计算机研究与发展
Journal of Computer Research and Development
ISSN 1000-1239/CN 11-1777/TP
42(11):1969~1976, 2005

一种资源优化的双最小均衡 Web 集群区分服务调度算法

刘安丰　陈志刚　龙国平　曾志文
(中南大学信息科学与工程学院　长沙　410083)
(anfengliu@tom.com)

A Resource Optimizing Scheduling Algorithm of Differentiated Service of Double Minimum Balance in Web Clusters

Liu Anfeng, Chen Zhigang, Long Guoping, and Zeng Zhiwen
(*College of Information Science and Engineering, Central South University, Changsha* 410083)

Abstract　Based on a new Web cluster architecture, a resource optimizing scheduling algorithm of differentiated service of double minimum balance is proposed. First, Web requests are dispatched to each back end server by resource balancing in the front scheduler. Second, the priority of Web requests and resource balancing are aggregated to design the scheduling queue of Web requests in back end server. In order to estimate the performance of the algorithm, much simulating experiments are conducted. Compared with other notable scheduling policies such as the separate scheduling strategy, the results indicate that the scheduling algorithm of double minimum balance can dramatically enhance the Web request efficiency by 11% and differentiated service is realized very well, which demonstrates the universality of the resource optimizing scheduling algorithm.

Key words　Web cluster; double minimum balance; quality of service; differentiated service; scheduling policy

摘　要　在一种新的 Web 集群体系结构的基础上,提出了一种资源优化的双最小均衡区分服务调度算法:首先在前端调度器按资源均衡度将 Web 请求分配到各后台服务器.然后将 Web 请求的优先级与资源均衡度两个特征参数结合起来,综合设计后台服务器的 Web 请求调度顺序,为了评估该算法的性能,进行了大量的模拟实验.在与其他著名调度策略如分离式调度的对比结果显示:双最小均衡调度算法使 Web 请求的效率提高了 11%,同时很好地实现了区分服务.证实了资源优化调度策略具有一定的普遍意义.

关键词　Web 集群;双最小均衡;服务质量;区分服务;调度策略

中图法分类号　TP393

1　引　　言

基于单一系统映像的 Web 服务器集群系统是满足当前应用而用得最广泛的一种方法,请求分配和选择方案是 Web 集群技术的重要研究内容[1].目前已提出了很多请求分配算法[2~4],但这些算法存在一定的局限性.从 Web 服务的发展过程中可以发现,采用多处理机(multiprocessor)技术和多线程技术可以使得系统资源真正得到并行利用[5].近来的

收稿日期:2004-05-09;修回日期:2005-02-03
基金项目:国家教育部博士点基金项目(20040533036);湖南省自然科学基金项目(03JJY4054)

研究一般将 Web 集群系统抽象为一个资源容器[6],而近几十年来的各种调度处理研究表明,任务的调度运行是以资源的满足为前提的,如果能恰当调度请求分配与选择任务,使系统资源能同时(并行)得到利用,无疑这时系统性能最高.

基于以上认识,本文的调度算法就是从资源优化的物质主义哲学观点出发,进行资源的二次最小均衡优化调度:第 1 步,请求分配. 先计算各请求对各种资源的需求,将这些请求按资源需求情况'均匀合理'地分配到各后台服务器,使各后台服务器自身拥有各维资源与分配给它的各维资源需求相一致. 第 2 步是选择调度. 在后台服务器进行合理的调度使系统资源得到最大程度的并行利用,使系统性能最高.

2 系统组成和工作原理

Web 集群体系结构是 Web 集群的重要研究内容,已经有了不少的研究[7]. 我们提出了一种新的基于内容的可扩展性透明 Web 集群体系结构[8]. 其设计目标是:保证系统可扩展性的同时做到对后台服务器的透明性,易于集成化和硬件独立化、产品化,从而获得较高的实用价值. 我们把 Switch, Dispatcher 和 Distributor 三部分跟后台 Server 分离开,形成一个独立的实体,我们称之为调度分发器. 同时它又要保证对后台服务器的透明性,即任意的 Web 服务器都不需任何修改和配置就可加入到该集群系统中,从而获得广泛的可使用性. 为达到可扩展性,我们将模块做成 4,8,16,32,64 口的前端分发器,满足用户所希望的路由器(router)一样的系统,接上电就能工作. 系统结构如图 1 所示:

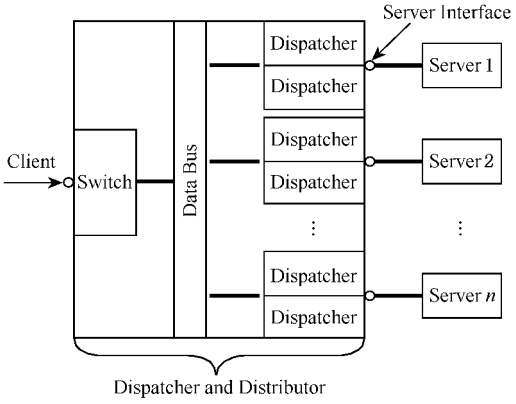

Fig. 1 A scalable content-aware Web cluster framework.

