分布式虚拟环境中的一种场景动态划分算法

——基于竞争的演化算法

A Partitioning Algorithm for a Distributed Virtual Environment

范 哲 顾毓清 陶占红

（中科院软件研究所 北京100080）

Abstract With the development of networking technologies and computer graphics, distributed virtual environment (DVE) has become an important research area in computer science. In a DVE system, clients at different places can simultaneously explore a virtual world and interact with each other. How to build an efficient DVE system that can support large number of concurrent users is a challenge. In the paper, we discuss the way of partitioning the DVE world to balance the workload among different servers as well as minimize the inter-server communication. We present the existent mathematics mode and algorithms at first, and then give our algorithm, partitioning algorithm based on competition. Experiments are carried out to illustrate the effectiveness of our proposed algorithm.

Keywords Distributed virtual environment, Virtual reality, Dynamic partitioning, Workload cost, Communication cost, Competition

1. 简介

虚拟环境，也称为虚拟现实，是一种完全沉浸式的计算机交互界面。用户好像真的处在计算机生成的世界中，所见、所闻、所感都像在真实世界里一样，而且用户还可以通过完全自然的方式向计算机发出命令。

分布式虚拟环境（简称 DVE）结合了虚拟环境和计算机网络技术。多个用户通过联网计算机，进入一个虚拟的3维世界，相互交流和作用[1]。DVE 可应用在虚拟战场游戏、电子教室、虚拟商店、虚拟医院、乃至虚拟社会等，其发展很可能给人们工作、生活、交流的方式带来巨大的变革。目前，快速环境建模和实时图形绘制，高速计算机网络，分布式数据库系统，人机交互等各项技术的发展，使得创建 DVE 系统已成为可能。然而，设计和实现一个高性能、实际可用的 DVE 系统是一项具有挑战性的工作，这其中有一些重要的问题，譬如：

·对象的一致性 虚拟世界中同一个3维物体（或称对象）在不同用户的观测下状态的一致性。对多个用户操纵同一物体的情况需要引入比较复杂的并发控制机制。

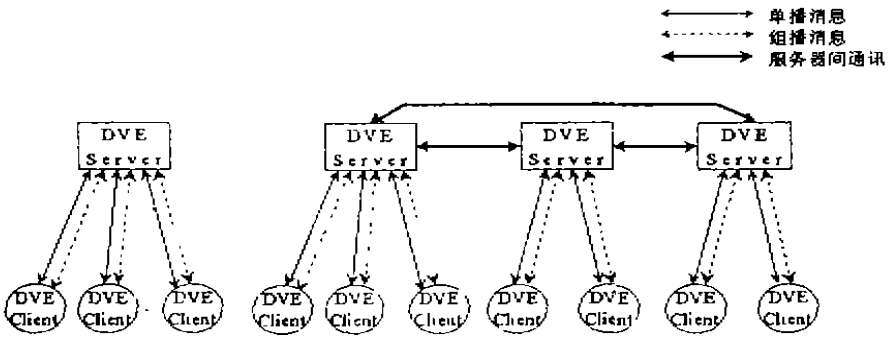

图1 两种系统结构

·观测的一致性 用户在虚拟世界中的活动应该连续地通知环境中的其他用户，以保证各个用户观察的一致性。用户很多情况下，会需要很大的通讯带宽。组播技术有助于降低这种带宽需求。

·平衡工作量与减少通讯量 一般在DVE系统的实现中，用户的活动发送给服务器，由它来计算然后将环境的改变广播给所有用户。DVE系统的两种结构如图1所示：单服务器DVE(SSDVE)和多服务器DVE(MSDVE)。若要支持大量用户，一般须采用多服务器结构。在一个MSDVE中，虚拟世界的整个区域被划分为多个分区，每个分区分别交给一台相应的服务器处理，从而划分了工作量，然而由于分区与分区交界处的影响，不可避免地又额外增加服务器与服务器之间的通讯量。因此，如何划分分区，以均衡地分配各服务器工作量，并将服务器间的通讯维持在较低的水平上，关系到DVE系统的性能和效率，也是本文要讨论的中心问题。

香港中文大学的John C.S.Lui，M.F Chan，OldField K.Y.So，T S Than在文[2]中，提出了场景划分的数学模型和Bisection划分算法。本文基于他们的数学模型，提出一个新的算法——基于竞争的演化算法。

