Web集群中文档组织分布的优化策略*)

熊 智 晏蒲柳 郭成城
(武汉大学电子信息学院 武汉 430079)

摘 要 Web集群服务器已被广泛用来提高Web服务器的性能。对于如今内容海量级的大型网站来说,如何在Web集群服务器上组织和分布Web文档是一个急需解决的问题。本文提出了一种Web集群服务器中文档组织和分布的优化策略,其目是减少集群系统的平均响应时间。通过对Web服务器日志的分析,挖掘客户的访问模式,将关联度高的网页聚类成网页簇,然后根据网页簇的负载计算其拷贝份数,最后在集群中优化分布各网页簇的拷贝。以这种方法组织和分布文档,可以减少服务器端的TCP连接迁移开销,可以实现集群内的负载均衡,从而减小集群系统的平均响应时间;且相对于内容全镜像的文档分布方案,节约了存储空间,减少了维护各服务器文档一致性的开销。

关键词 Web集群服务器,网页簇,文档组织,文档分布

Optimization Strategy for Organization and Distribution of Documents in Web Server Cluster

XIONG Zhi YAN Pu-Liu GUO Cheng-Cheng
(Department of Electronics Information, Wuhan University, Wuhan 430079)

Abstract Web server cluster has been widely used to increase the performance of Web servers. For a large website, how to organize and distribute the Web documents is imperative. This paper proposes a strategy to organize and distribute Web documents in Web server cluster, whose aim is to reduce the average response time of cluster system. Through the analysis of Web servers' logs, discover the client's access patterns, then group the webpages closely related into webpage clusters, then calculate every webpage cluster's copy number according to their load, finally distribute the copies of webpage clusters among servers. Such strategy can reduce the overhead of TCP connection migration, and achieve load balancing among servers, thereby reducing the average response time of cluster system. In addition, compared with the method of mirror image, our strategy saves storage space, and reduces the overhead of maintaining the coherence of the documents.

Keywords Web server cluster, Webpage cluster, Document organization, Document distribution

1 引言

Web集群服务器(简称Web集群)[1]已成为当前应用最为广泛的一种提高Web服务器性能的解决方法。Web文档的类型各种各样,对于如今内容海量级的大型网站来说,如何在Web集群上组织和分布Web文档是一个急需解决的问题。本文提出了一种Web集群中文档组织和分布的优化策略,其目的是通过将Web文档进行优化组织和优化分布来减小集群系统的平均响应时间。它通过对Web服务器日志的分析,挖掘客户的访问模式,将关联度高的网页聚类成网页簇,然后根据网页簇的负载计算其拷贝份数,最后在集群中优化分布各网页簇的拷贝。

2 相关工作

2.1 Web集群服务器

随着Internet的蓬勃发展,Web服务正成为最主要和最流行的网络服务。当用户急剧增多时,传统的Web服务器已经不能满足需求了。为了满足需求,Web集群服务器成为当前应用最为广泛的一种解决方法。

人们提出了很多集群的结构,包括基于客户端的、基于DNS的、基于服务器端的和基于分配器的[2]。其中,基于分配器的集群是最好[2]和使用最广泛的。基于分配器的集群由分配器和若干后台服务器构成。分配器是集中的全局的调度器,它负责接收客户的请求,并把请求分发给最适合的后台服务器。我们的集群采用的就是基于分配器的结构。

根据分配器工作在OSI模型的哪一层,Web集群可分为Layer4集群和Layer7集群。在Layer7集群中,分配器可以先分析HTTP请求的内容,然后再决定选择哪一台服务器来服务该请求。与Layer4集群相比,Layer7集群有很多的优势,例如,可以提高缓存命中率,节省存储空间等等[3]。

为了实现基于内容的请求转发,可以采用如下的转发机制:TCP网关、TCP拼接或TCP传递[3]。在前两种机制中,进入和流出集群的数据包都要经过分配器,因此分配器很容易成为系统的瓶颈。所以,我们系统采用的是TCP传递的转发机制。

