维普资讯 http://www.cqvip.com

异构 Web 集群中的比例伸展因子区分服务*)

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。如何保证 Web 服务的服务质量(QoS)是一个迫切需要解决的问题。区分服务已成为 QoS 研究领域中的一个焦点。本文分析了 Web 请求服务时间的数字特性,并使用 M/G/1 FCFS 排队模型对 Web 服务器及 Web 集群服务器进行了建模。在对模型进行分析的基础上,设计并实现了一种异构 Web 集群服务器中的比例伸展因子区分服务的方案,并提出了基于概率空间的请求调度算法。请求被分成多个类,无论系统的负载如何,系统确保每类请求的平均伸展因子与事先给定的参数成正比。实际测试表明,所设计的方案满足相对区分服务的可预测性和可控制性的要求。

关键词 Web 集群服务器,异构,伸展因子,相对区分服务,比例区分服务

Proportional Stretch Factor Differentiated Service in Heterogeneous 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 server. Moreover, the need for supporting Quality of Service (QoS) in Web service has become more strident recently. Differentiated service has been an active research topic in the QoS area. This paper analyzes the statistical characteristic of the request service time, and models the Web server and Web server cluster with M/G/1 FCFS queuing model. Based on the analysis of model, a solution to provide proportional stretch factor differentiated service in heterogeneous Web server cluster is designed and implemented, and a probability-space-based dispatching algorithm is proposed to dispatch requests. Requests are classified into several classes, and the average stretch factor ratios between requests classes are maintained according to their pre-specified differentiation parameters independent of the workloads. Realistic experiments show that the proposed solution satisfies the two important features of a relative differentiated service model, controllability and predictability.

Keywords Web server cluster, Heterogeneous, Stretch factor, Relative differentiated service, Proportional differentiated service

1 引言

随着 Internet 的蓬勃发展,Web 服务正在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。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案。研究人员提出了很多 Web 集群的结构,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。

Web 服务已由原来单纯的信息发布平台发展成了如今的电子商务、网上贸易等多功能的平台。如何保证 Web 服务的 QoS 已成为一个迫切需要解决的问题。在过去的 QoS 研究中,研究人员往往把重点放在网络 QoS 的研究上。然而,仅仅依靠网络的 QoS 并不能保证端到端(例如 Web)服务的 QoS。如果在 Web 服务器端没有 QoS 机制,即使高优先级的请求通过繁忙的网络到达了服务器,该请求也有可能被服务器丢掉。所以,我们必须在 Web 服务器上实现 QoS 机制,使其能区别对待不同优先级的请求。虽然目前 Web 服务器 QoS 的研究日益受到关注,但关于 Web 集群 QoS 的研究还很少。

区分服务,自从 1998 年被提出已来,已成为 QoS 研究领域中的一个焦点。区分服务可分为绝对区分服务和相对区分服务两种。绝对区分服务要求给每个服务类分配绝对的、固定的系统资源(例如带宽);而相对区分服务则是要保证高级别类得到的服务质量好于低级别类得到的服务质量,它没有资源预约,不保证绝对的端到端的性能,但保证各级别服务类服务质量之间的顺序不随系统状态的变化而变化。相对区分服务的两个重要特性是可控制性(能调整各级别类服务质量之间的差别)和可预测性(无论系统的负载如何,确保高级别类得到的服务不比低级别类得到的服务差)。不仅在网络传输中需要区分服务,在 Web 服务中也需要区分服务。例如,付费用户比不付费用户应该得到更好的服务;高优先级请求比低优先级请求应该得到更好的服务(例如,在商业网站中,购物对应请求的级别应该比浏览对应请求的级别高);内部客户比外部客户应该得到更好的服务等等。而目前普通的 Web 集群还不能提供区分服务。

在区分服务的基础上,研究人员提出了比例区分服务模型[3]。比例区分服务是指,无论系统的负载如何,各服务类的服务质量参数应当和事先设定的参数成比例。该模型已成为了一个公认的相对区分服务的模型,并且比例延时区分(Proportional Delay Differentiation)[4]模型已被用在网络数据包