图 1 基于内容的可扩展 Web 集群体系结构

根据以上体系结构,我们设计了基于双最小均衡调度策略的区分服务模型,如图 2 所示. 前端分发器执行请求分配的作用,执行第 1 次资源均衡调度;选择算法在后台服务器上执行,先依据请求优先级将任务置入不同的优先级队列,然后进行第 2 次资源均衡调度. 由于原有的 Web 服务器是先来先服务 FCFS,需要修改后台真实服务器代码以支持基于资源优化的调度策略,我们修改了 Apache3.1.14 服务器的代码,以支持我们的调度策略. 具体调度与实施策略将在第 5 节给出.

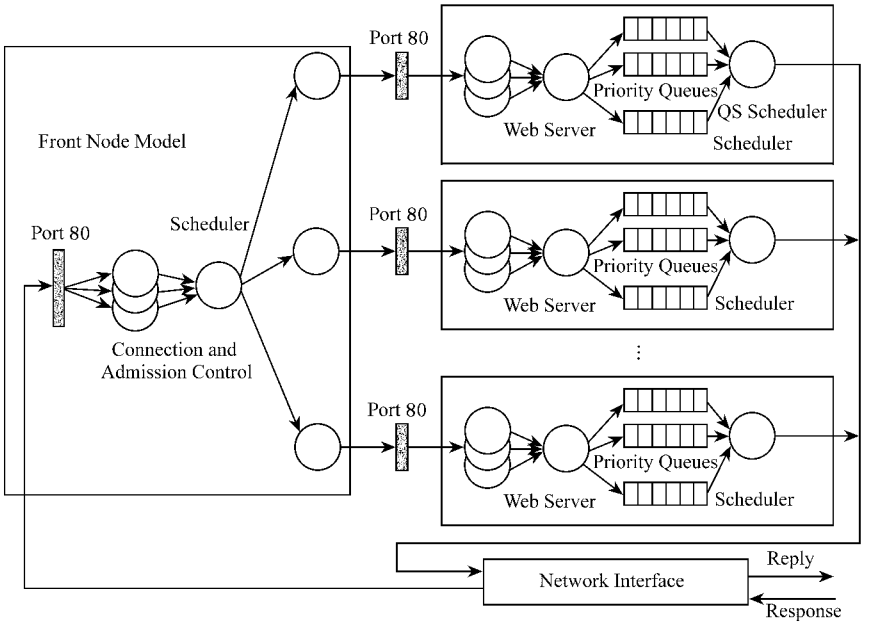

Fig. 2 The differentiated service architectures of Web cluster.

图 2 Web 集群区分服务体系结构

3 资源优化调度的有效性分析

3.1 资源优化调度策略有效性分析

假设有 m 个动态请求和 m 个静态请求,因为动态请求中最费时的是 CPU,我们设它所需的时间为 T_{cpu1},为便于对比,我们有如下等式:

$$T_{cpu1} = S_1 \times T_{io1} = S_1 \times P_1 \times T_{标}. \quad (1)$$

因为此时最费时的为 T_{cpu1},它是所需IO时间的 S_1 倍,显然 $S_1>1$,然后把它化成为一个基础的标准时间为 $S_1 \times P_1 \times T_{标}$,显然 $P_1>1$.

同样对于静态请求中最费时的是 IO,我们设它所需的时间为 T_{io2},为便于对比,我们化为下列等式:

$$T_{io2} = S_2 \times T_{cpu2} = S_2 \times P_2 \times T_{标}. \quad (2)$$

因为此时最费时的为 T_{io2},它是所需 CPU 时间的 S_2 倍,显然 $S_2>1$,然后把它化成为一个基础的标准时间为 $S_2 \times P_2 \times T_{标}$,显然 $P_2>1$.

