分布式 Web 服务器技术综述

A Survey on Distributed Web Server Techniques

马晓星 吕 建

(南京大学计算机科学与技术系 软件新技术国家重点实验室 南京 210093)

Abstract With the explosive growth of the World Wide Web, many popular Web sites are faced with the challenge of the overload of tremendous requests. The best way out is using distributed Web server systems for their good scalability and low costs. In this paper, we try to give a comprehensive survey on the underlying techniques of the distributed Web server systems: request-dispatching mechanisms, load-balancing algorithms, Web content replication and distribution schemas and other important aspects.

Keywords Web Server, Distributed System, Survey

World Wide Web (WWW,或简称 Web)是一个基于 Internet 的超文本分布信息系统。自十来年前诞生于欧洲粒子实验室以来,它以难以置信的速度迅速普及,现今许多重要的信息服务都通过 Web 来提供。随着 WWW 规模的爆炸性增长,许多 Web 站点的访问量都大大超出了普通 Web 服务器所能承担的范围。Web 服务器的过载会导致服务质量下降(响应时间过长,请求丢弃率过高等),甚至会使服务器宕机。

解决这一问题的直接办法是升级 Web 服务器的硬件,采用处理能力更强的计算机系统,如使用更快的 CPU、SMP 并行体系结构等。然而,这种方法的扩展性有限,性价比不高。更好的办法是使用分布式 Web 服务器系统。而若使用地理上分布的 Web 服务器系统则不仅能扩展 Web 服务器的客户请求处理能力,而且还可以节约网络带宽这一宝贵资源。

迄今为止,分布式 Web 服务器系统已经有许多成功的应用[1]。然而,要让一组 Web 服务器像一个服务器一样可靠并且高效地工作,有不少问题需要仔细考虑,其中某些问题还有待进一步研究。V. Cardellini 等人在文[2,18]中讨论了几种不同的分布式 Web 服务器及其负载均衡机制。本文将从一个更宽广的视角对各种分布式 Web 服务器技术做一个更全面的介绍和分析,特别是那些文[2,18]中未曾收入的新近的工作。

1. 分布式 Web 服务器

1.1 WWW 与 Web 服务器

WWW 采用 Client/Server 的基本结构,其基本工作原理非常简单。促使 WWW 获得全球性广泛流行的关键技术因素有三:一是一个简易友好的超文本语言 HTML,二是一个满足全球唯一性的命名规则 URL(更准确地说应是 URI(统一资源标识符,参见 RFC1630)。URL 是 URI 的一种形式,URI 还包括 URN(统一资源名)等。本文只涉及到 URL),三是一个轻便有效的通信协议 HTTP。

HTML,全称为超文本标记语言,是有史以来最成功的标记语言——有人估计 WWW 上有 1000 亿个 HTML 页面。HTML 本身规模较小且易于移植,可以很容易地在各种计算平台实现。HTML 可以在两维空间上展现文本或其他对象。HTML 也很简单易学,易于为用户接受。然而最重要的一点是 HTML 的超文本特性:HTML 页面中可以任意嵌入指向其他文档的指针(称为超链接),用户只需点击一下鼠标,就可以顺着超链接访问到也许位于地球另一端的另外一个页面。这构成了 WWW 的基础。

作为一个全球性的公共信息系统,WWW 需要一个系统的规则来命名其上的各个资源(包括 HTML 文档和其他对象),每个资源的名字应是全球唯一的。统一资源定位符 URL 利用服务器的名字来区分不同服务器上的资源,对于同一个服务器上的资源则用类似于 UNIX 文件系统的树形目录来命名。URL 还指明了访问该资源的协议和服务器端口,对于 WWW 来说通常是 HTTP 协议,端口缺省为 80。

超文本传输协议 HTTP[9]是一个用于分布协作超媒体信息系统的应用层协议,是 WWW 上 Web 服务器、代理服务器、浏览器等实体进行通信的主要协议。一个简单的 HTTP 交互的工作过程如下:浏览器首先和服务器建立一个 TCP 连接,再从这个通道向服务器发出一个 HTTP 请求,服务器再从这个通道发回他对这个请求的 HTTP 响应,最后连接关闭。HTTP 的一个重要特点是无状态,也就是各次 HTTP 交互之间是相互独立的。这意味着不论这些 HTTP 交互在实际内容上是否相关,都可以将它们分散到多个镜像服务器上进行处理,而对用户来说其效果与一台服务器是一样的。

实际上用户访问一个 WWW 资源的过程可能稍微复杂一些。图 1 展现一个典型的访问过程。用户要访问 http://www.w3.org/History.html,他的浏览器首先要获得 www.w3.org 的 IP 地址,于是就向本地的 DNS 发出一个域名解析请求;本地的 DNS 发现 www.w3.org 不是本地的域名,且本

