一种自适应遗传算法的 EJB 集群负载均衡策略①

An Adaptive Genetic Algorithm for Load Balancing in EJB Clusters

李　洋　刘万军　(辽宁工程技术大学 电子与信息工程学院 辽宁 葫芦岛 125105)

摘　要： 负载均衡技术是 EJB 集群系统任务调度的中心环节，但传统的负载均衡算法是一种静态分配负载均衡算法,因此不能很好的实现服务器端负载均衡。通过动态的获取服务器端系统的实时性参数，采用自适应的遗传算法来计算服务器的负载，从而保证系统长时间运行不会发生倾斜。实验结果表明，该方法降低了服务器端事务请求的响应时间，提高了系统的吞吐率，从而改善了系统性能。

关键词： EJB 集群　负载均衡　遗传算法

1　引言

随着 Internet 的飞速发展,导致了 Internet 日益膨胀，服务器处理能力成为网络访问的新瓶颈。一种解决方法是使用更快速的服务器，但服务器速度的提高也很难赶上用户数量的增长；而另一种解决方法是采用可扩展的服务器组，集群服务器，当用户数量增加时，只需要相应地增加服务器数即可满足用户对访问时延的要求。因此，一个重要问题是要设计合适的资源调度算法／请求分配算法，使每个服务器上的负载基本平衡。本文采用遗传算法来解决资源调度和请求分配。

集群就是一组应用服务器实例，它们作为一个逻辑实体一起工作。集群中的每个服务器实例都具有相同的配置，并且被部署了相同的应用程序。通过将服务器实例添加到集群来提高系统性能，从而实现了水平缩放[1]。负载均衡则是使集群中的各服务器尽量均匀地分摊客户请求，它是服务器集群的关键技术之一，主要解决服务器集群中由于部分服务器繁忙而部分服务器空闲而使集群的利用率低下的问题[1]。基于 EJB 集群的动态负载均衡，就是指根据集群中各服务器节点某一个时刻的负载状况，将任务动态地分配给服务器节点上，从而使负载较轻的服务器承担更多的新任务[2]。因此，如何选出相对“空闲”的服务器来处理新的请求，是负载均衡算法所要解决的问题。本文采用自适应遗传算法来实现负载均衡算法，用 EJB 集群的会话 Bean 来动态的获取集群中服务器节点负载信息，然后，根据负载均衡算法求出系统优化问题的最优解，从而选择负载最轻的服务器去处理请求，提高系统性能。EJB 集群负载均衡拓扑结构如图 1 所示。

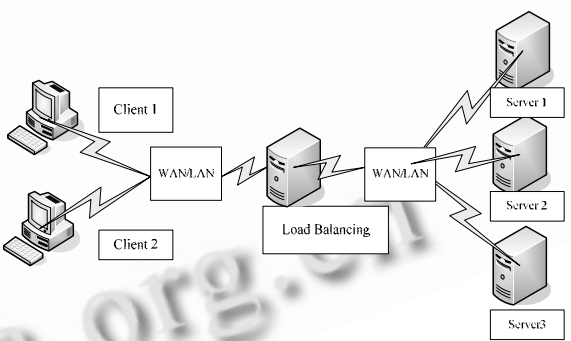

图 1　EJB 集群负载均衡拓扑结构

2　算法设计

随机分配(Random Allocation, RA)是遵循随机概率原理的一种分配方法。从长远的观点来看，连接分配到每台服务器上的概率是相等的。

轮询算法(Round Robin ,RR)把新的连接请求按顺序轮流分配到不同的服务器上，从而实现负载均衡。

加权最小连接算法(Weighted Least-Connection ，WLC)用相应的权值表示服务器的处理能力，将用户的请求分配给当前连接数与权值之比最小的服务器。

① 收稿时间:2009-02-24

遗传算法(Genetic Algorithm．GA[3,4])是一种借鉴生物界自然选择和自然遗传进化机制，是一个迭代过程，它模拟自然进化过程，反复将选择算子、交换算子、变异算子作用于群体，最终搜索到最优解的方法。

