

工作负载中应用的各种测试是通过下载一个典型的新闻网站 (www. dn. pt) 3 个层次的深度的文件。实验中, 请求所有的工作文件, 相当于客户机-服务器之间传输约 20 MB 的文件。具体地, 最大的字节数为 214 595, 最小字节数为 298, 平均字节数 29 672, 总大小为 19.8 MB。

客户端请求自动使用 Wget 实用程序。为了模拟一个真正的用户发起 Web 导航, 引入用户请求允许预取操作之间的空闲时间, 使用了 Wget 开关 “-w10” 和 “randomwait”, 主要为了引入一个 5、15 秒请求之间的随机延迟。

2.2 实验结果分析

实验一: 重点分析当执行数据去重时, 冗余检测效率对块大小的依赖情况。预取 (PF) 和数据去重 (DDP) 同时执行 (PFON-DDPON), 块的平均尺寸依次设置为 32、64、128、256、512、1 024 和 2 048 个字节。一般情况下, 设置最小块的大小等于平均块尺寸的 1/4, 块的最大尺寸等于平均块大小的 2 倍。实验结果参见图 4 和图 5。

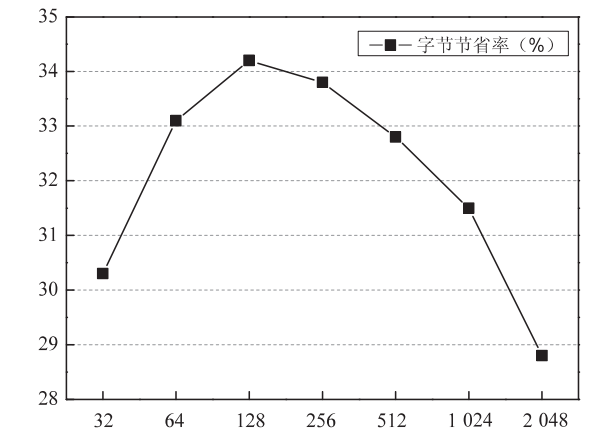


图 4 PFON-DDPON 相对于 PFON-DDPOFF 字节节省率 (带宽 54 Mbs)

图 4 显示了 PFON-DDPON 时刻相对于 PFON-DDPOFF 时刻的字节节省率。结果表明, 最好的冗余检测获得的块大小为 128 字节, 节省了 34.2% 的字节数量。 万方数据

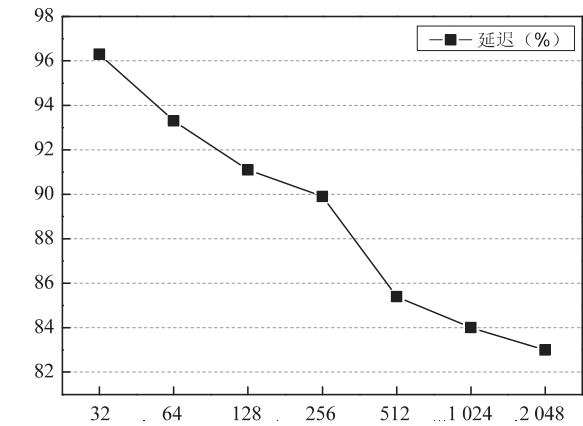


图 5 相对于标准 HTTP 传输情况 (PFON-DDPON, 带宽 54 Mbs)

图 5 显示延迟时间和块大小成反比关系。原因是随着块尺寸的增加, 块的数量减少, 那么数据去重的计算开销也要增加。因此数据去重算法执行时间也要兼顾。

为了获得更高的字节节省率, 设置块大小为 128 字节。具体实验对比了 PFOFF-DDPOFF 和 PFOFF-DDPON, PFON-DDPOFF 和 PFON-DDPON。带宽设置为 54 Mbs, 在字节节省率方面, PFOFF-DDPON 方法比 PFOFF-DDPOFF 方法节省了 34.1%, PFON-DDPON 方法比 PFON-DDPOFF 方法节省了 34.2%。结果表明, 引入数据去重技术可以节省 34% 的字节传输。

实验二: 使用该系统传输每种资源的延迟时间与标准 HTTP 传输时延迟的比值对比如图 6 所示。对比标准 HTTP 传输, 仅有去重时能够减少大约 8% 的延迟, 有预取时减少的延迟更明显, 分别减少 14.7% 和 8.9%。差别的原因在于数据去重算法存在一定的计算开销。综合考虑可见, 当使用预取和数据去重时, 数据去重保证了一定的字节节省率, 但是以适度增加的等待时间为代价, 保证了系统的整体性能。