6 Wetherall D, et al. ANTS: A toolkit for building and dynamically depoloying network protocols. In: Proc. IEEE OpenArch' 98, San Francisco, CA: Apr. 1998
7 Hicks M. PLANA--Programming Language for Active Network Architecture. http://www.cis.upenn.edu/~switchware/papers/plan.ps, 1997
8 Merugu S, et al. Bowman: A Node OS for Active Networks. IEEE INFOMCOM' 2000, 2000
9 Alexander D S, et al. Active Network Encapsulation Protocol (ANEP). http://www.cis.upenn.edu//~switchware/ANEP/docs/ANEP.txt, 1997
10 Legedza U, et al. Improving the performance of distributed applications using active network. IEEE INFOCOM' 98, 1998
11 陈晓林,等. 一个基于主动网络的大规模可靠组播协议的设计和实现. 电子学报,录用

地的缓存里也没有对应的有效记录，于是它就从高层 DNS 那儿获得 w3.org 这个域的 DNS 的地址，向它发出域名解析请求，得到回答"18.176.0.26"，再转发给客户浏览器。浏览器接着就要建立一个到这个地址的 TCP 连接，并通过这个连接发送请求"GET /History.html HTTP/1.1 \n HOST www.w3.org"。然而，在浏览器和 Web 服务器之间可能有一个或多个透明或不透明的代理服务器，如果这些代理服务器的本地缓存中有 History.html 的有效副本，那就直接返回给浏览器（图中虚线所示），否则就从 18.176.0.26 获得一个拷贝并返回给浏览器（图中实线所示）。

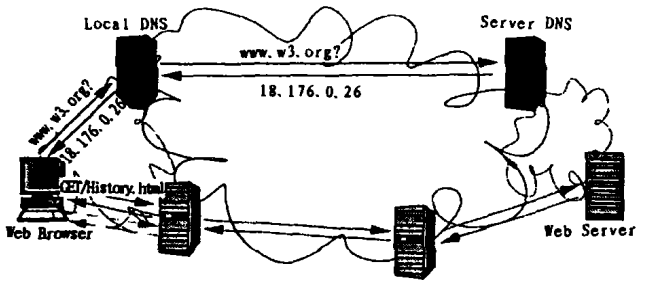


图 1　一个 WWW 访问过程

Web 服务器是指提供 Web 服务的软硬件系统。常见的 Web 服务器软件有开放源码软件 Apache、NCSA 的 HTTP 以及 Netscape、Microsoft、IBM 等公司的商业产品。Web 服务器软件的基本功能是响应客户浏览器发出的 HTTP 请求，把从文件系统或其他途径取得的 HTML 页面或其他对象发送给浏览器。现在的 Web 服务器还都扩充了对 CGI 和 Perl 等内嵌服务器端脚本的支持。实用的 Web 服务器在实现上都作了高度的优化，以同时支持大量的客户访问请求，然而单独一台服务器的能力总是有限的，为了满足不断增长的客户访问需求，可以使用分布式 Web 服务器。

1.2　分布式 Web 服务器系统

一个分布式 Web 服务器系统内部包含多个成员 Web 服务器，它们在某种机制的协调下合理地分担整个系统的负载。而这种分布性对用户应是基本透明的，整个分布式 Web 服务器系统只有一个唯一的域名。用户通过这个域名能够像访问通常的 Web 服务器那样来访问分布式 Web 服务器。现在有很多著名 Web 站点为了分流客户访问请求，在不同的地理位置提供了多个镜像站点，用户可以选择较近的站点访问。这并不能算是分布式 Web 服务器，因为这对用户不透明。而另有一些同样是很著名的 Web 站点，成功地使用了分布式 Web 技术，为全世界的大量用户提供了优质的服务。例如 1998 年长野冬奥会站点就创造了在 16 天内响应 6 亿多个请求和一分钟 11 万多次点击两项吉尼斯世界记录。

从地理分布的角度，分布式 Web 服务器可以分成两种类型：基于局域网的分布式 Web 服务器和基于广域网的分布式 Web 服务器。前者又称为 Web 服务器集群，用廉价的通用系统构建的规模较小的 Web 服务器集群常被用来替代较为昂贵的基于 SMP 的 Web 服务器系统，而大规模的高性能 Web 服务器集群则可达到单个 SMP 系统无法企及的性能指标。后者的各个成员服务器在地理上是分布的，因而除了可以提供很高的处理能力以外，还可以通过适当的调度、让距客户较近的服务器来响应客户请求，从而节约主干网带宽并缩短响应时间。这种地理上的分布性还可以进一步增强系统的可用性。

文[18]从系统结构的角度，依据分派客户请求的实体的不同，把分布式 Web 服务器分为基于客户的、基于 DNS 的、基于分配器的和基于服务器的四种类型。

现有各种分布式 Web 服务器工作原理虽不尽相同，但要解决的基本技术问题却大体一致：一是要有某种机制把客户的请求分派到各个成员服务器上，二是要有一个算法来指导请求分派以保证各个成员服务器的负载均衡，三是要在各个成员服务器恰当地复制和分布 Web 站点内容以维护其一致性并保证成员服务器的存取效率，此外还有整个分布式 Web 系统的可用性、可扩展性等其他问题。