2. DVE系统的数学模型

2.1 主要概念

Avatar 是虚拟世界中活动的一个3D对象，它代表用户在3D虚拟世界中所扮演的角色。

AOI：Area of Interest 术语AOI是指一个以Avatar的位置为中心的一个圆形区域，代表了Avatar当前能观察到的范围。

若一个Avatar处在另一个Avatar的AOI区域中，为了保证观测的一致性，前者的活动应当影响后者的所见，若这两个Avatar又分别处在两个不同的分区的话(分别在相邻两个分区的边缘)，则会引起对应的两台服务器之间的通讯。图2中，A1与A2的活动会互相影响对方的观测；而A1、A2的活动却不需要通知给A3。

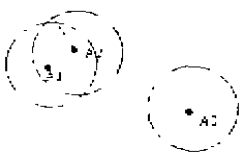

图2 Avatars与AOI

Cell 为方便程序实现，整个虚拟世界被划分为许多S * S的正方形单元，称为Cell。这种划分对用户来说是透明的，实现中，Cell的划分简化了对服务器间通讯量的估计。

假设各Avatar的AOI大小都相等，选取Cell的尺寸S，使：S/2<Radius(AOI)<S。

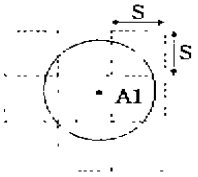

图3 Cell and AOI

从图3可看出，Avatar的AOI范围总是被它所处的Cell及邻近的八个Cell所包含。因此，判断两个Avatar是否会互相影响的问题化简为判断它俩是否处在同一个或者相邻(包括斜线相邻)的两个Cell中。

2.2 图表示模型

DVE世界可用一个有权图G=(V,E)来表示，其中V是节点集合，E是边集合。

1)每一个Cell用一个节点u_i来表示，i为下标。

2)节点u_i的权值W_i为该Cell的工作量，令它等于u_i中Avatar的数目(或预计数目)。

3)如果Cell u_i和u_j相邻(包括斜线相邻)，则图G中相应地有一条边E_{ij}连接u_i和u_j。

4)边E_{ij}的权值L_{ij}代表u_i和u_j之间引起的服务器通讯量的大小。令

$$\begin{cases} L_{ij}=W_i+W_j, & \text{若 } u_i,u_j \text{ 不在同一分区} \\ L_{ij}=0, & \text{若 } u_i,u_j \text{ 在同一分区} \end{cases}$$

图4为一个DVE的图表示模型示例。

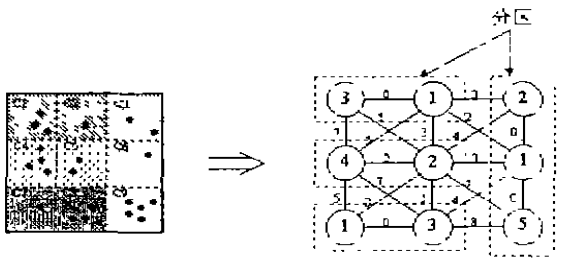

图4 DVE的有权图模型

3. 问题描述

符号定义：N--整个虚拟环境中的Cell数；P--分区数(也即服务器的数目)；n--整个虚拟环境中的Avatar数目；W_i--Cell u_i中的工作量(即u_i中的Avatar数)，i=1,2,…,N；L_{ij}--Cell u_i与u_j之间引起的服务器通讯量，1<=i,j<=N；w_i--一个非负数，代表平衡各服务

器工作量的重要性权值；w_2---一个非负数，代表减少服务器之间通讯量的重要性权值（注：$w_1+w_2=1.0$）；V_i--分区 i，$i=1,2,\cdots,P$；p--划分格局（$V_1,V_2,\cdots,V_P$），对图 $G=(V,R)$，p 将它划分为 P 个分区，对应顶点集分别为 V_1、V_2、…、V_P。当 $i\neq j$，$V_i\cap V_j=\Phi$；$\bigcup_{i=1}^{P}V_i=V$；C_p^W--划分 p 的工作量平衡代价；C_p^L--划分 p 的服务器通讯量代价；C_p--划分 p 的总代价。