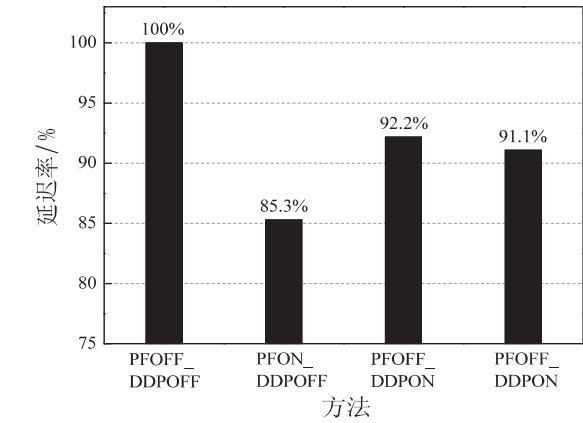


图 6 相对于标准 HTTP 传输延迟情况

实验三: 评估在可达到的冗余值中对不同带宽的影响效果。分别在 LAN 条件 (54 Mbs) 和蓝牙条件

(2.1 Mbs)下进行 PFON_DDPON 操作的比较测试,如表 1 所示。

表 1 不同带宽下相对于 HTTP 标准传输的延迟 %

指标	LAN	Bluetooth
字节节省率	34.2	34.3
资源延迟	91.1	95.6

结果表明,带宽对字节节省率基本无影响。在蓝牙情况下每个资源的延迟时间减少的幅度是 LAN 情况下减少的一半,分别为 4.4% 和 8.9%。由于预取频率依赖于用户请求之间的空闲时间,增加的延迟意味着请求需要更长的时间完成,因此可以减少空闲时间,从而减少预取请求启动的机会。

3 结束语

提出一种基于客户-服务器端双端去重的 Web 预取系统的设计方法,旨在提高网络访问效率,改善用户预期的延迟。实验结果表明,该系统一方面相对于正常传输的字节数传输量显著减少,节约了大约 34%;另一方面可以实现减少大约 8% 的用户预期的延迟,即使因为网络的客观条件影响实际效果,也能够保证降低 4% 的延迟。未来将进一步研究如何更准确地判定数据去重阶段的相似块以及如何降低数据去重算法的时间开销。

参考文献:

[1] 程龙泉. 基于预测模型和缓存替换策略的网络资源访问研究[J]. 科技通报,2017,33(10):134-136.

[2] 姚瑶,王战红,石磊. 一种基于页面聚类的 Web 概念化建模新方法[J]. 微电子学与计算机,2015,32(1):156-

160.

[3] ARONOVICHA L, ASHERB R, HARNIK D, et al. Similarity based deduplication with small data chunks[J]. Discrete Applied Mathematics, 2016, 212: 10-22.

[4] ZHANG Panfeng, HUANG Ping, HE Xubin, et al. Resemblance and mergence based indexing for high performance data deduplication[J]. Journal of Systems and Software, 2017, 128: 11-24.

[5] HIRSCHA M, ISH-SHALOMA A, KLEINB S T. Optimal partitioning of data chunks in deduplication systems[J]. Discrete Applied Mathematics, 2016, 212: 104-114.

[6] WIDODO R N S, LIM H, ATIQUZZAMAN M. SDM: smart deduplication for mobile cloud storage[J]. Future Generation Computer Systems, 2017, 70: 64-73.

[7] 王闪, 谭良. Web 大数据环境下的相似重复数据清理[J]. 计算机工程与设计, 2017, 38(3): 646-651.

[8] 熊忠阳, 牙漫, 张玉芳. 基于网页正文结构和特征串的相似网页去重算法[J]. 计算机应用, 2013, 33(2): 554-557.

[9] 李超, 王树鹏, 云晓春, 等. 一种基于流水线的重复数据删除系统读性能优化方法[J]. 计算机研究与发展, 2013, 50(1): 90-100.

[10] 孙爱玲, 冉禄纯. 一种基于重复数据删除的网络文件备份系统设计与实现[J]. 计算机应用与软件, 2014, 31(10): 86-90.

[11] WIDODO R N S, LIM H, ATIQUZZAMAN Z. A new content-defined chunking algorithm for data deduplication in cloud storage[J]. Future Generation Computer Systems, 2017, 71: 145-156.

[12] 王红梅, 李芬田, 王泽儒. 基于滑动窗口数据流频繁项集挖掘模型综述[J]. 长春工业大学学报, 2017, 38(5): 484-490.

[13] 王歧, 卢毓海, 刘洋, 等. 支持模式串动态更新的多模式匹配 Karp-Rabin 算法[J]. 计算机工程与应用, 2017, 53(4): 39-44.