2. 请求分派机制

如何把客户请求分派到组成系统的各成员服务器上是分布式 Web 服务器首先要解决的问题。这种请求分派的工作可以在下述不同的层次上进行：

2.1　DNS 分派

上面我们提到，用户使用 URL 访问 WWW 上的特定资源时，用户的浏览器一般首先要从这个 URL 中取出拥有这个资源的主机的域名，再从 DNS 服务器取得这个域名对应的 IP 地址，然后才能与该主机通信。如果我们让负责解析分布式 Web 服务器系统的域名的 DNS 服务器在不同时刻把这个域名分别解析为各个不同的成员 Web 服务器的 IP 地址，那么用户的访问请求就会被导向各个 Web 服务器。这就是基于 DNS 的客户请求分派技术。早期的分布式 Web 服务器如 NCSA 的系统[5]就采用了这种方法。

通过 DNS 来分派用户的访问请求是一种间接的办法。出于节约带宽和提高响应速度的考虑，用户的浏览器和本地 DNS 服务器会对先前的查询进行缓存，这样随后一段时间内对同一域名的查询将直接返回缓存中的结果。这个效应会导致 DNS 分派的效果大打折扣。把域名纪录的有效期（TTL）缩短有助于缓解这个问题，但是这会加大 DNS 的流量和负载，并且过短的 TTL 会被客户的 DNS 服务器忽略。

2.2　HTTP 重定向

前面我们提到，Web 浏览器通过 HTTP 协议[9]和 Web 服务器通信。在 HTTP 协议中定义了一种重定向机制。当一个浏览器向某个服务器发出一个 HTTP 请求，而得到的 HTTP 响应的状态码为 3XX（重定向）时，浏览器就必须采取进一步的动作才能获得服务。如果这个进一步的动作是"GET"或者"HEAD"，浏览器就可以不通知用户干预而自动地执行。比如一个常见的情况是某服务器上的某个页面被搬到了一个新的地方（可能在另外的 Web 服务器上），Web 服务器就可以使用重定向机制，让用户用老的 URL 访问这个页面时，会自动重定向到新的 URL 上去。这种技术也可用于分布式 Web 服务器，比如当某个成员服务器的负载太高时，对这个服务器的访问请求会被自动重定向到负载较低的服务器上去。UCSB 的 SWEB[10]就使用了 HTTP 重定向技术。这种技术的主要优点是实现简单，兼容性好。然而进行 URL 重定向本身也需要一定的开销，它还会浪费一些网络流量，并且用户等待时间被拉长了。

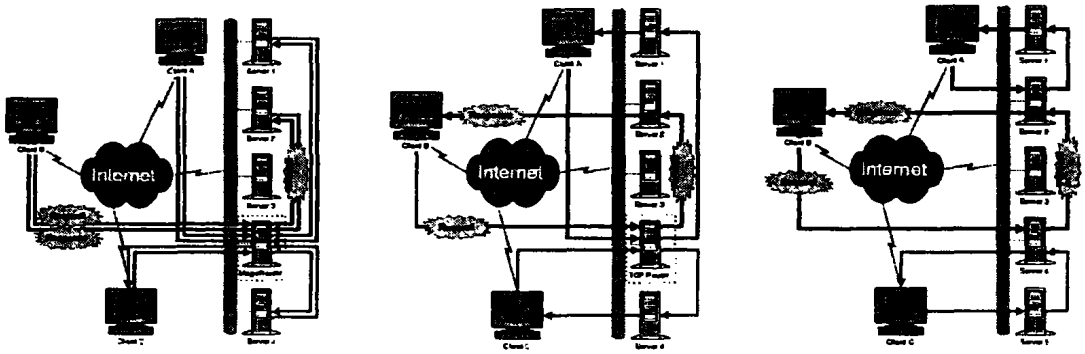
2.3　动态页面迁移

实际上，WWW 本身就是一种分布式的技术。我们很容易有这样的想法：如果一个 Web 服务器的负载太高，那么我们何不把它上面的内容分成几部分，分别放到几个服务器上？这原则上是可行的，然而有两个问题需要解决：一是我们无法预先得知各个页面的访问频率，实际上它们可能相差极大，从

而难以合理地把它们划分为总访问率相当的几个部分；二是这种划分可能会对用户造成额外负担，他们记不住许多个URL。Arizona 大学的 Scott M. Baker 和 Bongki Moon[11]注意到一般用户访问一个网站总是从少数几个重要的URL（称为入口）开始，并且他们很少关心页面中内嵌对象（如图片等）和框架内嵌页面的 URL。他们开发了分布协作 Web 服务器系统（Distributed Cooperative Web Server System，DCWS），在这个系统中，当服务器负载过高时，就根据访问频率，选择其上的某个非入口文档，动态迁移到系统中负载最低的服务器上，而相应的指向这个文档的超链接会被动态修改。

