# Web 集群的区分服务与负载均衡策略研究

高 昂* 慕德俊 胡延苏
(西北工业大学自动化学院 西安 710072)

**摘 要**：该文从区分服务和负载均衡两方面研究Web集群系统，设计并实现了一种支持区分服务的负载均衡集群模型，通过系统辨识和资源最优控制，对后端资源统一调度，动态调整每台后端节点不同优先级业务类的资源配额；同时设计了基于业务类的最大空闲节点优先的负载均衡策略以保证各个后端节点同一业务类资源被公平消耗，从而控制端到端延迟，实现比例延迟保证。经实验验证，无论采用轮询调度、最少连接数调度还是最大空闲节点优先的均衡策略，资源最优控制器均能取得良好的区分效果，扩展实验还表明，相对于轮询调度和最少连接数调度，采用最大空闲节点优先的均衡策略，能够使该集群系统在实现比例延迟保证的同时获得最大的性能优化，整体吞吐量提高了25%，平均延迟降低了33%。

**关键词**： Web 集群；负载均衡；区分服务；最大空闲节点优先
**中图分类号**：TP393 **文献标识码**： A **文章编号**：1009-5896(2011)03-0555-08
**DOI**: 10.3724/SP.J.1146.2010.00409

# Differentiated Service and Load Balancing in Web Cluster

Gao Ang Mu De-jun Hu Yan-su
(*College of Automation*, *Northwestern Polytechnical University*, *Xi'an* 710072, *China*)

**Abstract**: From the view of differentiated service and load balancing, this paper proposes and implements a load balancing cluster architecture supported differentiated service. By system identification and resource optimal control, the front-end dispatcher can adjust the resource quotas assigned to different classes in every single back-end server, and Multi-class based Maximum Idle First (MIF) load balancing strategy is designed to ensure every class a fair consumption among back-end nodes. As a result, the end-to-end delay is controlled and proportional delay is guaranteed. The experiments demonstrate that no matter using Round-Robin (RR), Least Connection Schedule (LCS) or Maximum Idle First load balancing strategy, the proposed resource optimal controller can hold the relationship between different classes. Compared to Round-Robin and Least-Connection First Schedule, Maximum Idle First strategy increases the total throughput by 25% and reduces the average delay by 33%.

**Key words**: Web cluster; Load balancing; Differentiated service; Maximum Idle First (MIF)

## 1 引言

部署集群系统以应对快速增长的用户对Web资源可用性和稳定性的要求，是当前主流的网站结构。考虑到ISP(Internet Service Provider)的商业利益，集群规模不可能无限制扩充，并且各个站点的负载分布往往受到用户的浏览习惯，地理分布以及突发事件的影响，即使是大规模的集群系统，也会出现系统过载的情况(over-loaded)。对于电子商务和商业站点，用户的结算请求相对于普通的浏览请求，受服务品质协议 SLA (Service Level Agreement) 保护的付费用户相对于普通用户，都应享有更好的服务质量。由于负载的不可预测，为不同的 Web 业务提供区分服务以适应异质化的 Web 应用和客户需求，成为各运营商收益最大化的重要手段[1]。区分服务技术一方面要求为不同业务类提供相应服务等级所需的资源，尤其是满足高优先级类，另一方面要求提高系统的资源利用率，以尽量保护低优先级类业务。

文献[1]提出的DDSD(Demand-Driven Service Differentiation)区分策略以及它的变形[2]，通过建立合适的排队模型(M/M/1或 M/G/1)，选取伸展因子为性能指标，将资源的重新分配归结为一个以SLA为约束的最优化问题，但由于DDSD是建立在请求的泊松到达和处理时间的指数分布之上，而当Web负载存在突发变化，业务流到达速率和离开速率不相等时，该排队模型并不能准确反映业务流的特征[3]；文献[4]提出的DynamicPart方法，通过反馈控制，动态划分主机实现资源调度，不同业务类之间

2010-04-23 收到，2010-06-20 改回
*通信作者：高昂 snailgao@gmail.com

按照单独占有主机的方式进行性能隔离，而没有考虑主机处理能力的差异，资源划分的细粒度不够，不利于集群资源的充分利用和在线扩展。

在 Web 应用中，服务器驻留时间是影响用户感受的主要因素，即连接延迟(排队时间)和处理延迟(服务时间)之和，它是指从前端节点建立 TCP 连接，开始接受 HTTP 请求直到将应答响应移交给 TCP 连接的传输层协议所经历的时间。本文以端对端延迟作为性能指标研究基于 Web 集群的区分服务和负载均衡。从 3 个方面来支持基于集群系统的比例延迟服务 PDDS (Proportional Delay Differentiated Service)，弥补上述两种方法的不足。首先，考虑到 Web 服务器结构以及 HTTP 请求处理流程[4]，本文设计并实现了一种实现比例延迟保证的负载均衡集群模型；其次结合控制理论，通过系统辨识和最优控制，对各个业务类所占用的集群资源周期性修正，来适应负载的突发变化，保持不同业务类延迟的比例关系恒定；最后采用最大空闲节点优先的负载均衡策略，以保证各后端节点同一业务类资源被公平消耗，提高系统资源的利用率。

## 2 Web 集群模型

### 2.1 支持区分服务的负载均衡集群模型

Web 集群通常由分布在高速局域网上的前端调度器(front-end dispatcher)和多台后端节点服务器(back-end server)组成，通过将用户的 HTTP 请求按特定的分发策略重定向到不同的后端节点达到负载平衡。单台服务器 QoS 的研究往往采用基于线程/进程调度的资源分配方法，根据应用的需求将 HTTP 请求划分为若干个业务类[3,5−8]，调整服务不同业务类的线程/进程数量，从而达到服务分级的目的。本文将所有后端节点的线程/进程资源由前端统一分配，同时在请求分发时，保证每台后端的同类资源被公平消耗。为此，设计了如图 1 所示的集群模型，分两个层面实现区分服务：资源调度层和请求分配层[9]。对于资源调度层，在每个采样时刻，最优控制器根据 QoS 观测器得到的各类业务流的平均延迟，动态调整每台后端节点相应业务类的资源配额；对于请求分配层，分类器根据 HTTP 请求头字段中的 URL 信息或者 cookie 将请求分为不同的业务流，其优先级关系由 ISP 指定的服务品质协议保证, 然后通过负载分发器，按照指定的均衡策略，从后端中挑选合适的节点为当前请求服务。