2.2 HTTP/1.1与TCP连接迁移

现在大部分的Web服务器使用的都是HTTP/1.1协议。如果使用HTTP/1.0,在一个会话中用户向服务器请求

*)本课题得到国家自然科学基金(90204008)、武汉市重大科技攻关项目(20001001004)的资助。熊 智 博士研究生,主要研究领域包括服务器集群系统、网络管理;晏蒲柳 教授、博导,主要研究领域包括信息网络、智能信息处理;郭成城 副教授、博士,主要研究领域包括信息网络、服务器集群系统。

一连串的网页时，客户每取一个网页都要和服务器建立一次TCP连接，其效率非常低。为了提高性能，HTTP/1.1支持使用一个TCP连接请求多个网页，即每当客户取回一个网页时，不用关闭TCP连接，可以用它继续请求其它的网页，这大大减少了建立和销毁TCP连接的开销。

HTTP/1.1的这种支持同一TCP连接请求多个网页的机制，虽然减少了与客户端的连接开销，但对于采用TCP传递机制[3]的集群服务器来说，却有可能产生服务器端的TCP连接迁移。如果客户在一个TCP连接中访问的多个网页不在同一台服务器上，系统就需要将与客户建立的TCP连接从一台服务器迁移到另一台服务器上。每进行一次TCP连接的迁移，都要进行前后台服务器的通讯，TCP连接的克隆，TCP连接的销毁等工作，开销很大。TCP连接迁移对集群系统性能的影响是十分明显的，随着迁移数量的增多，这种影响也相应加大。

如果通过分析Web日志，发现在客户的会话中总是同时请求某些网页，则可以将它们放置在同一台服务器上，这样既保持了会话的连续性，同时又减轻了服务器的开销。

2.3 Web文档的组织与分布

Web文档的类型各种各样，如何组织和分布这些文档是比较棘手的问题。最简单的办法是把整个网站的所有内容在各服务器间进行镜像拷贝。但是对于内容海量级的大型网站来说，不说单台服务器有可能容纳不下所有的内容，光是文档在所有服务器间的更新就会给文档管理带来极大的开销。

可能的一种解决办法是把所有的内容放到一个集中式的网络文件系统(NFS)上。但该方案非常脆弱，共享文件系统的故障将会导致整个集群系统的不可访问；且通过共享文件系统访问数据会因为远程的I/O操作造成客户的访问延时和局域网的拥塞。

上述的两种方案对于大型的异构的集群服务器系统来说都不很理想，它们都忽略了服务器和内容的异构性。前面提到Web文档类型的复杂性，不同类型的文档对系统资源的要求是不一样的。文[4]提出了文档按照类型进行分布的方案。然而，通常在一个请求中，包含了多种类型的文档对象，该方案使客户和服务器间的TCP连接需要在后台服务间进行多次迁移，增加了系统的开销。

到目前为止，虽然有一些关于Web文档组织和分布的研究[5,6]，但它们大多研究的是在广域网中如何组织和分布文档，并且大部分都是以文档为单位进行分布，而且主要目标都是使文档的通信开销最小或增强可靠性。关于Web集群中的文档组织和分布的研究还很少，将文档组织分布与集群系统性能联系起来的研究更少。

3 网页组簇

这里，先要区别“文档”和“网页”的概念。文档指的是Web服务器上的文件。网页则是用户指定访问的Web页面，它与URL相对应。一般用户有两种方法向服务器请求网页，一是直接在浏览器地址栏中输入URL地址，二是用户点击网页中的超级链接。一个网页可能由多个文档组成，例如一个网页中可能嵌有图片，声音，动画等等。

需要说明的是，我们组簇的对象是网页而不是文档，因为我们是要根据用户的访问模式进行组簇，而用户通常显式访问的只是网页，嵌在网页中的对象是浏览器根据HTML的标记自动下载的。另外，文档的数量远远大于网页的数量，对大型网站的文档进行组簇，工作量巨大，不太实际。

通常用户都有一定的浏览模式，且在一段时间范围内(一般是几个星期内)是不变的[6]。在用户的浏览过程中，存在着一些关联规则。所谓关联规则就是用户会话中存在着经常被用户一起访的Web页面的集合，这些页面之间并没有顺序关系。网页聚类就是通过挖掘Web日志，分析网页被用户访问的情况，寻找在同一会话中经常被一起访问的网页，将其归为一簇。