2.4 TCP 层包重写

这种技术的出发点是在 TCP 层次上进行客户请求的分派，对高层应用透明。这样一方面分派代码可以在系统核心层实现，或者用专用系统实现，从而保证效率；另一方面可以尽量避免高层应用软件的修改。这样整个 Web 服务器对外可以只有唯一的一个 IP 地址。这里需要解决两个问题，一是如何将客户端向这个地址发送的请求的 TCP 报文分派给各个成员 Web 服务器，二是保证响应请求的 TCP 报文的源 IP 地址都是这个外部可见的唯一地址。实现上可以采用包重写（Packet Rewriting）技术来解决这两个问题。



a. 双向包重写　　b. 单向包重写　　c. 分布式包重写

图 2　包重写（本图引自文[8]）

包重写技术可以上溯到网络地址转换（Network Address Translation，NAT）技术[3]。NAT 的原意是为了节约紧张的 IP 资源，让一个局域网内的各个主机使用对外不可见的私有地址，而只以一个公共 IP 地址对外通讯。NAT 技术不改变既有的网络协议，而仅以 NAT 服务器替换一般的路由器。NAT 服务器充当局域网内主机与外部主机 TCP 连接的中介，它负责将内部主机发向外部的包（Packet）中的源地址修改为公共地址，而将外部发送来的包中的目的地址从公共地址修改为恰当的内部主机地址，并调整相应的校验和（checksum）。很明显，这个技术稍作调整即可用于分布式 Web 服务器：一组通常的 Web 服务器被置于一个局域网内，它们拥有各自的私有地址；客户将向唯一的公共地址发送访问请求，而 NAT 服务器（可称为前端机）将按照某种负载均衡策略（下文讨论）将这些请求分发给内部各个 Web 服务器，并将其响应作相应修改后发给客户，如图 2a 所示。采用这种技术的例子有 UC Berkeley 的 MagicRouter，Cisco 的 LocalDirector 和 IBM 的 Network Dispatcher。

上述基于 NAT 的系统中，负责分配客户连接的前端机还要负责重写响应请求的包，而对 WWW 服务而言，一般请求的响应比请求本身大得多，这会导致前端机负载太重而成为整个系统的瓶颈。为解决这个问题，可让各成员 Web 服务器直接将发给客户端的 TCP 包的源地址设为公共地址。这就大大地减轻了前端机的负载，如图 2b 所示。这种方法称为单向包重写，相应地，前述给予 NAT 的方法称为双向包重写。单向包重写的代价是要求修改各成员 Web 服务器的网络协议实现。在文[6]中，这里的前端机称为 TCP Router。

单向包重写技术仍然需要一个集中的前端机来分配客户连接，在系统规模较大时这仍会成为瓶颈。为此，有人提出了分布式包重写技术（Distributed Packet Rewriting，DP-R）[8,4]。其基本想法是，取消这个前端机，而让系统中的某些甚至全部 Web 服务器除了提供 Web 服务外还要承担请求分派的工作。这样这个系统对外有 m 个 IP 地址，而拥有 n（n>=m）个 Web 服务器。再使用前面介绍的 DNS 分派技术，就可以把针对系统域名的连接请求分配到这 m 个 IP 地址上，然后再经过包重写，分派到最终提供服务的 n 个 Web 服务器上。

2.5 IP 层的分派

除了包重写技术以外，Damani 等人提出了一种称为 ONE-IP 的技术[7]，也可以把发向同一个 IP 地址的包分派给不同的成员 Web 服务器。这样的系统对外只有唯一的公共 IP 地址；再利用 IP 别名技术让每个成员服务器除了自己独有的 IP 地址外，还都把这个公共 IP 设为自己的第二地址。客户发向这个公共 IP 的包将会被分派给其中的某一个服务器。分派的依据是根据客户的 IP 地址，进行一次 Hash 变换，得到唯一的某个服务器的私有地址。这样做的理由是同一个客户的所有的 IP 包将由同一个服务器来处理，从而保证 TCP 连接不会中断。在实现上，文[7]提出了两种办法，一是使用一个特殊的路由器，负责进行 Hash 变换并将 IP 包送给某服务器，这种办法无需改动各个服务器的网络实现；二是采用广播的方式，让所有的服务器都可以收到 IP 包，但只有一个服务器会真的接受这个包，这就需要改动服务器的网络驱动程序，加入依据 IP 包源地址来决定是否接受这个包的代码。

最后值得一提的是，实际的大规模分布式 Web 服务器往往综合使用了前面讨论的多种技术。比如我们可以首先使用 DNS 分派技术将请求分派到地理上相距极远的各个成员服务器上，而每个成员服务器本身又是一个使用包重写技术的分布式 Web 服务器。这方面的例子包括美国 NASA 喷气推进实验室报道火星登陆的网站和长野冬奥会的官方网站。

3. 调度

上面我们讨论了分派客户请求的机制，接下来的问题就是当一个客户请求到来时，应该将其分派给哪一个成员服务

器?与其他分布式系统一样,各成员服务器的负载是否平衡是分布式 Web 服务器能否达到预期性能的一个关键所在。因此,尽量均衡各成员服务器的负载就成为分布式 Web 服务器系统调度的主要目标。人们以往对各种分布式系统的调度问题已经有大量卓有成效的研究[12,13],为分布式 Web 服务器系统提供了良好的基础,而各种分布式 Web 服务器在这方面也有不少独特之处。

在下面的讨论里,我们约定:整个分布式 Web 服务器系统拥有 n 各成员服务器 $S_0,S_1,\cdots,S_{n-1}$;系统按照先来先服务(FIFO)的顺序处理客户请求。