2.1 自适应遗传算法设计

本文采用自适应遗传算法的主要构造过程示意图如图 2 所示。在计算负载时,我们主要使用负载信息为服务器指标。服务器指标是指服务器上的各种负载信息。对于不同的应用，会有不同的负载情况，,这里我们引入各个负载信息的系数，来表示各个负载信息的权重。系统管理员根据不同应用的需求，调整各个负载信息的系数。根据文献[2]初始权重设为{0.3,0.1,0.3,0.2,0.1}。

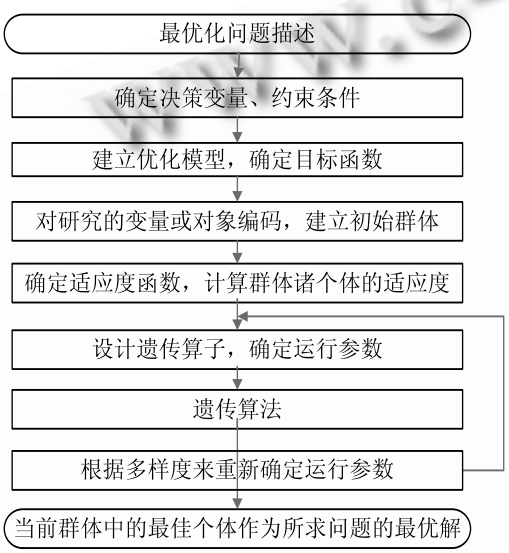

图 2 遗传算法的主要构造过程

① 决策变量的确定：决策变量指的是服务器的各项指标，包括 CPU 占用率为 cpu，内存占用率为 memory，网络利用率 net， 进程占用率 process,负载信息的系数即权值 wi，另一个重要的服务器指标是服务器所提供服务的响应时间为 response,它能比较好地反映服务器上请求等待队列的长度和请求的处理时间。

② 目标函数的确定：由于服务器负载是由服务器的各项指标决定的，因此是一个多目标优化问题，各个子目标函数 fi (x) (i=1,2,…,n-1) 赋予不同的权值 wi (i=1,2,…,n-1),其中 wi 大小代表相应子目标在多目标优化问题中的重要程度，即前面所说的权重。此外，由于服务器硬件配置的不同，配置较高的服务器，分配任务会相对配置低的服务器多，因此，设置初始 3 台服务器节点的分配任务比重 Cj 为{2：1：1}。则目标函数可表示为：

$$u(f(x))=\sum_{i=1}^{n}\sum_{j=1}^{3}(w_i f_i(x)/c_j),\quad \sum_{i=1}^{n} w_i = 1 \tag{1}$$

展开式表示为：

u(f(j))=w1*cpu[j]/cj+w2*memory[j]/cj+w3*net [j]/cj +w4*response [j] /cj +w5*process[j] /cj，其中 0＜cpu [j]≤1，0＜memory [j]≤1，0＜net [j]≤1，0＜response [j]≤100，0＜process [j]≤1 (2)

多目标优化问题的本质在于，在很多情况下，各个子目标是存在相互冲突的，一个子目标的改善有可能会引起另一个子目标性能的降低，各子目标同时达到最优是不可能的[3]。这里，响应时间和 CPU 占用率、网络利用率，就是相互冲突成反比关系。response = c / (cpu*net)；c 为常数。因此，目标函数变量转换为四个。于是，负载均衡服务器会定期获取这四项服务器节点的信息，然后将这些信息参数传入算法中进行编码。

③ 编码：编码采用数值型单变元和符号型变元相结合对四项服务器节点的信息即多目标问题的可行解(cpu, memory, net, process)编码为二进制串，根据 2mi＜(bi-ai)*10j≤2mi -1，mi 为子串长，j 取决于变量的精度，[ai，bi]为变量取值域，确定各可行解的长度 mi，染色体总长度即为所有可行解变量长度之和。变量的实际值=(ai +标记数 N* bi-ai)/(2mi -1),标记数 N 为将变量取值域等分后的分点值[4]。利用目标函数确定适应度函数，则适应度函数为

$$u(f(x))=k-\sum_{i=1}^{n}\sum_{j=1}^{3}(w_i f_i(x)/c_j)\ ；其中\ k \geq \max\{\sum_{i=1}^{n} w_i f_i(x) \mid x \in X\} \tag{3}$$

