文章编号:1000 - 582X(2005)11 - 0039 - 04

MMOG 的网络负载均衡算法*

徐光侠,祝伟华,杨 丹,谭亚竹
(重庆大学 软件学院,重庆 400030)

摘 要:随着网络游戏的迅猛发展,大型多人在线网络游戏(Massively Multiplayer Online Games, MMOG)对服务器的要求越来越高.通过对服务器集群和网络负载均衡算法的深入剖析,结合动态网络负载均衡算法——加权循环算法,给出了实现对大型多人在线网游环境下服务器集群的动态负载均衡算法.

关键词:集群;网络负载均衡;动态均衡算法;加权循环算法
中图分类号:TP301.6 **文献标识码**:A

网络游戏是近年来兴起的一种网络服务新模式,据报道,国外每年由电子娱乐行业带动的产业链,产值高达十亿美元.在国内,自从大型在线游戏引入后,经过近几年的发展,也已形成一定规模,其经济效益非常巨大.专家估计,网络游戏在未来几年内市场份额将迅速扩大,成为网络信息服务商获取利润的重要来源.

MMOG 的迅猛发展,对服务器提出了更高的要求,它的稳定性和安全性必须得到有力的保障.一旦在线游戏出现宕机,不仅对玩家而言意味着游戏中断,而且对运营商而言意味着利益损失和用户满意度下降.因此,基于 MMOG 的服务器集群平台必须提供高度的可用性、伸缩性与易管理性,才能有力保障游戏的正常运行.在 MMOG 中,服务器集群是"中枢神经",而负载均衡是实现集群的关键技术.

1 网络游戏服务器的负载均衡集群

1.1 负载均衡集群

集群(Cluster),是一种并行或分布式处理系统,由很多通过高速网络连在一起的计算机组成,像一个单独集成的计算资源一样协同工作.它主要分为 3 类:容错集群(Fail-Over Cluster),负载均衡集群(Load-Balancing Cluster),高性能计算集群(High Performance Computing).负载均衡集群一般用于相应网络请求的网页服务器,代理服务器.这种集群可以在接到请求时,检查接受请求较少、不繁忙的服务器,并把请求转到这些服务器上[1].

网络负载均衡有两方面的含义:首先,大量的并发访问或数据流量分摊到多台结点设备上分别处理,减少用户等待响应的时间;其次,单个重负载的运算分担到多台结点设备上做并行处理,每个结点设备处理结束后,将结果汇总,返回给用户,系统处理能力得到大幅度提高.本质上讲,它是分布式作业调度系统的一种实现.负载均衡器作为作业分配的控制者,根据集群结点的当前处理能力,集中地或分布地对作业进行调配,并且在作业的生命周期里监控执行结点的有效状态[2].

网络负载均衡集群为 MMOG 的需求提供更实用的系统.该系统使各结点的负载流量可以在服务器集群中尽可能平均合理地分摊处理,该负载需要均衡计算的应用程序来处理端口负载或网络流量负载.这样的系统非常适合于运行同一组应用程序的大量用户,很适合 MMOG 的访问模式.每个结点都可以处理一部分负载,并且可以在结点之间动态分配负载,以实现均衡.对于网络流量也如此,通常,网络服务器应用程序接受了大量入网流量,无法迅速处理,这就需要将流量发送给其它结点.负载均衡算法可以根据每个结点不同的可用资源或网络的特殊环境来进行优化.

1.2 多人在线游戏

多人在线游戏以主从架构的模式进行处理,所有的玩家都会连接到服务器,而不是玩家一对一的连接计算机,服务器负责将并发访问或数据流量分摊到多台结点上分别处理.多人在线模式分为轮流型和随机

* 收稿日期:2005 - 06 - 15
基金项目:重庆市自然科学基金资助项目(CSTC,2004BB2182)
作者简介:徐光侠(1974 -),女,重庆人,讲师,重庆大学硕士研究生,主要从事软件工程方面的研究.

型 2 种模式[3-4].

1) 轮流型:有一些游戏本身的玩法就是轮流型的,像麻将、接龙等游戏. 玩家是根据某种顺序轮流对游戏作出响应,然后等待其他玩家进行,如图 1 所示.

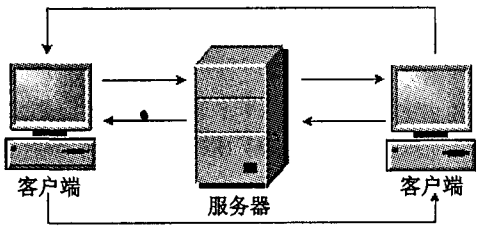

图 1 轮流型模式

2) 随机型:轮流型并不适用于所有的游戏,如果想实现的游戏像网络版的双人小蜜蜂那样,轮流型模式就不适用,此类游戏采用的典型模式是随机型模式. 某一方的玩家会将本身的状态传递给服务器程序,然后由服务器来更新另一方玩家的状态,如图 2 所示.