3.1 静态调度

静态调度相对比较简单,它不管各个成员服务器运行时刻的负载情况,而只根据预先的决策进行客户请求的分配。

静态调度中的最简单的算法是随机选择。若客户请求满足泊松过程,且平均服务时间满足指数分布,各个服务器的队列就都是 M/M/1 排队系统,系统是稳定的。如果各个成员服务器是异构的,处理能力分别为 $C_1,C_2,\cdots,C_n$,则以概率

$$P_k=C_k/\sum_{i=1}^{n-1}C_i,\quad k=0,1,\cdots,n-1$$

为服务器 S_k 分配请求。

轮转(Round Robin)选择就是把客户请求依次分派给各个成员服务器:将第 i 个请求分配给 S_k,$k=i \bmod n$。对于异构系统,也可采用与上述类似的按处理能力比例分配的处理方法。使用轮转法的最有名的例子就是 NCAS 的 Round-robin DNS。

静态调度也可以利用一些其他的信息,例如客户的地址。前面我们介绍的 ONE-IP 系统利用客户 IP 作散列来选择服务器。而对于地理上分布的 Web 服务器系统来说,客户地址就更加有用了,可以尽量让距离这个客户较近的那个服务器来为这个客户提供服务。

3.2 动态调度

影响 WWW 客户访问频率的因素很多,且难以预测。大多数分布式 Web 服务器系统中静态调度往往不能令人满意。此时可以考虑根据各服务器运行时刻的负载情况及其他信息进行动态的负载均衡。

最直观的想法是将客户请求分配给当前负载最低的成员 Web 服务器。许多已有的分布式 Web 服务器系统的确采用了这个算法。这个算法的问题在于:收集各个服务器的当前负载会导致额外的开销,因而这只能间歇地进行;而另一方面,过时的信息很可能会误导分配算法,从而破坏负载均衡。

如何利用不那么及时的信息进行负载均衡引起了许多有趣的研究。现在我们知道,在这种情况下引入一定的随机性反而会改善整个系统的总体性能。比如,先随机地选取 2 个成员服务器,而后再从中选择负载较低的一个分配客户请求。这称为 Pick-2。推而广之,可得到 Pick-K,即先随机选择 K 各成员服务器,再从中选一个负载最低的。这样,K=n 就是总选择整个系统中负载最低之服务器,而 K=1 就是前面我们所讨论的随机分配。Mitzenmacher[17]对 Pick-K 进行了较详细的分析和模拟实验,他得出结论认为在大多数情况下 K=2 是较好的选择。

文[16]提出了一个对 Pick-K 的改进,称为 Pick-KX,它进一步加入了一些受控的随机性:首先从 $S_0,S_1,\cdots,S_{n-1}$ 中选出 K 个服务器 $W_0,\cdots,W_{K-1}$,$1\leqslant K\leqslant n$;设其负载分别为 $L_0,\cdots,L_{K-1}$;记

$$L_{total}=\sum_{i=0}^{K-1}L_i;$$

$$X_j=(L_{total}-L_j)/L_{total}\quad j=0,\cdots,K-1$$

则以概率

$$P_j=X_j/\sum_{i=0}^{K-1}X_i$$

将当前请求分配给服务器 W_K。

文[19]中也提出了对 Pick-2 的改进的两个算法,称为 Basic LI 和 Aggressive LI。记负载信息更新时间间隔为 T,成员服务器数目为 n,平均每服务器的请求到达率为 λ,服务器 S_i 的负载为 $L_i(i=0,\cdots,n-1)$,预计 T 时间间隔内到达的客户请求数为 $arrive_T=T*\lambda*n$,系统总负载 $L_{tot}=\sum_{i=0}^{n-1}L_i$,将当前客户请求分配给服务器 S_i 的概率为 P_i。Basic LI 的基本目标是决定以怎样的比例分配客户请求以使在本阶段结束后各成员服务器的负载尽量均衡。如果

$$\forall i,(\frac{L_{tot}+arrive_T}{n}>L_i)$$

那么 $$P_i=(\frac{L_{tot}+arrive_T}{n}-L_i)/arrive_T$$

否则就表示各服务器之间负载不均衡度较高,而时间间隔 T 又较短,不足以均衡各个服务器。此时的处理较为简单,选取负载最小的那些成员服务器,并以一定的随机性向其分配客户请求即可。而 Aggressive LI 更进一步,它针对比较长的负载信息更新间隔。其基本想法是,在这个时间段的开始阶段就试图平衡各个成员服务器的负载,而后均匀分配客户请求。不失一般性,设 L_i 已从小到大排好序,并令 $L_n=\infty$ 作为终止标记;划分整个时间间隔为 n 个区间,在区间 j 内,在服务器 $S_0,\cdots,S_j$ 之间均匀地分配客户请求,以使它们的负载达到 S_{j+1} 的负载 L_{j+1}。这样,区间 j 的时间长度:

$$T_j=j*(L_{j+1}-L_j)/\lambda n,$$

而区间 j 内当前客户请求分配给服务器 S_i 的概率为

$$P_{ij}=\begin{cases}\dfrac{1}{j+1} & (i\leqslant j)\\ 0 & (i>j)\end{cases}$$

模拟实验表明[16,17,19],当负载信息很及时的时候,Pick-N 效果较好,当这个信息不那么及时时,它甚至不如 Pick-1。而 Pick-2 相对 Pick-1 有明显的改善。在此基础上,LI 和 Pick-KX 的性能又有进一步的提高。

