一种基于内容的Web集群系统负载均衡算法*)

李双庆　程代杰　何　玲
(重庆大学计算机科学与工程学院　重庆400044)

A Content-Aware Load Balancing Algorithm in Web Cluster System

LI Shuang-Qing　CHENG Dai-Jie
(School of Computer Science and Engineering, Chongqing University, Chongqing 400044)

Abstract　Web Cluster is an effective mechanism used in Web site construction to deal with the system capacity problem. Researchers proposed several strategies or algorithms, which improve the performance and scalability of the Web cluster system. In this paper, a content-based load distribution algorithm is proposed. It takes the processing ability of back-end severs and the request load weight into account, and ensure the request locality. The emulation results illustrate that this algorithm performs better in different kinds of Web site, comparing with other relative algorithms.
Keywords　Load balancing, Web cluster, Dispatching algorithm

1. 引言

Internet的普及为人们的工作带来许多方便，人们通过它获取信息或者开展电子商务活动。一个成功的Web站点面临不断增长的访问量和日益复杂的内容处理的挑战。这一切似乎都集中到如何扩充Web站点的处理能力上。早期依靠更高处理性能的服务器系统来解决问题的办法显得笨拙而昂贵。人们开始寻找更加灵活而廉价的技术手段。一些站点采用镜象(mirroring)的方式，在多个服务器上复制相同信息，以不同的URL供用户选择。但是这种方式对用户缺乏透明性，且系统无法控制用户访问，不能达到负载均衡分配的目的。

更有效的办法是在一个基于LAN的分布式体系结构下实现负载均衡，所有来自客户端的请求被透明地分配到若干服务器上。对用户而言，整个分布式系统仿佛是一台单一的逻辑服务器，我们称之为Web服务器集群(Web cluster)。这样的集群系统能够提供较强的可扩展性和较好的吞吐性能。从商业角度而言，不仅可以保护原来的投资，而且也可以通过廉价的集群系统获得高性能计算机所能达到的处理能力。当然，要实现这样的集群系统，必须面临许多技术上的挑战，已有一些研究人员在该领域进行了研究，并取得了一定的进展[3,4]。

Web服务器集群的负载均衡策略有两大类：一类是基于DNS的分配，如RR-DNS和WRR-DNS，它们在标准DNS的基础上扩充动态映射的功能[11,12]。其负载均衡能力有限，且对用户端不透明，本文不对其进行讨论；另一类是基于IP/TCP/HTTP重定向的分配[6~9]。一般需要一个特殊的前端节点，分配器(dispatcher)。所有的客户端请求都经过分配器并由它分配到后端服务器处理。本文将讨论此类负载均衡策略。

2. 基于分配器的请求分配机制

基于分配器的请求分配机制一般要求对客户端透明。目前主要有两种机制[5]，一种称为中继机制(relaying)，如图1a所示，客户端请求到达分配器后，由分配器按一定的负载分配算法，将请求传递给后端被选中的服务器。服务器处理后的结果传回至分配器，再由分配器转发给客户端。分配器的工作通常在操作系统的应用层完成。也有修改操作系统核心直接支持中继机制的系统，其性能会好一些，这种优化方法称为TCP衔接(TCP splicing)；另外一种机制称为TCP传递(TCP Handoff)，如图1b所示，客户端的请求经过分配器分配到达后端服务器，服务器处理后，将结果不经过分配器而直接发送给客户端。

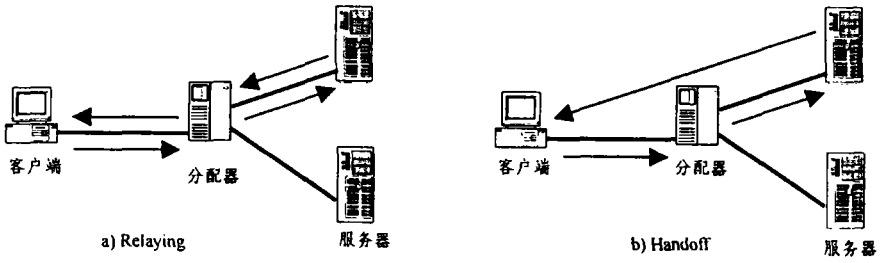

图1　基于分配器的请求分配机制

中继或TCP衔接机制要求所有的通信均要经过分配器(特别是处理结果信息量更大)，因此容易形成通信瓶颈；TCP传递机制避免了这一问题，因此性能更好，但是需要修改前端和后端节点的网络部件，以支持TCP Handoff协议。