*)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。

维普资讯 http://www.cqvip.com

的调度上。目前各种比例区分服务的研究已相继展开[5,6]。

虽然请求的响应时间和排队时间都是服务器端重要的性能参数，但为了反映“客户对大请求可以容忍较长的等待时间，而对小请求可以容忍的等待时间较短”这样一个事实，我们定义了伸展因子。伸展因子定义为：请求等待时间与请求服务时间的比值。请求的伸展因子与响应时间和排队时间一样，其越高意味着系统负载越重。伸展因子（或者它的变形），在近来支持 QoS 的系统的研究和设计中[7,8]，已被作为一个基本的并且重要的性能参数。

本文分析了 Web 请求服务时间的数字特性，并使用 M/G/1 FCFS 排队模型对 Web 服务器及 Web 集群进行了建模。在对模型进行分析的基础上，设计并实现了一种异构 Web 集群服务器中的比例伸展因子区分服务的方案，并提出了基于概率空间的请求调度算法。请求被分成多个类，无论系统的负载如何，系统确保每类请求的平均伸展因子与事先给定的参数成正比。

2 方案的设计与实现

设 Web 集群由分配器和 N 个后台服务器 $RS_1, RS_2, \cdots, RS_N$ 构成。

传统 Web 集群的分配器几乎没有任何 QoS 机制。为了实现比例伸展因子区分服务，我们在分配器上实现了如下 QoS 机制：区分服务、性能隔离、服务器资源动态划分和接纳控制。这些机制的大致关系如图 1 所示。

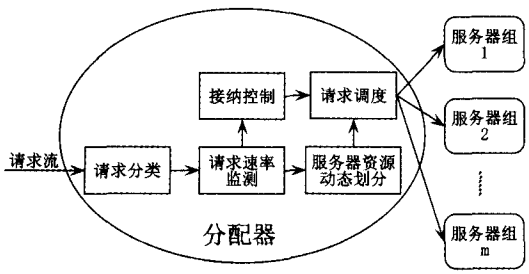

图 1 分配器中各 QoS 机制的关系

2.1 请求分类与性能隔离

我们的系统支持多个服务类别。为了实现请求的分类，可以编写一些模式匹配规则，然后根据这些规则来分类请求。匹配规则可以匹配如下内容：URI（不带参数的）、一些参数名和参数值、Cookies 和源地址等等。

设请求被分为 $M(>1)$ 类，$C_1, C_2, \cdots, C_M$。我们将 N 台服务器资源相应地分成 M 个组 $G_1, G_2, \cdots, G_M$，其中组 G_i ($1 \leqslant i \leqslant M$) 服务 C_i 类的请求，设组 G_i 中的服务器资源为 n_i。需要注意的是，n_i 不一定是整数，可能存在某些服务器属于多个组，它们需要为多个类别的请求提供服务。静态分组显然不能适应负载的变化，因此分配器周期性地，根据实际情况动态调整各组的服务器资源。

2.2 Web 服务器及 Web 集群的服务模型

我们假设到达服务器的请求流是泊松（Poisson）流。实际上，Web 访问分布是长范围相关的（Long-range dependent）、自相似的（Self-similar）和分形的（Fractal）[9]。但由于我们将系统时间划分为一小段一小段的时间，在这一小段时间内可以认为请求的到达服从泊松分布[10]。

研究表明，Web 请求的大小服从重尾（Heavy-tailed）分布。相应地，请求服务时间也服从重尾分布[7,11]。但在实际中，服务时间是有上下界的，因此请求服务时间服从有界 Pareto（Bounded Pareto）分布[7]。其特性由形状参数 α、上界 p 和下界 k 决定，其概率密度函数如下：

$$f(x)=\frac{\alpha k^{\alpha}}{1-(k/p)^{\alpha}}x^{-\alpha-1} \tag{1}$$

其中 $0<\alpha<2, 0<k\leqslant x\leqslant p$。设

$$\Psi(\alpha,k,p)=\alpha k^{\alpha}/1-(k/p)^{\alpha} \tag{2}$$

那么请求服务时间分布的概率密度为：

$$f(x)=\Psi(\alpha,k,p)x^{-\alpha-1} \tag{3}$$

根据概率密度可计算得到

$$E\{X\}=\int_k^p xf(x)\mathrm{d}x=\begin{cases}\Psi(\alpha,k,p)/\Psi(\alpha-1,k,p) & \alpha\neq 1\\ (\ln p-\ln k)\Psi(\alpha,k,p) & \alpha=1\end{cases} \tag{4}$$