前面我们提到,在使用 DNS 来分派客户请求的系统中,由于中间 DNS 服务器和客户端本身对域名解析的缓存,服务器 DNS 对客户请求的控制大为削弱。虽然减小域名解析的有效期(TTL)有助于改善这个问题,但作用有限,因而在这种系统中仅使用上述各种动态负载均衡还不能令人满意。文[14]提出了称为 adaptive TTL 的办法。这里考虑了如下因素:首先,客户请求来自不同的域,由于域 DNS 的缓存,该域在一个 TTL 内发出的请求(称为隐负载,hide load weight)将被集中导向同一个成员服务器。由于域的大小等因素的差别,各个域在同样的 TTL 发出的请求数目差别很大;其次,各个域的隐负载随着 TTL 的延长而增大。因此,要均衡系统的负载,除了可以在各成员服务器之间进行选择分派以外,还可以调整每次域名解析 TTL 的长短:对于请求率高的域的域名解析应赋以较低的 TTL,另一方面对指向负载较高或处理能力较弱的服务器的域名解析,也应给以较短的 TTL。

在使用 Web 页面动态迁移机制来平衡负载的分布式 Web 系统中,要统计对每个页面的请求的密度,以此判断每

个页面引起的负载的大小，从而选择适当的页面并将其搬到负载较低的服务器上去。

4. Web 内容的复制与分布

Web 内容的复制和分布是分布式 Web 服务器系统中的另一个重要问题。各个成员服务器在逻辑上共享一份完整的内容，而这些内容的物理存放方式将在很大程度上决定访问的效率，从而影响整个系统的性能。

对于基于高速局域网的分布式 Web 服务器系统（又称为 Web 服务器集群）来说，这个问题比较容易解决。由于高速局域网间通信的带宽高、延迟小，我们可以让各个成员服务器通过某种网络文件系统或分布式文件系统（如 NFS、AFS 等）来访问共享的 Web 文件。这种办法实现简单，便于管理，性能也能满足要求。

而对于地理上分布的基于广域网络的分布式 Web 服务器系统来说，共享文件的办法是行不通的，这会带来难以接受的延迟和网络开销。此时各个成员服务器需要访问的 Web 文件应当尽量存放在本地磁盘上。这里有两种基本手段，一是采用分布的办法，整个 Web 内容分布在各个成员服务器上，而在分配客户请求时尽量让拥有所请求的文档的服务器来处理这个客户请求。二是采用复制的办法，让所有的成员服务器都拥有一份 Web 数据的拷贝，这常常被称为“镜像”。对于前者，出于负载均衡的考虑，Web 内容的分布不能是静态的，而必须能够动态调整，如前面提到的 DCWS 系统[11]。而对于后者来说，必须设法保证各个拷贝之间的一致性，对于那些实时更新的内容这更为重要。解决这个问题的一个较好的办法是使用代理缓存（Proxy-Cache）技术。

Internet 上代理服务器十分常见，现在通过代理服务器已成为访问 Internet 的主要方式。WWW 代理服务器通常是由客户机构或者客户的 ISP 提供的。使用 WWW 代理服务器除了能保障内部网络安全、节约 IP 地址以外，还能节约网络带宽、缩短响应时间，这是因为通常 WWW 代理服务器都带有一个缓存（如 Squid[14]），用户浏览器通过代理访问外部的 Web 服务器上的 Web 对象时，这些对象可以拷贝一份存放在缓存中，当以后又有对这些对象的访问请求时，代理服务器就可直接返回缓存中的拷贝。这种技术也可以用于构建分布式 Web 服务器系统：为了分担一个 Web 服务器的负载，Web 服务的提供者也可以再设立多个代理服务器，并通过前面讨论的技术把客户的访问请求分派到这些代理服务器上，再由这些代理服务器向原始的 Web 服务器请求服务——由于缓存的作用，绝大部分访问请求将被这些代理服务器直接处理。这种代理服务器被称为“Web 服务器加速器”或“httpd 加速器”。其实这种“加速器”无论是在外部效果上还是在内部实现上都是地道的 Web 服务器，不过它必须能够从其他 Web 服务器而不是文件系统获取 Web 内容，并管理缓存。为了让缓存中的对象与原始 Web 服务器上的对象保持一致，每个缓存的对象都有一个生存期，超出这个期限后，对这个对象的访问就会促使 Web 服务器加速器向原始服务器再确认这个对象的有效性或更新这个对象[9,15]。原始服务器根据对象是否需要经常更新来决定其生存期的长短。Web 服务器加速器的加速效果在很大程度上决定于缓存的命中率。通过某些精心设计的优化措施如 DUP[15]和热点预取等可使缓存命中率提高到 97%以上[1]。