3. 现有负载分配算法

下面分析两种与本文直接相关的两种算法：CAP和LARD。

*)项目资助：重庆市科技攻关项目(99.6715)。李双庆　博士研究生，主要研究方向：电子商务，多代理技术，分布式处理。程代杰　教授，博士生导师，主要研究方向：电子商务，计算机网络，并行处理。何　玲　讲师。

3.1 CAP 算法

CAP(Client-Aware Policy)[2]算法基于客户端信息进行负载分配。它面向逻辑同构的 Web 服务器集群系统，即集群中各服务器的内容可通过某种机制，如复制操作，保持一致。CAP 将 Web 服务分成四种类型：

1)发布型：主要提供静态信息(如 HTML 页面及嵌入的对象)。

2)事务型：结果主要来自动态数据库查询，且查询条件一般由用户通过一个 HTML 页面动态提供。由于它需要进行密集的磁盘访问，因此也称为磁盘密集型服务。

3)电子商务型：提供静态、动态和安全信息传输，面向电子商务应用。在进行安全信息传输时，加解密操作需要消耗大量的 CPU 资源，数据库访问时需要密集访问磁盘，因此也称为磁盘/CPU 密集型服务。

4)多媒体型：提供实时的音频和视频服务。此类服务在 Web 站点中一般通过特殊的服务器和网络协议连接，因此，CAP 没有对此类型服务进行讨论。

CAP 的核心思想是通过分配器识别客户端请求的类型，然后将同类请求均匀地分配给后端服务器。这样，每个服务器得到的各类请求量大致相同，从而达到负载基本均衡。CAP 算法不需要获取服务器端状态，算法开销较小。

CAP 算法的不足之处是忽略了内存 cache 命中率对整个系统性能的影响，因此，尽管可以保证负载均衡，但对整个系统的吞吐能力和响应性能提高不大。

3.2 LARD 算法

LARD(Locality-Aware Request Distribution)[1]是一种基于客户端内容和服务器状态的负载分配算法，其核心思想是对集群中的服务器进行服务分区，让每台服务器对某些类型的请求进行响应。这样，相同请求提交给同一台服务器处理，只有当负载明显不均衡时，才对请求进行再分配。这样可以获得较高的 cache 命中率，从而提高整个集群系统的吞吐率和响应性能。

在 LARD 算法中，分配器可以监控每台后端服务器的活动连接数，以衡量其负载情况。后端服务器的内容是同构的，每台后端服务器都具备响应各类请求的能力，但分配器也许只要求每台后端服务器响应某一类或某几类请求。

LARD 算法的不足之处是简单地将服务器的活动连接数作为度量服务器负载的指标，而不能区分不同连接请求的负载差异，因此在负载代价差异较大的系统中，该算法的负载均衡能力较差，导致响应性能下降。

4. CAWLB 算法

我们在分析了 CAP 和 LARD 算法的特点后，提出一种改进的 CAWLB(Content-Aware Weighted Load Balancing)算法。该算法吸取了前两种算法的优点，同时通过一定的方法，弥补了各自的不足之处。

4.1 算法思想

1)采用加权连接代价度量服务器负载状态。由 CAP 算法对 Web 服务的分类可知，不同类型的服务所消耗的系统资源(如内存、CPU 时间和磁盘访问时间等)存在很大的差异。以文[2]中的安全负载模型为例，在直接命中内存且信息传输率为 100Mbps 的情况下，若采用 256 位 RSA、Triple-DES 和 MD5算法加密后的信息吞吐率分别为38.5Kbps、4.69Mbps 和33.1Mbps。可见，不能简单用服务器的活动连接数来度量服务器负载状态，而应根据请求类型对每个请求加权，以获得服务器更准确的负载状态。