$$E\{X^2\}=\Psi(\alpha,k,p)/\Psi(\alpha-2,k,p) \tag{5}$$

$$E\{X^3\}=\Psi(\alpha,k,p)/\Psi(\alpha-3,k,p) \tag{6}$$

$$E\{X^{-1}\}=\Psi(\alpha,k,p)/\Psi(\alpha+1,k,p) \tag{7}$$

从上面的讨论可知，对于一台 Web 服务器，任务的到达是泊松过程，并且存在服务时间的 1，2 和 3 阶矩且有限。因此，可假设 Web 服务器的服务模型是 M/G/1 FCFS 排队模型。

再考虑 Web 集群的情况。设一个 Web 集群由分配器和 m 台同构的后台服务器 D_i ($1\leqslant i\leqslant m$) 组成，分配器使用随机算法分发请求，请求到达分配器是强度为 λ 的请求流。我们不考虑分配的分发消耗，以及集群内的网络延时。由于分配器使用随机算法分发请求，所以每台服务器分到请求的概率相同，均为 $1/m$。

对任意 $t\geqslant 0$，用 $D(t)$ 表示分配器在 t 时刻得到的请求数，$D_i(t)$ 表示服务器 D_i 在 t 时刻得到的请求数。由全概率定理可知：

$$P(D_i(t)=n_i)=\sum_{n=0}^{\infty}(P(D_i(t)=n_i\,|\,D(t)=n)*P(D(t)=n)) \tag{8}$$

显然，条件概率 $P(D_i(t)=n_i\,|\,D(t)=n)$ 只可能发生在 $n\geqslant n_i$ 的条件下。因为每台服务器分配到请求的概率为 $1/m$，并且每个请求被分配到哪台服务器处理与其它请求的分配情况无关，因此当 $D(t)=n$ 时，随机变量 $D_i(t)=n_i$ 服从参数为 $(n,1/m)$ 的二项分布，故

$$P(D_i(t)=n_i\,|\,D(t)=n)=\binom{n}{n_i}(\frac{1}{m})^{n_i}(1-\frac{1}{m})^{n-n_i} \tag{9}$$

因为假设到达分配器的任务是强度为 λ 的泊松流，所以

$$P(D(t)=n)=\frac{(\lambda t)^n}{n!}\mathrm{e}^{-\lambda t} \tag{10}$$

根据式(9)和(10)，可得到

$$P(D_i(t)=n_i)=\sum_{n=n_i}^{\infty}\left(\binom{n}{n_i}(\frac{1}{m})^{n_i}(1-\frac{1}{m})^{n-n_i}\frac{(\lambda t)^n}{n!}\mathrm{e}^{-\lambda t}\right)=\frac{(\frac{\lambda}{m}t)^{n_i}}{n_i!}\mathrm{e}^{-\lambda t/m} \tag{11}$$

可见，$D_i(t)$ 为泊松过程，其强度为 λ/m，即到达每台服务器的请求流都是强度为 λ/m 的泊松流。因此，集群中每台服务器的服务模型都是 M/G/1 FCFS 排队模型。

2.3 平均伸展因子的计算

若一台服务器的服务模型是 M/G/1 FCFS 队列，设 λ 是请求到达的平均速率，X 是请求的服务时间，W 是请求的等待时间，S 是请求的伸展因子。根据排队论公式[12]，M/G/1

