文章编号:1001 - 9081(2006)04 - 0850 - 03

EJB 集群系统负载均衡策略的研究与实现

方 巍,孙 涌,张书奎
(苏州大学 江苏省计算机信息处理技术重点实验室,江苏 苏州 215006)
(hsfangwei@ sina. com)

摘 要:分析了常用 EJB 集群系统的设计方法与负载均衡算法,设计了一种软件 Proxy 系统,并针对常用算法的不足提出了一种基于 BP 神经网络的负载均衡算法。该算法具有较强学习能力、容错性强和自适应能力强的特点。仿真实验证明,基于该算法的集群系统具有较好的适应能力和伸缩性。

关键词:EJB 集群;负载均衡;BP 神经网络
中图分类号: TP311.5 **文献标识码**:A

Research and realization on load balancing strategy in EJB cluster system

FANG Wei, SUN Yong, ZHANG Shu-kui
(*Jiangsu Province Key Laboratory of Computer Information Processing Technology, Soochow University, Jiangsu Suzhou* 215006, *China*)

Abstract: The design methods and load balancing algorithm of EJB clustering system was analyzed, and a new approach based on the BP neural networks was presented to solve the load balancing algorithm problem. How to design the BP Neural Network model was researched. The example shows that the system based on this algorithm is proved to reach better effects.

Key words: EJB clustering; load balancing; BP neural network

0 引言

负载均衡是集群系统中的重要技术,它通过在由高速网络连接起来的各个节点间平衡系统负载来提高集群系统的性能。可是,最优的负载均衡策略很难实现,即使在简化的情况下,也是一个 NP 完全问题[1]。影响负载均衡的因素有三个,分别是算法、网络拓扑以及负载均衡的粒度。而负载均衡算法包括静态负载均衡和动态负载均衡算法。静态的负载均衡,也就是作业的初始放置问题,已经得到了广泛的研究,可是由于在一给定时刻获取全局状态的精确信息是不可能的,而且根据该不精确信息做出的调度又不能更改,因而其效果有限。研究表明,大多数情况下,动态负载均衡要比静态负载均衡性能提高大约 30%[2,6]。动态负载均衡,也就是根据处理结点的负载情况动态地改变分配给它的作业量策略,希望能最好地解决各个节点之间的均衡。

目前,随着 J2EE 应用服务器的大量部署和客户访问量的急剧增加,企业对 J2EE 系统的可伸缩性和高可用性要求越来越高。如何设计和构建一个具有可伸缩的、高可用性的 J2EE 集群应用服务器,成为 J2EE 应用服务器设计必须考虑的问题。但 J2EE 应用服务器的集群是基于 EJB 组件的,与普通 Web Server 集群技术有很大的不同,实现的方法也不相同。

本文提出了一种基于 BP 神经网络的动态负载均衡算法,采用软件 Proxy 技术,在开放源码基于 J2EE 规范的 JBOSS 应用服务器上实现了 EJB 集群中的负载均衡问题。

1 负载均衡技术简介

1.1 常用系统设计方法

1) 服务器端的循环 DNS[5]

基本思想是:通过轮转 DNS 服务器把域名轮流解析到一组服务器的不同 IP 地址,将负载分到各台服务器上,从而提高整个系统的性能。当一个 Client 访问时,给请求 JNDI 的 InitialContext 客户端传递一个 DNS 名,作为命名服务器的 URL,每个 DNS 名字被转换成一个不同的地址,使用这个技术,每个客户端 InitialContext 请求就被直接发送到不同的服务器上。这种方法的缺点是:域名服务器无法知道服务结点是否有效,如果服务结点失效,域名系统依然会将域名解析到该节点上,造成用户访问失效。

2) 基于客户端的软件 Proxy[5]

客户端程序首先向一服务器查询负载情况,选出负载轻的服务器,再向该服务器发服务请求;当服务器失效时,客户端程序会尝试其他服务器。软件 Proxy 维护连接到一系列服务器上的打开连接。当一个 Client 访问服务器时,先要经过这个软件代理,这个代理能通过一些负载均衡的算法(如采用类似 DNS Round-robin、随机方法、访问权衡算法)把一个用户的访问重新定向到一个服务器。这个软件代理方法能够及时发现服务器死机或没有响应,有效地避免了 DNS round-robin 方法中出现的故障访问。