2)保证局部性(locality)。算法通过一个 hash 算法将页面请求的 URL 转换为一定长度，如32位的编码。绝大多数情况下，不同页面请求会映射为不同编码值。分配器将同一编码值的请求分配到同一台(或组)服务器处理，可以获得较高的 cache 命中率。

3)适应后端服务器异构性。算法允许后端服务器具有不同的 CPU 处理能力、内存容量和磁盘访问速度。相同负载量情况下，异构服务器响应性可能会有明显的差异。算法采用相对负载量来度量服务器的负载状态，能够在异构型的服务器集群中均衡负载。

4.2 算法

设集群服务器集合 $S=\{s_1,s_2,\cdots,s_m\}$，请求集合 $R=\{r_1,r_2,\cdots,r_n\}$，请求编码集合 $H=\{h_1,h_2,\cdots,h_l\}$，请求类型集合 $T=\{t_1,t_2,\cdots,t_j\}$，请求处理代价集合 $W=\{w_1,w_2,\cdots,w_j\}$。

定义1(请求编码函数 $H(r)$) 对请求集合 R，请求编码集合 H，存在散列函数 $H(r)$，有：

$\forall r\in R,\exists h\in H$，使得 $h=H(r)$，且 $\forall r1,r2\in R,r1\neq r2$，有 $H(r1)\neq H(r2)$，则 $H(r)$ 为请求编码函数。

定义2(集群能力向量 C) 对集群服务器集合 S 中任意服务器 s_i，其处理能力为 c_i，有向量 $C=\{c_1,c_2,\cdots,c_m\}$，其中 $i=1,\cdots,m$，m 为集群中服务器数量，则称 C 为集群系统的能力向量。

定义3(请求分类函数 $T(t)$) 对请求编码集合 H，请求类型集合 T，有：

$\forall h\in H,\exists t\in T$，使得 $t=T(h)$，则称 $T(t)$ 为请求分类函数。

定义4(请求类型加权函数 $W(t)$) 对请求类型集合 T，请求处理代价集合 W，有：

$\forall t\in T,\exists w\in W$，使得 $w=W(t)$，则称 $W(t)$ 为请求类型加权函数。

定义5(服务器相对负载函数 $Q_i(t)$) 集群中服务器 $s_i,i=1,\cdots,m$，在 t 时刻有 n 个活动连接，对应请求为 $r_i0,r_i1,\cdots,r_in-1$，有：$Q_i(t)=\sum_{k=0}^{n-1}W(T(H(r_{ik})))/c_i$，则称 $Q_i(t)$ 为服务器相对负载函数。

根据上述定义，我们给出改进的负载分配算法 CAWLB：

```
fetch next request r;
h = H(r);
if serverSet(h)= null then //首次分配
    n, serverSet(h)= { server with least Q };
else {
    n= { server with least Q in serverSet(h)};
    m= {server with highest Q in serverSet(h)};
    if(n.load>T_high&&∃ s_i∈S,Q_i<T_low)||n.load≥2T_high then {//
       负载再分配
       p={server with least Q};
       add p to serverSet(h);
       n=p;
    }
    if|serverSet(h)|>1&&time()-serverSet(h).lastMod>K then
       remove n from serverSet(h);
}
send r to n
Modify Q of n
if serverSet(h)changed in this iteration then
    serverSet(h).lastModi = time();
}
```

下面对算法进行说明。我们采用了度量服务器负载状态的两个门限参数 T_{high} 和 T_{low}。如果服务器负载量高于 T_{high} 表示负载较重，需要负载再分配。如果服务器负载量低于 T_{low} 表