前端节点是整个集群系统的核心，需要同时和客户端以及后端服务器建立 TCP 连接，系统开销较大，实际应用中往往采用专用硬件设备避免成为系统瓶颈。后端节点采用线程/进程每连接结构(thread/processor per connection)，为了实现不同优先级业务的性能隔离[4](performance isolation)，后端的线程/进程池被对应分为若干部分，线程/进程池容纳服务线程/进程的个数称为线程/进程配额，不同业务类在相互隔离的池中接受服务。每台后端部署相同的内容，可处理该集群定义的所有业务类，使用与前端一致的业务分类准则，承载不同业务的 TCP 连接进入对应的完成队列中，以 FIFO 的方式等待线程/进程服务。前后端节点通过心跳检测原理(heartbeat)实现可靠的消息通讯，在每个心跳时刻，后点节点通过状态报告(status reporter)向前端返回自己的状态信息，线程调度器(process scheduler)根据前端节点返回的控制命令，调整当前节点服务各个业务类的线程/进程配额。

应该指出，本文关注的是如何在集群过载情况下，实现高效的区分服务，保证高优先级业务类享有相对于低优先级业务类更低的延迟，同时保护低优先级业务类不被过度的“牺牲”(over-sacrificed)。而节点的单点失效和冗余，节点间的内容一致性以及准入控制等不在本文的论述范围内。

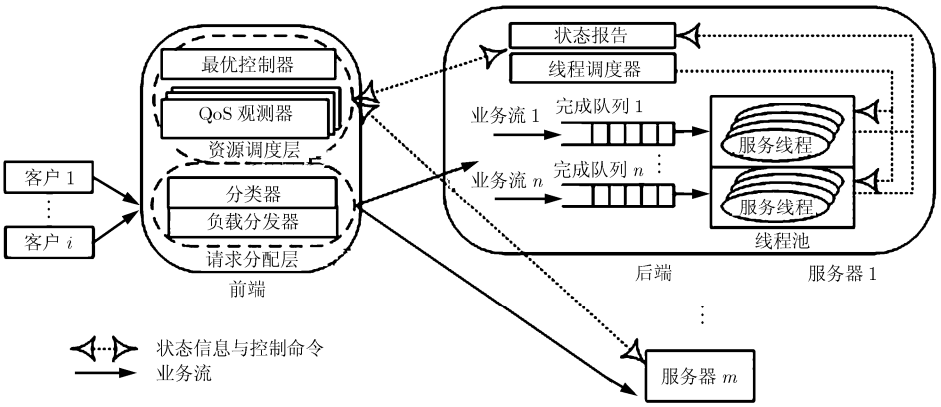



图 1 一种实现比例延迟保证的负载均衡集群模型

### 2.2 集群软件结构与节点信息交互

虽然存在更加高效的内核级别的TCP hand-off和TCP splicing转发方式，考虑到7层集群技术在支持区分用户的服务方面所具有的优势，本文采用Apache作为服务器软件，图2所示为集群软件结构以及前后端的信息交互，为Apache启用mod_proxy作为反向代理，通过mod_proxy_banlancer将用户请求URL映射为后端节点的绝对地址(absolute URL)。具体如下：

(1)配置修改过的mod_cluster实现消息传输，各个后端节点分别与前端建立TCP连接，在每个心跳时刻，主动向前端发送状态信息；同时，前端mod_cluster上实现了资源调度层的所有功能，通过最优控制，返回给各个节点更新后的业务类线程/进程配额；

(2)配置修改过的mod_proxy_banlancer模块实现请求分配层负载分发的功能，根据mod_cluster提供的后端节点的状态信息动态判断后端节点的忙闲程度，重写URL，然后由Apache httpd根据重写后的URL从后端节点获得响应。

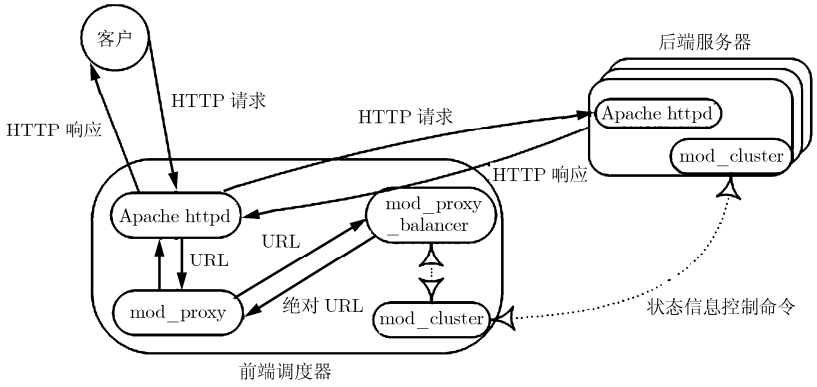



图 2 集群软件结构与节点信息交互

## 3 集群系统资源调度

假设 Web 集群由 $M$ 台后端节点为 $N$ 类业务提供服务，每台服务器共并发 $C_j$ ( $j=1,\cdots,M$ )个服务线程/进程，

$$C_j=\sum_{i=1}^{N}u_{i,j},\quad 1\le i\le N,\ 1\le j\le M \tag{1}$$

其中 $u_{i,j}$ 表示第 $j$ 台后端节点，服务于第 $i$ 类业务的线程/进程配额。为了保证不同业务流的延迟比恒定，最优控制器在每个采样时刻(采样周期为 $T$ )，调整每台后端节点不同业务类的线程配额 $u_{i,j}$，使式(2)成立。

$$\frac{\bar{\omega}_i}{\bar{\omega}_l}=\frac{\delta_i}{\delta_l},\quad 1\le i\le N,1\le l\le N \tag{2}$$

$\delta_i$ 是由 SLA 保证的固有优先级，$\delta_i$ 越小，优先级越高；$\bar{\omega}_i$ 是业务流 $i$ 平均延迟的期望值。图 3 所示为系统结构图。被控对象是整个集群系统的线程/进程资源，前后端节点通过信息交互进行通信。由于式(1)的约束，因此被控对象的输入有 $I=(N-1)\times M$ 个独立变量，在第 $k$ 个采样时刻，取输入变量

$$\boldsymbol{U}(k)=[u_{1,1}(k),u_{1,2}(k),\cdots,u_{1,M}(k),\cdots,u_{N-1,1}(k),u_{N-1,2}(k),\cdots,u_{N-1,M}(k)]^{\mathrm{T}}$$