对于分离式调度所需总的时间为

$$T_{separate} = T_{cpu1} + T_{io2} = (S_1 \times P_1 + S_2 \times P_2) \times T_{标}, \quad (3)$$

那么混合式优化调度所需 IO 时间为

$$T_{io1} + T_{io2} = P_1 \times T_{标} + S_2 \times P_2 \times T_{标} = (P_1 + S_2P_2) \times T_{标}.$$

它所需 CPU 时间为

$$T_{cpu1} + T_{cpu2} = S_1 \times P_1 \times T_{标} + P_2 \times T_{标} = (S_1P_1 + P_2) \times T_{标},$$

那么总的时间为 IO 或 CPU 二者中最大的时间,即

$$T_{mix} = \max((P_1 + S_2P_2) \times T_{标}, (S_1P_1 + P_2) \times T_{标}). \quad (4)$$

显然,由于 S_1, S_2, P_1, P_2 都大于 1,所以式(4)小于式(3),即

$$T_{mix} < T_{separate}. \quad (5)$$

基于以上的分析,我们通过下面的实验来验证基于资源优化策略的有效性.

3.2 资源优化调度策略有效性测试

为了证实我们提出的资源优化调度策略的有效性,我们与分离式调度算法[4]作对比,分离式调度算法主要思想是将动态请求与静态请求分别调度到不同的服务器上执行.

实验采用两台 Web 服务器,对同样的任务集合,首先采用分离式调度策略将动态请求与静态请求分别调往不同的服务器,然后用资源优化调度方案进行了不同的测试,测试负载为一旦有任务完成,立即调度一个新任务,即负载较重. 我们采用的测试工具是 LoadRuner[9].

测试方案结果如表 2 所示,其中方案 1 是先发两个 A 组,两个 B 组,两个 C 组. 请求顺序为:组内动态与静态页面同时发出,静态页面是动态页面的 2 倍. 重复 10 次;方案 2 是先发一个 A 组,一个 B 组,一个 C 组. 请求顺序同上. 重复 20 次. 动态页面是由 ASP 程序循环加法运算产生.

Table 1 Test Case

表 1 测试用例

Group ID	Static Request		Dynamic Request	
	Size of Page(KB)	Number of Page	Number of Addition	Number of Request
A	501	3	50000	3
B	197	14	500000	14
C	11	3	5000000	3

Table2 Comparison of Process Time Between the Resource Optimisn Schedule Algorithm and the Distributed Schedule Algorithm

表 2 资源优化调度与分离式调度算法执行时间对比 s

Time used of separate schedule algorithm	Time used resource optimism schedule algorithm	
	Scheme 1	Scheme 1
14.78	14.29	14.32

从表 2 我们可以看出,资源优化策略可以提高系统总的效率,这就证实了我们的策略是有效的. 下面给出双最小均衡算法的具体策略.

4 双最小均衡度算法(DMB)

4.1 相关定义

定义 1. 在一个调度中,若某任务的时间约束和资源需求都可以满足,则称该任务在这个调度中是可行的.

定义 2. 资源均衡度 p_r:均衡度是衡量系统资源是否被均衡使用的指标,假设真实服务器 i 的某维资源是 r_{ij},那么此台机器的总资源为 $\sum r_{ij}$, $j \in (1,2,\cdots,m)$. 每维资源与总资源的比为 $p_j = r_{ij} / \sum r_{ij}$, $j \in (1,2,\cdots,m)$. r'_{ij} 表示此台机器已经分配的资源,同样 $p'_j = r'_{ij} / \sum r'_{ij}$, $j \in (1,2,\cdots,m)$ 表示每维已经分配资源与已经分配总资源的比.

资源均衡度 $p_r = \sum_{j=1}^{m} (p'_j - p_j)^2$.

可以证明当资源均衡度 p_r 越小时,系统性能越好:当 $p_r=0$,表示分配给服务器的各维资源与服务器固有的各维处理资源能力最相适配,这时系统各维资源能充分地并行得到利用,任务完成时间为理想的最短完成时间,当 $p_r\neq 0$ 时,则表示资源之间不匹配,由于资源之间的约束关系使其不能同时得到并行利用,这时任务完成时间必然大于最短完成时间.

定义3. 任务的并行执行度:事务并行执行请求任务的个数,在我们的调度模型中,同时并行执行的线程个数是确定的,线程之间是分时并行、资源共享的;而一个线程对应一个请求,一个线程只有完成当前请求后才接受下一个请求. 假设服务器线程个数为 $\rho(\rho\geqslant 2)$.

4.2 请求分配

DMB请求分配算法基本思想:先选取剩余资源最大的服务器,这时表示此服务器剩余处理能力最大;再从任务集合中选取能使此服务器 p_r 最少的任务,这时表示服务器越能并行处理任务.

输入:有 c 台计算机,每台计算机的系统资源向量为 $\boldsymbol{R}(R_1,R_2,\cdots,R_m)$,有 m 维资源.

任务集合 $\{T_1,T_2,\cdots,T_n\}$;有 n 个请求. 每个任务需要资源为 $\boldsymbol{r}_i=\{r_{i1},r_{i2},\cdots,r_{im}\}$,$i\in\{1,2,\cdots,n\}$.

