

基于混合策略的集群负载均衡算法研究

郑 祺

(浙江科技学院 信息与工程学院,杭州 310023)

摘 要: 传统的负载均衡策略仅对某些类型的站点有效,而不能满足各种站点类型的要求。在分析已有方法的基础上,针对不同类型的用户请求特点,提出了一种基于混合策略的负载均衡算法。该算法通过对用户请求的分类调度来获得较高的 cache 命中率,同时还引入了会话保持技术和反馈环节来解决会话失效和负载不均衡的问题。实验结果表明,这种该算法能有效提高集群系统的整体性能。

关键词: 集群;负载均衡;内容分类;动态反馈;会话保持

中图分类号: TP393.02

文献标志码: A

文章编号: 1671-8798(2013)05-0361-05

Research on combined load balancing algorithm in cluster

ZHENG Qi

(School of Information and Electronic Engineering, Zhejiang University of
Science and Technology, Hangzhou 310023, China)

Abstract: Traditional strategies are just effective to some kinds of Web sites, but not suitable for all. Based on the existing algorithms and the features of different kinds of users requests, a combined load balancing algorithm is proposed. By using different methods to dispatch users requests, a higher cache hit rate is obtained. Furthermore, a session maintenance technology and feedback are brought in to solve the problem of session ineffectiveness and load imbalance. Experiment results show this algorithm can improve the performance of the cluster.

Key words: cluster; load balancing; contentclassification; dynamic feedback; session maintenance

随着 Internet 及媒体技术的飞速发展,网络访问量和数据流量也快速增加,致使网络服务器的处理能力成为网络访问的新瓶颈,仅靠价格昂贵的高性能主机已无法从根本上满足网络服务快速发展的需要。集群以其高可靠性、高可扩展性和高性价比,为网络服务器带来了新的解决方案^[1]。集群负载均衡算法是提高集群整体性能的关键,其根本目的是按照节点服务器(简称节点)的性能来分配对应的任务,以使应用程序的执行时间最小化,从而最大限度地利用各节点的处理能力达到提高集群系统整体性能的目的。

收稿日期: 2013-07-12

作者简介: 郑祺(1974—),男,江苏省无锡人,讲师,博士研究生,主要从事计算机网络并行处理研究。

1 算法研究现状

随着集群的广泛应用,Web 服务器的负载均衡算法已成为研究的热点。Web 服务器的负载均衡算法主要分为两大类:一类是静态均衡算法,如最小连接数法(Least-connection)、轮转法(Round-robin)等,这类算法均未考虑集群内各节点间的性能差异及用户请求任务间的差异,无法有效地解决集群负载均衡问题;另一类是动态均衡算法,其主要优点是能够根据各节点的实际性能和负载状况动态分配对应的任务,可以较好地解决集群负载均衡问题。目前的研究主要围绕动态负载均衡算法展开,其中基于内容的负载均衡算法是研究的热点之一。典型的算法包括 CAP(Client aware policy)^[2]、LARD(Locality aware request distributes)^[3]、IQRD(Intelligent queue-based request dispatcher)^[4]和 LOCEP(Lexicographic ordering and content equally partitioning)^[5]等。

CAP 是一种基于用户请求内容的负载均衡算法,其核心思想是通过均衡器识别用户请求的类型,然后再将同类型的请求平均分配给各节点,以使各节点获得的各类请求数量大致相同,从而实现负载均衡。CAP 算法的不足之处是未考虑集群内各节点的性能差异及实际负载情况,且忽略了内存 cache 命中率对整个集群系统性能的影响。

LARD 也是一种基于请求内容的负载均衡算法,但它引入了节点负载状态的概念,其核心思想是先对集群内的所有节点进行服务分区,每个节点只会对某些特定类型的用户请求进行响应,均衡器在调度用户请求时会尽量将相同的请求分配给同一节点处理,从而获得较高的 cache 命中率,提高集群的整体性能。只有当节点负载状态出现明显不平衡时,均衡器才会对用户请求进行再次分配。LARD 算法的缺点是计算量较大且仅简单地使用各节点的活动连接数来度量节点负载状况,而未考虑到不同类型的用户请求之间存在的巨大负载差异,容易出现节点负载失衡情况,算法的负载均衡能力较差。

