CS 结构下的 DVE 规模扩展方法

张 炯 雷小永 戴树岭

(北京航空航天大学 自动化科学与电气工程学院, 北京 100191)

摘 要: 针对分布式虚拟环境下的多服务器结构,将规模扩展问题转化为多节点与多服务器的最优分配问题,并提出了节点划分方案的评估方程.评估方程综合考虑系统负载、系统延迟,以达到对分配方案的有效评估.根据分布式虚拟环境实际应用的特点,将节点划分分为预分配和实时调节两个步骤.预分配通过密度优先策略和贪婪自适应搜索算法实现对预分配节点快速、有效地划分,利用评估方程对评估结果进行修订;实时调节严格限制节点分配的条件,达到实时分配的目的.相比于传统方法,很好的处理了分配的实时性和完备性之间的矛盾,具有较高的实用价值.通过实验比较,证明该方法的有效性,并得出结论.

关 键 词: 规模扩展; 节点分配; 预处理和实时处理

中图分类号: TP 391.9

文献标识码: A **文 章 编 号**: 1001-5965(2010)12-1422-05

Method of scale extension in distributed virtual environment under CS structure

Zhang Jiong Lei Xiaoyong Dai Shuling

(School of Automation Science and Electrical Engineering, Beijing University of Aeronautics and Astronautics, Beijing 100191, China)

Abstract: According to the client server (CS) architecture in distributed virtual environment (DVE), it extended the problem of scale extension into the optimization of node partition between multi-node and multi-server. An evaluation equation of the node partition was also proposed. Based on the characteristics of DVE application, the whole process was divided into two steps, predisposition and runtime regulation. The process of predisposition was achieved partition rapidly and efficiently with density primary strategies and greedy randomized adaptive search procedures(GRASP), and the evaluation equation was used to adjust partition result. In real-time regulation process, the condition of node partition was restricted strictly to achieve real time distribution. Compared with the traditional methods, this method has better solved the contradiction between the real-time quality and the maturity of distribution, and has been proved to be of great practical value. The experimental results illustrate the effectiveness of this method and a conclusion was drawn.

Key words: scale extension; node partitioning; predisposition and real-time regulation

在网络交互迅速发展的当今社会里,只能提供几十或上百用户同时交互的 DVE(Distributed Virtual Environment)系统显然远远不能满足需求,规模扩展已成为 DVE 系统需要解决的一个关键性问题.在多服务器的结构下规模扩展的问题可以归结为:解决节点和服务器之间资源分配的问题,即"Partitioning Problem"[1]节点划分问题.目前节点的划分可分为全局策略及局部策略两种.

全局策略考虑系统整体的负载平衡度,力求达到系统最优.文献[1]提出一种全局负载评估公式;文献[2]采用 GRASP(Greedy Randomized Adaptive Search Procedures)进行节点划分处理.全局方法可以实现系统节点的最优分配,但由于

收稿日期: 2009-10-14

作者简介: 张 炯(1981—),男,陕西汉中人,博士生,zj572_1981@163.com.

计算过于庞大,缺乏实时性.局部策略只针对超出负载极限的服务器,进行节点再分配.文献[3]利用局部策略,降低了计算复杂度.文献[4]提出了密度划分法则.局部方法只将服务器承载节点数目作为负载计算的依据,缺乏全面性.

针对两种策略的优缺点,本文构建全面的负载评估方程,提高分配效率,达到规模扩展的目地.