根据公式(3)求出可行解中最优染色体。

④ 选择复制：复制是遗传算法的基本算子，对当前群体实施复制操作的意图就是使适应度高的优良个

体尽可能的在下一代新群体中得以繁值，体现“适者生存”的自然选择原则。本文采用 J.Holland 设计的轮盘法进行选择复制。

⑤ 交换：通过两个个体的交换可以产生两个新的个体，称为子代。通常子代与父代不同，但包含着两个父代的遗传信息。这里采用单点交换算子，交换概率 Pi 初始设为 0.5[4]。交换过程如下：

while (i<= 群体数){

Pi = [0,1]分布的随机数；

if (Pi < 0.5){选择交换的双亲进行交换；}

i++;}

根据前面编码中利用公式 2mi＜(bi-ai)*10j≤2mi -1，令 j=3，求出子串长度为 10 位，则整个染色体长度为 40 位，在从 1—40 位中按均匀分布产生一个随机整数作为交换点。

⑥ 变异：是指将个体染色体编码串中的某些基因座上的基因值用其他基因座上的基因值替换，从而形成一个新的个体。其目的有两个，一是改善遗传算法的局部搜索能力。二是维持群体多样性，防止出现早熟现象(早熟是指过早的收敛到局部最优解)[4]。

另外，在迭代操作之后，要根据群体的多样度函数公式(4)来计算多样度大小，衡量多样度合理性，若不合理，重新设定遗传算子。还可以重新设定每个服务器指标的权重，把相对重要的指标系数调大。然后从步骤 4 开始进行重新操作。

⑦ 运算终止准则：设定终止迭代次数 T=500，它表示遗传算法运行到指定的迭代次数后就停止运行，返回当前群体中的最佳个体作为所求问题的最优解。

2.2 遗传算子的改进

针对遗传算法可能过早地减少多样性，陷入局部极值点，提出一种自适应的调整遗传算法中遗传算子的方法，它能够度量群体的多样性，对遗传操作过程中的每一代群体可以进行个体多样性检验。以下为定义群体的多样度的公式：

设 X 是规模为 n 的一代群体，其个体分别记为 A_1，A_2，…，A_n,个体字符串长度为 L。其中，$A_i=a_{i1},a_{i2},\cdots a_{iL}$，$a_{ij}\in\{0,1\}$,i=1,2,…,n；j=1,2,…,L。X 用矩阵表示为：

$$X=\begin{pmatrix} a_{11} & \cdots & a_{1L} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nL} \end{pmatrix}$$

设 Y 为 X 的重心，即：

$$Y=\frac{1}{n}\sum_{i=1}^{n}A_i=(y_1,y_2,\ldots,y_L)$$

$$=\left(\frac{1}{n}\sum_{i=1}^{n}a_{i1},\frac{1}{n}\sum_{i=1}^{n}a_{i2},\ldots,\frac{1}{n}\sum_{i=1}^{n}a_{iL}\right)$$

群体 X 中个体之间的距离为

$d(A_1,A_j)=\bigvee_{k=1}^{L}|a_{ik}-a_{jk}|$，∨为取最大运算。

群体的多样度为

$$D(X)=\sum_{i=1}^{n}d(A_i,Y)=\sum_{i=1}^{n}\left(\bigvee_{k=1}^{L}|a_{ik}-y_k|\right),\quad(4)$$

$$0\le D(X)\le 1$$

由于群体的多样度与遗传算法的收敛有直接关系，多样度减小，导致算法的收敛；若多样度减小过快，可能导致早熟。而多样度与复制概率成反比，复制概率大将导致多样度减小；与变异概率一般成正比关系，变异概率大会导致多样度增大。所以，可以根据群体的多样度来动态调整遗传算子,然后再进行复制，交换，变异循环操作，从而提高遗传算法的最终性能和效率。

3 测试实例及结果分析

整个集群环境搭建在 Window2003 Server 平台上，测试环境中有 3 台服务器，1 台 Intel Pentium D 3.0 GHz,内存 2G 和 2 台 P4 3.0GHz，内存 512MB。系统网络环境为 100 Mbit/s。Web 服务器采用 JBOSS4 部署 EJB 组件，构建系统集群。负载均衡服务器也采用 EJB 组件的形式，用集群 EJB 组件的会话 Bean 动态的获取服务器端系统的实时性参数，完成搜集服务器端节点的负载信息，然后根据负载均衡算法选取调度的服务器节点处理客户请求。这里，用于获取系统信息类的 EJB 会话 Bean 的集群方式是基于 Round-Robin 负载均衡策略的。而每次的客户发送请求信息根据算法定制的动态负载均衡来查找高可用性 Java 命名和目录接口(High Availability—Java Naming and Directory Interface，HA—JNDI)。通过 HA—JNDI 查找或创建绑定 JBOSS 集群的 EJB 服务器节点[9,10]。