5. 可伸缩性、可用性及其它

良好的可伸缩性是分布式 Web 服务器系统获得高性能的本质保障。它使得可以通过向系统中添加成员服务器的办法来不断地提高系统的性能。许多分布式 Web 服务器系统就称为“可伸缩的 Web 服务器”[6,8,10,11]。各种分布式 Web 服务器系统都有一定的可伸缩性，但某些系统中存在一些潜在的瓶颈。在采用一个集中的设备来分派客户请求时，这个设备（称为分派器）就有可能成为瓶颈。例如，采用 DNS 分派技术的系统中的 DNS 服务器、采用包重写技术的系统中的前端机等。而分派器的负载高低与分派的粒度有关，粒度越小，负载均衡的效果就越好，可是分派器的负载就越高，可伸缩性就越差。不过，集中的分派器的功能一般较为单一，可以做专门的优化。例如文[1]中提到 IBM 的 Network Dispatcher 可运行在一个针对通信优化了的嵌入式 OS 上，可以每秒钟处理 10000 个 HTTP 请求。

分布式 Web 系统的又一个优点是可以提供更高的可用性。系统中一个或几个成员 Web 服务器的失效不会导致整个系统的失效或不合理的性能严重下降。要达到这个目标，首先要有一种失败检测手段，能够发现失效的成员服务器；然后再将失效的服务器从请求分配机制和负载均衡算法中剔除出去，待其恢复后再添加进去。对于 Web 服务而言，丢弃少量的客户请求是允许的（用户可以重发这些请求），因而上述措施的实现较为容易。某些分布式 Web 系统的可用性还受限于系统中的集中控制部分的可用性。例如使用单向包重写技术的 TCP router。此时可以在这些关键部位使用专用的高可靠性系统或冗余设备。

在第 2 部分我们提到使用地理上分布的 Web 服务器系统不但可以提高处理能力，还能节约主干网带宽。DNS 分派、HTTP 重定向、动态页面迁移等较高层的技术可以用于地理分布的 Web 服务器系统，而在 TCP/IP 层次上进行分派的技术则主要用于基于局域网的分布式 Web 服务器系统。虽然分布式包重写技术理论上可用于地理上分布的系统，但其重定向会导致较高的延迟。

在设计分布式 Web 系统时，对已有协议和系统的兼容性也是一个要考虑的重要问题。系统对外的协议兼容性是决定系统能否实用的前提。例如某些系统将 DNS 的 TTL 设为 0，这就不为许多 DNS 服务器接受；又如某些系统对客户浏览器有特殊要求，也难以广泛应用。系统内部采用标准的协议则可以避免对已有软硬件系统的修改，从而降低成本。

总结与展望 分布式 Web 服务器技术具有良好的伸缩性、可用性和性价比，是解决因流量不断增长而导致 Web 站点过载问题的最佳途径。本文较为全面地分析和介绍了分布式 Web 服务器系统所涉及的请求分派机制、负载均衡算法和内容复制分布等方面的技术问题，并在此基础上对分布式 Web 服务器的可伸缩性、可用性、地理分布性、协议兼容性等性质作了进一步的讨论。

人们对分布式 Web 服务器技术已有相当深入的了解，部分技术已获得商业应用。然而，还有不少问题有待进一步的研究。对于地理上分布的系统，由于网络的延迟，负载信息很不及时，如何在现有工作的基础上进一步均衡各成员服务器的负载？又如何将成员服务器和客户的地理位置信息引入负载均衡算法，以尽量避免请求分派中“南辕北辙”的情况？而更重要的挑战来自于 Web 本身的发展。例如 WWW 原本主要用

于信息发布，其上的绝大多数内容都是只读的，而现在WWW逐渐成为一个网络计算的平台，“只读”的假设不再成立，此时保证分布式Web服务器系统中各成员服务器的一致性就要付出更大代价。又如，随着网络带宽的提高，WWW上的大粒度对象（如视频对象）越来越多，它们必将对分布式Web服务器系统的请求分配、负载均衡和内容分布产生重大影响。这些都是迫切需要研究解决的问题。