维普资讯 http://www.cqvip.com

FCFS排队模型的平均等待时间为：

$$E\{W\}=\frac{\lambda E\{X^2\}}{2(1-\lambda E\{X\})} \tag{12}$$

在排队模型中，任务的到达与服务时间相互独立，并且采用的是FCFS排队模型，所以任务的等待时间与任务的服务时间也相互独立，因此

$$E\{S\}=E\{\frac{W}{X}\}=E\{W\}E\{X^{-1}\}$$

$$=\frac{\lambda E\{X^2\}E\{X^{-1}\}}{2(1-\lambda E\{X\})} \tag{13}$$

下面考虑集群中的情况。设集群由分配器和 m 台同构的后台服务器组成，分配器使用随机算法分发请求。设请求的到达过程服从泊松分布，λ 是请求到达的平均速率，X 是请求的服务时间，W 是请求的排队延时，S 是请求的伸展因子。那么，根据前面的分析我们可以知道，每台服务器的服务模型都是M/G/1 FCFS排队模型，并且请求到达各服务器的平均速率均为 λ/m，根据式(13)，我们可以得到

$$E\{S\}=\frac{(\lambda/m)E\{X^2\}E\{X^{-1}\}}{2(1-(\lambda/m)E\{X\})}$$

$$=\frac{\lambda E\{X^2\}E\{X^{-1}\}}{2(m-\lambda E\{X\})} \tag{14}$$

2.4 比例伸展因子区分模型

设 $Q_i(1\leqslant i\leqslant M)$ 为 C_i 类请求的某个QoS参数，δ_i 为事先定义的比例值，比例区分模型的目标为

$$\frac{Q_j}{Q_k}=\frac{\delta_j}{\delta_k},1\leqslant j,k\leqslant M \tag{15}$$

例如，Q_i 代表请求的丢弃率，$\delta_1/\delta_2=1/2$，则代表 C_2 类请求的丢弃率是 C_1 类请求丢弃率的两倍。

比例伸展因子区分模型的目标就是，按照事先定义的比例值 $\delta_i(1\leqslant i\leqslant M)$ 控制各类请求的平均伸展因子 $E[S_i]$ 之间的比例，即

$$\frac{E\{S_j\}}{E\{S_k\}}=\frac{\delta_j}{\delta_k},1\leqslant j,k\leqslant M \tag{16}$$

区分模型的可预测性要求高级别请求得到较好的服务（至少不比低级别的差），也就是说高级别请求的平均伸展因子应该较低。不失一般性，假设 C_{i-1} 类的级别高于 C_i 类，那么，$0<\delta_1\leqslant\delta_2\leqslant\cdots\leqslant\delta_M$。

不同类别请求的平均到达速率可能不同，设 C_i 类请求的平均到达速率为 λ_i。设 C_i 类请求的服务时间为 X_i，伸展因子为 S_i。还暂时假设所有的后台服务器同构，且在同一个服务器组内分配器随机分发请求。那么，根据式(14)和(16)可得到方程组：

$$\begin{cases}E\{S_i\}=\dfrac{\lambda_i E\{X_1^2\}E\{X_i^{-1}\}}{2(n_i-\lambda_i E\{X_i\})}, & 1\leqslant i\leqslant M\\ \dfrac{E\{S_j\}}{E\{S_k\}}=\dfrac{\delta_j}{\delta_k}, & 1\leqslant j,k\leqslant M\\ \sum\limits_{i=1}^{M}n_i=N, & 0<n_i<N\end{cases} \tag{17}$$

如果各类请求的服务时间分布不完全相同，那么方程组(17)解的形式较为复杂。实际上，一般在使用Cookies或源地址对请求进行分类的系统中，各类请求的服务时间有着相同的分布，那么设

$$E\{X\}=E\{X_i\} \tag{18}$$

$$E\{X^{-1}\}=E\{X_i^{-1}\} \tag{19}$$

$$E\{X^2\}=E\{X_i^2\} \tag{20}$$

由式(17)、(18)、(19)和(20)可解得：

$$n_i=\lambda_i E\{X\}+\frac{\lambda_i/\delta_i(N-E\{X\}\sum\limits_{i=1}^{M}\lambda_i)}{\sum\limits_{i=1}^{M}\lambda_i/\delta_i} \tag{21}$$

式(21)中，前一部分是为了保证组中 C_i 的服务器资源有足够的能力服务 C_i 类的请求，而不至于过载；后一部分是集群系统剩余的服务能力，按 λ_i/δ_i 成比例地分配给各个服务器组。

分配器每隔一段时间就根据式(21)重新计算各 n_i 的值。式(21)中，参数 δ_i 是预设的；N 是后台服务器的台数；$E\{X\}$ 可根据式(4)计算得到，计算所需的参数 α、p 和 k 可根据实际文档的大小分布和测试事先得到，这些参数也可以通过分析历史日志得到；λ_i 则是由请求速率监测部件监测到的。