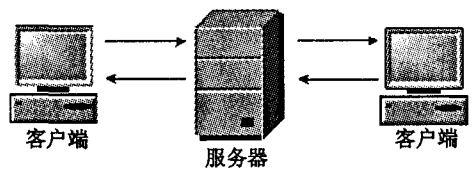

图 2 随机型模式

在实现随机型网络游戏时会遇到问题,如果把每一个联机都视为一个游戏的副本,那么该应用应确保这些副本之间的状态都能保持一致,也就是说"同步",问题是如何在有限的带宽与服务器负载下确保每个游戏的状态都能保持一致. 这是该类游戏在设计时的关键和服务器集群应考虑的问题.

2 网络负载均衡算法

将大型网络游戏安装到多台服务器上,将这些服务器配置为共享工作负荷,该类型的配置就是负载均衡集群. 负载均衡技术(通称"负载均衡器")可以接收用户端传入的请求,并根据需要将它们重定向到特定主机. 有负载均衡功能的主机能够同时响应不同客户端请求,甚至是来自同一客户端的多个请求,这就分散了负载,加快了处理速度,并缩短了对客户端的响应时间.

负载均衡的实现有 2 种主要类别:基于软件的负载均衡和基于硬件的负载均衡. 基于软件的均衡包括在负载均衡群集中安装在服务器上的特殊软件. 软件根据不同的算法分派或接受客户端向服务器发出的请求. 算法可以是简单的循环算法,也可以是考虑服务器性能的更复杂的算法,该类算法采用负载均衡器的方式来实现. 负载均衡器使用不同的算法控制通信流量,这些算法基于软件的方式来分散负载,最大限度地利用群集内的所有服务器. 当然也可以采用基于硬件的方式来实现负载均衡. 不过它也是以软件为其提供负载均衡功能的专用交换机或路由器组成的.

大多数负载作业的调度都是以连接为粒度的,在 HTTP 协议中,每个对象从服务器上获取都需要建立一个 TCP 连接,同一用户的不同请求通常应该会被调度到不同的服务器上,这种细粒度的调度在一定程度上可以避免单个用户访问的突发性引起服务器间的负载不平衡. 连接调度算法的实现,直接关系到负载均衡器的开销,是实现集群可扩展目的的关键制约因素.

负载均衡器上通常使用的连接调度算法有:循环法、随机均衡法、最少连接法、最低缺失法、散列法、最快响应法、加权法. 由于篇幅有限,以下将详细分析循环法和加权调度算法.

2.1 循环法

循环法也称轮转法,就是以循环的方式依次请求调度到不同的服务器,即每次调度执行 $i=(i+1) \bmod n$,并选出第 i 台服务器. 算法优点是将负载均衡地分配给每台服务器,而不考虑当前的连接数或响应时间. 循环法适合于群集中的服务器具有相同处理能力的情况. 循环算法是所有调度算法中最简单也最容易实现的一种方法. 在一个任务队列里,队列的每个成员都具有相同的地位,循环法简单地在这组成员中顺序循环选择. 在负载均衡环境中,均衡器将新的请求轮流发给结点队列中的下一结点,如此循环,每个集群的结点都在相等的地位下被轮流选择.

循环法的活动是可预知的,每个结点被选择的机会是 $1/n$,因此很容易计算出结点的负载分布. 循环法在实际应用中,一般将它与其他简单方法联合使用时比较有效. 循环法的算法流程如下:

假如有一组服务器 $S=\{S_0,S_1,\cdots,S_{n-1}\}$,指示变量 i 表示上一次选择的服务器,$W(S_i)$ 表示服务器 S_i 的权值. 变量 i 被初始化为 $n-1$,其中 $n>0$.

```
j=i;
do {
    j=(j+1) mod n;
    if (W(Sj) >0) {
        i=j;
        return Si;
    }
} while(j! =i);
return NULL;
```

循环法相对简单,不适用于服务器组中处理性能不相同的情况,而且当请求服务时间变化比较大时,容易导致服务器间的负载不均衡.

2.2 加权法

加权法只能与其他方法合用,是其它算法的补充.

加权法根据结点的优先级或当前的负载状况(即权值)来构成负载均衡的多优先级队列,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循环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结点能力的一个估计值.