1　节点划分评估方程

1.1　系统负载

系统的总负载包含计算负载和通信负载.计算负载是系统中节点对服务器造成 CPU、内存资源的消耗.计算负载 C_{wcp} 表达式为

$$C_{\mathrm{wcp}}=\sum_{i=1}^{m}\left|\sum_{a_j\in Z_i}[W_{\mathrm{cp}}(a_j)+W_{\mathrm{cmp}}(a_j,S_i)]-w^*\right| \tag{1}$$

式中,W_{cp} 为单个节点对管理服务器带来的计算负载;W_{cmp} 为单个节点对管理服务器由于通信过程带来的额外计算负载;m 为系统服务器数量;Z_i 为属于服务器 i 的节点集合;a_j 为第 j 个节点;S_i 为服务器;$w^*=\sum_{j=1}^{n}[W_{\mathrm{cp}}(a_j)+W_{\mathrm{cmp}}(a_j,S_i)]/m$ 为系统平均负载.对于任意服务器应当满足计算负载总和不超出其计算负载上限 R_{cp},即满足:

$$R_{\mathrm{cp}}\geqslant\sum_{a_j\in Z_i}[W_{\mathrm{cp}}(a_j)+W_{\mathrm{cmp}}(a_j,S_i)] \tag{2}$$

通信负载是节点不同划分结果对服务器之间的通信负载影响,通信负载 C_{wcm} 为

$$C_{\mathrm{wcm}}=\sum_{0<i<m}\sum_{0<j<m}W_{\mathrm{sm}}(S_i,S_j) \tag{3}$$

式中,W_{sm} 为相互通信的服务器之间的通信负载;S 为服务器集合.对于任意服务器应当满足通信负载总和不超出其负载上限 R_{cm},即满足:

$$R_{\mathrm{cm}}\geqslant\sum_{a_j\in Z_i}W_{\mathrm{cm}}(a_j)+\sum_{0<j<m}W_{\mathrm{sm}}(S_i,S_j) \tag{4}$$

式中,W_{cm} 为节点通信产生的通信负载.

计算负载和通信负载并不同性,例如:节点呈集中式分布时,平均分配节点可以减小计算负载,但会极大增加系统的通信负载.故系统总负载 C_{w} 定义为

$$C_{\mathrm{w}}=C_{\mathrm{w1}}C_{\mathrm{wcp}}+C_{\mathrm{w2}}C_{\mathrm{wcm}} \tag{5}$$

式中,C_{w1},C_{w2} 为非负系数,$C_{\mathrm{w1}}+C_{\mathrm{w2}}=1.0$.当服务器资源较少,服务器计算承载能力低(CPU,内存等限制较大),适当增大系数 C_{w1} 的值;当系统网络环境差,带宽不足,通信负载是成为制约系统性能的主要因素,则应当提高系数 C_{w2}.

1.2　系统延迟

网络延迟是造成系统时空不一致,影响用户体验感的重要原因.在多服务器结构下,节点之间的延迟大小与节点划分的结果具有相关性,信息传递路径越长,延迟的可能性就越大.本文称系统中数据延迟大于延迟极限的节点数目为系统延迟度 C_{delay},表达为

$$C_{\mathrm{delay}}=\sum_{i=1}^{m}\sum_{i=2,j>i}^{m}x(i,j) \tag{6}$$

式中,$x(i,j)=\begin{cases}0 & t_{\mathrm{d}}(i,j)<D\\ 1 & t_{\mathrm{d}}(i,j)\geqslant D\end{cases}$,$D$ 为延迟极限,表示用户可以忍受的延迟最大值,$t_{\mathrm{d}}(i,j)$ 为节点 i,j 之间的通信延迟时间.

1.3　系统评估方程

评估方程由 C_{w} 和系统延迟度 C_{d} 决定.因为 C_{d} 和 C_{w} 是不同类型的数据,不能简单加和,本文采用负载和延迟度的变化率作为判定依据.假设节点分配前后的负载和延迟率分别为 $C_{\mathrm{w}}^{\mathrm{o}}$,$C_{\mathrm{d}}^{\mathrm{o}}$ 和 $C_{\mathrm{w}}^{\mathrm{n}}$,$C_{\mathrm{d}}^{\mathrm{n}}$.$\mathrm{d}C_{\mathrm{w}}$ 和 $\mathrm{d}C_{\mathrm{d}}$ 表示负载和延迟度的变化率,则系统节点划分评估 C 的表达式为

