基于标记的缓存协作分布式 Web 服务器系统*

林曼筠[1,2+], 钱华林[1]

[1](中国科学院 计算机网络信息中心,北京 100080)

[2](中国科学院 计算技术研究所,北京 100080)

A Distributed Web Server System Based on Cooperative Cache Using Tag

LIN Man-Yun[1,2+], QIAN Hua-Lin[1]

[1](Computer Network Information Center, The Chinese Academy of Sciences, Beijing 100080, China)

[2](Institute of Computing Technology, The Chinese Academy of Sciences, Beijing 100080, China)

+ Corresponding author: Phn: 86-10-68409547, E-mail: lmanyun@yahoo.com

http://www.cstnet.net.cn

Received 2001-08-13; Accepted 2002-04-10

Lin MY, Qian HL. A distributed Web server system based on cooperative cache using tag. *Journal of Software*, 2003,14(1):117~123.

Abstract: In this paper, the state-of-the-art DWSS (distributed Web server system) technologies are described, including their mechanisms, advantages and disadvantages. Then a new algorithm TB-CCRD (tag based cache cooperative Web requests distribution) for Web requests distribution is presented. Based on it, a scalable high performance DWSS is constructed. In this new architecture of DWSS, the front-end server organizes the caches of each backend Web server into a virtual large cache to improve the hit rate of cache and the response time of the DWSS. The operation of TCP connection handoff is distributed among the backend servers to prevent the front-end from becoming a bottleneck of the system. Tags are used to indicate where the content of a URL is cached, so as to avoid extra communications within the system. Through this mechanism, the TB-CCRD DWSS achieves both higher performance and better scalability.

Key words: DWSS (distributed Web server system); load balancing; cooperative Web caching; Web requests distribution algorithm; Web performance analysis

摘 要: 介绍了提高 Web 服务器性能的前沿技术——分布式 Web 服务器系统,讨论了现有各种方案的优缺点,在此基础上提出一种新的分布式 Web 服务器系统.该系统使用基于标记的缓存协作用户请求分发方法(tag based cache cooperative Web requests distribution,简称 TB-CCRD),通过前端机把系统中各个 Web 服务器的缓存组织成一个大的虚拟缓存系统,提高系统的整体缓存命中率,缩短了请求的响应时间;通过分布式处理 TCP 连接转交来消除前端机的性能瓶颈;利用标记通告 URL 在缓存中的位置,避免了额外的系统内部通信.从而得到了一个可扩展的高性能分布式 Web 服务器系统.

* Supported by the National High Technology Development 863 Program of China under Grant No.863-306-ZD08-01-1 (国家 863 高科技发展计划)

第一作者简介: 林曼筠(1968－),女,海南琼山人,博士,工程师,主要研究领域为计算机网络技术.

关键词: 分布式 Web 服务器系统;负载平衡;协作式 Web 缓存;Web 请求分发算法;Web 性能分析
中图法分类号: TP393 文献标识码: A

提高 Web 服务器性能的传统方法是升级单机系统.这种做法简单、易行,但由于业务的发展和多种随机事件的发生,准确预测网站的访问量几乎是不可能的,因此,满足当前性能要求的服务器有可能很快就需要进一步的升级.另外,被替换下来的机器很可能被闲置,从而造成资源和资金的浪费.

分布在 LAN 或者 WAN 上的多台 Web 服务器主机通过自组织方式或者由专门的设备负责组织调度的方式进行协同工作,组成一个 Web 站点,共同分担用户对该站点的 Web 请求负荷,我们称这样的系统为分布式 Web 服务器系统(distributed Web server system,简称 DWSS).DWSS 是目前最前沿的服务器性能提升技术之一,其关键技术在于系统的组织结构、系统各部分之间的协作方法以及相应的任务分配——用户请求分发算法.请求分发的目的在于通过负载平衡增加系统的吞吐量,缩短用户响应时间.本文提出了一种新的算法,并据此构造了一个 DWSS 系统,在性能和可扩展性方面均有显著的提高.