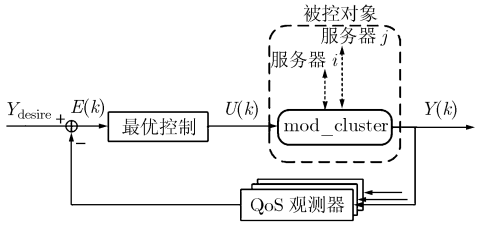



图 3 基于最优控制的资源调度结构图

**定义**

$$y_i(k)=\frac{\omega_i(k)}{\sum_{l=1}^{N}\omega_l(k)},y_{i_{\mathrm{desire}}}=\frac{\delta_i}{\sum_{l=1}^{N}\delta_l},\ 1\le i\le N$$

其中 $\omega_i(k)$ 为第 $k$ 个采样时刻 QoS 观测器获得的业务流 $i$ 的平均延迟，$y_i(k)$, $y_{i_{\mathrm{desire}}}$ 分别为为规格化延迟和其期望值。因 $\sum_{i=1}^{N}y_i(k)$=1, $\sum_{i=1}^{N}y_{i_{\mathrm{desire}}}$=1，因此共有 $O=N-1$ 个独立输出变量，故可取 $\boldsymbol{Y}(k)=[y_1(k),y_2(k),\cdots,y_{N-1}(k)]^{\mathrm{T}}$ 为系统的测量输出，最优控制器根据 $\boldsymbol{Y}(k)$ 与期望值 $\boldsymbol{Y}_{\mathrm{desire}}$ 的残差 $\boldsymbol{E}(k)$，调整不同业务的资源配额，从而保证不同业务流的优先级关系。

### 3.1 系统辨识及检验

为了设计合理的控制器，实现集群系统的对不同业务类的比例延迟保证，必须确定被控系统的数学模型。虽然这种基于性能隔离的区分模型严格地说是非线性系统[10]，但为了简化控制器设计，本文

采用线性模型近似，假定 $r$ 阶线性模型能达到很好的近似程度，那么相应的 MIMO 系统的差分方程可写为

$$\boldsymbol{A}(z^{-1})\boldsymbol{Y}(k)=\boldsymbol{B}(z^{-1})\boldsymbol{U}(k)+\boldsymbol{\varepsilon}(k) \tag{3}$$

其中

$$\left.\begin{aligned}\boldsymbol{A}(z^{-1})&=\boldsymbol{I}-\boldsymbol{A}_1z^{-1}-\cdots-\boldsymbol{A}_rz^{-r}\\ \boldsymbol{B}(z^{-1})&=\boldsymbol{B}_0z^{-1}+\cdots+\boldsymbol{B}_{r-1}z^{-r}\end{aligned}\right\} \tag{4}$$

且 $\boldsymbol{A}_i\in R^{O\times O}$，$\boldsymbol{B}_j\in R^{O\times I}$，$0<i\le r$，$0\le j<r$，由于 $\boldsymbol{Y}(k)$ 存在观测误差，并且系统内部也存在噪声，因此假设 $\boldsymbol{\varepsilon}(k)=[\varepsilon_1(k),\varepsilon_2(k),\cdots,\varepsilon_{N-1}(k)]^{\mathrm{T}}$ 为 $O$ 阶不相关白噪声序列，其均值为零。式(3)可重写为

$$\boldsymbol{Y}(k+1)=\boldsymbol{\theta}\boldsymbol{\Phi}(k)+\boldsymbol{\varepsilon}(k+1) \tag{5}$$

其中

$$\left.\begin{aligned}\boldsymbol{\theta}&=[\boldsymbol{B}_0,\cdots,\boldsymbol{B}_{r-1},\boldsymbol{A}_1,\cdots,\boldsymbol{A}_r],\ \ k\ge r-1\\ \boldsymbol{\Phi}(k)&=[\boldsymbol{U}^{\mathrm{T}}(k),\cdots,\boldsymbol{U}^{\mathrm{T}}(k-r+1),\\ &\quad\ \boldsymbol{Y}^{\mathrm{T}}(k),\cdots,\boldsymbol{Y}^{\mathrm{T}}(k-r+1)]^{\mathrm{T}}\end{aligned}\right\} \tag{6}$$

$\boldsymbol{\theta}\in R^{O\times[O\times(M+1)\times r]}$，$\boldsymbol{\Phi}$ 是 $O\times(M+1)\times r$ 维列向量，采用递推最小二乘法确定该 $r$ 阶 MIMO 系统模型参数，设由前 $q(q\ge r-1)$ 次采样数据得到的 $\boldsymbol{\theta}$ 的最小二乘估计为 $\hat{\boldsymbol{\theta}}_q$，则在第 $q+1$ 次采样后，$\hat{\boldsymbol{\theta}}_q$ 可修正为

$$\hat{\boldsymbol{\theta}}_{q+1}=\hat{\boldsymbol{\theta}}_q+\frac{[\boldsymbol{Y}(k+1)-\hat{\boldsymbol{\theta}}_q\boldsymbol{\Phi}(k)]\boldsymbol{\Phi}^{\mathrm{T}}(k)\boldsymbol{P}_q}{\lambda+\boldsymbol{\Phi}^{\mathrm{T}}(k)\boldsymbol{P}_q\boldsymbol{\Phi}(k)} \tag{7}$$

其中

$$\boldsymbol{P}_{q+1}^{-1}=\boldsymbol{P}_q^{-1}+\left[1+(\lambda-1)\frac{\boldsymbol{\Phi}^{\mathrm{T}}(k)\boldsymbol{P}_q\boldsymbol{\Phi}(k)}{(\boldsymbol{\Phi}^{\mathrm{T}}(k)\boldsymbol{\Phi}(k))^2}\right]\boldsymbol{\Phi}(k)\boldsymbol{\Phi}^{\mathrm{T}}(k) \tag{8}$$