$$\left.\begin{aligned}\mathrm{d}C_{\mathrm{w}}&=\frac{C_{\mathrm{w}}^{\mathrm{n}}-C_{\mathrm{w}}^{\mathrm{o}}}{C_{\mathrm{w}}^{\mathrm{o}}}\\ \mathrm{d}C_{\mathrm{delay}}&=\frac{C_{\mathrm{d}}^{\mathrm{n}}-C_{\mathrm{d}}^{\mathrm{o}}}{C_{\mathrm{d}}^{\mathrm{o}}}\\ C&=w_1\,\mathrm{d}C_{\mathrm{w}}+w_2\,\mathrm{d}C_{\mathrm{d}}\end{aligned}\right\} \tag{7}$$

式中,w_1,w_2 为非负系数且 $w_1+w_2=1.0$,当 C_{w} 较小,带宽高,连接的节点较少,增大 w_2,增加 $\mathrm{d}C_{\mathrm{d}}$ 比重;当 C_{w} 大,连接节点多时,增大 w_1,增加 $\mathrm{d}C_{\mathrm{w}}$ 的比重.评估结果 $C<0$,表明新的划分结果优于旧的划分结果,系统采用新划分方式;相反 $C>0$ 时,系统保持原有划分方式.

2　节点划分策略

节点划分问题属于 NP-Hard[2] 问题,很难实现对所有分配结果进行评估.在实际应用中,节点不会同时加入系统,一般是初始化时部分节点连入系统,在运行过程中节点动态加入或退出.因此本文节点分配方法采取两步骤,分别为初始分配阶段和实时调节阶段.在初始阶段进行快速的节点分配,获得近似最优的分配结果.在实时调节阶段根据节点的加入、退出或位置改变,实时调整分配结果.

2.1　初始阶段分配法则

IPP (Initial Partition Phase)是对系统预设的所有节点进行的预分配,应当达到如下目标:

1) 预设节点完全分配完毕;

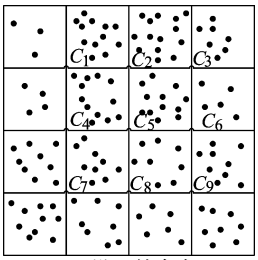
2) 分配结果达到或逼近最小系统负载.避免运行时阶段做过多的调整;

3) 初始阶段虽然属于"offline"过程,实时性要求不高,但为避免体验者长时间等待,初始阶段也要尽量减小时间复杂度.

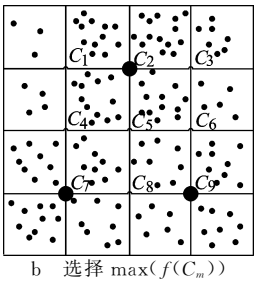
本文在初始分配采用密度优先分配以及GRASP进行初始节点的划分.

密度优先分配核心是选择节点密集的区域,把该区域内的节点划分至同一服务器之下.因为节点的交互对象绝大多数集中在其AOI(Area Of Interest)范围之内,将密度集中的节点分配至同一服务器下,可有效地减少系统的通信负载和计算负载.

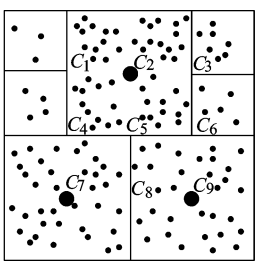
密度优先分配过程是:将节点有效虚拟区域划分为若干等面积的矩形区域,矩形区域的数量一般设置在$(4\times m,6\times m)$范围内,设定区域的结合点从左至右,从上至下依次为$C_1,C_2,\cdots,C_n$,边界点除外,如图1a.每个结合点相邻的4个区域内的节点数目为$f(C_m)$,选择$\max(f(C_m))$的结合点如图1b,并将其周围的4个区域中的节点放到统一的服务器下管理.重复上述过程,直到服务器分配完毕,如图1c.



a 设置结合点



b 选择$\max(f(C_m))$



c 划分至同一服务器

·—节点;•—$\max(f(C_m))$;C_2—结合点.