2.5 服务器资源的分配及调度算法

需要注意的是，虽然用式(21)计算得到 n_i 是在各后台服务器同构的前提下推导得到的，但是其结果依然可用于异构集群的情况。不同的是，如果是在同构的集群中，n_i 代表组 G_i 中服务器的实际台数；如果是在异构的集群中，n_i 代表的则是各组中服务器资源的相对比例，即组 G_j 中服务器资源与组 G_k 中服务器资源的比例应为 n_j/n_k。由于同构集群是异构集群的一个特例，因此下面我们只讨论在异构集群中服务器资源的具体分配及相应的调度算法。

设在Web集群中，各后台服务器按各自不同的性能（考虑CPU、内存、硬盘等）各赋予一个权值，设服务器 $RS_i(1\leqslant i\leqslant N)$ 的权值为 W_i。设

$$W=\sum_{i=1}^{N}W_i \tag{22}$$

那么，各服务器 RS_i 的归一化权值为 $w_i=W_i/W_N$。如果各服务器是同构的，则 $w_1=w_2=\cdots=w_N=1/N$。

由于 $n_i(1\leqslant i\leqslant M)$ 代表的是各组中服务器资源的相对比例，那么 G_i 组中所分得的服务器资源的归一化权值应该为：

$$n'_i=\frac{n_i}{\sum\limits_{i=1}^{M}n_i}*\sum_{i=1}^{N}w_i=\frac{n_i}{N} \tag{23}$$

我们采用基于概率空间的调度算法来分发请求。如图2，每台服务器的概率空间的大小就是其归一化的权值 w_i，每个请求类的概率空间的大小就是其应分得的服务器资源的归一化权值 n_i，整个概率空间为：

$$\sum_{i=1}^{N}w_i=\sum_{i=1}^{M}n'_i=1 \tag{24}$$

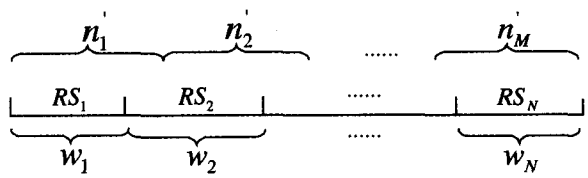

图2 各服务器的分配概率空间分布

当一个 $C_i(1\leqslant i\leqslant M)$ 类的请求到达时，分配器临时产生一个 $[0,n'_i)$ 范围内的随机数 r，然后计算

$$r'=\sum_{j=1}^{i-1}n'_j+r \tag{25}$$

分析 r' 在概率空间中的落点，看其落在哪个服务器的概率空间内，就将该请求分配给那个服务器。

这种基于概率空间的调度算法是合理的，这是因为：1）对于具有较大权值的服务器，因其概率空间也较大，自然具有较高的分配几率，即被分配更多的请求任务；2）对于应该占有较多服务器资源的请求类别，因其概率空间也较大，从而占有更

维普资讯 http://www.cqvip.com

多的归一化权值的服务器资源，即有更多的资源为其服务；3）每次分配都是一个独立的事件，既不受上次分配的影响，也不影响下一次的分配操作，从而可以达到更为均匀合理的分配效果。

需要说明的是，当服务器资源重新分配后，假设原来为 C_i 类请求提供服务的一些服务器资源不再服务于 C_i 类请求。这些服务器资源被重新分配后，它们除了为新类别的请求提供服务外，还继续服务原来未完成的类 C_i 的请求。当然，分配器可以采取一些更强硬的措施，例如丢掉原来未完成的类的请求或迁移走这些请求，但我们不这样做。因为前者会使系统有很高的请求丢弃率，后者的开销太大。

2.6 接纳控制

为了确保服务器组 $G_i(1\leqslant i\leqslant M)$ 中的服务器资源提供的服务能力能满足 C_i 类请求的需求，显然我们必须保证：

$$\lambda_i E\{X\}\leqslant n_i \tag{26}$$

式(26)可以作为接纳控制的依据。请求速率监测部件实时监测各请求流的强度，当式(26)不能得到满足时，接纳控制部件开始拒绝相应类别的新请求。过了一段时间，当式(26)重新得到满足时，接纳控制部件又恢复接纳相应类别的新请求。