第 1 节介绍几种典型的 DWSS,并分析它们的优点和不足.第 2 节介绍我们提出的基于 TB-CCRD(tag based cache cooperative Web requests distribution)的 DWSS.第 3 节通过性能的数值比较和仿真实验研究新系统的扩展性和性能.第 4 节是结论.

1 典型的分布式 Web 服务器系统

使用镜像技术的 DWSS[1,2],由置于网络上的服务器的多个拷贝共同来提供服务,分担负荷,提高站点的处理能力.该系统中的多台主机对用户是可见的,不了解各服务器的处理能力和拥挤程度的用户必须首先指定某一镜像站点为其提供服务,即便在某些服务器仍然空闲的情况下,用户的请求也可能会因发往某个已是满负荷运行的镜像点而得不到服务;另一方面,站点的运营者无法控制负载的分布.

以 DNS 为中心的 DWSS[3,4],由服务器端 DNS 按照某种调度方法,将用户请求的 URL 中的主机名解释为 DWSS 中某个服务器的 IP,使相应的客户请求发往该服务器.常用的 DNS 调度方法有 DNS 轮询(DNS round-robin)和加权 DNS 轮询调度算法(weighted round-robin scheduling,简称 WRR)[4].然而,由于 URL 和 IP 的对应关系会在浏览器及许多中间媒介的 DNS 中缓存,因此,只有部分用户请求可以由服务器端来调度;而服务器端 DNS 每次选中系统中的某个服务器,都有可能在某一时段(TTL)内给该服务器带来难以预测的突发流量.

以分配器为中心的 DWSS[5~7]以分配器为前端机,它不但可以控制所有的用户请求流量,而且还可以为站点提供单一的 IP 映像.一般情况下,所有的用户请求都首先发往分配器,由它按照某种调度策略将请求转发给系统中的某一个 Web 服务器主机(又称为后端机)来处理.分配器上使用的用户请求分发方法有多种.简单轮询(round robin)、URL-Hash 等静态调度方法不考虑服务器的实际负载等状况,难以实现真正的负载平衡;HTTP 重定向、最小连接法(least-connection scheduling)以及带权重的最小连接法等动态调度算法,利用定期收集的服务器负载等状态参数动态调整负载的分布,从而改善负载平衡效果.但是,http 重定向使用户必须重新建立 TCP 连接,增加了延迟,也增加了网络流量,而分配器转发过程中对数据包的改写会引入额外的延迟,如果处理不当,则可能使分配器成为系统的瓶颈.

在基于分布式用户请求调度策略的 DWSS 中,组成系统的每一个服务器都参与用户请求的调度,它克服了集中调度方式中因核心设备负载过重而易成为系统瓶颈的缺点[8].使用 HTTP 重定向方式控制负载的分布[9];利用特制的 Web 服务器软件对 Web 页面中嵌入的 URL 进行操作并完成相应的 URL 内容的自动拷贝和转存,从而实现对负载分布的控制.这些方法的弱点是增加了网络通信负荷,并引入了较明显的延迟.

在上述 DWSS 中,用户请求的分发方法都只考虑了多个服务器的负载平衡,而没有考虑 Cache 的命中率,长时间来看,系统中的每一个服务器都倾向于提供该站点的全部内容,而这些内容往往会超出单个服务器的缓存能力;另一方面,相同的内容可能在多个服务器中重复缓存,资源利用率低,系统的整体缓存命中率不高[10,11].实际上,CPU 速度的提高远远要比磁盘存取速度的提高快得多,因此,当磁盘存取速度远远不能满足 CPU 的需求

时,提高 Cache 的命中率对于改善 DWSS 的性能将是十分有益的.另外,使用“识别内容的请求调度策略(content-aware requests distribution,简称 CARD)”[11~13],有利于分布式存储策略的实施,也便于使用专门的服务器为特定的一些客户提供服务,以确保服务质量.多方面的用途使 CARD 日益受到关注.