参考文献

1 Iyengar A，et al. High-Performance Web Site Design Techniques. IEEE Computing，March-April 2000. 17～26
2 Cardellini V，Colajanni M，Yu P S. High Performace Web-server Systems. In：Proc. of 13th Int. Sym. On Computer and Information Sciences (ISCIS'98)，Ankara，Turkey，Oct. 1998. 286～293
3 Egevang K，Francis P. The IP Network Address Translator (NAT)，RFC1631，1994
4 Aversa L，Bestavros A. Load Balancing a Cluster of Web Servers Using Distributed Packet Rewriting. In：Proc. of IPCCC'2000：The IEEE International Performance，Computing，and Communication Conference，Phoenix，AZ，Feb. 2000. 24～29
5 Kwan T T，McGrath R E，Reed D A. NCSA's World Wide Web Server：Design and Performance. Computer，1995，28(Nov)：68～74
6 Dias D，et al. A scalable and highly available web server. In：Proc. of the IEEE Computer Conference (COMPCON)，Santa Clara，March 1996
7 Damani O P，et al. ONE-IP：Techniques for hosting a service on a cluster of machines. Computer Networks and ISDN Systems，1997，29：1019～1027
8 Bestavros A，et al. Distributed Packet Rewriting and its Application to Scalable Server Architectures. In：Proc. of the Intl. Conf. on Network Protocols，Austin，Texas，USA，Oct. 1998
9 Fielding R，et al. Hypertext Transfer Protocol--HTTP/1.1，Network Working Group，RFC2616
10 Andresen D，et al. SWEB：Towards a Scalable World Wide Web Server on Multicomputers：[Technical Report TRCS95-17]. Department of Computer Science，University of California，Santa Barbara，1995
11 Baker S，Moon B. Scalable Web Server Design for Distributed Data Management：[Technical Report 98-8]. Department of Computer Science，The University of Arizona，Aug. 1998
12 Shirazi B A，Hurson A R，Kavi K M. Scheduling and load balancing in parallel and distributed systems. IEEE Computer Society Press，Los Alamitos，Calif. 1995
13 Casavant T L，Kuhl J G. A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems. IEEE Transactions on Software Engineering，Feb. 1988. 141～154
14 See http://www.squid-cache.org/
15 Wessels D，Claffy K. Internet Cache Protocol (ICP) Version 2. RFC2186，Sep. 1997
16 Genova Z，Christensen K J. Challenges in URL Switching for Implementing Globally Distributed Web Sites. In：Proc of the Workshop on Scalable Web Services，Aug. 2000. 89～94
17 Mitzenmacher M. How Useful is Old Information. In：Proc. of the 16th Annual ACM Symposium on Principles of Distributed Computing，1997. 83～91
18 Cardellini V，Colajanni M，Yu P S. Dynamic load balancing on Web-server systems. IEEE Internet Computing，1999，3(3)：28～39
19 Dahlin M. Interpreting Stale Load Information. In：Proc. of the 19th Intl. Conf. on Distributed Computing Systems，May 1999
20 Colajanni M，Yu P S，Cardellini V. Dynamic load balancing in geographically distributed heterogeneous Web-servers. In：Proc. of the 18th IEEE Intl. Conf. on Distributed Computing Systems，Amsterdam，The Netherlands，May 1998. 295～302

（上接第4页）

科中的科学问题、核心概念、数学方法以及系统科学方法，其实它还隐含了学科中一些有趣的问题。其中一个问题就是计算领域的工作者应该具备什么能力？报告指出有两类能力：

1. 面向学科的思维能力：发现本领域新的特性的能力。这些特性将导致新的活动方式和新的工具，以便使这些特性能被其他人所利用。

2. 使用工具的能力：使用本领域的工具有效地进行其他领域实践活动的能力。

报告建议：把面向学科的思维能力作为大学计算专业课程设计的主要目的。同时，计算专业工作者必须充分熟悉工具，以便与其他学科的人们有效地合作，进行那些学科的设计活动。

在面向学科的思维方式这类问题上，世界著名计算机科学家、图灵奖获得者Dijkstra教授在充分的论述后，告诫我们在计算科学的教学过程中不要用拟人化的术语，而是要用数学的形式化方法。我国学者赵致琢博士也提出了思维方式的数学化的思想。我们认为，面向学科的思维能力应包含两层意思：其一是面向学科方法论的思维能力；其二是面向学科的数学思维能力。

现在，计算学科有了自己的学科方法论，有了一个关于学科认知领域的理论体系。为此，我们向学术界推荐这种计算认知领域的理论体系，并认为，随着这种理论体系的不断完善，必将提供给人们一种完全超出自己粗放的想象的认识计算学科的能力。

参考文献

1 Denning P J，et al. Computing as a discipline. Communications of the ACM，1989，32(1)：9～23
2 董荣胜. 计算教育哲学初探. 计算机科学，2000，27(1)：93～97
3 Turner A J，et al. A Summary of the ACM/IEEE-CS Joint Curriculum Task Force Report：Computing curricula 1991. Communications of the ACM，1991，34(6)：68～84
4 IEEE-CS/ACM Joint Task Force. Computing curricula 2001 (draft). http://www.computer.org/education/cc2001
5 Chang C K. Curricula 2001：Bringing the Future to the Classroom. Computer，1999，32(9)：85～88
6 国家教委社会科学研究与艺术教育司. 自然辩证法概论（修订版）. 高等教育出版社，1991
7 Denning P J. A debate on teaching computing science. Communications of the ACM，1989，32(12)：1397～1397
8 Dijkstra E W. On the cruelty of really teaching computing science. Communications of the ACM，1989，32(12)：1397～1404
9 Scherlis W L，et al. Colleagues respond to dijkstra's comments. Communications of the ACM，1989，32(12)：1405～1413
10 Dijkstra E W. Dijkstra's reply to comments. Communications of the ACM，1989，32(12)：1414～1414
11 赵致琢. 关于计算机科学与技术认知问题的研究简报（Ⅰ，Ⅱ）. 计算机研究与发展，2001，38(1)：1～15
12 王浩. 数理逻辑通俗讲话. 科学出版社，1983
13 吴允曾选集. 北京科学技术出版社，1991
14 莫绍揆. 递归论. 科学出版社，1987
15 Computing Sciences Accreditation Board Inc. Annual Report，Stanford，Conn.，1998