当接纳控制部件拒绝一个请求时，最简单的手段是直接断开 socket 连接。但用户因为不了解在服务器端发生了什么，可能会不断地刷新页面，这样会对服务器造成不必要的访问负荷。尤其在重载情况下，这种负荷也是很大的。因此，更友好的拒绝手段是事先定义一个拒绝内容，告诉用户系统现在很忙，可以稍后再来访问。

根据式(26)，我们可得到

$$\sum_{i=1}^{M}\lambda_i E\{X\}\leqslant N \tag{27}$$

当重新分配服务器资源时，若式(27)得不到满足，则说明整个集群系统提供的服务能力已不能满足 M 类请求的总需求。分配器应在日志中记录下当时的情况，并通知管理员。

3 测试

3.1 负载的模拟

为了模拟一些客户向 Web 服务器发送 HTTP 请求，我们使用 SURGE[13] 来产生 Web 文档和 HTTP 负载流。SURGE 在请求流的大小分布、服务器上文档的大小分布、各文档相对的访问频率、请求访问的局部性和各个请求的到达时间间隔等方面与实际情况下的规律相符。

SURGE 使用人工线程来产生请求，每个线程是一个循环，它交替地发送请求和睡眠。每个人工线程称为一个等效客户(User Equivalent)。SURGE 所产生负载的大小可以通过改变等效客户的个数来调节。我们使用 SURGE 产生了2,000个文档，最小的为 77 字节，最大的为 3,119,822 字节，平均大小为 25,896 字节。

3.2 测试环境

被测试的 Web 集群服务器由 1 个分配器和 12 个后台服务器构成，其中 2 台服务器的权值设为 1，8 台的权值设为 2，另外 2 台的权值设为 3。使用若干台测试机来发送 HTTP 请求。各机器的配置如表 1。网络环境为 100M。

通过测试与分析，服务时间(以秒为单位)所服从的有界 Pareto 分布的下界 $k=0.833*10^{-3}$，上界 $p=0.264$，形状参数 $\alpha=1.4$。

表 1 机器配置

机器	CPU	内存	硬盘	操作系统
分配器	Intel P3 500M	256M	8G IDE	优化了的 Linux 2.4.20-8
权值为 1 的服务器	Intel P3 866M	256M	18G SCSI	Linux 2.4.20 Apache 2.0.40
权值为 2 的服务器	Intel P4 1.7G	256M	60G IDE	Linux 2.4.20 Apache 2.0.40
权值为 3 的服务器	AMD Althon MP 1.8×2	256M	40G SCSI	Linux 2.4.20 Apache 2.0.40
测试机	Intel P3 700M～1G	128M	40G IDE	Linux 2.4.20 SURGE

3.3 测试结果

测试时，系统每 5min 调整一次各组中的服务器资源。另外，每次测试时间为 105min，其中前 5min 是用来预热系统。并且系统每分钟计算一次在前一分钟内各类请求的平均伸展因子，那么一共可计算得到 100 组平均伸展因子。

3.3.1 两类请求的平均伸展因子

设请求类别数 $M=2$。请求根据源地址来分类，假设 2 台测试机发送属于 C_1 类的请求，3 台测试机发送属于 C_2 类的请求。预设参数 $\delta_1=1$ 和 $\delta_2=2$。图 3 给出了在各种负载情况下的 100 组数据的平均值。