CARD 首先了解用户请求的文档内容,然后再选择适当的服务器来处理该请求.由于 HTTP 协议是建立在面向连接的 TCP/IP 协议上的,因此,必须首先经过 3 次握手,建立用户与 Web 服务器端的 TCP 连接,才能了解用户请求的内容.为了保持调度方法对用户透明且开销最小,必须有某种方法,使实际处理用户请求的服务器无须再次建立与用户的 TCP 连接,而是接受已有的连接,直接处理用户的请求并给出回应[11,12].TCP 连接转交(TCP hand off)[11]是目前较好的方案.基于 LARD(local aware requests distribution)[11]的 DWSS 使用 TCP 连接转交方法获得了比 WRR 更高的吞吐率.Mohit Aron 等人[12]指出,该系统的前端机是潜在的性能瓶颈,并提出了一种改进的方法 SCARD.SCARD 通过使用多个设备分担前端机的工作,以改善系统的可扩展性.但是,该系统中前端机性能的改善是以增加系统内部各设备之间的通信量为代价的.分发器需要与分配器进行通信,以便了解 URL 的分布情况.在完成 TCP 转交或者连接中断时,分发器需要与前端机通信,以便于更改连接信息.这些内部通信不但增加了参与通信的各个部件的负担,也占用了系统内部宝贵的网络带宽资源,使系统的扩展性受到限制.我们建议分发器使用批量查询的方法,即当收到多个用户的 URL 请求之后再向分配器查询其分布情况,而不是为每一个用户的 URL 请求单独发送一个查询包.显然,这种方法将增加用户的响应延迟.

2 基于标记的缓存协作分布式 Web 服务器系统

在基于 LARD 的 DWSS 中,前端机的工作主要包括 TCP/IP 连接的建立、数据包的转发、URL 分发和 TCP 连接的转交.实验表明[11,12],TCP/IP 连接的建立和转交的延迟都远远大于 URL 分发和数据包转发的延迟,因此,TCP 连接的建立和转交是系统扩展时前端机不堪重负的主要原因.

为了保持 LARD 和 SCARD 的优点,克服其不足,我们设计了一种新的 DWSS,即基于 TB-CCRD 请求调度方法的 DWSS.该系统通过集中管理 URL 在各个服务器 Cache 的分布来提高 Cache 的命中率;利用标记传递 URL 分布信息,避免额外的网络通信量;用分布式处理 TCP 连接转交消除系统的瓶颈.

2.1 系统框架结构及工作原理

基于 TB-CCRD 的新系统采用与 Linux 直接路由式虚拟服务器相同的框架结构.如图 1 所示,DWSS 的各个后端服务器通过高速以太网相互连接,它们屏蔽 ARP 协议,并拥有与前端机相同的 IP(记为 vIP)和 Web 服务端口号(记为 vPort),在用户看来,这个系统就相当于一个 IP 地址为 vIP,服务端口号为 vPort 的 Web 服务器.在新系统中,前端机负责接收来自用户的数据包,发放用于指示处理该数据包最适当的服务器 ID(可以是后端机在系统内部的标识符,也可以是它的 MAC 地址)的标记,并转发数据包;后端机则负责处理由前端机转发而来的用户数据包并直接回应用户,具体包括建立/拆除与用户的 TCP 连接、TCP 连接转交以及提供 URL 内容等工作.

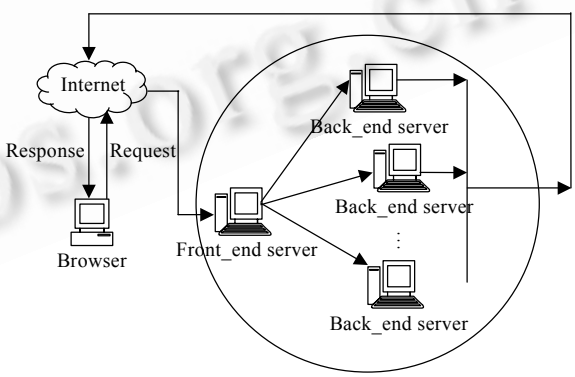

Fig.1 The framework of the TB-CCRD based DWSS
图 1 基于标记的缓存协作 DWSS 系统框架

2.2 系统工作流程

2.2.1 连接状态表和 URL 分布表

为了转发用户数据包和实现对后端机缓存系统的统一管理,前端机需要维护两张表:连接状态表和 URL 分布表.连接状态表通过某一连接的用户端标识(即用户的 IP 地址和端口号,记为(IP_c,$PORT_c$))检索出服务器系统中负责处理该连接的后端机的 MAC 地址、该连接的最近使用时间和连接状态.此外,根据某个后端服务器的

MAC 地址检索该表,还可得到该服务器当前负荷情况(由当前的活动连接数表示).