输出:任务集合 $\{T_1,T_2,\cdots,T_n\}$ 中的 c 个互不相交的子集 $S_1,S_2,\cdots,S_c$,和各服务器的剩余资源 $\{l_1^i,l_2^i,\cdots,l_m^i\}$,$i\in\{1,2,\cdots,c\}$.

目标:从任务集中选取合适的任务,分配到各服务器上,使服务器所分配的任务最多,而使服务器不能分配出的剩余资源 $\{l_1^i,l_2^i,\cdots,l_m^i\}$,$i\in\{1,2,\cdots,c\}$ 最小.

Step1. 初始化

Step1.1. let a_i 为 $r_i/\left(\sum_{i=1}^{m}r_i\right)$,$i\in\{1,2,\cdots,m\}$;/* 每维资源在总资源中的比值(机器自身的资源比例) */

Step1.2. let $C_{ij}=(r'_{ij})/\left(\sum_{j=1}^{m}r'_{ij}\right)$,$i\in\{1,2,\cdots,c\}$,$j\in\{1,2,\cdots,m\}$;/* r'_{ij} 是第 i 台服务器已经分配第 j 维资源量,这个比值表示:每台服务器,每维已经分配资源与此台机器已经分配总资源量的比值 */

Step2. 重复下列步骤,直到服务器集合或任务集合为空时:

Step2.1. 从服务器集合中选择剩余资源量总和最大的服务器,选取的服务器设为 p;

Step2.2. 从剩余任务集合中选择任务 k 能放入 p 服务器,重新计算 p 台服务器加入任务 k 的资源均衡度 p_{pk},在所有 p_{pk},$k\in\{1,2,\cdots,n\}$ 中选取 p_{pk} 最小;

Step2.3. 如果没有 p_{pk},表示所有任务都不能再放入 p 服务器,从服务器集合中去掉 p 服务器;否则,将此任务加入到 p 服务器,从任务集合中去掉此任务;

Step3. 输出各台服务的任务,分配资源,剩余资源,资源利用率;

Step4. 请求分配算法结束.

4.3 选择(调度设计)

当任务分配到后台服务器后,在这一层主要对任务进行区分服务与资源优化. 我们的选择调度算法描述如下:当一个任务到达后,首先确定它的优先级,然后,插入到此优先级队列的适当位置,使插入任务后,包含此任务以及比它优先级高的所有任务的 p_{pk} 最小,而调度时,只要按优先级顺序调度即可. 基于此,优先级队列的数据结构描述如下:

```
typedef struct PriorTaskArray {
int tasked[n];/* 此优先级队列的任务数组 */
int resource[m];/* 此队列已经分配的各维资源 */
} PriorTaskArray[p];/* p 个优先级队列 */
```

它有 p 个优先级,每个优先级队列由当前队列的已经分配的任务数组与已经分配的各维资源量组成.

当一个任务到达后的算法过程描述如下:

Step1. 新分配给后台服务一个任务 T_i;

Step2. 确定 T_i 的优先级别,假设为 P_i;

Step3. for $k=1$ to m do

for $j=0$ to P_i-1 do /* 0 为最高级 */

$Resource[k]+=PriorTaskArray[j].resource[m]$;

/* Step3 求出比 P_i 优先级高的各维资源和 */

Step4. while ($PriorTaskArray[P_i].tasked[k]!=0$)

/* 只要第 P_i 优先级队列不为空,k 初始为 0 */

{将任务 T_i 试插入到第 P_i 个优先级队列的第 k 位置后计算 p_{pk},并将最小的 p_{pk} 所对应的插入位置记入 s;}

Step5. 将 T_i 插入到第 P_i 级优先级队列的第 s 位置;

Step6. for(int $j=0$;$j<m$;$j++$)

{$PriorTaskArray[P_i].resource[m]+=r_{ij}$;}

/* 更新第 P_i 级优先级队列的各维资源量,r_{ij} 为任务 T_i 的第 j 维资源 */

Step7. 算法结束.

当调度任务执行时也要更新各维资源量,过程与 Step6 类似,在此不再赘述.

5 双最小均衡算法的设计与实现

5.1 前端分发器

基于上面的 Web 集群体系框架,前端分发器与我们在文献[8]中提出的前端分发器基本相同,只是增加了我们提出的双最小均衡调度算法的请求分配调度策略第一部分算法.

5.2 优化调度系统结构

为了使后台服务器支持选择调度算法,我们首先修改 Web 服务器代码,使其支持我们的调度策略.

我们修改了 apache1.3.14 服务器代码[10]使其支持区分服务,主要修改了 http_core.c 和 http_protocol. c 文件. 如图 3 所示:系统在 80 端口监听请求的到来.