$\boldsymbol{P}_q$ 为协方差阵，$\lambda$ 为遗忘系数。以上各式中 $\boldsymbol{\Phi}(k)$ 与 $\boldsymbol{Y}(k)$ 均指实验获得的数据。选取适当的初值 $\hat{\boldsymbol{\theta}}_0$ 和 $\boldsymbol{P}_0$，即可得到系统参数的估计值 $\hat{\boldsymbol{\theta}}$，由于估计值与真值存在误差，定义损失函数：

$$j(r)=\sum_{k=r}^{R+r-1}||\boldsymbol{Y}(k+1)-\hat{\boldsymbol{\theta}}_q\boldsymbol{\Phi}(k)||^2 \tag{9}$$

其中 $||\bullet||$ 为向量2范数，$R$为样本数目。采用$F$检验确定系统阶数。设 $r_1$ 和 $r_2$ $(r_1<r_2)$ 是模型的两个相邻阶数，构造统计量：

$$H(r_1,r_2)=\frac{j(r_1)-j(r_2)}{j(r_2)}\cdot\frac{R-2r_2}{2(r_2-r_1)} \tag{10}$$

当 $W$ 充分大且 $r_2>r_1$ 时，$H(r_1,r_2)$ 服从分布：$H\sim F(2(r_2-r_1),R-2r_2)$。

表1为系统参数辨识和损失函数$j(r)$的伪代码，显然，将得到序列 $\boldsymbol{J}=[j(1),j(2),\cdots,j(r_{\mathrm{MAX}})]^{\mathrm{T}}$，按式(10)依次计算 $H(r_1,r_2)$ 的值。对于本文搭建的Test-bed(见5.1节)，由4台后端节点组成集群，每台并发100的服务线程，对两类业务提供服务，即 $N=2$，$M=4$，取 $\boldsymbol{Y}_{\mathrm{desire}}=[1/3]$，每台节点根据伪随机序列当前的值，调整下一周期该节点这两类业务的线程配额 $\boldsymbol{U}(k+1)=[u_{1,1}(k+1),u_{1,2}(k+1),\cdots,u_{1,4}(k+1)]^{\mathrm{T}}$，输入的伪随机序列由 $\xi(k)=\xi(k-p)+\xi(k-q)(\mathrm{mod}\,4)$ 产生。取 $p=7$，$q=13$，$\xi(k)$ 与 $u_{i,j}(k+1)$ 具体对应关系如表2所示。

**表1 系统参数辨识和损失函数 $j(r)$ 的伪代码**

| |
|---|
| (1)取 $r$ 初值为系统阶数的可能的最大值，$r$=$r_{\mathrm{MAX}}$； |
| (2) $q=0$，$k=r-1$，选取合适的 $\hat{\boldsymbol{\theta}}_0$，$\boldsymbol{P}_0$； |
| (3)取前 $[k-r+1,k]$ 组样本数据，按式(6)构造 $\boldsymbol{\Phi}(k)$； |
| (4)由式(7)计算 $\hat{\boldsymbol{\theta}}_{q+1}$，$\boldsymbol{P}_{q+1}$； |
| (5) $k=k+1$；$q=q+1$； |
| (6)如果 $k\le R$，跳转到(3)；否则，此时的 $\hat{\boldsymbol{\theta}}_{R-r+1}$ 则为该 $r$ 阶系统的最小二乘估计； |
| (7)按式(9)计算 $j(r)$； |
| (8) $r=r-1$； |
| (9)如果，$r\ge1$，跳转 (2)；否则，结束。 |

**表2 $\xi(k)$ 与 $u_{i,j}(k+1)$ 对应关系**

| $\xi(k)$ | $u_{i,j}(k+1)$ |
|---|---|
| 0 | 25 |
| 1 | 40 |
| 2 | 60 |
| 3 | 75 |

取采样周期 $T$ =30 s，实验共进行5000 s，获得150组有效数据，对于0.05的显著性水平，$F_{0.05}(2,144)\approx3.05$，因为 $H(2,3)=2.29<F_{0.05}(2,144)$，即当模型阶次为2和3时 $j(r)$ 已无显著变化，所以 $r=2$ 时，式(5)达到了足够的近似程度。因此原系统可认为是二阶系统，对应系统参数的最小二乘辨识为

$$\left.\begin{aligned}\hat{\boldsymbol{\theta}}&=[\hat{\boldsymbol{B}}_0,\hat{\boldsymbol{B}}_1,\hat{\boldsymbol{A}}_1,\hat{\boldsymbol{A}}_2]\\ \hat{\boldsymbol{B}}_0&=[b_{0,1},b_{0,2},b_{0,3},b_{0,4}]\\ &=[-0.0004,-0.0005,-0.0004,-0.0004]\\ \hat{\boldsymbol{B}}_1&=[b_{1,1},b_{1,2},b_{1,3},b_{1,4}]=[0.0049,0.0060,0.0056,0.0050]\\ \hat{\boldsymbol{A}}_1&=a_1=0.4528,\ \ \hat{\boldsymbol{A}}_2=a_2=0.0922\end{aligned}\right\} \tag{11}$$

验证结果如图4所示。可见辨识值较好地近似于实际测量值，所以用二阶线性MIMO模型描述区分模型是合适的。

### 3.2 资源最优控制

后端节点的线程/进程的调度可看作是一个最优化问题，一方面希望各类业务规格化延迟 $y_i$ 与期

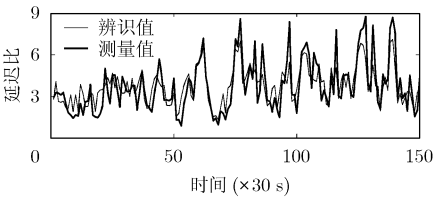



图4 系统辨识值与实际测量值比较

望值 $y_{i\text{desire}}$ 的偏差尽量的小，另一方面尽可能减少控制量 $\boldsymbol{U}(k)$ 的变化率，以减小系统开销，构造损失函数为

$$L = E\{||\boldsymbol{W}[\boldsymbol{Y}(k+1)-\boldsymbol{Y}_{\text{desire}}]||^2 + ||\boldsymbol{Q}[\boldsymbol{U}(k)-\boldsymbol{U}(k-1)]||^2\} \tag{12}$$