3) 服务器端的硬件均衡器

网络地址转换为在内部地址和外部地址之间进行转换,以便具备内部地址的计算机能访问外部网络,而当外部网络中的计算机访问地址转换网关拥有的某一外部地址时,地址转换网关能将其转发到一个映射的内部地址上。因此如果地址转换网关能将每个连接均匀转换为不同的内部服务器地址,此后外部网络中的计算机就各自与自己转换得到的地址上服务器进行通信,从而达到负载分担的目的。

1.2 常用负载均衡算法

1) 随机选择算法

若客户请求的到达服从泊松分布,且平均服务时间满足

收稿日期:2005 - 10 - 18;**修订日期**:2005 - 12 - 19

作者简介:方巍(1975 -),男,安徽黄山人,硕士研究生,主要研究方向:软件工程、分布式计算; 孙涌(1958 -),男,江西南昌人,副教授,博士研究生,主要研究方向:现代软件工程、计算机网络与数据库; 张书奎(1966 -),男,内蒙古伊克昭盟人,高级工程师,主要研究方向:分布式数据库、分布式网络应用.

指数分布,则各个成员服务器的队列就都是 M/M/1 排队系统,如果这些成员服务器是异构的,并且处理能力分别为 C_1, $C_2,\cdots,C_N$,则以概率为 $P_k = C_k \Big/ \sum_{i=1}^{N} C_i$ 为服务器 S_k 分配请求。它是静态算法中最简单的一种算法,有可能造成个别服务器负载过重。

2) 轮转选择算法[3]

轮转算法(Round Robin)是以循环的方式把同一个域名依次解析成不同服务器的 IP 地址。例如,把第 i 个请求分配给服务器 S_k,$k = i \bmod N$。对于异构系统,也可以采用上述类似的按处理能力比例分配的处理方法。这个算法在 DNS 域名轮询中被广泛使用。轮转法的活动是可预知的,每个节点被选择的机会是 $1/N$, 因此很容易计算出节点的负载分布。该算法简单易实现,但不适用于集群中性能不一的情况,而且当请求服务时间变化比较大时,轮转算法易导致服务器间的负载不平衡。

3) 散列法

散列法也叫哈希法(Hash),通过单射不可逆的 Hash 函数,按照某种规则将网络请求发往集群节点。哈希法在其他几类平衡算法不是很有效时会显示出特别的优势。例如,在 UDP 会话的情况下,由于轮转法和其他几类基于连接信息的算法,无法识别出会话的起止标记,会引起应用混乱。而采取基于数据包源地址的哈希映射可以在一定程度上解决这个问题:将具有相同源地址的数据包发给同一服务器节点,这使得基于高层会话的事务可以以适当的方式运行。

4) 加权法

加权方法只能与其他方法合用,是它们的一个很好的补充。加权算法根据节点的优先级或当前的负载状况(即权值)来构成负载平衡的多优先级队列,队列中的每个等待处理的连接都具有相同处理等级,这样在同一个队列里可以按照前面的轮转法或者最少连接法进行均衡,而队列之间按照优先级的先后顺序进行均衡处理。在这里权值是基于各节点能力的一个估计值。

2　负载动态均衡系统的设计

2.1　负载均衡系统设计

根据上面的分析,我们提出了一种基于软件 Proxy 方法和 BP 神经网络算法的 EJB 集群均衡系统模型,如图 1。整个系统采用软件 Proxy 方法,主要从以下几方面考虑:1)与 1.1 中所述第一种方案相比,它避免了由于中介服务器失效而引发的全线崩溃;2)除非客户端崩溃,负载平衡和失效转发策略才会失败。这样客户一般不会对此产生过多报怨;3)从性能方面来考虑,这种方案也是最优的,因为所有策略都发生在客户端(使用代理),省去了第一、三种方案由服务器来管理带来的瓶颈。