这种以网页簇为单位的内容组织方法主要有以下优点：1)把一些相关性大的网页放在一起，TCP连接迁移的次数将大大减少；2)抽象并简化了文档分布的单位；3)缓存时可以以簇为单位，节省了服务器资源。

通过Web日志挖掘，将网页聚类的主要步骤包括：数据预处理，计算相关矩阵和聚类。

3.1 数据预处理

数据预处理主要完成将原始日志经过过滤、筛选以及重组后，将其转变为适合挖掘的数据格式。数据预处理的主要过程如下：

3.1.1 数据净化

数据净化是指删除Web日志中与网页聚类无关的数据。因为我们组簇的对象是网页，因此我们需要从日志中删除浏览器自动下载对应的请求，只留下用户显式访问的请求，通常只有HTML文件。

另外，在聚类时其实并不用考虑所有的网页。很多研究都表明，在一段时期内(与每个网站的面向的具体应用相关)热门网页基本上保持很高的稳定性，它们包含了大多数的客户请求。一般来说，一天内访问最为频繁的约10%的网页能够包含接下来一周内约80%的客户请求[6]。我们的研究对象就是这些约10%的热门网页，这样能极大提高网页分析和聚类的效率。因此，我们在统计出每个网页的访问频率后，删除那些访问频率较低的非热门网页。

3.1.2 用户识别

我们根据IP地址来识别用户。如果用户的IP地址相同，但是相应的代理日志表明用户的浏览器类型或操作系统改变，则认为不同的代理表示不同的用户。如果没有代理日志，则仅以IP地址作为识别用户的根据。

3.1.3 会话识别

在跨越时间区段较大的日志中，用户有可能多次访问了该站点。会话识别的目的就是将用户的访问记录分为单个的会话。通常采用超时的方法识别用户会话，即如果两相邻网页的请求时间间隔超过一定的界限就认为用户开始了一个新的会话。文[7]提出了一种确定超时阈值的方法，超时阈值是这样一个值，当适当改变它的大小时，不会引起会话数量的剧烈波动。

3.1.4 路径补充

由于本地缓存和代理服务器缓存的存在，使得服务器的日志会遗漏一些重要的页面请求，路径补充的任务就是将这些遗漏的请求补充到用户会话中。如果当前请求的页面与用户上一次请求的页面之间没有超文本链接，那么用户很可能使用了浏览器上的“后退”按钮调用缓存在本机中的页面。我们可以通过检查引用日志来确定当前请求来自哪一个页面，如果在用户的历史访问记录上有多个页面都包含与当前请求页的链接，则将请求时间最接近当前请求页的页面作为当前请求的来源。

3.2 相关矩阵

如何表示网页间的相关距离是网页聚类的关键步骤，以后的聚类算法都是围绕网页间的相关距离进行的。我们从时间的角度来确定网页间的相关距离，它的模型最接近人们实际的访问模式，因而有更高的准确度。我们利用两个网页在一个会话内同时出现的概率来计算它们之间的相关距离。这反映了它们在时间局域性上的相似度。

假设通过分析日志，将请求划分成了 p 个会话。网页 A，B 在第 i 个会话中同时出现的次数表示为 $f_i(A,B)=f_i(B,A)$，它等于在一个会话中(A,B)或(B,A)出现的次数且不必彼此相邻。此外，用 $o(A)$和 $o(B)$分别表示网页 A 和网页 B 在日志中出现的次数。网页 A 和 B 之间的相关距离定义如下：

$$dist(A,B)=1-\frac{\sum_{i=1}^{p} f_i(A,B)}{o(A)+o(B)}$$

显然 $dist(A,B)=dist(B,A)$，$dist(A,A)=0$，所以得到的相关距离矩阵是一个上三角的矩阵。这个矩阵将作为下一步网页聚类算法的输入。

3.3 网页聚类

聚类是一个将数据集划分为若干组或类的过程，并使得同一个组内的数据对象具有较高的相似度；而不同组中的数据对象是不相似的。相似或不相似的描述是基于对象属性的取值来确定的，通常就是利用距离来进行表示的。我们需要的就是根据网页间的相关距离用聚类算法将网页聚类成簇。