其中权重矩阵 $\boldsymbol{W}$ 是为 $O\times O$ 维正定对角阵，罚矩阵 $\boldsymbol{Q}$ 是 $I\times I$ 维单位阵，矩阵 $\boldsymbol{W}$ 对角线上的元素代表了不同业务类的优先级关系：$w_{ii}$ 越小，相应的业务流在资源调度的时候越受“歧视”。

若 $\widetilde{\boldsymbol{\Phi}}(k)=[\boldsymbol{0},\boldsymbol{U}^{\mathrm{T}}(k-1),\cdots,\boldsymbol{U}^{\mathrm{T}}(k-r+1),\boldsymbol{Y}^{\mathrm{T}}(k),\cdots,\boldsymbol{Y}^{\mathrm{T}}(k-r+1)]^{\mathrm{T}}$，则有

$$\begin{aligned}L =& E\{||\boldsymbol{W}[\boldsymbol{Y}(k+1)-\boldsymbol{Y}_{\text{desire}}]||^2 + ||\boldsymbol{Q}[\boldsymbol{U}(k)\\ &-\boldsymbol{U}(k-1)]||^2\} = E\{||\boldsymbol{W}[\hat{\boldsymbol{\theta}}\widetilde{\boldsymbol{\Phi}}(k)+\widehat{\boldsymbol{B}}_0\boldsymbol{U}(k)\\ &+\boldsymbol{\varepsilon}(k+1)-\boldsymbol{Y}_{\text{desire}}]||^2\}+||\boldsymbol{Q}[\boldsymbol{U}(k)-\boldsymbol{U}(k-1)]||^2\\ =& ||\boldsymbol{W}[\hat{\boldsymbol{\theta}}\widetilde{\boldsymbol{\Phi}}(k)-\boldsymbol{Y}_{\text{desire}}]||^2 + ||\boldsymbol{W}\boldsymbol{B}_0\boldsymbol{U}(k)||^2\\ &+2\boldsymbol{U}^{\mathrm{T}}(k)\widehat{\boldsymbol{B}}_0^{\mathrm{T}}\boldsymbol{W}^{\mathrm{T}}\boldsymbol{W}[\hat{\boldsymbol{\theta}}\widetilde{\boldsymbol{\Phi}}(k)-\boldsymbol{Y}_{\text{desire}}]+||\boldsymbol{Q}\boldsymbol{U}(k)||^2\\ &+||\boldsymbol{Q}\boldsymbol{U}(k-1)||^2 -2\boldsymbol{U}^{\mathrm{T}}(k-1)\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{U}(k)\\ &+E\{||\boldsymbol{W}\boldsymbol{\varepsilon}(k+1)||^2\}\end{aligned}$$

当 $L$ 取得最小值时，其关于 $\boldsymbol{U}$ 的导数为零，因此：

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{U}(k)} =& 2(\boldsymbol{W}\widehat{\boldsymbol{B}}_0)^{\mathrm{T}}\boldsymbol{W}[\hat{\boldsymbol{\theta}}\widetilde{\boldsymbol{\Phi}}(k)-\boldsymbol{Y}_{\text{desire}}]+2(\boldsymbol{W}\widehat{\boldsymbol{B}}_0)^{\mathrm{T}}\\ &\cdot\boldsymbol{W}\widehat{\boldsymbol{B}}_0\boldsymbol{U}(k)+2\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{U}(k)-2\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{U}(k-1)\\ =&0\end{aligned} \tag{13}$$

求解式(13)，得到第 $k$ 个采样时刻，最优控制器输出的控制量为

$$\begin{aligned}\boldsymbol{U}^*(k)=&[(\boldsymbol{W}\widehat{\boldsymbol{B}}_0)^{\mathrm{T}}\boldsymbol{W}\widehat{\boldsymbol{B}}_0+\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q}]^{-1}\{(\boldsymbol{W}\widehat{\boldsymbol{B}}_0)^{\mathrm{T}}\\ &\cdot\boldsymbol{W}[\boldsymbol{Y}_{\text{desire}}-\hat{\boldsymbol{\theta}}\widetilde{\boldsymbol{\Phi}}(k)]+\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{U}(k-1)\}\end{aligned} \tag{14}$$

## 4　最大空闲节点优先的负载均衡策略

资源最优控制主要解决如何为各个后端节点指定不同业务类的线程/进程配额，而负载均衡策略则保证各个节点业务类配额被平均消耗，避免有的后端节点过载或“饥饿”，大多数均衡策略主要是建立在访问泊松到达和服务时间指数分布的假设上，如轮询调度 RR (Round-Robin)，最少连接数调度 LCS (Least Connection Schedule)等，但是，请求页面中包含越来越多的动态嵌入对象以及数据库操作，使得用户的服务时间难以满足指数分布，并且这些静态均衡策略均不支持区分服务，因此为了准确地评估每台后端节点各个业务类的负载状况进而分配新的请求任务，本文采用基于业务类的最大空闲节点优先 MIF (Maximum Idle First)的负载均衡策略，考虑到影响服务器驻留时间的主要因素，后端节点 $j$ 返回的状态信息是由各个业务类对应的完成队列长度 $\ell_{i,j}$ 和线程池中空闲的服务器线程个数 $\tau_{i,j}$ 组成的 $N$ 个二元组 $<\ell_{i,j},\tau_{i,j}>$，$i=1,\cdots,N$。服务器驻留时间 $\bar{\omega}_{i,j}$ 正比于完成队列长度，反比于并发处理该业务类的线程数目，即空闲的该业务类线程数[3]。对于支持 HTTP1.1 的客户端，一个 Web 会话(Web session)包含的多个嵌入请求(embedded requests)通常是在一个持续TCP连接上以Pipeline方式完成的。

$$\left.\begin{aligned}&\bar{\omega}_{i,j}=\alpha\ell_{i,j}E[s_{i,j}]/\tau_{i,j}\\ &\text{idle}_{i,j}\propto 1/\bar{\omega}_{i,j}\end{aligned}\right\} \tag{15}$$