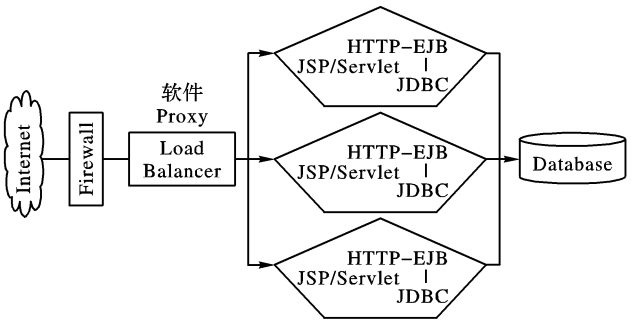

图 1　EJB 容器集群负载均衡设计

EJB 服务器的集群是基于组件的一种集群方式,和普通 Web Server 集群技术有很大的不同,实现的方法也不相同。又由于 EJB 规范中没有提供任何有关支持集群的标准,即使有的厂商在 EJB 服务器中提供了集群特性,但如何具体实现集群也是由厂商自己确定。他们实现的方法也各不相同。目前,大多数 J2EE 应用服务器都提供了集群功能,如 Bea WebLogic 应用服务器,开放源码的 JBoss 应用服务器。

EJB 服务器的核心是提供 EJB 使用的一个或者多个 EJB 容器。EJB 容器管理它所包含的 EJB,为 EJB 组件的生存和执行提供了运行环境,同时也负责 EJB 的事务管理、安全管理、资源访问控制和一些异常处理。EJB 容器不允许 J2EE 的客户端程序直接访问容器中 EJB 对象,当一个客户端用户想访问一个 EJB,EJB 规范中要求客户使用 Java 名字和目录接口 JNDI(Java Naming and Directory Interface)API 来定位 Bean 的 home 接口。

在 EJB 集群系统设计中,因为每个应用服务器上有专门的 EJB 容器对各种 EJB 进行管理维护,我们采用 JNDI 树代理,集群中每个 J2EE 应用服务器都维护一个自己的本地私有 JNDI 树。当 Client 访问 EJB 对象时,首先查询服务器负载情况,按动态算法选出负载最轻的服务器,通过服务器的 JNDI 代理树,可以重定向访问集群中所有的 EJB 对象。这样实现起来简单,只要设计好 JNDI Prxoy 树就可以了。对每个应用服务器要求不高,而且系统可伸缩性好,升级维护很容易,只要简单增加服务器就可以了。

2.2　负载量的预测

所谓时间序列是指一变量 y 在不同的时间 t 经观察得到的序列值,可表示为 $\{y_t\}t \in T$,通常 T 是平等分隔的,对事务按时间段 T 的访问次数作为负载强度的指标,在 T 时间间隔内事务的负载强度可以用访问次数与处理此事务的负载量的积表示。

采用 ARMA(p,q) 时间序列模型[4] 的数学表示如下:

$$\alpha(L)y_t = \theta + \beta(L)\delta_t$$

$$y_t = \theta + \sum_{j=i}^{p}\alpha_j y_{t-j} + \varepsilon_t + \sum_{i=1}^{q}\beta_i\varepsilon_{t-i}, \varepsilon \sim WN(0,\sigma^2)$$

$$\alpha(L) = 1 - \alpha_1 L - \cdots - \alpha_p L^p, \alpha_p \neq 0$$

$$\beta(L) = 1 + \beta_1 L + \cdots + \beta_q L^q, \beta_1 \neq 0$$

ARMA(p,q)模型阶数一般选择能保证具有较好的模拟效果为原则。一般情况,阶数选得越高,模拟效果越好,但当阶数达到一定数值后,则阶数的增大仅在原有 ARMA(p,q)模型中引入一些系数趋近于 0 的高阶项,对模拟的精度影响不大。因此,ARMA(p,q)模型阶数的选择确定应综合考虑运算量和总体模拟精度因素。