2 CCAP 算法

在深入研究上述几种集群负载均衡算法的基础上,笔者提出一种基于混合策略的动态负载均衡算法 CCAP(Combined content-aware policy),该算法能够较好解决各种不同负载状况下的负载均衡问题。

算法思路:均衡器收到用户请求后,首先根据其 URL 进行分类,然后按类型发送到相应的队列进行调度,为提高 cache 命中率并尽可能满足异构集群的需求,均衡器对不同队列内的用户请求会采用不同的调度策略,对于静态内容的用户请求将采用基于 URL 散列的方法进行调度,其他队列内的用户请求采用加权轮转法(WRR)进行调度,以使各节点分配到的各类用户请求数大致与其当前权值相对应。同时,均衡器会周期性地监测各节点的实际负载状况,然后调整节点当前实际权值,并根据实际权值调整分发比例,以达到较好的负载均衡效果。此外,考虑到在一些应用中需要客户端与服务器经过多次交互过程才能完成一次交易,若均衡器把属于同一个交易的请求分配到不同的节点上会导致会话丢失,因此,算法也引入了会话保持机制,以保证会话的完整性。

2.1 用户请求分类

由于在 Web 网站中既存在大量静态内容,又包含了大量使用动态对象嵌入技术的动态内容,这使得不同的用户请求所需要的系统资源差异可能很大。于是,为了更好地平衡节点负载,首先需要对用户请求进行分类。借鉴文献[2]和[6],可将 Web 服务分成以下 4 类。

1) 发布型(Web publishing):主要提供静态信息,可利用内存 cache 技术缩短响应时间,提升性能。

2) 事务型(Web transaction):结果大多来自于数据库查询,查询条件通常由用户动态提供,需要进行大量的磁盘访问。

3) 电子商务型(Web commerce):主要针对电子商务应用,一般包括静态、动态和安全信息传输。出于安全原因,大部分情况下会使用 SSL 协议,将动态生成的数据通过安全途径进行传送。加解密处理通常需要消耗大量的 CPU 资源,而数据库访问又需要进行密集的磁盘访问,因此,也可称之为磁盘/CPU

密集型服务。

4) 多媒体型(Web stream):提供音视频流媒体服务,通常需要使用特殊的服务器和网络协议。

2.2 节点权值计算

在基于 Intel 处理器的 Linux 服务器环境下,各类硬件资源对应用程序运行的具体影响比例^[7]如图 1 所示。

参照图 1,在计算各节点的初始权值时主要使用 CPU 的计算能力、系统的 I/O 速率、内存的总容量及网络接口速率等参数,具体计算公式为:

$$W_{\text{init}}(N_i) = R_1 K_i \frac{L_{\text{CPU}}(N_i)}{\text{BASE}_{\text{CPU}}} + R_2 \frac{L_{\text{I/O}}(N_i)}{\text{BASE}_{\text{I/O}}} + R_3 \frac{L_{\text{M}}(N_i)}{\text{BASE}_{\text{M}}} + R_4 \frac{L_{\text{NET}}(N_i)}{\text{BASE}_{\text{NET}}} \quad (1)$$

其中 $L_f(N_i)$ 表示节点 N_i 的某一参数的值,公式(1)中依次表示为:CPU 的计算能力、系统的 I/O 速率、内存总容量及网络接口速率, K_i 是节点 i 的处理器个数。 BASE_f 表示对应参数的基准值,依次为基准的 CPU 计算能力、系统 I/O 速率、内存容量和网络接口速率,基准值可通过对各节点的实际情况进行统计确定,也可简单地以某个节点参数作为基准值。 R_i 是对应的一个权重系数,可用于区分各类硬件资源的重要程度,其中 $\sum R_i = 1$ 。一般可采用: (0.4, 0.1, 0.2, 0.3)^[8]。