URL 分布表通过 URL-hash,检索对应的 URL 内容在哪一个(或一组)后端机的本地磁盘或者 cache 中(后端机由它的 MAC 地址代表、本地磁盘命中标志的“True”/“False”表示该 URL 是/否在硬盘中,cache 命中标志的“True”/“False”表示该 URL 是/否在 cache 中).该表还记录各个 URL 的大小(url_size)及其在 cache 中最后一次被访问的时间.在站点发布新的内容时,产生或者更新 URL 分布表.初始状态,磁盘命中标志(Disk_hit)根据 URL 是否在本地磁盘中存储而分别设为“True”或“False”,而所有的缓存(Cache_hit)命中标志都设为“False”,当 URL 内容进、出后端机的 cache 时,缓存标志作相应的修改.

2.2.2 系统的工作流程

TB-CCRD 方法需要前端机和后端机相互配合来实现.基于 TB-CCRD 的 DWSS 的工作流程如图 2 所示.

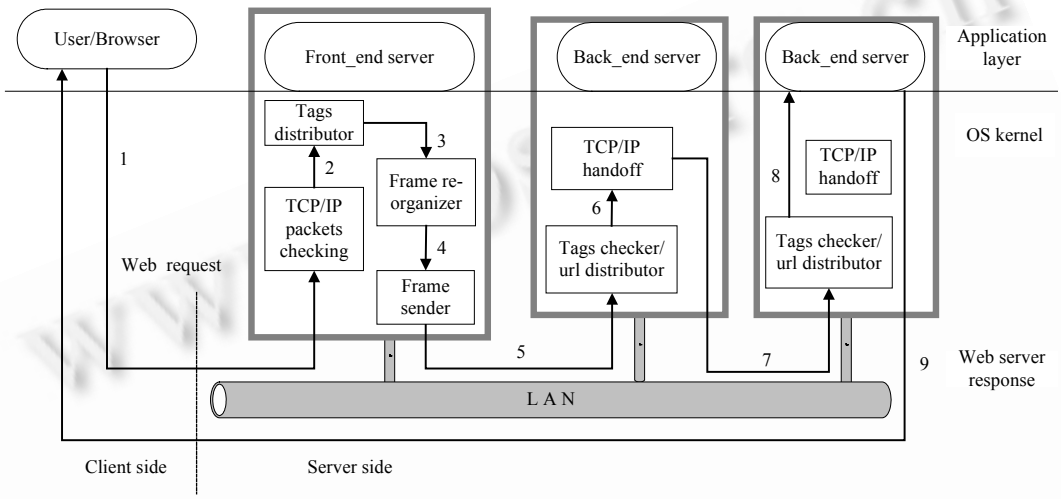

Fig.2 The operation flow chart of the TB-CCRD based DWSS

图 2 基于标记的缓存协作 DWSS 的工作流程图

为了描述方便,我们按照 TCP 标志位和 TCP 数据部分长度的不同特点,将 DWSS 收到的来自用户的 IP 数据包分为 4 类:

(1) TCP 连接建立请求包.用于向 Web 服务器请求建立 TCP 连接,特点是 TCP 的 syn 标志位置 1,简称为 syn 包.

(2) 信息确认包.这是指用户收到来自服务器的数据包(比如连接请求应答包,即 ack 和 syn 都置 1 的包,或者包含用户所请求的 URL 内容的数据包)之后发给服务器的确认数据包,其特点是 TCP 标志位 ack 置 1,且 TCP 的数据部分长度为 0,简称为 ack 包.