预测某事务的在时间段内的访问次数,则 $t+1$ 步预测值等于其以为 α 参数的前 P 步某事务访问次数之和加上其以 β 为参数的前 q 步的所有预测误差之和,而 $t+2$ 步预测值等于其以 α 为参数的 $t+1$ 步预测值与前 $p-1$ 步事务访问次数之和再加上其以 β 为参数的 $t+1$ 步预测误差与前 $q-1$ 步预测差之和,依此类推。

2.3　动态负载均衡算法

针对目前常用的 Round Robin 算法的不足,我们采用了基于 BP 神经网络的负载均衡算法,它指基于误差反向传播算法的多层前向神经网络,采用 Delta 规则(梯度下降法)由导师的训练方式,根据期望输出与实际输出的差值,不断调整各神经元之间的连接权值 W_{ij},使最终 W_{ij} 所确定的输出能以要求的精度逼进期望值,从而较好地解决多层网络的学习问题。

理论已经证明,BP 神经网络隐含层节点数量可以根据实际需要进行自由设置的前提下,三层前向神经网络可以实现

以任意精度逼近任意连续函数的功能,因此本系统采用标准的三层网络拓扑结构形式组成输入层、隐含层、输出层三部分,如图3所示。

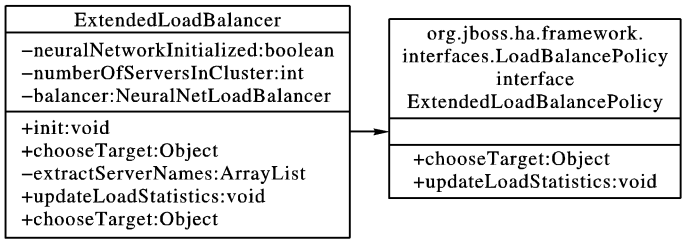

图2 BP动态负载均衡设计结构

其中,每个输入节点对应一种连接请求输入,每个输出节点对应一种负载输出。网络中相邻结点采用全相联系方式连接,同层各神经元之间没有任何连接,输入与输出层之间也没有直接的联系。

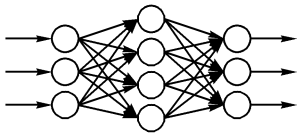

图3 BP神经网络结构

正是由于上述特性以及能够通过经验进行学习,三层前馈神经网络才具备了较强的计算能力。三层前馈神经网络一般都采用BP算法作为学习算法,BP算法的步骤如下:

1) 置各权值和阈值的初始值:$w_{ji}(0)$,$\theta_j(0)$ 为小的随机数值;

2) 提供训练样本:输入向量 $X_l(l = 1,2,\cdots,P)$,期望输出 $d_l(l = 1,2,\cdots,P)$,对每个输入样本进行下面3) ~ 5) 的迭代;

3) 网络的输出层和隐层神经元的输出值:

$$o_{k,j} = f_j\left(\sum_i w_{ji}o_{k,i} + \theta_j\right) \tag{1}$$

4) 计算输出层和隐层神经元的输出值误差:

输出层: $$\delta_{k,j} = o_{k,j}(1 - o_{k,j})(t_{k,j} - o_{k,j}) \tag{2}$$

隐层: $$\delta_{k,j} = o_{k,j}(1 - o_{k,j})\sum_m \delta_{k,m}w_{mj} \tag{3}$$

5) 调整权值和阈值:

$$w_{ji}(t+1) = w_{ji}(t) + \eta\delta_j o_{k,i} \tag{4}$$

$$\theta_j(t+1) = \theta_j(t) + \eta\delta_j \tag{5}$$

当 t 由1变为 P 后,一轮学习过程结束,判断网络输出值误差的平方和 E 是否满足精度要求,即 $E \leqslant \varepsilon$,若满足精度要求,则学习过程结束,否则,转至2),开始下一轮学习。