均衡算法设计的好坏直接决定了集群在负载均衡上的表现,不好的算法,会导致集群的负载失衡. 一般的均衡算法主要任务是决定如何选择下一个集群结点,然后将新的服务请求转发给它. 有些简单均衡方法可以独立使用,有些必须和其它简单或高级方法组合使用. 好的负载均衡算法不是万能的,一般只在某些特殊的应用环境下才能发挥最大效用. 因此在考察负载均衡算法时,应注意算法本身的适用面. 在采取集群部署时根据集群自身的特点作综合考虑,把不同的算法和技术结合起来使用.

3　网络游戏的动态负载均衡算法实现

以动态负载均衡技术为支撑,利用网络游戏接入平台,实现以动态负载均衡技术为基础的分布式服务器集群结构,使得同一游戏可以允许多达 10 万人同时在线. 在该技术支持下,游戏中的对战可以真正达到千军万马混战的磅礴气势.

在网络游戏的应用中,对任务请求处理时间的不同可能会导致处理结点利用率的倾斜,即处理结点的负载不均衡. 例如,有些结点已经超负荷运行,而其他结点基本是闲置着. 与此同时,有些结点可能已经忙不过来,有很长的请求队列,还不断地收到新的请求. 因此,有必要采用一种机制,使得均衡器能够实时地了解各个结点的负载状况,并根据负载的变化做出调整. 具体的做法是采用基于负反馈机制的动态负载均衡算法,该算法考虑集群中每一个结点的实时负载和响应能力,不断调整任务分布的比例,以避免有些结点过载时依然收到大量请求,从而提高服务器集群的整体吞吐率.

3.1　加权循环算法

在集群内,负载均衡器上运行服务端监控进程,监控进程负责监视和收集集群内各个结点的负载信息;而每个结点上运行客户端进程,负责定时向均衡器报告自身的负载状况. 监控进程根据收到的全部结点的负载信息来进行同步操作,对将要分配的任务按照权值的比例重新进行分布.

加权循环算法用相应的权值表示结点的处理性能,根据权值的高低顺序并按照循环的方式将任务请求分配到各结点. 权值高的结点比权值低的结点处理更多的任务请求,相同权值的结点处理相同份额的请求,权值低的则相应减少任务的分配.

加权循环算法的基本原理可描述为:假设某集群内有一组结点 $N = \{N_0, N_1, \cdots, N_{n-1}\}$,用 $W(N_i)$ 表示结点 N_i 的权值,指示变量 i 表示上一次选择的服务器,$T(N_i)$ 表示结点 N_i 当前所分配的任务量. $\sum T(N_i)$ 表示当前同步周期需要处理的任务总量,$\sum W(N_i)$ 表示结点的权值总和. 则 $W(N_i)/\sum W(N_i) = T(N_i)/\sum T(N_i)$ 表示任务分配是按照各个结点权值占权值总和的比例来进行分配.

3.2　权值计算

当集群的结点初次投入系统中使用时,系统管理员根据结点的硬件配置情况对每个结点都设定一个初始权值 $W_0(N_i)$(通常根据结点的硬件配置来定义,硬件配置越高的结点默认值越高),在负载均衡器上先使用默认权值. 然后,随着结点负载的变化,均衡器对权值进行调整. 结点的动态权值是在群集运行时根据结点的相关参数计算出来的,选取的最重要几项参数包括内存容量,系统响应时间,CPU 频率,I/O 状况,当前进程数等信息作为计算公式的因子. 结合比较每个结点当前的权值,可以计算出新的权值大小.

为方便在系统运行过程中针对不同的应用对各个参数的比例进行适当调整,给每一参数设定一个常量系数 R_i,用以表示各个负载参数的重要程度,其中 $\sum R_i = 1$. 因此,任何一个结点 N_i 的动态权值公式就可以描述为[5]:

$$\mathrm{LOAD}(N_i) = R_1 * \mathrm{Lmemory}(N_i) + R_2 * \mathrm{Lresponse}(N_i) + R_3 * \mathrm{Lcpu}(N_i) + R_4 * \mathrm{Lio}(N_i) + R_5 * \mathrm{Lprocess}(N_i), \quad (1)$$

其中 $L(N_i)$ 表示结点 N_i 当前某一项参数的负载值,式(1)中依次表示为:内存使用率、系统响应时间、CPU 使用率、I/O 状况以及进程总数.

在服务器集群中,采用 $R_i = \{0.2, 0.2, 0.3, 0.1, 0.2\}, i \in [1,5]$. 这里认为服务器的 CPU 较其他参数重要一些,若当前的系数 R_i 不能很好地反映应用的负载,可以对系数不断地修正,直到找到贴近当前应用的一组系数. 设置实时采集权值的周期,周期短则频繁采集,会给均衡器和结点带来负担,增加额外的网络负荷. 在实践过程中要适当地调整采集负载信息的周期,一般在 5 ~ 10 s,但在群集的不同应用实践中周期的设置有较大差别,对实时性要求较高的采集周期较短. 采集的权值曲线应尽量表现为比较平滑曲线,这样在负反馈机制的调整效果上就会比较好.