(3) URL 请求包.这是 TCP 连接建立之后,用户用于向服务器索取 URL 内容的数据包,其特点是 ack 位置 1,且 TCP 的数据部分包含用户希望索取信息的 URL 字段等信息,长度不为 0,简称为 url 请求包.

(4) TCP 连接结束数据包.这类数据包在用户希望结束 TCP 连接或者收到不属于当前 TCP 连接的服务器端数据包时使用,其特点是,TCP 的 fin 或者 rst 位置 1,简称为 fin/rst 包.

在新系统中,前端机位于用户通往真正提供服务的后端 Web 服务器的逻辑路径上,每一个来自用户的 IP 数据包(记为 P)都必须经由它中转.为了避免重新计算 IP checksum,我们将标记插在以太网帧头和 IP 数据包头之间.考虑到 MAC 地址能够方便地惟一识别一台后端机,我们采用后端机的 MAC 地址作为标记,并将特殊的 MAC 地址“0x0000 0000 0000”称为 0 标记.

若 P 为 syn 包,则前端机根据 WRR 规则为它选择一个后端机(记为 BE_{tmp},其 MAC 地址为 MAC_{tmp}),向 BE_{tmp} 转交由 P 和 0 标记构成的数据帧,并将相应 TCP 连接的状态信息存入连接状态表中;若 P 是 url 请求包,则首先由前端机的标记分发模块为它选择一个标记 tag(记为 MAC_{best},相应的后端机记为 BE_{best}),然后,前端机向当前处理该连接的后端机转交由 P 和相应标记(即 MAC_{bes})构成的数据帧,并修改连接状态表中相应 TCP 连接的状态

信息(更新“最后通信时间”、“连接状态”,并将“处理该连接的后端机 MAC 地址”改为标记分发模块新选中的 MAC_{best});若 P 是 ack 包或者 fin/rst 包,则前端机根据连接状态表读取处理该连接的后端机 MAC 地址(MAC_{tmp} 或 MAC_{best}),向它转交由 P 和 0 标记构成的数据帧,对 ack 包,前端机还要修改连接状态表中相应 TCP 连接的状态信息(更新“最后通信时间”、“连接状态”等信息),对于 fin/rst 包,前端机还要删除连接状态表中相应的记录.

标记分发模块产生标记 tag 的方法如下:

(1) 根据 P.url 查询 URL 分布表和连接状态表,获取 Cache_hit=“True”,且当前活跃连接数目最小的后端机,记为 BE_{best}.

(2) 如果所有后端机缓存中都没有 P.url,或者(1)选中的后端机已经满负载运行,则

(a) 另选一个对 P.url 而言 Disk_hit=“True”,且当前活跃连接数最小的后端机,记为 BE_{best};

(b) 更改 URL 分布表中相应于(P.url,BE_{best})的记录,把 P.url 在 BE_{best} 中的 Cache 命中标志置为“True”;

(c) 检查 BE_{best} 是否有足够的 cache 空间存放 P.url,如果空间不足($\sum$url_size >MAX_CACHE_SIZE),则按照预先约定的缓存替换策略,把即将被替换出缓存的 url 的 Cache_hit 置为“False”.

(3) 将所选中的最佳后端机 BE_{best} 的 MAC 地址 MAC_{best} 作为标记,即 tag=MAC_{best}.更新 URL 分布表中相应于(P.url,BE_{best})的记录的最后访问时间.

如果站点内容可以全部存在于一个后端机的本地硬盘中,则这是上述算法的一个特例,只要将 URL 分布表中的“本地磁盘命中标志”都置为“True”即可.

HTTP1.1 允许在一次 TCP 连接过程中提交多个 URL 请求,这些 URL 可能缓存于不同的后端机中,这时,系统需要进行多次 TCP 连接转交.上述算法同样适用.