对一个划分 p，工作量平衡代价表示为：

$$C_p^W=\sum_{j=1}^{P}\left|\sum_{u\in V_j}W_i-n/P\right| \tag{1}$$

式(1)中，n/P 是工作量完全平衡时每个分区的工作量。$\sum_{u\in V_j}W_i$ 是分区 j 的实际工作量。所以，工作量平衡代价就是划分 p 下，各分区工作量与理想情况下偏差的总和。

下面定义函数 ADJ 用来判断 Cell u 与分区 V_i 是否相邻：

$$ADJ(u,V_i)=\begin{cases}1 & \text{如果 } u\notin V_i\text{；且}\exists\ v\in V_i\text{、}\\ & \quad u\text{ 和 }v\text{ 在图 G 中有一条边相连}(E_{uv}\in E)\text{；}\\ 0 & \text{否则。}\end{cases}$$

给定一个划分格局 p，定义 C_{ij} 为分区 V_i 与分区 V_j 之间的通讯量，其表示为：

$$C_{ij}=\sum_{u\in V_i}W_u*ADJ(u,V_j)+\sum_{u\in V_j}W_u*ADJ(u,V_i) \tag{2}$$

令：

$$C_p^L=\sum_{i=1}^{P}\sum_{j>i}^{P}C_{ij} \tag{3}$$

$$C_p=w_1*C_p^W+w_2*C_p^L \tag{4}$$

求解的最终目标就是要找到一个划分格局 p^*，使之满足：

$$C_{p^*}=\min_p(C_p) \tag{5}$$

可以证明，这是一个 NP 完全问题。若使用穷举法求解，其时间复杂度为 $\Theta(P^N)$。

4. 场景划分算法

4.1 基线划分算法(BP)

BP 算法最简单，它假定 Avatar 在整个虚拟世界中基本上均匀分布，这样的话，要平衡工作量，就要给各分区分配相同的面积。同时，为了减少分区之间通讯量，还应该考虑分区的形状。直观上，分区的交界处引起服务器间的通讯，因此应该使得各分区的边界尽可能地短。

给定面积，圆的周长将比其他所有图形都小。然而，是无法用一些不相交的圆来覆盖一个区域的，所以自然的设想是使分区呈六边形，如图5。蜜蜂把蜂窝建成许多六边形格以使用最少的材料（与这些六边形的周长之和成正比），获得最大的空间。

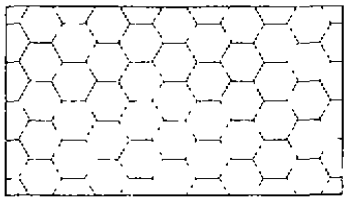

图5 用六边形覆盖区域

因为 DVE 系统由正方形的 Cell 组成，不可能拼凑出完全的六边形，所以比较方便的做法是还是使用正方形分区，给定区域面积 A，若使用面积为"a"的正方形作为分区，则一个分区的周长为 $4\sqrt{a}$，总周长为 $4\frac{A}{a}\sqrt{a}$。若使用面积为"a"的六边形，一个分区的周长为 $\sqrt{\frac{24a}{\sqrt{3}}}$，那么总周长为 $3.72\frac{A}{a}\sqrt{a}$，可见使用正方形与六边形相比通讯量代价相差7%。

做法：若服务器数目 P 不是个完全平方数时，则使用一个比 P 小的完全平方数 P' 划分大部分区域，剩下部分 Cell 则平均分配给剩下的服务器。其算法时间复杂度为 O(1)，图6中示范了一个 5 * 5 Cells 的 DVE 世界，P=5情况下 BP 算法的划分结果。

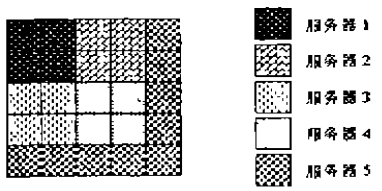

图6 BP 算法划分结果

4.2 二分划分算法(Bi-P)[3]