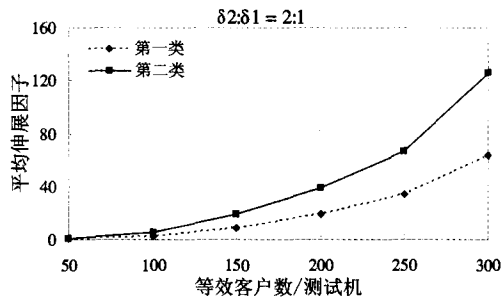

图 3 两类请求时的平均伸展因子

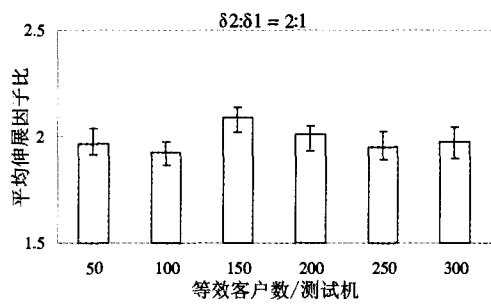

图 4 两类请求时的平均伸展因子比

图 3 表明：1）各类请求的平均伸展因子随着负载的加重而增加；2）在各种负载条件下，C_2 类请求的平均伸展因子都高于 C_1 类请求的平均伸展因子。这满足相对区分服务模型的可预测性要求。

图 4 反映了在各种负载情况下 100 个平均伸展因子比的波动情况。图中，柱状的值是这 100 个平均伸展因子比的平均值，柱状以上横线的值是 3%的分位点，即将 100 个平均伸展因子从大到小排列后排第 3 位的值，柱状以下横线的值是 97%分位点。这三个值之间的差值可反映平均伸展因子比的波动情况。

从图 4 可以看出，两类请求平均伸展因子比的均值都在 2 附近，并且上下的波动不是很大，向上和向下的波动都不超

维普资讯 http://www.cqvip.com

过0.094。这满足相对区分服务模型的可控制性要求。

3.3.2 三类请求的平均伸展因子

设请求类别数$M=3$。请求根据源地址来分类。假设2台测试机发送属于C_1类的请求，3台测试机发送属于C_2类的请求，4台测试机发送属于C_3类的请求。预设参数$\delta_1=1$，$\delta_2=2$和$\delta_3=3$。图5给出了在各种负载情况下的100组数据的平均值。图5表明，在各种负载情况下，C_3类请求的平均伸展因子大于C_2类请求的平均伸展因子；C_2类请求的平均伸展因子大于C_1类请求的平均伸展因子。这也满足相对区分服务模型的可预测性要求。

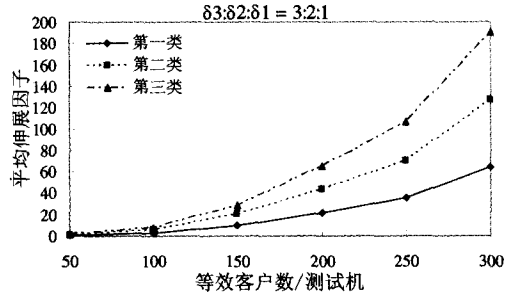

图5 三类请求时的平均伸展因子

100组平均伸展因子比的波动情况如图6。从图6可以看出，C_3类请求与C_1类请求的平均伸展因子比的均值都在3附近，并且上下的波动不是很大，向上和向下的波动都不超过0.091；C_2类请求与C_1类请求的平均伸展因子比的均值都在2附近，并且上下的波动不是很大，向上和向下的波动都不超过0.103。这也满足相对区分服务模型的可控制性要求。

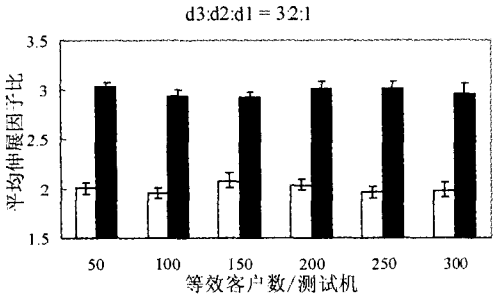

图6 三类请求时的平均伸展因子比

3.3.3 平均伸展因子比与服务器数量的关系

图7给出了请求类别数为2、2台测试机发送属于C_1类的请求、3台测试机发送属于C_2类的请求、每台测试机的等效客户数量为200、预设参数$\delta_1=1$和$\delta_2=2$时，平均伸展因子比与后台服务器数量的关系(测试时，最多用到了16台服务器，除了前面用到的8台权值为2的机器外，另外的8台与它们的配置虽然不完全一样，但差别不大，性能差不多，因此16台机器的权值都设为2)。