$\text{idle}_{i,j}$ 表示节点 $j$ 服务于业务流 $i$ 的忙闲度，与服务器驻留时间成反比，$\alpha$ 为传输系数，是与系统相关的常数。$E[s_{i,j}]$ 是会话大小的数学期望，$E[s_{i,j}]=E[v_{i,j}]E[g_{i,j}]$, $1\le i\le N$,$1\le j\le M$，$v_{i,j}$，$g_{i,j}$ 分别表示组成会话的嵌入请求的大小和个数。由于集群后端节点部署的内容相同以及不同业务流请求的自相似性，$E[s_{i,j}]$ 为定值，当新的业务流 $i$ 请求到达前端节点，mod_proxy_banlancer计算各个后端节点的 $\text{idle}_{i,j}$ 值，将请求交给 $\text{idle}_{i,j}$ 最大的节点处理。

## 5　实验验证

### 5.1　实验配置与性能评价

Test-bed 由 9 台运行于 1000 Mbps Ethernet 的 PC 组成，其中 4 台为后端节点($M=4$)，服务器软件为 windows 平台上 Apache2.0(Httpd–ver2.0.53)，每台后端并发 100 个服务线程 $C_j=100$；另外 4 台为 Linux 系统(kernel-2.6.27)，运行 SURGE (ver 1.00a)作为负载发生器，每台模拟 120 个客户向前端节点并发请求；前端节点为 windows 平台上的 Apache2.2 (Httpd–ver2.2.63)，并发足够多的服务线程，避免其成为系统性能瓶颈，实验中设为 1000，按 2.2 节配置修改过的 mod_cluster 和 mod_poxy_balancer 实现资源调度层和请求分配层的功能，按式(14)设计资源最优控制器。在本文的实验中，根据客户端 IP 地址将请求分为业务流 1 和业务流 2，两者的期望延迟比取 $\bar{\omega}_2/\bar{\omega}_1=\delta_2/\delta_1=2$, $N=2$，即 $\boldsymbol{Y}_{\text{desire}}=[1/3]$。

每台后端生成大小服从重尾分布的 20000 个文

件，这些文件作为嵌入请求组成了 90054 个 Web 会话，数据的统计结果如图 5 所示，分别表示嵌入请求个数 $g$ (图 5(a))，嵌入文件大小 $v$ (图 5(b))以及实测 Web 会话大小 $s$ (图 5(c))的概率密度分布。从图 5(b)可以看出，嵌入请求大于 $10^4$ byte 的概率小于 $10^{-2}$，并且 Web 会话大小的分布也是具有重尾特性，符合网络通讯实际情况。定义相对方差[11]式(16)作为评价控制系统的性能指标，$R(Y)$ 的值越小，说明控制器对于规格化延迟 $\boldsymbol{Y}(k)$ 稳定在 $\boldsymbol{Y}_{\text{desire}}$ 的控制能力越强，区分效果越好。

$$R(Y)=\frac{\sqrt{\sum_{k=1}^{I}||\boldsymbol{Y}(k)-\boldsymbol{Y}_{\text{desire}}||^2/\boldsymbol{I}}}{||\boldsymbol{Y}_{\text{desire}}||} \tag{16}$$

### 5.2 实验验证与结果分析

为了验证基于最优控制的资源调度方法和最大空闲节点优先的负载均衡策略，本文设计了两组对比实验，分别比较资源最优控制器在不同负载分发策略下的区分效果，以及相应的系统吞吐量和延迟。第 1 组实验取采样周期 $T$ =30 s，结果如图 6，图 7 所示，其中图 6(a)，6(b)，6(c)分别为 RR，LCS 以及 MIF 均衡策略下，不同业务类延迟及延迟比变化情况，图 7 为相应的线程配额变化情况。以 RR 为例说明，前 800 s 资源最优控制器关闭，后端服务器分配给两类业务的线程配额 $u_{1j}=u_{2j}=50(1\le j\le 4)$，相应的延迟基本一致(图 6(a))，无区分服务可言；800 s 之后，在控制器的作用下，各后端服务器分配给不同业务类的线程配额随之调整(图 7(a)，其中一台后端的配额变化)，对应的实际延迟之比 $\omega_2(k)/\omega_1(k)$ 也基本稳定在 2 附近。对比图 6 和图 7 的结果，还可以看出一下几点：

(1)无论是哪种负载均衡策略，哪种采样周期，资源最优控制器均能够较好的实现比例延迟保证，验证了本文设计的负载均衡集群模型的有效性；

(2)比较 800 s 前后(控制器开启)的延迟 $\omega_i^{-800}$ 和 $\omega_i^{+800}$ (无论是那种均衡策略)，显然有

$$\omega_2^{+800}-\omega_2^{-800}\ge\omega_1^{-800}-\omega_1^{+800} \tag{17}$$

也即，低优先级业务流 2 延迟增加的程度要大于高优先级业务流 1 延迟减小的程度，尽管与此，鉴于第 1 节的原因，实现比例延迟保证还是很有意义的；

(3)比较稳定状态下( $\boldsymbol{Y}=\boldsymbol{Y}_{\text{desire}}$ )各类业务的延迟，有

$$\left.\begin{aligned}\omega_{1,\text{MIF}}^{+800}&<\omega_{1,\text{RR}}^{+800}\approx\omega_{1,\text{LCS}}^{+800}\\ \omega_{2,\text{MIF}}^{+800}&<\omega_{2,\text{RR}}^{+800}\approx\omega_{2,\text{LCS}}^{+800}\end{aligned}\right\} \tag{18}$$

其中 $\omega_{i,\text{RR}}^{+800}$，$\omega_{i,\text{LCS}}^{+800}$，$\omega_{i,\text{MIF}}^{+800}$ 分别表示 RR，LCS，MIF 均衡策略下，稳定状态时业务流 $i$ 的延迟，式(18)表明，MIF 在实现比例延迟保证的前提下具有相对更小的延迟；

(4)考虑到采样周期对于控制器的影响，在 $T$ =20 s 的情况下，重复实验 1，得到类似的结果。按式(16)计算控制器开启后的相对方差，如表 3 所示：虽然不同采样周期下，均可实现区分服务，但是当 $T$ =20 s 时，延迟比抖动明显。这主要是由于个别巨型文件的存在导致了处理延迟抖动，而这种抖动在较小采样周期下表现的更为明显；同时可以看到，资源最优控制器在 MIF 均衡策略下，更好地保持了系统的性能。