研究人员提出了很多聚类算法，需要根据应用所涉及的数据类型、聚类的目的以及具体应用要求来选择合适的聚类算法。RDBC[8]是一种基于密度的聚类方法，它是当前面向大数据集聚类算法中较快的一种。我们采用的就是 RDBC 聚类算法。

不包含在任何聚类中的对象称为噪声数据。我们将所有孤立点归为一个新的聚类，称为“孤立簇”。最后，还要将在数据净化时删除的那些非热门网页组成一个“非热门簇”。

4 网页簇的分布

通过 Web 日志挖掘和网页聚类得到的网页簇是我们文档分配的基本单位。本节主要介绍如何在集群服务器间对这些网页簇进行分布，以减小集群系统的平均响应时间。

对网页簇进行分布的前提条件是：客户的访问模式在未来的一段时间内(通常是几周)保持与分配前基本相同。

设有 N 个后台服务器：$S_1,S_2,\cdots,S_N$；M 个网页簇：$C_1,C_2,\cdots,C_M$。

4.1 Web 服务器的服务模型

由于 Apache 使用最为广泛，所以本文仅考虑 Apache 服务器。Apache 是多进程结构的，主要特点如下：1)处理 HTTP 请求的多个进程以分得时间片的形式轮流执行；2)一个进程服务一个 HTTP 请求，直到结束，该进程才能服务下一个 HTTP 请求；3)一个进程的阻塞挂起不会导致其他进程的继续执行；4)并发进程数受操作系统的限制，即并发处理的 HTTP 请求数有限。

因此至少可以得到两个结论：1) Apache 服务器同时服务的请求数是有限的；2)请求的服务模式不是先来先服务(FCSF)而更像是共享处理器(Processor Sharing，PS)。

新推出的 Apache 2.0 可以通过不同的 MPM(Multi-Processing Modules)运行在一种多进程与多线程相混合的模式下。但其仍然符合上面的两个结论。

研究表明，Web 访问分布是长范围相关的、自相似的和分形的，但由于我们将时间划分为一小段一小段的，在这一小段时间内可以认请求的到达服从泊松分布[9]。另外，有研究表明，Web 请求的服务时间并不服从负指数分布[10]。

综合以上的结论，我们使用 M/G/1/K PS 排队模型对 Apache 服务器进行建模。

在该模型中，请求(任务)的到达过程是泊松过程，设请求平均到达速率为 λ；请求的服务时间服从一般分布，设平均服务时间为 x；如果系统中的请求数量已达到预设定的值 K，那么新到来的请求将被丢弃；系统用轮转的方式服务各个请求，当某请求用完所分得的时间片后将被挂起，直到其它的请求也都用完所分到的相等的时间片。

用 n 表示系统中的任务个数，那么 n 的分布如下[11]：

$$P\{n=i\}=\frac{(1-\lambda x)(\lambda x)^i}{1-(\lambda x)^{K+1}}$$

系统的损失率为：

$$P_b=P\{n=K\}=\frac{(1-\lambda x)(\lambda x)^K}{1-(\lambda x)^{K+1}}$$

系统的请求进入率为：

$$\bar{\lambda}=\lambda(1-P_b)$$

根据 Little 公式，请求的平均逗留时间为：

$$T_t=E\{n\}/\bar{\lambda}$$

因为在 PS(处理器共享)模式中，请求的等待时间为零，所以请求的平均响应时间 T 就是系统的平均逗留时间 T_t。又显然，请求的平均响应时间是 λ 和 x 的函数。通过计算可得到请求的平均响应时间为：

$$T(\lambda,x)=\frac{(\lambda x)^{K+1}(K\lambda x-K-1)+\lambda x}{\lambda(1-(\lambda x)^K)(1-\lambda x)} \quad (1)$$