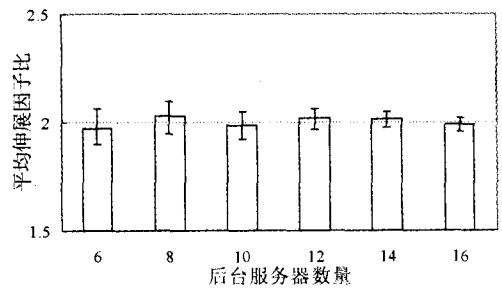

图7 平均伸展因子比与服务器数量的关系

图7表明：1)随着后台服务器数量的增加，平均伸展因子比越来越接近$\delta_2:\delta_1=2$；2)随着后台服务器数量的增加，平均伸展因子比的波动越来越小。

平均伸展因子比不能精确等于预先设定比例的一个重要原因就是，当某台服务器被多个类别请求共享时，不能准确有效地控制该服务器资源的分配。例如，有3台服务器，其归一化权值分别为$w_1=0.1$，$w_2=0.4$和$w_3=0.5$。请求被分为两类，通过计算得到$n_1=0.2$和$n_2=0.8$，那么也就是说，服务器RS_1完全为第一类请求提供服务，服务器RS_3完全为第二类请求提供服务，服务器RS_2要用其$(0.2-0.1)/0.4=25\%$的资源服务于第一类请求，而用其$(0.8-0.5)/0.4=75\%$的资源服务于第二类请求，但是我们不能保证服务器RS_2在任何负载的状况下始终保持其25%的资源服务于第一类请求，其75%的资源服务于第二类请求。但是随着后台服务器台数的增加，这种影响逐渐减弱，因此平均伸展因子的比越来越接近预设的比值，并且波动越来越小。

结论和未来的工作 本文分析了Web请求服务时间的数字特性，并使用M/G/1 FCFS排队模型对Web服务器及Web集群进行了建模。在对模型进行分析的基础上，设计并实现了一种异构Web集群服务器中的比例伸展因子区分服务的方案，并提出了基于概率空间的请求调度算法。请求被分成多个类，无论系统的负载如何，系统确保每类请求的平均伸展因子与事先给定的参数成正比。实际测试表明，所设计的方案满足相对区分服务的可预测性和可控制性的要求。

当一台服务器服务多个类别的请求时如何分配资源，是我们即将研究的内容。

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 Dovrolis C, Ramanathan P. A Case for Relative Differentiated Services and the Proportional Differentiation Model [J]. IEEE Network, 1999, 13(5): 26～34

4 Dovrolis C, Stiliadis D, Ramanathan P. Proportional Differentiated Services: Delay Differentiation and Packet Scheduling [J]. IEEE/ACM Transaction on Networking, 2002, 10(1): 12～26

5 Lee S C M, Lui J C S, Yau D K Y. A Proportional Delay Diffserv-enabled Web Server: Admission Control and Dynamic Adaptation [J]. IEEE Transaction on Parallel and Distributed Systems, 2004, 15(5): 385～400

6 Wei J, Xu C Z, Zhou X. A Robust Packet Scheduling Algorithm for Proportional Delay Differentiation Services [C]. In: Proceedings of IEEE GlobeCom, Dallas, Texas, USA, 2004, 2: 697～701

7 Harchol-Balter M. Task Assignment with Unknown Duration [J]. Journal of ACM, 2002, 29(2): 260～288

8 Chekuri C, Goel A, Khanna S, et al. Multi-processor Scheduling to Minimize Flow Time with Resource Augmentation [C]. In: Proceedings of ACM Symposium on Theory of Computing, Chicago, USA, 2004. 363～372

9 Crovella M, Bestavros A. Self-similarity in World Wide Web Traffic: Evidence and Possible Causes [J]. IEEE/ACM Transactions on Networking, 1997, 5(6): 835～846

10 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

11 Riska A, Sun W, Smirni E, et al. ADAPTLOAD: Effective Balancing in Clustered Web Servers under Transient Load Conditions. In: Proceedings of 22nd International Conference on Distributed Computer Systems, Vienna, Austria, 2002. 104～112

12 孙荣恒，李建平. 排队论基础[M]. 北京：科学出版社，2002. 116～119

13 Barford P, Crovella M. Generating Representative Web Workloads for Network and Server Performance Evaluation [C]. In: Proceedings of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, Madison, USA, 1998. 151～160