示负载过轻。如果分配器不限制集群中的连接总数，当所有服务器的负载量均超过$2T_{high}$，系统相当于采用 WRR 算法，响应性能将明显下降。CAWLB 中分配器允许的最大负载量为$S=(n-1)*T_{high}+T_{low}-1$，其中 n 为服务器数量。这保证了当没有服务器负载量$\leqslant T_{low}$时，最多有 n－2台服务器负载$\geqslant T_{high}$。当负载再分配时，目标服务器的负载差至少为$T_{high}-T_{low}$，最大为$2T_{high}-T_{low}$。

显然设置T_{high}和T_{low}的值对算法十分重要。T_{low}根据服务器的处理能力确定，处理能力越强，取值越大，反之则越小。$T_{high}-T_{low}$的取值与响应的局部性和延迟差相关。取值过大，则局部性好而延迟差大，取值过小，则局部性差而延迟差小。因此，$T_{high}-T_{low}$的取值不能过高或过低。在给定T_{low}后，如果期望的最大延迟差为 D，平均请求服务时间为 R，则T_{high}的取值为$(T_{low}+D/R)/2$，且满足$T_{high}>T_{low}$。在仿真实验中，典型的取值是$T_{high}=475$和$T_{low}=75$。

此外，算法还考虑了同一类 URL 请求在一台服务器处理不能满足响应要求的情况，将一类请求映射给一个服务器集(server set)。服务器集中的服务器数量有1到 n 台，并且根据负载情况动态变化。当某请求所对应的服务器集中所有服务器都过载时，分配器给该服务器集增加一台服务器；如果经过时间 K 服务器集中没有发生服务器的增减，则减少其中一台服务器。为防止系统出现抖动现象，K 值设置不应过小。在仿真实验中，K＝15秒。服务器集是一种逻辑分配，同一台服务器同时被分配到一个或多个服务器集中。

4.3 仿真实验

为了研究不同负载均衡策略在不同集群规模、不同 CPU 处理速度和内存容量情况下的系统特性，我们设计开发了 Web 服务器集群仿真器。仿真器对网络本身的传输带宽和特性不作限制，且以 450MHz Pentium II 微机运行 FreeBSD 2.2.5获得的测试数据[10]为依据，建立服务器集群处理和响应模型。当文件大小为17kB 时，全部命中内存 cache 的吞吐率为1000req/sec。cache 替换算法采用 LRU 算法，cache 大小为16MB。对动态页面文件，不允许内存 cache，并假设磁盘文件的访问速度为内存访问速度的1/10。对磁盘密集的访问，例如数据库查询操作，其速度根据结果数据大小确定，我们设定为同等大小内存文件访问速度的1/100。对系统加密处理的开销，不同的加密算法开销差异较大，我们假设采用 RSA 和 Triple-DES 算法，内存文件访问速度下降为原来的1/100。

同时我们建立了三种典型 Web 站点的负载模型：

1)发布型站点：所有页面请求均为静态请求。

2)事务型站点：包含75% 的静态请求和25%的动态请求。

3)电子商务型站点：包含60% 的静态请求、20%磁盘密集的动态请求、15%CPU 密集型请求以及5%CPU/磁盘密集型请求。

此外，整个站点的规模为20000个 URL。整个负载模型中90%的请求会访问集中在整个站点10%的 URL 上。请求结果页面的大小满足正态分布，且平均大小为17.3kB。