(1) 当连接建立阶段,请求 socket 被赋予最高级别以保证所有的请求连接都被系统接受;

(2) 当连接建立后,先根据区分服务的原则,确定请求的优先级. 然后根据资源均衡原则插入到相应的请求优先级队列中.

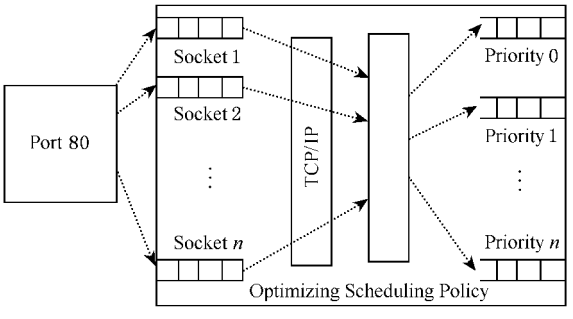

Fig. 3 Back-end server priority schedule structure.

图 3 后台服务器优先级调度结构图

6 实验结果

6.1 测试环境

由于对双最小均衡调度算法进行理论解析比较困难,但在实际的环境中,Web 请求的各维资源不容易取得,为了证实双最小均衡调度策略的有效性,我们采用了大量的模拟数据进行了仿真. 测试数据按如下条件设定:

(1) 请求任务按指数分布产生.

(2) 调度对象模型. 为了便于分析,我们对调度对象做一些假设:我们仅对影响系统性能最大的 3 种资源:CPU、内存、I/O进行分析,请求任务我们用模拟数据表示,这样可以排除其他因素影响,清楚地对比算法的优劣.

模拟数据按以下原则产生:系统资源假设 CPU、内存、I/O 各为 1000 个单位;对于静态页面,我们先按大(>500KB,占 15%),中(<500KB 并且 >20KB,占 70%),小(<20KB,占 15%)随机产生. 对于 CPU 我们按文件大小按如下公式计算:$1+0.01\times Filesize$. 对于 I/O 按如下公式计算:$1+0.2\times Filesize$,内存按 $50+1\times Filesize$. 对于动态页面我们要给出两个指标:一是 CPU 指标,二是 I/O 指标. 我们分为如下几种:高 CPU,低 I/O 占 15%;高 CPU,高 I/O 占 15%;中 CPU,中 I/O 占 70%.

6.2 双最小均衡 Web 集群调度算法的有效性测试

在给出对比结果之前,先给对比的指标下一个明确的定义.

定义 4. 队列关键资源最长完成时间(L_{resouce}):在一个队列的调度中,队列关键资源最长完成时间为 $L_{\text{resouce}}=\max(\sum q_i/C_i)$,其中 $\sum q_i$ 是队列中某维资源的和,C_i 是此服务器处理此维资源的能力.

定义 4 表明任何调度算法最好完成时间不可能好于队列关键资源最长完成时间(L_{resouce}). 因此我们将此 L_{resouce} 执行时间与其他算法的执行时间的比值称为算法优化率 O_{ratio}. O_{ratio} 越接近 1 表明优化算法越接近最优值.

(1) 与其他算法的对比情况

图 4 表明了 FCFS(先来先服务)、SS(分离式调度)、DMB(双最小资源优化调度)3 种算法在不同请求率下的每个连接平均所需执行时间对比,从图 4 中可以看出我们的算法最好,FCFS 算法最差.

图 5 表明了 FCFS,SS,DMB 三种算法在不同请求率下的 O_{ratio},结果对比表明:DMB 算法比分离式调度算法平均提高效率达 11%.

(2) 用户响应时间以及对区分服务的支持

用户的响应时间是重要的性能指标,图 6 表明 DMB 算法在不过载情况下高优先级任务与低优先

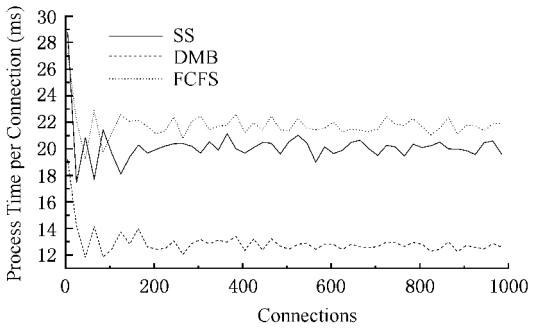

Fig. 4 Comparison of average process time of FCFS,SS,DMB.