基于负反馈机制的动态均衡算法能够通过周期性的反馈信息,不断调整用户请求分发的比例,充分考虑到每个节点的负载水平和响应能力,从而尽可能地避免因节点负载不均衡而引起的部分节点过载的情况,可有效提高集群的整体性能^[9]。同时,为降低均衡器的系统资源消耗,可由各节点自行计算当前权值,然后发送给均衡器。权值计算方式如下:

1) 计算各节点的负载比例 $L(N_i)$

$$L(N_i) = R_1 U_{\text{CPU}} + R_2 U_{\text{I/O}} + R_3 U_{\text{M}} + R_4 U_{\text{NET}} \quad (2)$$

R_i 的取值同上, U_{CPU} 、 $U_{\text{I/O}}$ 、 U_{M} 和 U_{NET} 分别表示 CPU 占用率、系统 I/O 占用率、内存占用率和网络带宽占用率,其取值范围均为 $[0, 1]$ 。

2) 计算各节点权值 $W(N_i)$

$$W(N_i) = L(N_i) \cdot W_{\text{init}}(N_i) \quad (3)$$

从理论上讲,一方面,实时采集和反馈节点的信息可准确反映节点的负载状况,从而保证负载的均衡分配,但节点信息的采集和反馈同样会消耗均衡器和节点的资源,实时或过于频繁的采集和反馈会给均衡器和节点带来很多额外负担,也会增加不必要的网络负荷,严重影响集群的整体性能;另一方面,通过周期性采集和反馈的节点信息只能反映采集时刻节点的负载状况,若采集周期过长则无法真实反映节点的实际负载状况。因此,需要选取适当的采集周期来解决上述问题,采集周期一般可取 5~10 s。

2.3 用户请求分发

cache 的访问速度远远高于磁盘文件的访问速度,因而通过提高 cache 命中率可有效提升集群的响应能力和吞吐量,但除了发布型服务外,其他服务大多不支持内存 cache,因此,需要对不同队列内的用户请求采用不同的调度方法。

对于发布型的用户请求,由于该类服务的页面可存放在内存 cache 中且需消耗的服务器资源很小,因此,调度时主要考虑如何提高 cache 命中率问题。通过对一些基于 URL 散列算法的实验对比^[10],最终选择 HfIp 算法作为调度算法。具体算法:先对用户请求的 URL 进行散列,然后检查目标节点的当前负载状况,如果目标节点的负载比例与平均负载比例的偏差不超过规定阈值,则将用户请求转发给目标节点,否则转发给当前负载比例最低的节点,以保证集群内各节点间的均衡。HfIp 算法如下:

```
unsigned int HfIp(char url, int size)
{
    unsigned int n=0;
```

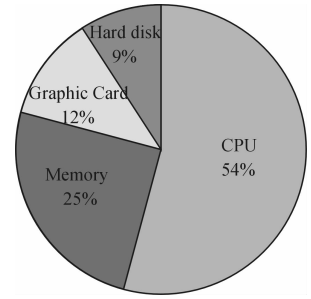


图 1 硬件资源对应用程序的影响

Fig. 1 Effects of hardware resources to application

```

char c=(char)&n;
for(int i=0;i<strlen(url);i++)
c[i%4]=url[i];
return n%size;
}

```

其中, size 为当前集群内的节点数。

对于其他 3 种类型的用户请求的调度则采用加权轮转法(WRR),即按节点的当前权值采用加权轮转的方式分发用户请求,权值越高的节点得到的用户请求就越多。

2.4 会话保持

在大多数电子商务系统及一些需要进行用户身份认证的系统中,往往需要客户与服务器间经过若干次交互才能完成一笔交易或认证。由于这些交互过程是密切相关的,服务器在执行某一交互步骤时,通常需要了解前一次或前几次交互过程的结果,这就要求所有这些相关的交互过程都应由一台服务器完成,而不能被均衡器分发到不同的服务器上。由于 Web 技术所依赖的 HTTP 是无状态的,用户与 Web 应用交互过程信息(如购物车信息、用户状态、用户记录等)就只能依靠 session 技术。当 Web 应用运行于集群环境中时,如果没有会话保持,每次的 HTTP 请求可能会被分配到不同的 Web 应用服务器并导致会话丢失^[11],因此,在算法中引入了 cookie 会话保持技术,以保证会话的完整性。