在以上模型的基础上，我们进行了仿真实验，并获得以下结果(图2～图4)。

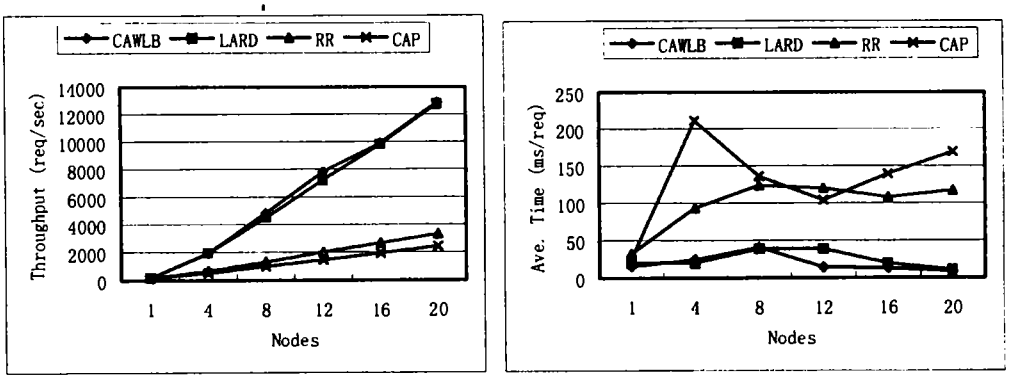

图2 发布型站点仿真结果

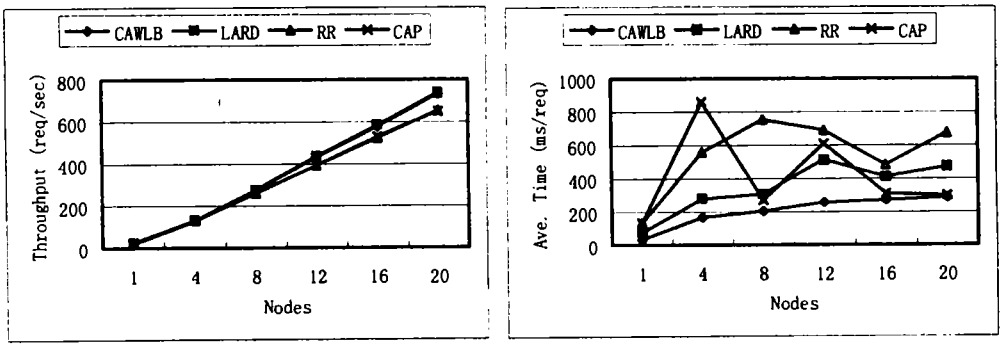

图3 事务型站点仿真结果

(下转第163页)

Software Architecture with the Chemical Abstract Machine. http://www.ics.uci.edu/~djr/rosatea/attendees.html. pp. 1~4
3 Stafford J A, et al. Chaining: A Software Architecture Dependence Analysis Technique: [Technical Report CU-CS-845-97]. University of Colorado, Sep. 1997. 1~12
4 Bertolino A, et al. Deriving Test Plans from Architectural Descriptions. In: ACM Proc. Int. Conf. On Software Engineering (ICSE2001), June 2000. 220~229
5 Harrold M J. Architecture-Based Regression Testing of Evolving System. http://www.ics.uci.edu/~djr/rosatea/attendees.html. pp. 1~5
6 Young M. Testing Complex Architectural Conformance Relations. http://www.ics.uci.edu/~djr/rosatea/attendees.html. pp. 1~4
7 Anderson S O, et al. Type System, Software Architecture and Testing. http://www.ics.uci.edu/~djr/rosatea/attendees.html. pp: 1~4
8 Bertolino A, et al. Reaction Graphs for the Testing and Analysis of Software Architecture. http://www.ics.uci.edu/~djr/rosatea/attendees.html. pp. 1~6
9 Rosenblum D S. Challenges in Exploiting Architectural Models For Software Testing. http://www.ics.uci.edu/~djr/rosatea/attendees.html. pp. 1~4
10 Richardson D J, Stafford J A, Wolf A L. A Formal Approach to Architecture-Based Software Testing. National Science Foundation Research Proposal. pp. 1~24
11 郑人杰. 计算机软件测试技术. 清华大学出版社, 1992
12 周芝英, 等. 计算机软件测试技巧. 清华大学出版社, 1985

(上接第140页)