实际上，在一个虚拟世界中，Avatar 更多可能是不均匀分布的[3]，BP 算法对于这种情况已不再适合，这时可以使用基于二分法的 Bi-P 算法。

假定 Cell 数为 N，不失一般性，先考虑服务器数 P=2的情况。

令 V_i^k 为划分给第 k 个服务器的分区，它包含 m 个 Cell。首先令：

$$V_1^N=V=\{u_1,u_2,\cdots,u_N\};\quad V_2^0=\Phi \tag{6}$$

令 p_i 为第 i 个划分格局，其总代价为 Cp_i。初始格局 p_0 为 (V_1^N,V_2^0)，其总代价为 Cp_0。从 p_i 格局开始，在 V_1^N 分区中选择一个 Cell，交给分区 V_2^0，得到一个新的格局 p_{i+1}。需注意的是，选中的这个 Cell 是第一个分区所有 Cell 中，移到第二个分区后使得 Cp_{i+1} 最小的 Cell。这个选择过程相对 Cell 数 N 需要线性时间。

类似地，对 $i=0,1,\cdots,N-1$，由 $p_i=(V_1^{i-1},V_2^i)$ 可得：

$$p_{(i+1)}=(V_1^{i-1}-\{u_j\},V_2^i\cup\{u_j\})$$

其中 $u_j\in V_1^{i-1}$，且使得 $Cp_{(i+1)}$ 最小。当 $P=2$ 时 Bi-P 算法的目标为选择其中的一个二分格局 p_i 作为结果，它满足：

$$Cp_i=\min_{0\le j\le N}\{Cp_j\} \qquad (7)$$

而对于一个任意数 P，我们可以先对整个区域做前面所述的二分划分，以后每次找一个工作量最大的分区，以它为区域再做二分，重复 P－1 次，最后得到的一个划分为求解结果。该算法的时间复杂度为：

$$O(\sum_{i=0}^{\lceil\log_2 P\rceil-1}\frac{N}{2^i})$$

4.3 竞争划分算法(CP)

在已有模型和算法的基础和启示下，考虑一个新的划分分区的思路，把人工智能中进化计算[4]的思想类似应用到分区划分中来：首先初始化一个划分格局，然后利用一定的规则让各个分区不断地互相争夺区域从而改变各自大小和形状，这样经过一段时期的演化，最后达到一个比较稳定的划分状态。

需要设计特定的竞争规则，使得最后的划分结果总代价较小。

这里定义分区的重心为以 Cell 的工作量作为密度，分区区域的重心。前面介绍 Baseline 算法时已提到了图形形状与其面积边界长比的关系。因此，在本算法中，每次竞争总是使大的分区倾向于失去离自己重心最远的 Cell，小的分区则倾向于争夺离自己重心最近的 Cell，这样都有一种变圆的倾向。可以想象，演化的过程中，各个分区都有获得平均的工作量、并且收缩自己边界的趋势。预计这样得到的最终划分将趋于稳定、且总代价将较小。

(竞争演化算法1，CP1)

1. 初始化一个划分格局 p_0，并令 $p=p_0$，$i=0$。
2. 对于当前格局 p_i：
   1)找出工作量最大的分区 V_e；挑选 V_e 中离其重心最远的 Cell u，将 u 从 V_e 中删除，并入邻近它的工作量最小的分区；
   2)找出工作量最小的分区 V_f，
   挑选不属于 V_f，且离 V_f 的重心最近的 Cell v，
   将 v 从原来的分区中删除，并入 V_f。
   这样得到新的格局 p_{i+1}，
3. 如果 $Cp_{i+1}<C_p$，则 $p=p_{i+1}$。
4. 若 p 的划分已经令人满意，则选择划分 p 作为结果，结束。
   否则，$i=i+1$，回到第2步继续。 □

试验发现，CP1算法结果比预想要差。其服务器通讯代价 C_p^1 不如原来设想的那样理想，使得总代价 C_p 还是比较大。

这里分析其原因。从宏观上看，变圆的倾向使得每个分区趋向于规则、匀称的图形，理应减少了总的边界，然而，如图7所示，DVE 世界其实是划分为 Cell 的，斜线和曲线其实都是由折线组成的，其长度比理想中的长度实际上要大；同样地，圆周的实际长度与它的外切正方形周长相等，而该正方形面积显然大于圆。所以从直观上可看出，考虑到 Cell 的划分，要真正减小边界总长，应该尽量使得边界多为竖线和横线，分区趋向于方形(而不再是圆形)。