图1 密度优先策略划分示意

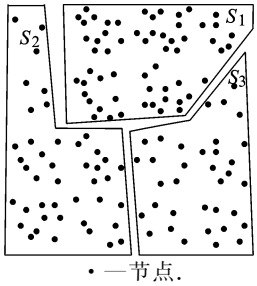
经过密度优先分配后,仍有一定的节点未分配,对于未分配的节点采取GRASP方法,利用式(7)作为迭代评估公式,将剩余节点划入服务器管理.

GRASP过程为:①构建算法的可行解.假设经过密度分配后剩余的节点数为nass,从中任取一个节点并将其归入任意的服务器之下,通过式(7)可计算该分配方式下的系统C.重复上述过程,计算每一个节点分配到每一个服务器上新产生的系统平衡度,并将结果存入列表ListnassC.列表存储方式为{节点编号,服务器编号,C},分配结束后列表共有nass$\times m$个元素.按照C从低到高进行排序后,截取整体的$1/m$作为其子序列ListTopnassC.②进行领域搜索部分.从ListTopnassC取得任意一个元素,获得节点编号,将该节点归入该服务器管理,并从列表中删除该元素.从与该节点发生AOI区域重叠的节点中,挑选任意节点并依次分配给各个服务器,选取系统C结果最小的节点,进行服务器分配,并删除ListnassC或ListTopnassC中对应的元素,重复该过程直到AOI区域重叠的节点完全分配完毕.回到领域搜索的开端,重复上述过程,直到ListTopnassC为空.重新执行步骤①,从ListnassC中重新构建ListTopnassC列表,重复领域搜索,直到节点完全分配.GRASP的核心代码如下:

```
Get not assigned node=nass;
For(i=0 to i=nass)  do
    {For(j=0 to j=m) do
    {Assignserver(i,j) then compute C;
    Insert (i, j, C) to ListnassC cancel assign i;}}
QuickSort(ListnassC) then get ListTopnassC;
Random get element from ListTopnassC;
While(have node i ∈ node-a && not assigned )
{ For(i∈node's AOI && i not assigned)
  {For(j=0 to j=m)
    { Assignserver(i,j) then compute C;
    tmpC=C;
    If(min_C> tmpC)
      { min_C= tmp_C; savei=i, savej=j;}
    cancel assign i;}}
    Assignserver(i,j)}
```

上述的贪婪搜索过程可以反复迭代,每次迭

代的结果得到的系统 C 不同,可经过多次迭代后,选择最优的分配方案,作为系统的初始分配方案.图 2 是剩余节点通过 GRASP 分配完得到的初始化分配结果,共 136 个节点,3 台服务器.



·—节点.

图 2　GRASP 划分示意

2.2　实时调节分配法则

RPP (Runtime Partition Phase) 在运行过程中,根据节点的变更,动态调整分配结果,实现系统分配优化.该过程属于"Online"阶段,必须满足实时分配.

运行时分配在初始分配的基础上,针对新加入节点或与非自身管理服务器下的其他节点发生信息交互对象变化的节点,进行再分配.

对于符合上述规定的节点,RPP 将节点依次分配至各个服务器,同时通过式(7)计算不同分配方案下的系统 C,选择系统 C 最小的方案作为实际方案结果.

负责节点分配的计算机在结构上独立于用户节点,只与各个服务器进行数据交互.在获取信息时,采用增量式获取方式,只传输可能改变系统负载的信息,减少了信息数据量,因此节点分配计算机给系统带来的负载可忽略不计.