K 是服务器系统一个重要参数，可以预设 λ 和 x，测试得到 T，然后根据(1)计算得到 K 的值。

4.2 网页负载的评价

假设访问每个网页的请求流都是独立的泊松流。在这里先给出一个定理，其证明参见文[3]。

定理 1 n 个强度分别为 $\lambda_1,\lambda_2,\cdots,\lambda_n$ 的独立泊松流的合成流仍为泊松流，且强度为 $\lambda=\lambda_1+\lambda_2+\cdots+\lambda_n$。

设 W 为一些网页的集合(可为单个网页)，为了叙述方便，我们将“访问 W 中网页的请求的平均到达速率”简称为“W 的平均到达速率”，将“服务器服务访问 W 中网页的请求的平均服务时间”简称为“服务器服务 W 的平均服务时间”；将“访问 W 中网页的请求的平均响应时间”简称为“W 的平均响应时间”。

设 P 为一个网页。我们用 $R(P)$表示网页 P 的平均到达速率，用 $X_i(P)$表示服务器 S_i 服务网页 P 的平均服务时间。那么，系统服务网页 P 的平均服务时间 $X(P)$计算如下：

$$X(P)=\sum_{i=1}^{N} X_i(P)/N$$

一个网页所带来的负载主要由两个因素决定：一个是网页本身的负载(大小)，另一个是访问该网页的请求的平均到达速率。因此，网页的负载定义如下：

定义 3 网页 P 的负载为网页 P 的平均到达速率与系统服务网页 P 的平均服务时间的乘积，即，$H(P)=R(P)\cdot X(P)$。

4.3 网页簇负载的评价

设第 k 个网页簇 C_k 中有 g_k 个网页，该网页簇用集合 $C_k=\{p_k^1,p_k^2,\cdots,p_k^{g_k}\}$ 表示。

根据定理 1 可知，网页簇 C_k 的平均到达速率为：

$$R(C_k)=\sum_{i=1}^{g_k}R(p_k^i)$$

另外，服务器 S_i 服务网页簇 C_k 的平均服务时间为：

$$X_i(C_k)=\sum_{i=1}^{g_k}(R(p_k^i)\cdot X(p_k^i))/\sum_{i=1}^{g_k}R(p_k^i)$$

那么，系统服务网页簇 C_k 的平均服务时间为：

$$X(C_k)=\sum_{i=1}^{N}X_i(C_k)/N \tag{10}$$

类似地，我们可定义网页簇的负载。

定义 4 网页簇 C_k 的负载为网页簇 C_k 的平均到达速率与系统服务网页簇 C_k 的平均服务时间的乘积，即，$H(C_k)=R(C_k)\cdot X(C_k)$。

4.4 网页簇的拷贝数量

要想减小系统的平均响应时间，我们必须使各个服务器分得的负载较为均衡，避免出现有的过载而有的过闲。网页簇的负载相差太大不利于负载均衡的实现，自然也不利于系统整体性能的提升。因为那些热点网页所属的网页簇由于具有很高的访问频率，通常也是负载很高的网页簇，如果把这样的网页簇只是分配到某台服务器，该服务器很容易因为客户的持续访问而过载，从而造成负载失衡，影响集群系统的整体性能。

网页簇的拷贝就是为了平衡网页簇负载之间的差别，分散客户对热点网页的访问，从而达到负载均衡的目的。显然网页簇的负载越大，它拥有的拷贝份数就应该越多。设

$$H_{MAX}=\max\{H(C_i)\mid 1\leqslant i\leqslant M\}$$

网页簇 C_k 的拷贝份数计算如下：

$$Q_k=\left\lceil\frac{H(C_k)}{H_{MAX}}*N*\xi\right\rceil$$

($\lceil x\rceil$ 表示不小于 x 的最小整数) 其中 $0<\xi\leqslant 1$，我们称其为拷贝份数系数，它表示负载最大的那个网页簇的拷贝份数与服务器数量的比。显然，ξ 的值越大，拷贝份数就越多，管理开销就越大，但有利于负载均衡；ξ 的值越小，拷贝份数就越少，管理开销就越小，但不利于负载均衡。一般 ξ 可取 0.5～0.1，这里我们取 $\xi=0.8$。