**表 3 不同采样周期和负载均衡策略下 $R(Y)$ 的值**

| $R(Y)$ | RR | LCS | MIF |
|---|---|---|---|
| 20 s | 0.3843 | 0.3810 | 0.31925 |
| 30 s | 0.3831 | 0.3759 | 0.28000 |

为了进一步比负载均衡策略对集群系统吞吐量和延迟的影响，第 2 组实验在 $T$ =30 s 条件下进行，结果如图 8，图 9 所示，无论哪种均衡算法，集群系统的吞吐量(图 8)和延迟(图 9)都随着承载业务流的 TCP 连接请求到达率的增加而增加 ，同时两类业务依然保持了较好的区分关系。对于 RR 和 LCS 均衡策略来说，当 TCP 连接请求增长到 25 个/s 时，集群饱和，系统的吞吐量达到峰值，此时业务流 1 和业务流 2 的吞吐量约为 $\text{thr}_{1,\text{RR}}=\text{thr}_{1,\text{LCS}}=80$ 个

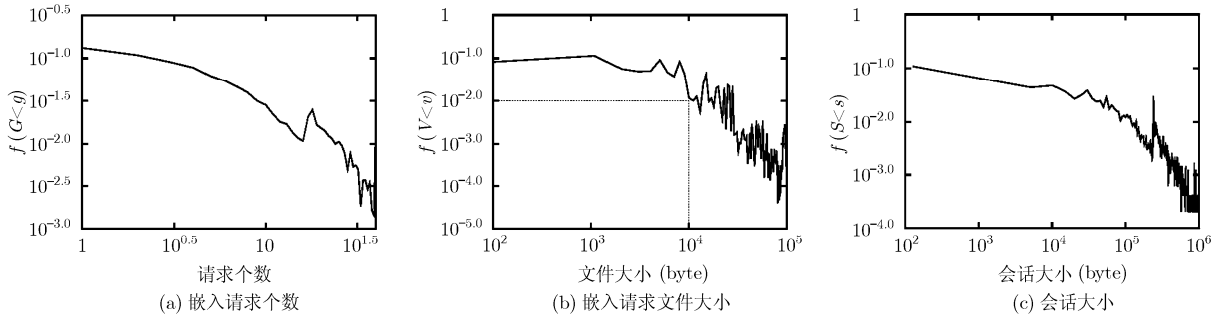
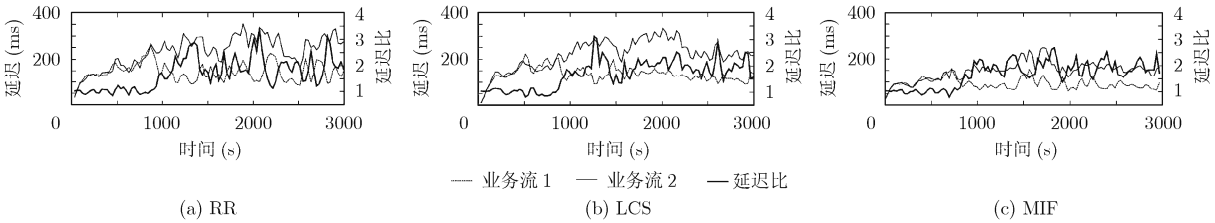
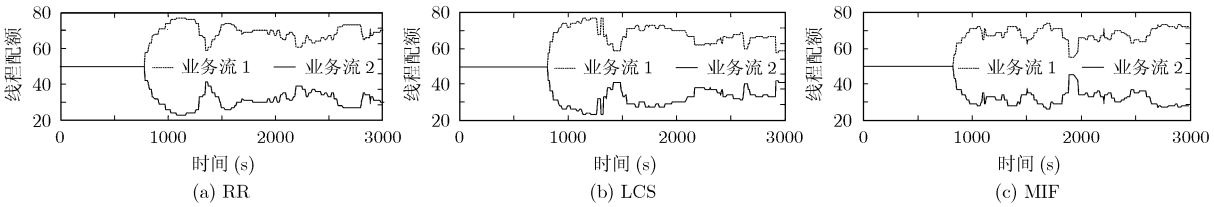