节点在为用户创建会话信息时,会生成一个集群内唯一的 session ID 作为该用户的 session 键值,此 session ID 会以 cookie 的形式发送到客户端,同时节点会把此 session ID 传给均衡器,均衡器会记录 session ID 和节点的对应关系,客户端下次请求时会带上此 session ID,均衡器收到请求后可根据请求首部中的 session ID 找到此用户 session 所在的节点并进行转发,从而达到会话保持的目的。

3 仿真试验

通常用平均响应时间和平均吞吐量作为衡量集群负载均衡算法的主要评价指标。测试集群内部网络连接采用 100 Mb/s 的交换机,由 1 台负载均衡器和 4 台节点服务器构成,具体硬件配置见表 1。另外,使用若干台 PC 作为客户端对集群进行模拟加压测试,测试工具采用 Linux 下的 Httpperf,请求事件流为符合负指数分布的泊松事件流,后台负载模型包括发布型和电子商务型两类,前者全部为静态请求,后者包含 35% 的静态请求、35% 的简单动态请求和 30% 的混合请求(主要以加解密运算和数据库操作为主),对比测试算法为 CAP 和 LARD。具体仿真对比测试结果如图 2~5 所示。

表 1 设备参数

设备	CPU		内存	硬盘
	型号	容量		
均衡器	2×Xeon 2.8		2	73
节点 1	P4 3		1	80
节点 2	P4 3		1	80
节点 3	2×Xeon 2.8		2	73
节点 4	Xeon 2.4		1	80