我们在各拷贝之间采用的是随机算法，所以，对于网页簇 C_k，其每份拷贝的平均到达速率为：

$$R_{per-copy}(C_k)=R(C_k)/Q_k$$

4.5 网页簇拷贝的分布

不失一般性，设 $H(C_1)\geqslant H(C_2)\geqslant\cdots\geqslant H(C_M)$。

我们的文档组织分布策略的目标是减小集群系统的平均响应时间，因此分配网页簇拷贝的方法如下：先将网页簇 C_1 的各份拷贝分布到合适的服务器上，使得系统所有请求的平均响应时间的增加最少；然后将网页簇 C_2 的各份拷贝分布到合适的服务器上，也使得系统所有请求的平均响应时间的增加最少；……；依此类推，直到最后将网页簇 C_M 的各份拷贝分布到合适的服务器上，使得系统所有请求的平均响应时间的增加最少。

具体的分布算法如下：($1\leqslant k\leqslant M$)

(a)各服务器 S_i 上已放置网页的平均到达速率 $\lambda_i=0$，平均服务时间 $x_i=0$，平均响应时间 $T_i=0$。

(b)$k=1$。

(c)前 $k-1$ 个网页簇的拷贝已经分布到各台服务器上，下面我们将分布网页簇 C_k 的 Q_k 份拷贝。

(d)设当前服务器 S_i 上已放置网页的平均到达速率为 λ_i，服务器 S_i 服务它上面已放置网页的平均服务时间为 x_i。那么，当前服务器 S_i 上已放置网页的平均响应时间为：

$$T_i=T(\lambda_i,x_i)$$

(e)如果将网页簇 C_k 的一份拷贝放在服务器 S_i 上，那么，服务器 S_i 上已放置网页的平均到达速率变为：

$$\lambda'_i=\lambda_i+R_{per-copy}(C_k)$$

服务器 S_i 服务它上面已放置网页的平均服务时间变为：

$$x'_i=\frac{\lambda_i\cdot x_i+R_{per-copy}(C_k)\cdot X(C_k)}{\lambda_i+R_{per-copy}(C_k)}$$

服务器 S_i 上已放置网页的平均响应时间变为：

$$T'_i=T(\lambda'_i,x'_i)$$

(f)若将网页簇 C_k 的一份拷贝放在服务器 S_i 上，那么，所有请求的平均响应时间的增量为：

$$\Delta T_i=(T'_i-T_i)\lambda_i+(T'_i-0)R_{per-copy}(C_k)$$

(g)根据(d)、(e)和(f)，我们可分别计算出 N 个 ΔT_i ($1\leqslant i\leqslant N$)，将它们从小到大排序，设排在前 Q_k 个 ΔT_i 的下标依次为 $i_1,i_2,\cdots,i_{Q_k}$，那么我们就将网页簇 C_k 的 Q_k 份拷贝分别放在服务器 $\{S_i\mid i\{i_1,i_2,\cdots,i_{Q_k}\}\}$ 上。

(h)如果 $k\neq M$，k 加 1，执行(c)；否则分布完毕。

网页组簇和网页簇拷贝分布的结果，被放置到前端分配器和后台的服务器上。前端分配器根据它们选择初始服务器；后台服务器则根据它们选择 TCP 连接迁移的服务器。如果被访问的网页有多个拷贝放在不同的服务器上，我们使用随机算法进行选择。

5 测试

本节将用实际的日志驱动对我们的文档组织和分布策略进行测试。并将采用我们的策略对文档进行组织分布的系统与采用以下文档组织分布方案的系统进行比较：

1)全镜像轮转：每台服务器上都放有所有文档的拷贝，分配器使用轮转算法分发请求；

2)组簇随机分布：将网页组簇后，不考虑簇的负载，每簇只有一份拷贝，并且拷贝被随机的放在后台服务器上；

3)文档基于类型分布：按文档的类型进行分类，同一类型的文档放在同一台服务器上。我们将文档分为两类：图片文档和其它文档，每类放在一半数量的服务器上(若服务器为奇数台，则放图片的少一台)。