后端机负责建立和拆除与用户之间的 TCP 连接,按照前端机给出的标记处理用户的 URL 请求,实现 URL 在 Cache 中的分布.根据标记的不同,当后端机收到数据包时,执行不同的操作:

- 若 tag 为 0 标记,则由本机的 Web 服务器进程来对数据包进行处理:建立与用户的 TCP/IP 连接,或者从本地 Cache 中获得相应内容并应答用户,若 Cache 不命中,则从磁盘获得数据并缓存,然后回应给用户;
- 若 tag 不是 0 标记,则将相应的 TCP 连接转交给 tag 所指示的最佳后端机 BE_{best},由它来接手处理该 URL 请求以及此后属于该连接的所有数据包.

从系统的算法流程中可以看到,前端机对系统缓存的集中管理,使其能够给出 TCP 的转交信息,后端服务器在完成连接转交之后无须向前端机通告连接的变更信息;同时,前端机通过在客户的 URL 请求数据包中插入标记的方式通告 TCP 的转交目标;这些措施使得新系统在使用分布式处理 TCP 连接转交的同时无须引入大量额外的系统内部通信开销,节省了宝贵的内部带宽资源.

3 性能的数值比较与仿真实验

LARD 通过提高 URL 请求命中率获得了较好的 Web 服务性能,但是,LARD 使用集中处理的方式完成 TCP 连接的建立和转交,导致前端机容易成为系统瓶颈,扩展性较差.SCARD 致力于改进前端机的处理能力,从而改进系统性能和扩展性,但是它的 URL 分发方案带来了随系统规模增长的不可忽视的系统开销,使系统的扩展性受到限制.WRR 系统虽然有更好的扩展性,但是 Web 服务性能较差.以上方法在系统的可扩展性和 Web 服务性能上不可兼得.DWSS 系统的扩展性主要体现在前端机的处理能力以及系统内部信道的开销上,而 Web 服务的性能主要体现在 URL 的 cache 命中率及吞吐率上.我们的 TB-CCRD 方法同时考虑了以上两个方面的问题,在维护系统良好的扩展性的同时获得较高的性能.下面,我们通过模拟实验和性能比较来分析 TB-CCRD 系统的扩展性和 Web 服务性能.

3.1 前端机扩展性比较

根据文献[11,12],在基于 LARD 的 DWSS 中,前端机处理一个请求所需的 CPU 时间为

T_1=连接建立时间(T_{conn})+TCP 转交时间($T_{handoff}$)+任务分派时间($T_{dispatch}$)+Pack 转发时间($T_{tranfer}$).

其中 $T_{handoff}$,T_{conn} 都在 $T_{transfer}$ 的 15 倍以上,而 $T_{dispatch}$ 为 $T_{handoff}$ 的 10%~20%.

对基于 TB-CCRD 的系统,在同等条件下,前端机处理一个请求所需的 CPU 时间为

T_2=P.syn/P.fin/rst/P.url/P.ack 的转发时间(4*$T_{transfer}$)+任务分派时间($T_{dispatch}$).

可见,TB-CCRD 系统前端机的处理能力是 LARD 系统的 5 倍以上.

3.2 系统内部信道开销比较

在 SCARD 系统中,分配器与每一个后端机之间建立一个持久的 TCP 连接,当后端机收到客户的 URL 请求之后,它将利用这个连接向分配器询问该 URL 该由哪一个后端机来处理,因此,了解一个 URL 的缓存位置所带来的系统内部信道开销包括从后端机发往分配器的查询数据包和相应的分配器的回应数据包.根据文献[12]提供的原型系统,后端机发往分配器的查询数据包的最小长度为

以太网帧报头/尾(26B)+IP 报文报头(20B)+应用层报文报头(22B)+URL_hash(16B)=84Bytes.

分配器的回应数据包最的小长度为

以太网帧报头/尾开销(26B)+IP 报文报头开销(20B)+应用层报文报头(26B)=72Bytes.

TB-CCRD 系统采取类似于 MPLS 夹缝头(shim header)的标记封装方式,在客户的 URL 请求包中加贴一个标记来传播 URL 的缓存分布信息.大多数的 HTTP 连接包含 5~6 个来自用户的数据包[12],因此,后端机了解一个 URL 的缓存位置所带来的系统内部通信开销为 6Bytes*6=36Bytes,小于 SCARD 系统通信开销的 1/4.