图 5 嵌入请求个数，文件大小和会话大小的概率密度



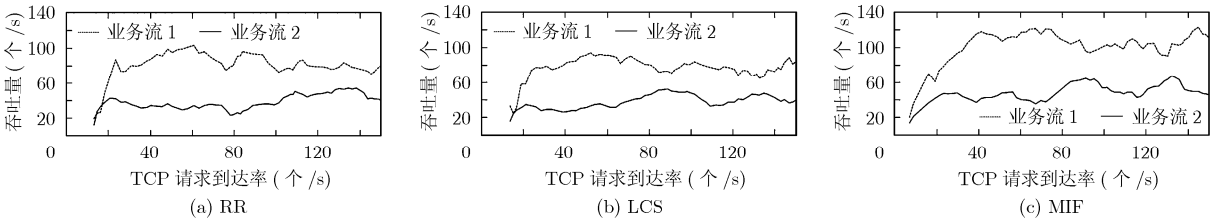

(a) RR　(b) LCS　(c) MIF

图 6　延迟对比图



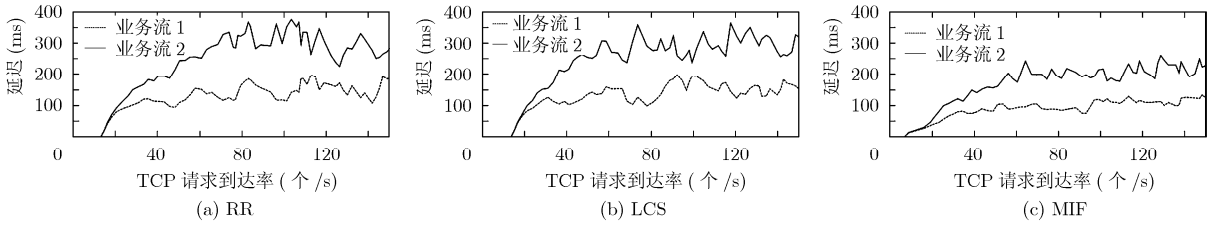

(a) RR　(b) LCS　(c) MIF

图 7　线程配额对比图




(a) RR　(b) LCS　(c) MIF

图 8　系统吞吐量随请求到达率变化




(a) RR　(b) LCS　(c) MIF

图 9 延迟随请求到达率变化

HTTP 请求/s 和 $\mathrm{thr}_{2,\mathrm{RR}} = \mathrm{thr}_{2,\mathrm{LCS}} = 40$ 个 HTTP 请求/s；当 TCP 连接请求增长到 60 个/s 时，集群处于过饱和状态，后端服务器节点通过准入控制策略限制 TCP 完成队列长度(本文采用“尾弃”策略)，此时业务流 1 和业务流 2 的平均延迟稳定在 $\omega_{1,\mathrm{RR}} = \omega_{1,\mathrm{LCS}} = 150$ ms 和 $\omega_{2,\mathrm{RR}} = \omega_{2,\mathrm{LCS}} = 300$ ms。而在 MIF 均衡策略下，业务流 1 的峰值吞吐量提高到 $\mathrm{thr}_{1,\mathrm{MIF}} = 110$ 个 HTTP 请求/s，业务流 1 和业务流 2 的平均延迟降低到 $\omega_{1,\mathrm{MIF}} = 100$ ms 和 $\omega_{2,\mathrm{MIF}} = 200$ ms，与(3)的结论一致，并且系统吞吐量提高了 25%，平均延迟降低了 33%。

$$[(\mathrm{thr}_{1,\mathrm{MIF}}+\mathrm{Thr}_{2,\mathrm{MIF}})-(\mathrm{thr}_{1,\mathrm{RR}}+\mathrm{thr}_{2,\mathrm{RR}})]/(\mathrm{thr}_{1,\mathrm{RR}}+\mathrm{thr}_{2,\mathrm{RR}})=25\%,$$

$$[(\omega_{1,\mathrm{RR}}+\omega_{2,\mathrm{RR}})-(\omega_{1,\mathrm{MIF}}+\omega_{2,\mathrm{MIF}})]/(\omega_{1,\mathrm{RR}}+\omega_{2,\mathrm{RR}})=33\%$$

这是由于MIF 负载均衡策略不仅考虑到队列长度和空闲服务器线程对驻留时间的影响，而且节点状态信息针细化到每个节点所服务的 $N$ 类业务，而其它分发算法虽然能够保证各台后端之间的公平调度，但后端节点资源不一定能够得到最优利用。举例来说：当后端节点第 $j$ 类业务的线程配额将要耗尽或者队列长度很长，而其它 $N-1$ 类业务资源非常空闲的时候，RR 和 LCS 调度策略往往将新到达的请求重定向到该节点，如果新到达的请求恰好为第 $j$ 类业务，会进一步增加该服务器对于此类业务的延迟，而此时，其它节点第 $j$ 类业务配额很有可能还有空闲，这种资源的不平等分配导致了各类业务流延迟的抖动，降低了系统的性能。

## 6　结束语

本文对基于集群系统的区分服务进行研究，提出并实现了一种支持比例延迟保证的负载均衡集群

模型，通过系统辨识建立该 MIMO 系统的数学模型，并在此基础上完成了资源最优控制器的设计；同时采用基于业务类的 MIF 负载均衡策略，保证各个后端节点资源公平消耗；最后，在 Windows 平台下，对 Apache2.2 的 mod_cluster 和 mod_poxy_balancer 模块进行了修改并分别在 RR，LCS 和 MIF 负载均衡策略下进行了实验，验证了该系统模型的有效性，实现了比例延迟保证；扩展实验还表明，采用 MIF 均衡策略，能够使该支持区分服务的负载均衡集群系统获得最大的性能优化。

## 参 考 文 献

[1] Zhu H, Tang H, and Yang T. Demand-driven service differentiation in cluster-based network servers[C]. Proceedings of IEEE INFOCOM, Anchorage, AK, United States, Apr. 24–26, 2001: 679–688.

[2] Wu X, Li M, and Wu Jian-xin. Enhanced demand-driven service differentiation algorithm in Web clusters[C]. Proceedings of IEEE ICEBE, Shanghai, China, Otc. 24–26, 2006: 386–391.

[3] Gao A, Mu D, Hu Y, and Pan W. Proportional delay guarantee in Web QoS based on predictive control[C]. Proceedings of International conference on information Secience and Engineering, Nanjing, China, Dec. 18–20, 2009: 1789–1792.

[4] Andreolini M, Casalicchio E, and Colajanni M. A cluster-based Web system providing differentiated and guaranteed services[J]. *Cluster Computing*, 2004, 7(1): 7–19.

[5] Pan W, Mu D, Wu H, and Sun Q. Proportional delay differentiation dervice in Web application servers: A feedback control approach[C]. Proceedings of International Colloquium on Computing, Communication, Control and Management, Guangzhou, China, Aug. 3–4, 2008: 600–604.

[6] Xiong Kai-qi and Harry P. SLA-based resource allocation in cluster computing systems[C]. Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium, Program and CD-ROM, Miami, FL, United States, Apr.14–18, 2008: 1–12.

[7] Xu Cheng-zhong, Liu Bo-jin, and Wei Jian-bin. Model predictive feedback control for QoS assurance in Web servers[J]. *Computer*, 2008, 41(3): 66–72.

[8] Xu Cheng-zhong. Quality assurance and adaptation of internet services: Early experience[C]. Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium, Program and CD-ROM, Miami, FL, United states, Apr.14–18, 2008: 1–5.

[9] Gao A, Zhou H, Hu Y, Mu D, and Hu W. Proportional delay differentiation service and load balancing in Web cluster systems[C]. Processings of IEEE InfoCom 2010, Student Workshop, San Diego, CA, USA, Mar.15–19, 2010: 1–2.

[10] Liu Xue, Zhu Xiao-yun, Pradeep P, Wang Zhi-kui, and Sharad S. Optimal multivariate control for differentiated services on a shared hosting platform[C]. Proceedings of the 46th IEEE Conference on Decision and Control, New Orleans, LA, United states, Dec. 12–14, 2007: 3792–3799.

[11] Wei J and Xu C. A self-tuning fuzzy control approach for end-to-end QoS guarantees in Web servers[C]. Proceedings of Quality of Service. IWQoS 2005, Passau, Germany, Jun. 21–23, 2005: 123–135.

高 昂： 男，1984 年生，博士，研究方向为网络 QoS 控制，网络化控制.

慕德俊： 男，1963 年生，教授，博士生导师，研究方向为计算机网络、信息安全.

胡延苏： 女，1985 年生，博士，研究方向为网络化控制.