3 实验结果分析

在设计好模型,并对BP神经网络系统进行样本训练后,采用OPNET网络仿真软件模拟实际物理网络中包的流动和实际网络协议中的组包和拆包过程。通过仿真实验进行系统测试。JBOSS应用服务器的集群配置遵循XML规范,下面是的一个普通EJB对象的典型集群配置:

```
<jboss>
  <enterprise-beans>
    <session> <ejb-name>MySessionBean</ejb-name>
      <clustered>True</clustered>
      <cluster-config>
        <partition-name>DefaultPartition</partition-name>
        <home-load-balance-policy >
          org.jboss.ha.framework.interfaces.BPNeuralNetwork
        </home-load-balance-policy>
        <bean-load-balance-policy>
          org.jboss.ha.framework.interfaces.FirstAvailable
        </bean-load-balance-policy>
      </cluster-config>
    </session>
  </enterprise-beans>
</jboss>
```

通过一个简单的无状态会话Bean来测试这种算法的性能。测试任务主要由1000个Remote请求分布在集群结点中,分别采用不同的负载均衡算法进行测试。主要测试算法有:轮转算法、随机算法和BP动态负载均衡算法。从图4中可以看出,使用这种BP神经网络的动态负载算法性能明显高于其他常用的静态算法。

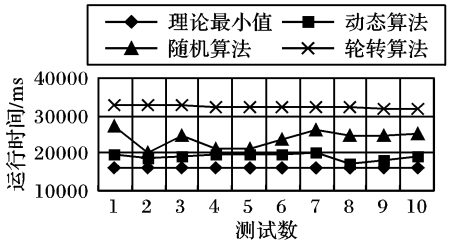

图4 常用算法的运行时间/测试数

图5表示完成远程请求数目。开始两种算法非常接近,在约30s后,该动态负载算法优于其他静态算法,因为神经网络算法需要一段时间进行学习。

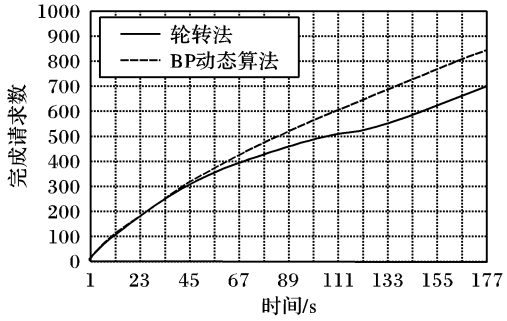

图5 完成请求数的响应时间

4 结语

实践表明,基于BP神经网络的EJB集群负载均衡方法是一种有效的方法,它具有较强的学习能力和容错性强、自适应性强以及进行联想、推测和记忆、类比等能力。今后的研究将更进一步对算法改进,使之能够做到:在事务之间内联度最高、外联度最小、服务器间内联度最高外联度最小的基础上达到负载均衡。然后,根据对比分析结果,进一步优化、完善EJB容器集群的自适应负载均衡服务功能。

参考文献:

[1] BUYYA R. High Performance Cluster Computing: Architectures and System[M]. Prentice Hall PTR, NJ, USA, 1999. 340 - 363.

[2] KAMEDA H, FATHY ES, RYU I, *et al*. A Performance Comparison of Dynamic vs. Static Load Balancing Policies in a Mainframe-Personal Computer Network Model[A]. Proceedings of IEEE CDC2000 [C]. Sydney, Australia, 2000. 1415 - 1420.

[3] BALASUBRAMANIAN J, SCHMIDT DC, DOWDY L, *et al*. Evaluating the Performance of MiddleWare Load Balancing Strategies[A]. Proceedings of the 8th IEEE intl Enterprise Distributed Object Computing Conference[C]. 2004. 1541 - 1577.

[4] KROLZING H-M. Introduction to Time-Series Analysis[M]. Hilary Term. 2002.

[5] 郭全生, 舒继开, 毛希平, 等. 基于LVS系统的负载动态平衡设计与实现[J]. 计算机研究与发展, 2004, 41(6): 923 - 929.

[6] 唐丹, 金海, 张永坤. 集群动态负载平衡系统的性能评价[J]. 计算机学报, 2004, 27(6): 808 - 810.