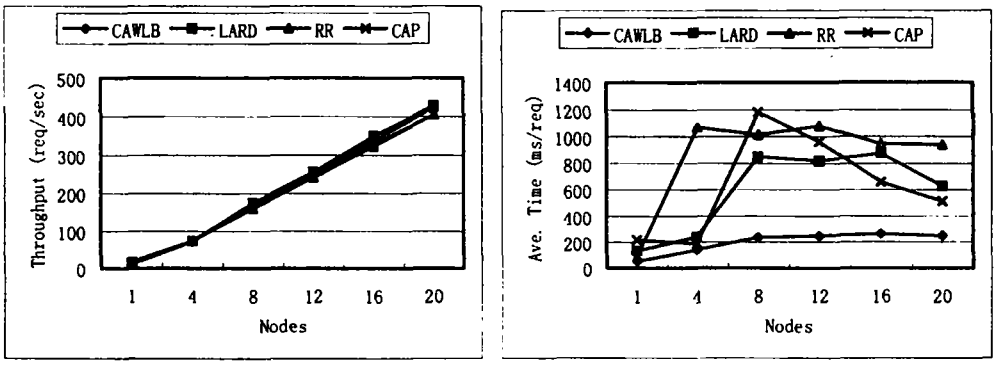

图4 电子商务型站点仿真结果

从仿真结果可以看出 LARD 与 CAWLB 算法对发布型站点在吞吐率和平均响应时间上有明显优势；对事务型或电子商务型站点，各种算法的吞吐率差异较小，但平均响应时间指标上 CAWLB 和 CAP 算法存在明显优势。综合以上仿真结果可知，CAWLB 对不同类型的站点，其吞吐率和平均响应时间均有优势。

总结 本文在分析了基于内容分配的 Web 集群算法 LARD 和 CAP 之后，提出了一种改进的负载分配算法 CAWLB。该算法根据请求内容分配后端服务器，可以获得较高的 cache 命中率，同时，根据不同请求类别的负载估算，计算后台服务器的负载量，为请求再分配提供了较准确的依据。另外，算法具有对服务器处理能力异构情况的适应性。仿真实验表明，该算法与其它相关算法相比，在系统吞吐率和平均响应时间等指标上存在优势，算法具有较好的实用价值。在实际应用中，不同负载的权重可以通过测试或分析 Web 日志文件获得。并且，可以考虑通过 ASIC 实现算法获得更快的执行速度。

参 考 文 献

1 Vivek S, Mohit A. Locality-Aware Request Distribution in Cluster-based Network Servers. In: Proc. of ASPLOS-VIII, ACM SIG-PLAN, 1998. 205~216
2 Emiliano C, Michele C. A Client-Aware Dispatching Algorithm for Web Clusters Providing Multiple Services. In: Proc. of 10th Int'l World Wide Web Conf. Hong Kong, May 2001
3 Trevor S, Steve G, Byrav R. Scalable Web Server Clustering Technologies. IEEE Network, 2000, 14(3): 38~45
4 Valeria C, Michele C, Philip S Y. Dynamic Load Balancing on Web-Server Systems. IEEE Internet Computing, 1999, 3(3): 28~39
5 Mohit A, Darren S, Peter D, Willy Z. Scalable Content-aware Request Distribution in Cluster-based Network Servers. In: Proc. of the 2000 Annual Usenix Technical Conf. San Diego, CA, June 2000
6 Ludmila C, Magnus K. Scalable Web Server Cluster Design with Workload-Aware Request Distribution Strategy WARD. In: Proc. of the 3rd Intl. Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems, Milpitas, CA USA, June 2001
7 Ludmila C. FLEX: Load Balancing and Management Strategy for Scalable Web Hosting Service: [HP Labs Technical Reports, HPL-1999-64R1, 991117]
8 Kai S, Tao Y. Cluster Load Balancing for Fine-grain Network Services. Accepted for publication at International Parallel and Distributed Processing Symposium (IPDPS'02), Fort Lauderdale FL, April 2002
9 Yong M, Rassul A. Comparison of Load Balancing Strategies on Cluster-based Web Servers. http://www.comp.nus.edu.sg
10 The Workload for the SPECweb96 Benchmark. http://www.specbench.org/osg/web96/workload.html
11 Srisuresh, Gan. Load Sharing using IP Network Address Translation (LSNAT), RFC2391, Aug. 1998
12 Brisco T. DNS Support for Load Balancing. RFC 1794, April 1995