被测试的 Web 集群服务器由分配器和 6 个同构的后台服务器构成。分配器(P3-500M，256M)运行优化了的 Linux 2.4.20-8 的内核；后台服务器(P4-1.7G，256M)运行 Linux 2.4.20 的内核和 Apache 2.0.40。网络环境为 100M。

测试时先从日志中分析出每个会话，然后将这些会话分给多台测试机。当一个会话开始时，测试机创建一个线程，该线程与服务器建立一个连接，并用此连接依次按时请求会话中的所有网页，当会话结束时，连接被关闭，线程被销毁。

5.1 日志来源和组簇结果

我们使用的是美国环保署 Web 服务器一天(1995-8-30)的日志[12]。日志是 ASCII 的格式，每一行对应一个请求，每个请求有如下属性：客户、请求时间、请求内容、返回代码和传输字节数。

在重建网站内容时，无论动静态文档，我们只是根据日志中记录的请求 URL 和传输字节数重建它们。

该日志共记录了47,748个请求，经过预处理后得到20,713个请求，3,054个不同的网页。选取时间域值T为30分钟，得到7,184个用户会话。选取合适的聚类参数，我们将所有网页组成了19个网页簇(包括"孤立簇"和"非热点簇")。

为缩短测试时间，我们将日志中的时间压缩了10倍，即请求间隔缩短了10倍。

5.2 负载均衡

我们使用负载均衡因子(LBM)[13]来衡量系统负载均衡的程度。周期性的收集各服务器的负载信息，并计算负载峰值和负载均值之间的比值，LBM就是负载峰值和负载均值之间比值的加权平均。这里，我们使用CPU利用率来衡量负载，每分钟采样一次。显然，LBM的取值范围是从1到N(服务器数量)，并且LBM越小表示系统负载越均衡。

测试时，改变后台服务器数量，各系统的负载均衡因子如图1。从图中可以看到，我们系统的负载均衡程度与全镜像轮转系统差不多，比它略好，并且比另外两个系统要好。这是因为我们在计算网页簇拷贝数量，以及分布网页簇拷贝时均考虑了网页簇负载的大小。注意，轮转算法是基于连接的算法，当使用HTTP/1.1协议时，连接中请求的个数可能差别很大，故不能取得很好的负载均衡效果。特别地，当存在动态请求时，轮转算法的负载均衡效果会更差。

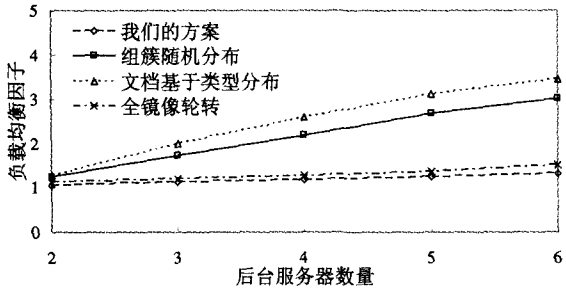

图1 负载均衡因子

5.3 平均响应时间

为了在不同负载强度的情况下测试比较系统的平均响应时间，我们定义，用原始日志驱动时的负载强度为1。可用如下的方法得到负载强度为2的日志：在原始日志的每个请求(每一行)后面复制一个请求(一行)，复制的请求除了客户与原请求不同外，其它的内容与原请求完全一样。用类似的方法，我们可以得到负载强度更大的日志。使用6台后台服务器，在不同的负载强度下，各系统的平均响应时间如图2。

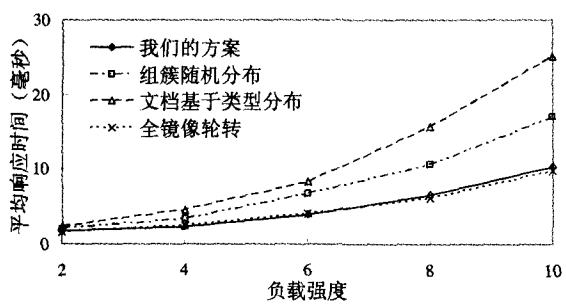

图2 平均响应时间

这里，影响系统平均响应时间的主要因素有两个：负载均衡和TCP连接迁移次数。很显然，TCP连接迁移次数从多到少的排列为：文档基于类型分布、组簇随机分布、我们的方案和全镜像轮转。负载均衡的测试结果上面已给出。下面就从这两个方面分析图2中的结果。