3.3 Web服务性能仿真实验

3.3.1 仿真模型及其参数

为了研究和比较不同的用户请求分发策略的效率和性能,并与文献[11]具有可比性,我们参照文献[11]给出的实验模型和参数,以软件方式定制了 DWSS 仿真模型,对 TB-CCRD 和 WRR 调度算法的性能进行比较.DWSS 仿真模型如图 3 所示.

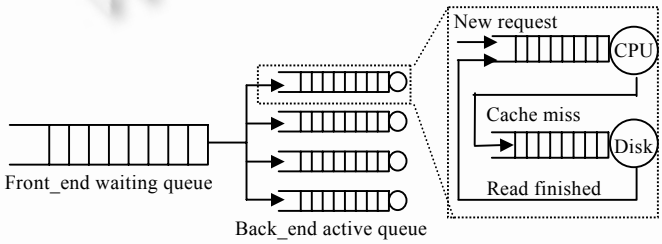

Fig.3 DWSS simulation Model

图 3 DWSS 仿真模型

在该模型中,每一个后端 Web 服务器都有自己的 CPU 和磁盘,用户的 Web 请求经过前端机的调度后进入各后端机的等待处理队列.对用户 Web 请求的处理包括连接建立、磁盘文件读取(Cache 不命中时)、数据传输、连接拆除几个步骤.我们使用与文献[11]相同的延迟参数.

我们在数值仿真实验中使用 LRU 缓存替换算法,大于 500KB 的文件不缓存.同一个请求的几个处理步骤必须顺序进行,不同请求的 CPU 处理时间、访存时间和磁盘存取时间可以重叠,并假设网络性能和前端机的速度都能够满足系统的最大吞吐率要求.

3.3.2 仿真实验的输入文件和输出结果

仿真实验的用户请求序列来自 Clarknet 的存档轨迹文件[14].系统的整体吞吐率以轨迹文件中请求的个数除以模型处理完这些请求所需要的时间而得到,Cache 不命中率由不命中的请求个数除以输入请求的总个数得到.图 4 和图 5 是仿真实验的结果.

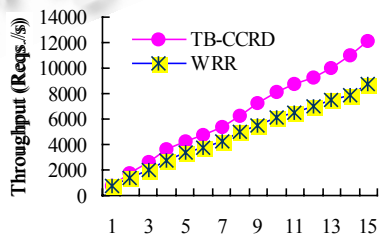

Fig.4 Throughput

图 4 吞吐率

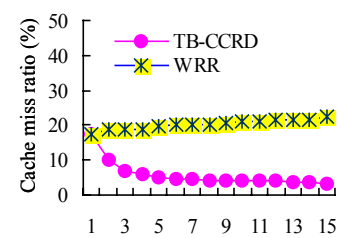

Fig.5 Cache miss ratio

图 5 Cache 不命中率

从图中可以看出,基于 TB-CCRD 的 DWSS 的命中率明显高于基于 WRR 的系统,而且随着系统规模的扩大,前者的性能越来越优于后者.数值仿真实验的结果表明,相对于 WRR 算法,尽管 TB-CCRD 算法引入了 URL 分发和 TCP 连接转交等额外的开销,但由于新系统显著提高了

缓存命中率,因此能够较显著地提高系统的吞吐量,从而缩短对用户请求的响应时间,提高 Web 服务性能.

4 结 论

本文在分析现有的分布式 Web 服务器系统技术的基础上,提出了一种基于标记的缓存协作分布式 Web 服务器系统,该系统通过前端机将系统中各个服务器的缓存组织成为一个大的虚拟缓存系统,以集中管理的方式控制 URL 在这个大的虚拟缓存系统中的分布,并利用在正常通信数据包中加贴标记的方式传递 URL 的分布信息.该系统的特点是:

(1) 使用识别内容的请求分发方法,在进行用户请求分发时不仅考虑负载分布情况,同时还考虑 URL 的 cache 命中率问题,从而提高系统的总体 cache 命中率,缩短用户请求响应时间,提高 Web 服务性能.