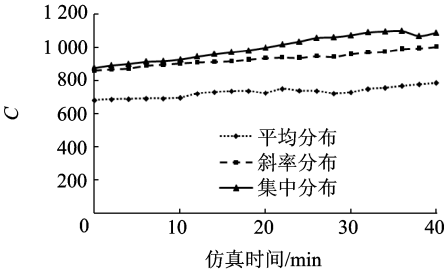
RPP 复杂度分析:对于每一个等待分配的节点,计算复杂度为 $O(m)$.假设同一时刻等待分配的节点数为 k,则复杂度为 $O(mk)$.在实际应用中,同一时刻符合运行时分配原则的节点 k 远远小于系统节点总数 n,因此 RPP 完全能达到实时性的要求.

3　实验及结果分析

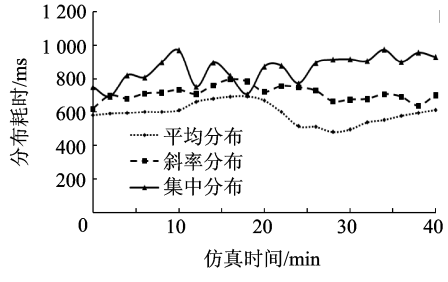
针对本文提出的方法,设计两部分的测试:①测试本文方法的实时性和稳定性,以及对负载的平衡作用;②测试采用本文方法,系统可以承受的最大节点数目.参数如下:节点数目:初始 1 500,每分钟增加 80;服务器数目:6;AOI:4 km;延迟极限 D:40 ms;节点分布方式:平均,斜率递增,集中;测试时间:40 min;测试次数:25;$W_{cp}=9$;$W_{cm}=7$;$R_{cp}=50\,000$;$R_{cm}=50\,000$;$C_{w1}=0.8$;$C_{w2}=0.2$;$w_1=0.7$;$w_2=0.3$.

3.1　测试 1 结果

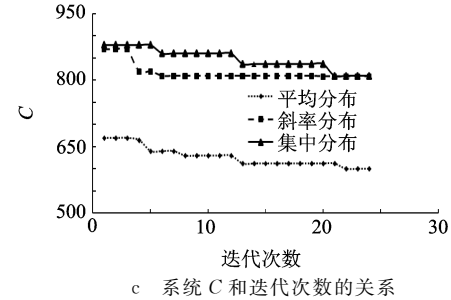
图 3 是测试 1 的结果,图中的数据结果是 25 次测试结果的均值.图 3a 可以看到,随着时间的增长,节点增多,系统 C 增长十分缓慢,表明本文方法优化了系统 C,使之趋于平衡.图 3b 是 RPP 阶段的分配耗时,可以看到,在任何情况下,分配耗时的均值不超过 45 ms,满足实时的要求.图 3c 是 GRASP 多次迭代的结果,迭代次数越多,负载平衡性越好.



a　系统 C 随仿真时间变化



b　RPP 分配耗时

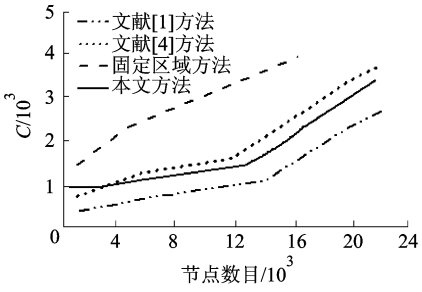


c　系统 C 和迭代次数的关系

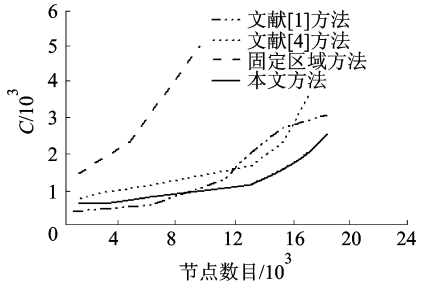
图 3　测试 1 结果

3.2　测试 2 结果

测试 2 在节点平均分布、集中分布的状态下,对比各种方法系统可以承载的最大节点数目,测试结果如图 4 所示.在平均分布下,文献[1,4]以及本文方法都可达到 20 000 节点,文献[1]的方法系统 C 最小;集中分布下,文献[1]和本文方法的最大承载节点都在 18 000 左右,文献[4]的节点数只有 16 700.文献[1]的方法因为其分配时间长,很难有实际应用价值,而本文方法是在实时的过程中完成的,并可达到很好的效果,具有实际应用价值.



a 节点平均分布



b 节点集中分布

图4 测试2结果

4 结束语

本文利用负载评估方程,通过预处理和实时调节方法解决了DVE系统在多服务器模式下的节点划分问题,方法不仅满足了DVE系统对划分实时性的要求,并有效地降低系统的负载度.实验证明该方法具有较强的实用价值,可有效的扩大DVE系统的规模.