图2表明，1)随着负载强度的增加，各系统的平均响应时间都增加；2)文档基于类型分布的系统的平均响应时间最长，这是由于它的TCP连接迁移次数过多和负载不均衡造成的；3)文档组簇随机分布的系统的平均响应时间也较长，虽然组簇后TCP连接迁移次数不多，但是由于每簇只有一份拷贝，热点簇所在的服务器容易重载甚至过载，从而影响了系统的总体性能；4)我们系统的平均响应时间与全镜像轮转系统差不多(我们的略小)，但明显比其它两个系统小，这主要是由于它们的TCP连接迁移次数很少，并且负载相对比较均衡。更主要的是，我们在分布网页簇的拷贝时使得系统的平均响应时间最短。

总结 本文提出了一种Web集群服务器中文档组织和分布的优化策略，其目的是通过将Web文档进行优化组织和优化分布来提高集群系统的性能。在我们的策略中，1)通过对Web服务器日志的分析，挖掘客户的访问模式，将关联度高的网页聚类成网页簇，并以网页簇为文档分布的单位，以减少TCP连接迁移次数；2)根据网页簇的负载，将网页簇进行多份拷贝，以均衡负载；3)优化的将网页簇的拷贝分布到各服务器上，以减小集群系统的平均响应时间。

测试表明，我们的策略能使集群系统有较少的TCP连接迁移次数，较好的负载均衡性能，和较小的平均响应时间。且相对于内容全镜像的文档分布方案，能节约存储空间，减少维护各服务器文档一致性的开销。

参考文献

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

2 Cardellini V, Colajanni M, Yu P S. Dynamic load balancing on Web-server systems [J]. IEEE Internet Computing, 1999, 3(3): 28～39

3 熊智. 高可用Web服务器集群系统负载均衡的研究与实现[D]: [硕士学位论文]. 武汉：武汉大学, 2003

4 Yang C S, Luo M Y. A content placement and management system for distributed Web-server systems [C]. In: Proceedings of the 20th International Conference on Distributed Computing Systems, Taipei, Taiwan, China, 2000. 691～698

5 Narendran B, Rangarajan S, Yajnik S. Data distribution algorithms for load balanced fault-tolerant Web access [C]. In: Proceedings of the 16th Symposium on Reliable Distributed Systems, 1997. 97～106

6 Chen Y, Qiu L, Chen W, et al. Clustering Web content for efficient replication [C]. In: Proceedings of the 10th International Conference on Network Protocols, Paris, France, 2002. 165～174

7 Adya A, Bahl P, Qiu L. Analyzing the browse patterns of mobile clients [C]. In: Proceedings of the ACMSIGCOMM Internet Measurement Workshop, San Francisco, USA, 2001. 189～194

8 Su Z, Yang Q, Zhang H H, et al. Correlation-based document clustering using Web logs [C]. In: Proceedings of the 34th Annual Hawaii International Conference on System Science, Hawaii, USA, 2001, 5: 5022

9 Zhu H, Tang H, Yang T. Demand-driven service differentiation for cluster-based network servers [C]. In: Proccedings of IEEE INFOCOM, Alaska, USA, 2001. 679～688

10 Balter M H. Task assignment with unknown duration [J]. Journal of ACM, 2002, 29(2): 260～288

11 Cao J, Andersson M, Nyberg C, et al. Web server performance modeling using an M/G/1/K * PS queue [C]. In: Proccedings of the 10th International Conference on Telecommunication, 2003, 2: 1501～1506

12 Traces available in the Internet traffic archive: DB/OL]. http://ita.ee.lbl.gov/html/traces.html

13 Bunt R B, Eager D L, Oster G M. Achieving load balance and effective caching in clustered Web servers [C]. In: Proccedings of the 4th International Web Caching Workshop, San Diego, USA, 1999. 159～169