图 4 FCFS,SS,DMB 三种算法的平均执行时间对比

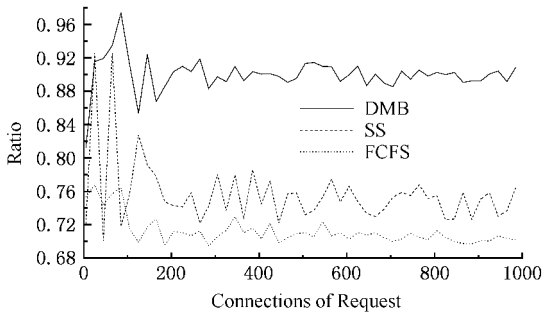

Fig. 5 O_{ratio} of FCFS,SS,DMB.

图 5 FCFS,SS,DMB 三种算法的 O_{ratio}

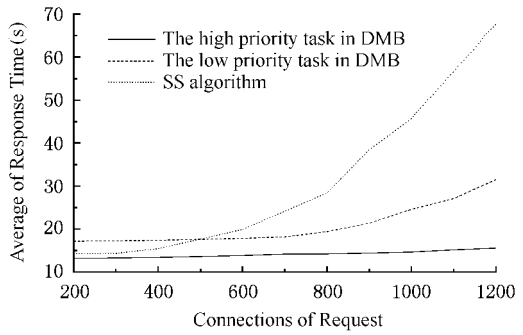

Fig. 6 Mean response time of different priority task of several algorithms.

图 6 不同算法的不同优先级任务的平均响应时间

级任务的平均响应时间与分离式调度算法大致相同,但在负载重时,分离式调度算法响应时间下降较快,而 DMB 算法中的高优先级任务仍然保持较快的响应时间,而低优先级任务响应时间急剧降低.

图 7 表明了不同优先级下的请求拒绝率,图 7 中表明在负载较低时不同优先级的拒绝率都很低,但随着请求数增多,低优先级请求拒绝率很快上升,而高优先级拒绝增长较慢,表明了我们算法对区分服务的支持,即在过载时低优先级任务得不到执行.

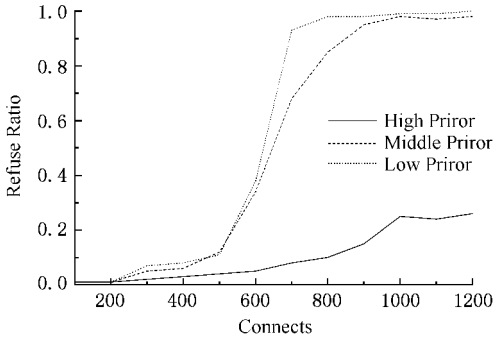

Fig. 7 Ratio of request of different priority.

图 7 不同优先级下的请求拒绝率

(3) 请求分配算法的有效性测试

请求分配的目的是将请求按资源情况‘均衡’地分到各后台服务器中去,图 8 结果表明,我们的请求分配算法资源利用率都超过 95%,它是在后台服务器为 8 台,资源维数为 6 的情况下的各台服务器各维资源分配情况.

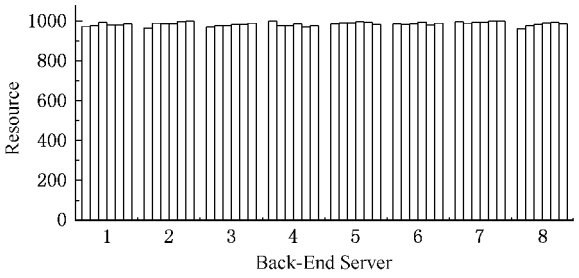

Fig. 8 Resource allocation utilization of request allocation algorithm.

图 8 请求分配算法的资源分配利用率

图 9 表明请求分配算法在不同资源粒度情况下的资源利用率,结果表明资源粒度越大,利用率降低.

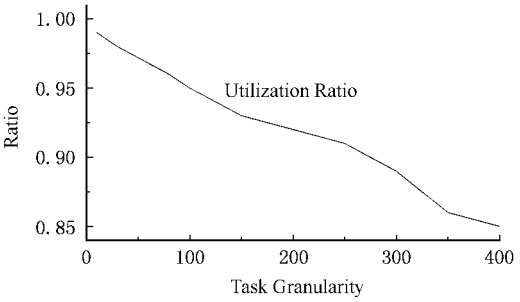

Fig. 9 Resource utilization ratio of different task granularity.

图 9 不同任务粒度下的资源利用率

(4) 选择算法的有效性测试

不同的线程处理数目,系统优化效率不同,当线程数为 3 和 4 时系统效率基本相同,当线程数为 5

时系统效率呈下降趋势,结果如图 10 所示:

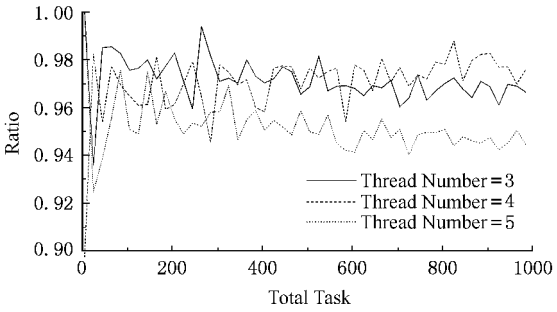

Fig. 10 Ratio in different schedule thread.