负载均衡的核心类包括：Algorithm，Genetic

Algorithm,ContextStrategy,IMonitorService,MonitorServiceImpl,ServerFactory,ServerService。Algorithm 表示算法的抽象类,GeneticAlgorithm 表示遗传算法类，OtherAlgorithm 包括加权最小连接算法和轮询算法，ContextStrategy 用于形成策略模式的上下文类，IMonitorService 表示获取系统信息的接口，MonitorServiceImpl 表示获取系统信息类，是 EJB 会话 Bean，ServerService 表示提供给会话 Bean 的工厂接口，ServerFactory 提供给会话 Bean 的工厂类，告诉 MonitorServiceImpl 类获取的是哪台服务器的信息。程序采用 Gof 提出的设计模式对负载均衡算法和相关的类进行设计，以提高程序扩展能力。本人采用工厂方法模式与策略模式相结合对负载均衡策略进行设计。其具体类图如图 4 所示。

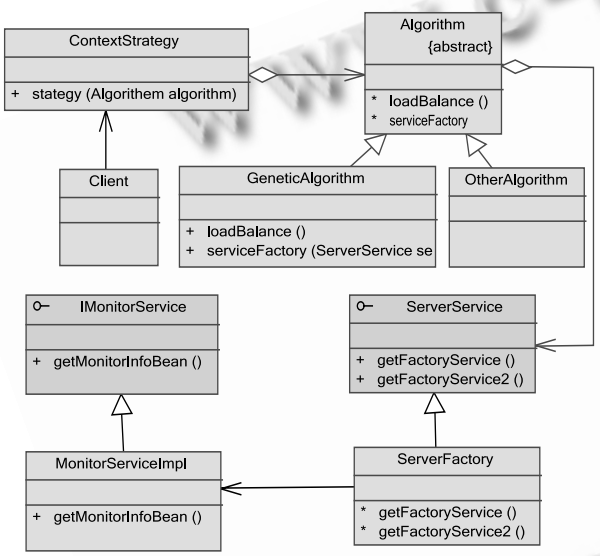

图 4 负载均衡设计类图

为验证算法的有效性，采用 Mercury LoadRunner 工业标准级负载测试工具,它通过模拟多个用户来实施并发负载来对整个企业架构进行测试。通过场景计划可以设置负载行为以精确地描绘用户行为，配置虚拟用户数量。并可以确定将负载应用于应用程序的速率、负载测试持续时间以及如何停止负载等。本文通过场景计划配置虚拟用户总量为 100，以每 30 秒增加 10 个用户对系统进行增压，同样以每 30 秒减少 10 个用户对系统进行减压停止负载。然后分别对随机分配，轮询算算法，加权最小连接算法和遗传算法这四种负载均衡算法进行测试比较。根据初始参数，采用遗传算法计算各台服务器负载，与其它算法比较所得的对比情况如图 5，图 6,图 7 所示。其中，Genetic 表示遗传算法,Weighted 表示加权最小接算法，Round 表示轮询算法,Random 表示随机算法。