(2) 把导致前端机性能瓶颈的 TCP 连接转交分布到各个后端机去完成,随着系统规模的扩大,参与完成 TCP 连接转交的主机数目也相应增多,从而使得前端机不会成为系统瓶颈.

(3) 把提高 URL Cache 命中率的分布式功能通过标记的方式实现,避免了额外的网络通信负荷.

这些特点很好地克服了 LARD,SCARD 和 WRR 的不足.性能比较和仿真实验结果表明,基于 TB-CCRD 的新系统比 LARD 和 SCARD 系统扩展性好,比代表当前先进水平的基于 WRR 的系统具有更高的 Web 服务性能.

References:

[1] Crovella ME, Carter RL. Dynamic server selection in the Internet. In: Proceedings of the 3rd IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems (HPCS'95). Piscataway: IEEE Communication Society Press, 1995. 158~162.

[2] Myers A, Dinda P, Zhang H. Performance characteristics of mirror servers on the Internet. In: Proceedings of the IEEE INFOCOM'99. Piscataway: IEEE Press, 1999. 304~312. http://dl.comsoc.org/cocoon/comsoc/index.html.

[3] Brisco T. DNS support for load balancing. RFC 1794, 1995.

[4] Cardellini V, Colajanni M, Yu KS. Dynamic load balancing on Web-server system. IEEE Internet Computing, 1999,3(3):28~39.

[5] Dias D, Kish W, Mukherjee R, Tewari R. A scalable and highly available server. In: Proceedings of the COMPCON Spring'96—the 41st IEEE International Computer Conference. Los Alamitos: IEEE Computer Society Press, 1996. 85~92. http://www.computer.org/proceedings/compcon/7414/7414toc.htm.

[6] Iyengar A, Challenger J, Dantzig P. High-Performance Web site design techniques. IEEE Internet Computing, 2000,4(2):17~26.

[7] Schroeder T, Goddard S, Ramamurthy B. Scalable Web server clustering technologies. IEEE Network, 2000,14(3):38~45.

[8] Andresen D, Yang T, Holmedahl V, Ibarra OH. SWEB: towards a scalable World Wide Web server on multicomputers. In: Proceedings of the IPPS'96. Los Alamitos: IEEE Computer Society Press, 1996. 850~856. http://computer.org/proceedings/ipps/7255/7255toc.htm.

[9] Baker SM, Moon B. Distributed cooperative Web servers. Computer Networks and ISDN Systems, 1999,31(11-16):1215~1229.

[10] Lin YW, Zhang DJ, Qian HL. A cooperative Web caching system based on concentrated management. Journal of Computer Research and Development, 2001,38(1):68~73 (in Chinese with English Abstract).

[11] Pai VS, Aron M, Banga G, *et al*. Locality-Aware request distribution in cluster-based network servers. In: Proceedings of the ASPLOS-VIII. New York: ACM Press, 1998. 205~216. http://computer.org/proceedings/ipps/7255/7255toc.htm.

[12] Aron M, Sanders D, Druschel P, *et al*. Scalable content-aware request distribution in cluster-based network servers. In: Proceedings of the USENIX 2000 Annual Technical Conference. Berkeley, CA: USENIX Association, 2000. 323~336. http://www.usenix.org/events/usenix2000/program.pdf.

[13] Cherkasova L. FLEX: load balancing and management strategy for scalable Web hosting service. In: Proceedings of the ISCC 2000. Los Alamitos: IEEE Computer Society, 2000. 8~13. http://computer.org/proceedings/iscc/0722/0722toc.htm?SMSESSION=NO.

[14] Danzig P, Mogul J, Paxson V, Schwartz M. Internet traffic archive. Berkeley, CA: Lawrence Berkeley National Laboratory, 2000. http://ita.ee.lbl.gov/html/contrib/ClarkNet-HTTP.html.

附中文参考文献:

[10] 林永旺,张大江,钱华林.一个基于集中管理的协作式 Web 缓存系统.计算机研究与发展,2001,38(1):68~73.