图 10 不同调度线程下的 O_{ratio}

在相同的线程数目(线程数为 3)下,我们的算法与理想最优情况执行时间对比如图 11 所示,表明我们的算法与最优情况比较接近.

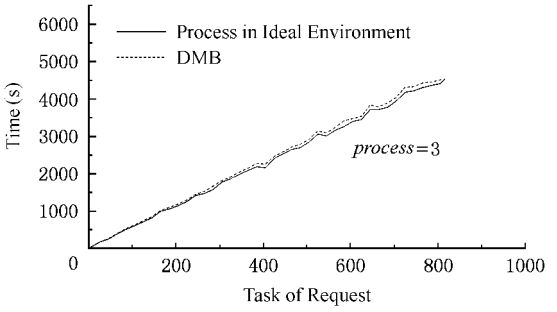

Fig. 11 Comparison of process time of ideal algorithm when scheduled thread is 3.

图 11 调度线程为 3 时与理想算法执行时间对比

同样,请求任务的粒度对调度的影响如图 12 所示,在图 12 中给出的条件是:粒度在 1~20;1~30;1~40,调度线程个数为 3 个. 结果表明请求粒度越大,我们算法的 O_{ratio} 呈下降的趋势.

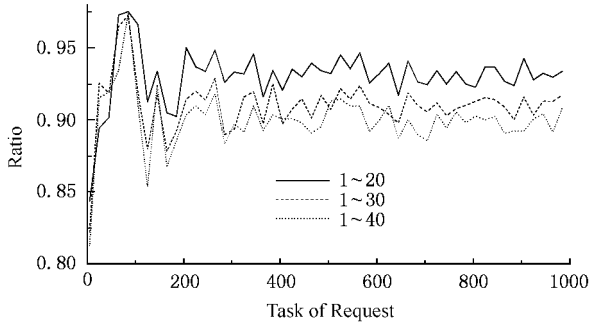

Fig. 12 Ratio of several algorithms in different granularity.

图 12 不同任务粒度下算法的 O_{ratio}

7 结 论

本文在一种新的 Web 服务器集群的基础上,提出了以资源优化为目标的请求分配与选择的双最小均衡算法,并通过实验与其他算法进行了比较,证实了 DMB 算法的有效性与普遍性.

参 考 文 献

1 V. Cardellini, M. Colajanni, P. S. Yu. The state of the art in locally distributed Web-server systems. ACM Computing Surveys, 2002, 34(2):1~49

2 M. Colajanni, P. S. Yu. Dynamic load balancing in geographically distributed heterogeneous Web servers. http://dlib.computer.org/conferen/icdcs/8292/pdf/82920295.pdf, 1998

3 E. Katz, M. Butler, R. McGrath. A scalable Web server: The NCSA Prototype. Computer Networks and ISDN Systems, 1994, 27(2):155~164

4 H. Zhu, B. Smith, T. Yang. Scheduling optimization for resource-intensive Web requests on server clusters. ACM Symposium on Parallel Algorithms and Architectures, Saint-Malo, 1999

5 D. Andresen, T. Yang. Multiprocessor scheduling with client resources to improve the response time of WWW applications. The 11th ACM SIGARCH Int'l Conf. Supercomputing (ICS'97), Vienna, Austria, 1997

6 Aron M. Gaurav Banga, Jeffrey C. Mogul. Resource containers: A new facility for resource management in server systems. The 3rd Symposium on Operating Systems Design and Implementation (OSDI'99), New Orleans, LA, 1999

7 Mohit Aron, Darren Sanders, Willy Zwaenepoel. Scalable content-aware request distribution in cluster-based network servers. The 2000 Annual Usenix Technical Conf., San Diego, CA, 2000

8 Jin Liang, Chen Zhigang, *et al*. Design of scalable content-aware Web cluster with transparency. Computer Engineering, 2004, 30(7):101~103 (in Chinese)
(金亮,陈志刚,等. 一种基于内容的可扩展性透明 Web Cluster 体系结构的设计. 计算机工程, 2004, 30(7):101~103)

9 LoadRunner. http://www.Mercuryinteractive.fr/products/loadrunner/, 2003

10 Mor Harchol-Balter, *et al*. Size-based scheduling to improve Web performance. ACM Trans. Computer Systems, 2003, 21(2):207~233

Liu Anfeng, born in 1971. Ph. D. candidate. His research interests include Web cluster architecture with high performance, and grid Web services.
刘安丰,1971 年生,博士研究生,主要研究方向为高性能 Web 集群系统、网格 Web services.