参考文献(References)

[1] Lui J C, Chan M. An efficient partitioning algorithm for distributed virtual environment systems [J]. IEEE Trans Parallel and Distributed Systems, 2002, 13(3): 193－211

[2] Morillo P, Ferna'ndez M. A GRASP-based algorithm for solving DVE partitioning problem [C]//Proceedings of the International Parallel and Distributed Processing Symposium. Los Alamitos: IEEE Computer Society, 2003: 60－68

[3] Lee K M, Lee D M. A scalable dynamic load distribution scheme for multi-server distributed virtual environment systems with highly-skewed user distribution [C]// Virtual Reality Software and Technology Proceedings of the ACM symposium on Virtual reality software and technology. New York: ACM, 2003: 160－168

[4] Hu Xiaomei, Zhai Zhengjun, Cai Xiaobin. A task-based clustering method for the dynamic partitioning management in CVE systems [J]. Wireless and Mobile Computing, Networking And Communications, 2005(4): 139－144

(编 辑:刘登敏)

(上接第1406页)

参考文献(References)

[1] 姚昌仁,唐国梁,宋廷伦. 火箭导弹发射动力学[M]. 北京:北京理工大学出版社,1996:6－7
Yao Changren, Tang Guoliang, Song Tinglun. Launching dynamics for rockets or missiles[M]. Beijing: Beijing Institute of Technology Press, 1996: 6－7(in Chinese)

[2] Toniolo M D, Tartabini P V, Pamadi B N, et al. Constraint force equation methodology for modeling multi-body stage separation dynamics[R]. AIAA-2008-219, 2008

[3] Tartabini P V, Roithmayr C, Toniolo M D, et al. Verification of the constraint force equation methodology for modeling multibody stage separation[R]. AIAA-2008-7039, 2008

[4] 薛明旭. 战术导弹发射动力学与仿真[D]. 西安:西北工业大学航天学院,2004
Xue Mingxu. Launch dynamics of tactical missile and simulation[D]. Xi'an: College of Astronautics, Northwestern Polytechnical University, 2004(in Chinese)

[5] 周慧钟,李忠应,王瑾玫. 有翼导弹飞行动力学[M]. 北京:航空专业教材编审组,1983:11－19
Zhou Huizhong, Li Zhongying, Wang Jinmei. Winged missile flight dynamics [M]. Beijing: Aeronautics Professional Teaching Material Edit Group, 1983: 11－19(in Chinese)

[6] 邵伟龙,陈万春. 吸气式导弹分离动力学建模与仿真分析[C]//2006年中国飞行力学年会论文集. 黄山市:中国宇航学会空气动力学与飞行力学专业委员会,2006:221－225
Shao Weilong, Chen Wanchun. Modeling, simulation and analysis of the air-breathing missile separation dynamics [C]//2006 China Flight Dynamics Annual Conferences. HuangShan: Chinese Society of Astronautics Aerodynamics and Flight Dynamics Committee, 2006: 221－225(in Chinese)

[7] 肖业伦,陈万春. 飞行器相对姿态运动的静力学、运动学和动力学方法[J]. 中国空间科学技术,2003,10(5):10－15
Xiao Yelun, Chen Wanchun. Statical, kinematical and dynamical approaches for treating the relative attitude motion of flight vehicles[J]. Chinese Space Science and Technology, 2003, 10(5): 10－15(in Chinese)

(编 辑:刘 晔)