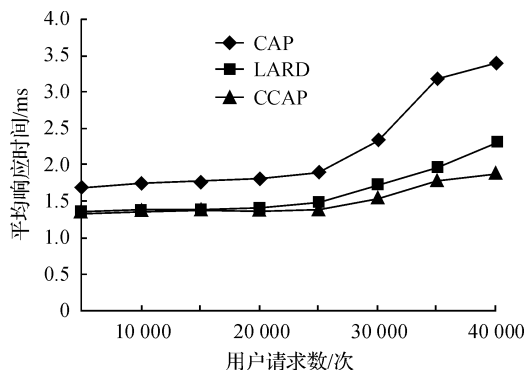


图 2 发布型负载下的平均响应时间

Fig. 2 Web publishing workload; average response time

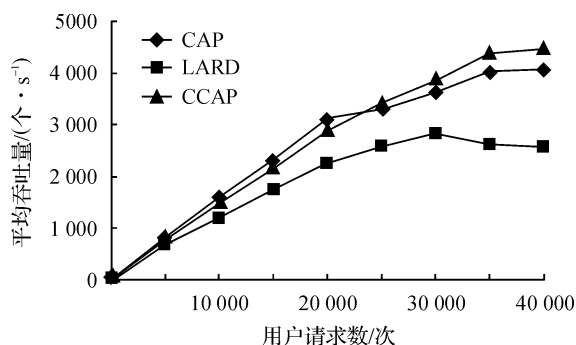


图 3 发布型负载下的平均吞吐量

Fig. 3 Web publishing workload; average throughput

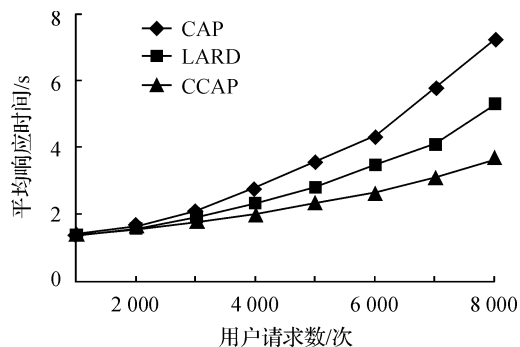


图4 电子商务型负载下的平均响应时间

Fig. 4 Web commerce workload: average response time

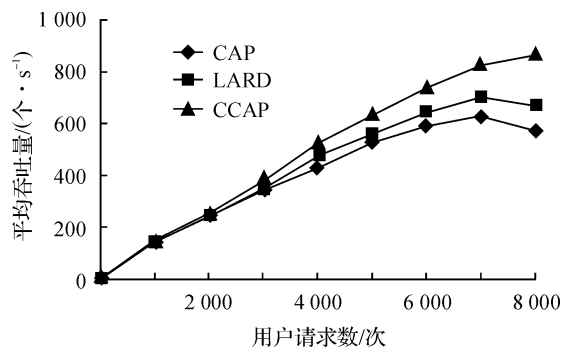


图5 电子商务型负载下的平均吞吐量

Fig. 5 Web commerce workload: average throughput

从仿真测试结果可以看出,对于发布型负载,CCAP与LARD算法无论在吞吐率还是平均响应时间上均明显优于CAP算法;对于电子商务型负载,CCAP算法的吞吐率最高,在平均响应时间上也优于LARD和CAP算法。仿真试验结果表明,算法在发布型和电子商务型两类负载环境下都能较好解决负载均衡问题,可有效提高集群的整体性能。

4 结 语

集群负载均衡算法是提高集群整体性能的关键,因此,调度时必须考虑不同类型的用户请求的特点及其差异。对于发布型的用户请求,应尽可能地通过提高cache命中率来获得更好的响应性能;对于其他类型的用户请求,需要考虑的是如何根据节点的负载能力将同一类型的请求进行均衡分配。结合上述思想,笔者在分析现有几种算法后,提出了一种基于内容分类的混合调度算法—CCAP。该算法充分考虑到了不同用户请求间的负载差异及异构集群内的节点性能差异,先通过均衡器识别用户请求的类型,然后按类型进行分类调度,并引入了反馈环节和会话保持技术,以更好地保持各节点负载的平衡,充分利用节点的资源,从而提高集群的整体性能。仿真测试结果表明,该算法与其他两种相关算法相比,在发布型和电子商务型两类负载环境下其平均响应时间和平均吞吐率都优于对比算法,具有较高的实用价值。

参考文献:

- [1] Chen L C, Choi H A. Approximation algorithms for data distribution with load balance of Web servers[C]// Proceedings of IEEE International Conference on Cluster Computing. Newport Beach: IEEE computer Society, 2001: 274-281.
- [2] Casalicchio E, Colajanni M. A client-aware dispatching algorithm for web clusters providing multiple services[C]// Proceedings of the 10th International World Wide Web Conference. Hong Kong: IWWW Conference Committee, 2001: 535-544.
- [3] Pai V S, Mohit A, Gaurav B, et al. Locality-aware request distribution in cluster-based network servers[J]. ACM SIGPLAN Notices, 1998, 33(11): 205-216.
- [4] Sharifian S, Motamedi S A, Akbari M K. A content-based load balancing algorithm with admission control for cluster web servers[J]. Future Generation Computer Systems, 2008, 24(8): 775-787.
- [5] 杨兴良, 华蓓, 高鹰. 一种应用于Web服务器集群系统的URL分配算法[J]. 系统仿真学报, 2007, 19(6): 1406-1409.
- [6] 李双庆, 程代杰, 何玲. 一种基于内容的Web集群系统负载均衡算法[J]. 计算机科学, 2003, 30(3): 138-163.
- [7] C Y. Performance improvement of cluster system by server status information[C]// Proceedings of the Fourth Annual ACIS International Conference on Computer and Information Science, Jeju Islana, South Korea: ACIS, 2005: 282-287.
- [8] 郑祺, 周广平. 基于内容分类的集群负载均衡算法[J]. 计算机系统应用, 2011, 20(5): 47-74.
- [9] 郑祺. 基于集群的Web服务器负载均衡算法研究[J]. 浙江科技学院学报, 2009, 21(1): 15-18.
- [10] 李晓明, 凤旺森. 两种对URL的散列效果很好的函数[J]. 软件学报, 2004, 15(2): 179-184.
- [11] 孙天昊, 陈飞, 邓俊昆. 一种基于cookie会话保持的LVS集群系统[J]. 计算机应用研究, 2013, 30(4): 1102-1104.