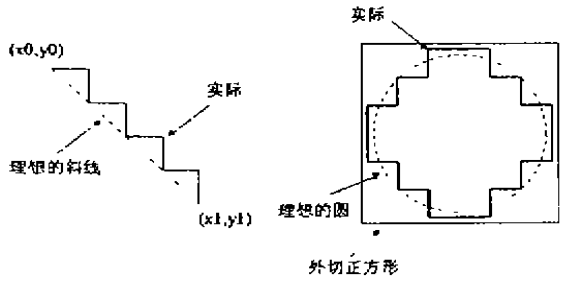

图7 基于 Cell 时，实际与理想的偏差

由于划分 Cell 的影响，还应该尽量消除边界的凹凸，图8(a)中，假设分区 V_1 是工作量最大的分区，可见 u_1 是其中离重心最远的 Cell。CP1算法首先将 u_1 从 V_1 中划出并入到邻近的分区，以减少 V_1 的工作量。然而，V_1 失去 u_1 后，其边界长度没有变化。若改进一下算法，使其首先将凸出的 Cell u_2 从 V_1 划出，则 V_1 的边界将减小2S，同样，工作量最小的分区争夺的 Cell 也不仅仅需要离自己重心近，而且应优先选择凸入自己区域的 Cell。如图8(b)中，将 u_2 并入 V_2 比将 u_1 并入 V_2 更好。下面的 CP2算法将针对划分 Cell 的实际情况进行改进。

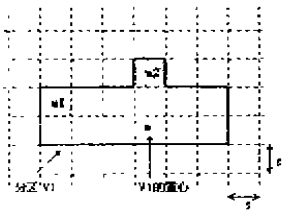

图8(a)

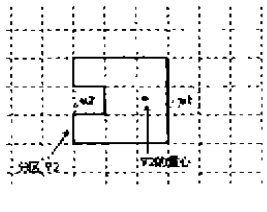

图8(b)

假设一个 K * K 的 DVE 世界，令 DADJ(u)为整个 DVE 世界中，与 Cell u 直接相邻(上下左右相邻，不包括斜线相邻)的 Cell 数：

$$DADJ(u)=\begin{cases}2,u=(1,1)(1,K)(K,1)或(K,K) & \text{即四个角上的 Cell}\\ 3,u=(1,y)(x,1)(K,y)或(x,K) & \text{即边上的 Cell}\\ \quad x\neq 1,K\ 且\ y\neq 1,K & \\ 4,其它 & \end{cases}$$

令 VADJ(u,V_i)为分区 V_i 中，与 Cell u 直接相邻（上下左右相邻，不包括斜线相邻）的 Cell 数。

（基于 Cell 的改进：竞争演化算法2，CP2）

1. 初始化一个划分格局 p_0，并令 $p=p_0$，$i=0$。
2. 对于当前格局 p_i：
   1)I)找出工作量最大的分区 V_e；
   II)对 V_e 中所有的 Cell u_k，计算 VADJ(u_k,V_e)/DADJ(u_k)的值；
   III)从该值最小的 Cell 集合中，挑选离 V_e 的重心最远的 Cell u；
   IV)从与 u 直接相邻的分区集合中，挑选 VADJ(u,V_j)/ DADJ(u)最大的分区 V_j；
   V)将 u 从 V_e 删除，并入 V_j；
   2)I)找出工作量最小的分区 V_f；
   II)对不在 V_f 中的所有的 Cell u_k，计算 DADJ(u_k,V_f)/ DADJ(u_k)；
   III)从该值最大的 Cell 集合中，挑选离 V_f 的重心最近的 Cell v；
   V)将 v 从原来的分区中删除，并入 V_f。
   这样得到新的格局 p_{i+1}。
3. 如果 $Cp_{i+1}<Cp$，则 $p=p_{i+1}$。
4. 若 p 的划分已经令人满意，则选择划分 p 作为结果，结束。
   否则，$i=i+1$，回到第2步继续。 □

实际实现时，为了防止死循环的出现（比如两个分区反复争夺一个 Cell 的情况），还需引入一些变异，比如在第1)I)步中，在一个很小的概率下，不是找出工作量最大的分区，而是次大的分区。同样地，在1)III)，2)I)，2)III)都可以做类似的变异。

CP2算法的服务器间通讯量代价比 CP1明显降低。

5. 实验比较

实验中令 DVE 世界为25 * 25个 Cell，Avatar 数为2500，令 $w_1=w_2=0.5$。为模拟实际上可能的 Avatar 分布，采用三种方法来产生 Avatar 在虚拟世界中的位置。