Chen Zhigang, born in 1964. Ph. D., professor and Ph. D. supervisor. His research interests include network processing and distributed computing.
陈志刚,1964 年生,博士,教授,博士生导师,主要研究方向为网络计算与分布式处理.

Zeng Zhiwen, born in 1970. Ph. D. candidate. His main research interest is grid computing.

曾志文,1968 年生,博士研究生,主要研究方向为分布式计算.

Long Guoping, born in 1982. Master candidate. His main research interest is computer architecture.

龙国平,1982 年生,硕士研究生,主要研究方向为计算机体系结构.

Research Background

This paper is partly supported by the National Research Foundation for the Doctoral Program of Higher Education of China project (The study of key techniques that are applied to the APN and heterogeneous system oriented DAG schedule theory (project No. 20040533036)) and the Natural Science Foundation of Hunan Province of China Project (The study of the control and optimism of Web cluster QoS (project No. 03JJY4054)). The main goal is to study the QoS control, task scheduling, load balancing, and so on).

中国计算机学会电子政务与办公自动化专委会
第 3 届全国 Web 信息系统及其应用学术会议(WISA2006)
征文通知

Web 在人类信息的存储和交换过程中发挥着日益重要的作用. 适用于网络平台、动态特性的 Web 技术层出不穷,提高办公效率、节约资源消耗和扩大信息共享的 Web 应用和服务蓬勃发展. 但随着 Web 规模的不断膨胀,Web 上数据资源以爆炸性的趋势飞速增长,而且信息的结构和内容越来越复杂,使得管理、查询和使用 Web 信息变得愈加困难.

全国 Web 信息系统及其应用会议(WISA)是中国计算机学会电子政务与办公自动化专委会主办的系列会议. 首届会议 WISA2004 于 2004 年 10 月在武汉圆满召开,会议共收到应征论文 368 篇,其中 54 篇论文在《武汉大学学报(英文)》(EI 源刊)作为正刊专辑发表(已经全部被 EI 收录). 第 2 届会议 WISA2005 于 2005 年 9 月在沈阳召开,会议共收到应征论文 606 篇,录用 239 篇,其中 62 篇论文在《武汉大学学报(英文)》(EI 源刊)作为正刊专辑发表,118 篇在《计算机科学》发表,59 篇在由清华大学出版社出版的会议论文集《Web 信息系统与技术》上发表.

WISA2006 将于 2006 年 10 月在南京召开. 会议将继续这一良好的传统,在 Web 技术、信息系统、电子政务与办公自动化等方面进行深入广泛的学术交流. 会议论文集仍将分两部分出版,录用论文中将选择 60 篇左右高水平论文以英文方式继续由《武汉大学学报(英文)》(EI 源刊)正刊专辑出版,中文论文集将由著名计算机核心期刊《计算机科学》专刊和中央级出版社出版. 会议期间除进行会议论文交流外,还将邀请著名学者作特邀报告. 本次会议仍将评选大会优秀学生论文.

征文范围(包括但不限于)

Web 信息挖掘与检索	语义 Web 与智能 Web	Web 站点逆向工程与维护技术
Web 测试与 Web 应用的质量保证	Web 与网格计算	多媒体数据管理
Web 与数据库技术	工作流模型	XML 与半结构化数据管理
组件与中间件技术	Web 信息系统环境与基础	代理技术及信息管理
Web 应用框架和体系结构	自动文本索引与分类技术	Web 与信息系统安全性
决策支持与分析技术	Web 信息系统开发工具	电子政务与电子商务框架及应用
Web 系统度量与分析技术	电子政务与办公自动化发展现状与趋势	

来稿要求

(1) 本次会议只接受 Email 投稿.

(2) 中英文稿均可,一般不超过 6000 字,为了便于出版论文集,来稿必须附中英文摘要、关键词、资助基金与主要参考文献,注明作者及主要联系人姓名、工作单位、详细通信地址(包括 Email 地址)与作者简介. 稿件要求采用 Word 或 PDF 格式.

联系信息

投稿地址:东北大学信息科学与工程学院 王国仁(wanggr@mail.neu.edu.cn)

会务情况:东南大学计算机科学与工程系 徐宝文 许蕾(xlei@seu.edu.cn)

大会网站:http://www.neu.edu.cn/wisa2006/

重要日期

征文截止日期:2006 年 3 月 25 日

录用通知发出日期:2006 年 4 月 15 日

正式论文提交日期:2006 年 4 月 30 日