服务器集群系统使用中,系统管理员对服务器都设定一个初始权值 $W_0(N_i)$,随着服务器负载的变化,对最终权值 W_i 进行调整. 为了避免最终权值 W_i 变成一个很大的值,规定 W_i 的值域为 $[W_0(N_i), \mathrm{SCALE} * W_0(N_i)]$,权值(SCALE)大小是可以调整的,它的缺省值为 10. 当 $W_0(N_i)$ 不为零,则查询该服务器的各负载

参数,并根据式(1)计算出动态权值 LOAD(N_i). 引入以下权值计算公式,结合结点的初始权值和采集的动态权值来计算最终权值的结果[6].

$$W_i \leftarrow \begin{cases} W_i + A * \sqrt[3]{0.95 - \mathrm{LOAD}(N_i)} & \text{当 LOAD}(N_i) \neq 0.95, \\ W_i & \text{当 LOAD}(N_i) = 0.95. \end{cases} \quad (2)$$

式(2)中,A 是一个可调整的参数(缺省值为 5),设置初始权值 $W_0(N_i) = 0.95$. 1)当动态权值 LOAD(N_i)为初始权值 0.95 时,则均衡器中该结点的最终权值不变,说明系统的负载状况刚好达到理想状况;2)如果动态权值 LOAD(N_i)大于初始权值 0.95 时,则均衡器中该结点的最终权值 W_i 升高,说明该结点负载较轻,均衡器将会增加分配给该结点的任务比率;3)如果动态权值 LOAD(N_i)低于初始权值 0.95 时,则均衡器中该结点的最终权值 W_i 降低,说明该结点开始处于过载状况,均衡器将会减少对该结点分配的任务.

在实际使用中,若发现所有结点的动态权值 LOAD(N_i)都小于 $W_0(N_i)$,则说明当前集群处于超载状态,这时需要加入新的结点到集群中以处理过载;反之,若所有结点的动态权值 LOAD(N_i)大大高于 $W_0(N_i)$,则说明当前系统的负载都比较轻,可以适当减少集群中的结点以节约资源.

4 结束语

在 MMOG 的服务器集群中,网络负载均衡诸多问题主要集中在负载均衡机制的有效性、通信机制的效率、集群 I/O 的一致性等几个方面上. 在实际应用中,由于它们的不可预测性,导致了在设计以及实现网络负载均衡集群时,尽量使用适合网络均衡条件的、比较简单的循环法或者最少连接算法. 当其效率不高或不能达到预期的负载均衡效果时,采用较为复杂的动态均衡算法. 但是无论采用基于简单算法还是负反馈动态权值的机制,都要尽量避免通信包的泛滥.

参考文献:

[1] RAJKUMAR BUYYA. High Performance Cluster Computing Architectures and System[M]. 北京: 人民邮电出版社, 2002.

[2] MOHAMMED A M IBRAHIM, LU XINDA. Utilization of Cluster of PCs for the Study of Dynamic Load Balancing[Z]. IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering, Beijing, China, 2002.

[3] DANTE TREGLIA 著. 游戏编程精萃 3[M]. 张磊 译. 北京: 人民邮电出版社, 2003. 424 - 425.

[4] 苏羽, 王媛媛. Visual C + + 网络游戏建模与实现[M]. 北京: 北京科海电子出版社, 2003. 13 - 16.

[5] MOHAMMED A M IBRAHIM, LU XINDA. Performance of Dynamic Load Balancing Algorithm on Cluster of Workstations and PCs[Z]. Fifth International Conference on Algorithms and Architectures for Parallel Processing, Beijing, China, 2002. 44 - 47.

[6] TA NGUYEN, BINH DUONG, ŻHOU SUIPING. A Dynamic Load Sharing Algorithm for Massively Multiplayer Online Games[Z]. The 11th IEEE International Conference, Sydney, Australia, 2003. 131 - 136.

Network Load-balance Algorithm Research of Massively Multiplayer Online Games

XU Guang-xia, ZHU Wei-hua, YANG Dan, TAN Ya-zhu

(College of Software Engineering, Chonqing University, Chongqing 400030, China)

Abstract: With the development of network game, Massively Multiplayer Online Games(MMOG) requires ever-higher levels of the servers. The Cluster of Servers and Network Load Balancing Algorithm is analyzed deeply. Combined with Dynamic Network Load Balancing Algorithm——Weighted Round-Robin Scheduling, an algorithm is presented to realize dynamic load balancing of Cluster of Servers under the circumstances of MMOG.

Key words: cluster; network load balancing; dynamic load balancing algorithm; weighted round-robin scheduling

(编辑 张 苹)