·均匀分布　令每个 Avatar 的位置(x,y)，x 值、y 值都为[0，25] 之间的随机数。

·倾斜分布　将2500个 Avatar 分为4个同样大小的组，G_i，$i=1,2,3,4$。

第 i 组中的 Avatar 的位置(x,y)，x 值、y 值都为[0,i * 25/4]之间的随机数。

·族状分布　首先随机产生 M 个点(x_1,y_1)，(x_2,y_2)，…，(x_M,y_M)，将 Avatar 分为 M 组 G_1，G_2，…，G_M。在 G_i 中的每个 Avatar，其位置(x,y)

$$x=\begin{cases}0 & \text{若 } x_i+dx*\Omega<0\\ 25 & \text{若 } x_i+dx*\Omega>25\\ x_i+dx*\Omega & \text{其它}\end{cases}$$

$$y=\begin{cases}0 & \text{若 } y_i+dy*\Omega<0\\ 25 & \text{若 } y_i+dy*\Omega>25\\ y_i+dy*\Omega & \text{其它}\end{cases}$$

我们令 M＝3；dx，dy 为(－1，1)之间的随机数；Ω＝3.0 。

结论　CP2算法与 Bisection 算法的比较

CP2是基于竞争、逐步演化的算法，它的求解时间可以根据对解的质量的需求而设定。而对于给定的 DVE 世界及 Avatar 分布，Bisection 算法的求解时间总是固定的。

·求解时间和求解质量　试验数据的对照：在均匀分布和族状分布两种情况下，CP2的效率比 Bisection 算法有所改进。要得到相同质量的解，CP2的求解时间更短。而延长 CP2算法的计算时间，还能得到相当程度下更好的解。表1示例了当分区数 P＝8时，两种分布下的结果（P 为其它数时对比关系类似）。

表1

分区数(P)	分布	算法	解的时间(T)	工作量平衡代价(C_p^w)	服务器通信代价(C_p^b)	总代价(C_p)
8	均匀	Bisection	44	48	1127	587.5
		CP2	13	130	1029	579.5
			48	16	979	497.5
			575	16	853	434.5
8	族状	Bisection	62	71	2163	1117.0
		CP2	16	59	2131	1095.0
			30	263	1726	994.5

表2

分区数(P)	分布	算法	解的时间(T)	工作量平衡代价(C_p^w)	服务器通信代价(C_p^b)	总代价(C_p)
4	倾斜	Bisection	46	8	704	356.0
		CP2	55	52	895	473.5
5	倾斜	Bisection	51	744	895	819.5
		CP2	41	8	894	451.0
6	倾斜	Bisection	57	824	985	904.5
		CP2	57	52	1310	681.0
7	倾斜	Bisection	58	529	1080	804.5
		CP2	55	31	1254	642.5
8	倾斜	Bisection	58	68	1318	693.0
		CP2	57	44	1563	803.5

在倾斜分布下，则：当 P 为2的幂时(2，4，8…)，CP2算法的结果不如Bisection算法理想；但是若P不

（下转第12页）

器工作量的重要性权值；w_2---一个非负数，代表减少服务器之间通讯量的重要性权值（注：$w_1+w_2=1.0$）；V_i--分区 i，$i=1,2,\cdots,P$；p--划分格局（$V_1,V_2,\cdots,V_P$），对图 $G=(V,R)$，p 将它划分为 P 个分区，对应顶点集分别为 V_1、V_2、…、V_P。当 $i\neq j$，$V_i\cap V_j=\Phi$；$\bigcup_{i=1}^{P}V_i=V$；C_p^W--划分 p 的工作量平衡代价；C_p^L--划分 p 的服务器通讯量代价；C_p--划分 p 的总代价。

对一个划分 p，工作量平衡代价表示为：

$$C_p^W=\sum_{j=1}^{P}\left|\sum_{u\in V_j}W_i-n/P\right| \tag{1}$$

式(1)中，n/P 是工作量完全平衡时每个分区的工作量。$\sum_{u\in V_j}W_i$ 是分区 j 的实际工作量。所以，工作量平衡代价就是划分 p 下，各分区工作量与理想情况下偏差的总和。

下面定义函数 ADJ 用来判断 Cell u 与分区 V_i 是否相邻：

$$ADJ(u,V_i)=\begin{cases}1 & \text{如果 } u\notin V_i\text{；且}\exists\ v\in V_i\text{、}\\ & \quad u\text{ 和 }v\text{ 在图 G 中有一条边相连}(E_{uv}\in E)\text{；}\\ 0 & \text{否则。}\end{cases}$$