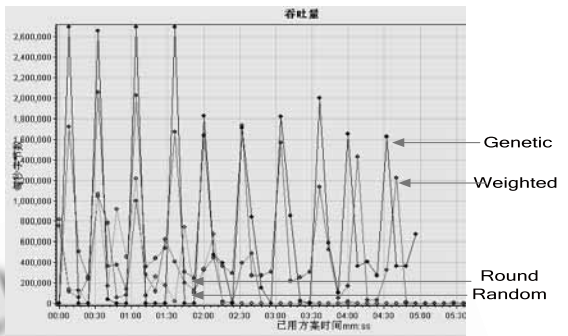

图 5 四种算法吞吐量比较图

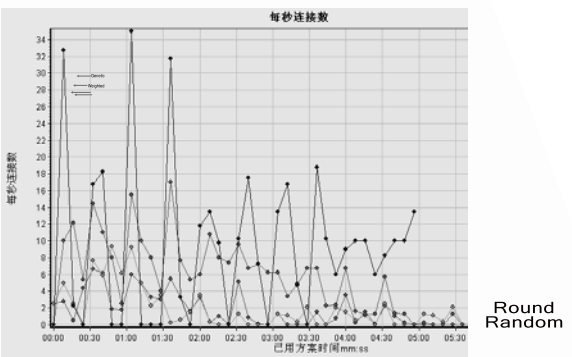

图 6 四种算法连接数比较图

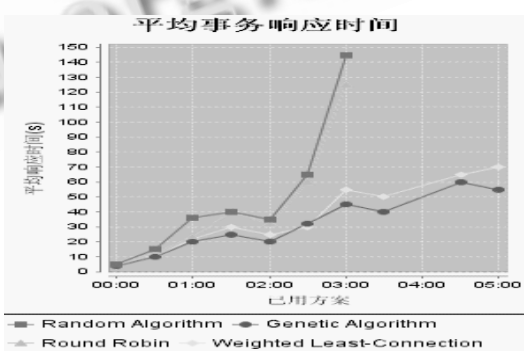

图 7 四种算法平均事务响应时间比较图

从图 5 场景运行的吞吐量图确定目前的网络带宽是否是瓶颈，可以看出当并发的用户数量很大时，遗传算法和加权最小连接算法所表示的曲线图呈高低起伏状态，表示并没有出现瓶颈,而相应的轮询算法和随机算法所表示的曲线图呈比较平的直线，表

示出现网络带宽瓶颈。图 6 场景运行的每秒连接数图显示在每一场景运行过程中用户每秒向服务器连接次数(HTTP 请求数)，从该图可以看出遗传算法所表示的曲线图呈高低起伏状态，可以很好的满足负载的要求，负载较高时，连接次数受到影响较小。图 7 场景运行的平均事务响应时间图可以看出，当用户请求数量较小时，四种分配策略所响应的时间差异不是很大，但随着户请求数量的增多，利用遗传算法负载均衡策略的响应时间明显比其它算法响应的时间少。另外，当系统运行 3 分钟以后，利用轮询算法和随机算法出现网络带宽瓶颈，响应时间直线上升，这时已经不能正常响应，而利用遗传算法负载均衡策略的响应时间依然比较平稳。

4 结语

本文通过用集群 EJB 组件的会话 Bean 动态的获取服务器端系统的实时性参数，采用对遗传算子的不断调整，改变复制概率和变异概率，来使遗传算法的性能更加优化，从而选择更合理的服务器来处理客户请求，实现集群系统的均衡负载。实验结果表明，该负载均衡算法能够有效地降低事务平均响应时间，增加吞吐量和连接数，从而提高集群系统的综合性能。其缺点就是要定时对服务器信息进行监测，要不断调整遗传算子，计算量略大，还要额外的时间来计算负载。

参考文献

1 Sun Java System Application Server 9.1 Installation Guide. http://dlc.sun.com/pdf/819-3670/819 - 3670.pdf. 2007 - 9.

2 章文嵩.Linux 服务器集群系统.LVS(Linux Virtual Server 项目.http: //www-128. ibm. com/developerworks/cn/linux/cluster/lvs/part4 /index.html

3 周明,孙树栋.遗传算法原理及应用.北京:国防工业出版社, 2000.

4 郭嗣琮,陈刚.信息科学中的软计算方法.沈阳:东北大学出版社, 2001.

5 陶洋,陈辉.一种基于遗传算法的负载均衡选播路由算法.计算机科学, 2006,33(1):35 - 37.

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.计算机科学, 2003,30(3):138 - 140.

7 任彦琦,彭勤科,胡保生.一种基于内容的 Web 集群服务器负载均衡算法.计算机工程, 2006,31(2):122 - 124.

8 朱利,张兴军.Web 服务器组的负载均衡方法研究.小型微型计算机系统, 2003,24(12):2096 - 2099.

9 Rima Patel sriganesh,Gerald Brose, Micah Silverman. 精通 EJB3.0.罗时飞,译.北京:电子工业出版社, 2006.

10 塔克(Stark,S).罗时飞,译. JBoss 管理与开发核心技术.北京:电子工业出版社, 2004.