给定一个划分格局 p，定义 C_{ij} 为分区 V_i 与分区 V_j 之间的通讯量，其表示为：

$$C_{ij}=\sum_{u\in V_i}W_u*ADJ(u,V_j)+\sum_{u\in V_j}W_u*ADJ(u,V_i) \tag{2}$$

令：

$$C_p^L=\sum_{i=1}^{P}\sum_{j>i}^{P}C_{ij} \tag{3}$$

$$C_p=w_1*C_p^W+w_2*C_p^L \tag{4}$$

求解的最终目标就是要找到一个划分格局 p^*，使之满足：

$$C_{p^*}=\min_p(C_p) \tag{5}$$

可以证明，这是一个 NP 完全问题。若使用穷举法求解，其时间复杂度为 $\Theta(P^N)$。

4. 场景划分算法

4.1 基线划分算法(BP)

BP 算法最简单，它假定 Avatar 在整个虚拟世界中基本上均匀分布，这样的话，要平衡工作量，就要给各分区分配相同的面积。同时，为了减少分区之间通讯量，还应该考虑分区的形状。直观上，分区的交界处引起服务器间的通讯，因此应该使得各分区的边界尽可能地短。

给定面积，圆的周长将比其他所有图形都小。然而，是无法用一些不相交的圆来覆盖一个区域的，所以自然的设想是使分区呈六边形，如图5。蜜蜂把蜂窝建成许多六边形格以使用最少的材料（与这些六边形的周长之和成正比），获得最大的空间。

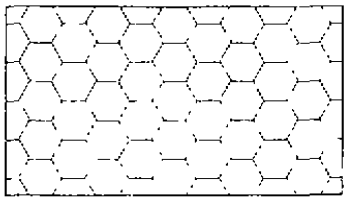

图5 用六边形覆盖区域

因为 DVE 系统由正方形的 Cell 组成，不可能拼凑出完全的六边形，所以比较方便的做法是还是使用正方形分区，给定区域面积 A，若使用面积为"a"的正方形作为分区，则一个分区的周长为 $4\sqrt{a}$，总周长为 $4\frac{A}{a}\sqrt{a}$。若使用面积为"a"的六边形，一个分区的周长为 $\sqrt{\frac{24a}{\sqrt{3}}}$，那么总周长为 $3.72\frac{A}{a}\sqrt{a}$，可见使用正方形与六边形相比通讯量代价相差7%。

做法：若服务器数目 P 不是个完全平方数时，则使用一个比 P 小的完全平方数 P' 划分大部分区域，剩下部分 Cell 则平均分配给剩下的服务器。其算法时间复杂度为 O(1)，图6中示范了一个 5 * 5 Cells 的 DVE 世界，P=5情况下 BP 算法的划分结果。

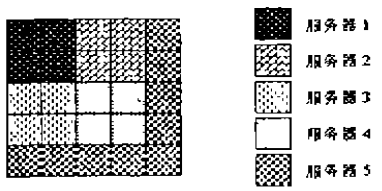

图6 BP 算法划分结果

4.2 二分划分算法(Bi-P)[3]

实际上，在一个虚拟世界中，Avatar 更多可能是不均匀分布的[3]，BP 算法对于这种情况已不再适合，这时可以使用基于二分法的 Bi-P 算法。

假定 Cell 数为 N，不失一般性，先考虑服务器数 P=2的情况。

令 V_i^k 为划分给第 k 个服务器的分区，它包含 m 个 Cell。首先令：

$$V_1^N=V=\{u_1,u_2,\cdots,u_N\};\quad V_2^0=\Phi \tag{6}$$

令 p_i 为第 i 个划分格局，其总代价为 Cp_i。初始格局 p_0 为 (V_1^N,V_2^0)，其总代价为 Cp_0。从 p_i 格局开始，在 V_1^N 分区中选择一个 Cell，交给分区 V_2^0，得到一个新的格局 p_{i+1}。需注意的是，选中的这个 Cell 是第一个分区所有 Cell 中，移到第二个分区后使得 Cp_{i+1} 最小的 Cell。这个选择过程相对 Cell 数 N 需要线性时间。