一种基于兴趣约束的新型组播地址分配方法

周 忠 赵沁平
(北京航空航天大学计算机学院 北京 100083)

摘 要 基于兴趣的约束关系提出了一种新型的组播地址分配方法——建立布种模型,通过布种模型的初始化和推演实现组播地址的静态和动态分配.论文结合二维点阵和 PR 四分树定义了布种模型的空间数据结构,并提出自适应生长/剪枝算法和检索算法实现组播地址的动态分配和快速搜索.算法分析和性能测试表明该方法效率高,可满足大规模分布式虚拟环境中的组播地址分配和检索.最后简介了该方法在分布式仿真运行平台 BH RTI 中的实现.

关键词 组播地址分配;兴趣约束;布种模型;分布式虚拟环境;RTI
中图法分类号 TP393

A Novel Multicast Address Allocation Approach Based on Interest Constrains

ZHOU Zhong ZHAO Qin-Ping
(*School of Computer Science & Technology, Beihang University, Beijing* 100083)

Abstract There exists a sharp conflict between the large amount of multicast addresses required in distributed virtual environment and the multicast address finiteness. In this paper, a novel multicast address allocation approach based on the interest constrains is proposed, which operates the allocation process with a seeding model. The initiation and run-time process of seeding model carry out the static and dynamic multicast address allocations individually. This paper combines 2-dimension lattice and PR quadtree structure to design the spatial data structure of seeding model and presents the adaptive growing and pruning algorithms for dynamic allocation, together with the searching algorithm for rapid searching the multicast address for a location. Algorithm analysis and performance experiments show that this approach has high efficiency and reach the requirements for multicast address allocation and search in large-scale distributed virtual environment. In the end a brief introduction to the implementation of this approach in BH-RTI, a run-time infrastructure, for distributed simulation is given.

Keywords multicast address allocation; interest constrains; seeding model; distributed virtual environment; RTI

1 引 言

兴趣技术是提高分布式虚拟环境规模化和持久性的一个基本的和重要的技术.兴趣技术主要包括两个方面的内容:一方面是仿真实体的兴趣表达,即仿真应用程序描述本地的仿真实体需要什么样的数据;另一方面是通道建立的过程,即需要一种方法能够有效地将仿真实体的数据从一个数据源传输到所有需要该数据的仿真节点.因为组播技术是大规模

收稿日期:2004-11-16;修改稿收到日期:2005-11-29.本课题得到国家"九七三"重点基础研究发展规划项目基金(2002CB312105)资助.
周 忠,男,1978 年生,博士,讲师,主要研究方向为分布式仿真和虚拟现实.E-mail:zz@vrlab.buaa.edu.cn.赵沁平,男,1948 年生,博士,教授,博士生导师,主要研究领域为虚拟现实、可视化、人工智能.

分布式虚拟环境所需要依赖的主要通信手段,组播地址分配也就成为兴趣技术中建立通道的关键所在.分布式虚拟环境对组播地址的需求巨大,由于路由器能支持的组播地址是有限的,主机能同时加入的组播地址也是有限的,分布式虚拟环境对组播地址的大量需求和组播地址资源的有限性形成了一对尖锐的矛盾,组播地址分配成为兴趣技术的一个重点和难点.

基于对象类的组播地址分配方法[1]易于实现,DMSO RTI 提供的简单策略[2]在属性集层次上进行网络带宽的划分,二者的过滤粒度过大,很少单独使用.在分布式虚拟环境的发展过程中,组播地址分配方法主要有四类,一是基于数据本身的内容进行的组播地址分配方法,典型例子就是各种形式的网格分配方法;二是基于数据发送方的组播地址分配方法;三是基于数据接收方的组播地址分配方法;另外还有一类是从全局通信连接信息出发,把组播地址分配问题转化为最优问题求解,根据算法求解进行组播地址分配的方法.

各种基于区域或场景静态划分的机制采用的都是基于数据内容的组播地址分配方法. HLA DDM[2,3]的网格过滤法将路径空间静态划分,每个单元对应到特定的组播地址,NPSNET-IV 将地形进行六边形网格划分并分配组播地址[4],DIVE 将组织成层次树的场景划分为不同的 Light Weight Group 并分配组播地址[5,6],MASSIVE-2 与 DIVE 类似,MASSIVE-3 将虚拟空间分为若干个 Locale 并分配组播地址[7]等都属于此类.这类方法的优点是计算量小,缺点是随着虚拟空间的增大,组播地址需求迅速增长,而且组播地址浪费严重. Boukerche 等在网格划分的基础上提出了一种集中式动态组播地址分配方法,有效利用了组播地址,解决了组播地址需求迅速增长和浪费的问题,缺点是中心服务器容易成为瓶颈[8].

基于发送方的组播地址分配方法是在数据的发送方为每个数据源分配组播地址.数据发送方根据所有的发布/订购信息进行匹配计算,并指导接收方加入相应的组播地址.这种方法一般采用一个对象实例分配一个组播地址,系统使用的组播地址数量和实体数量成线性关系,因此成本很高,除需要大量的组播地址外,路由器的处理延迟对整个系统的影响也很大.而由于要让仿真节点知道虚拟环境中周围的实体以及这些实体所使用的组播地址,每个实体需要定期组播或广播自己的状态信息,给网络上带来了额外的数据传输.另外,仿真主机加入组播组的数量的限制对这种方法的应用也存在制约.

基于接收方的组播地址分配方法是基于数据接收目标,为每种目的节点的组合分配组播地址.一旦发送方知道了目标节点列表,它就可以把数据向代表这些目标的组播地址发送.对于一个全连接的仿真来说,如果有 N 台主机,这种机制需要的组播地址的数量将是 $2^N - 1$.这种机制的优点是仿真节点不会收到不想接收的数据.为了让这种方法更实用,每个仿真节点可以为经常用到的目标节点保留固定数量的组播地址,同时定义一个包含所有节点的组播地址.一旦没有找到包含相同目标节点的组,就可以采用这个包含所有节点的组播地址发送数据.采用这种方法使过滤效率有所下降,但可以大大减少所需要的组播地址的数量.

还有一类基于最优问题求解的组播地址分配方法. Morse 提出根据通信连接图基于最优问题求解分配组播地址的方法[9].该方法是建立盟员之间的有向连接图,将地址分配问题等价为如下最优问题:将 n 个连接分配到 m 个组播地址中,使得每个数据包从盟员发出到其他盟员收到的延时都小于某个阈值. Morse 提出了 LOC(最大输出连接)算法和 IRLOC(输入限制的最大输出连接)算法. Adlery 等将通道问题定义为数据源与组播地址的对应关系和组播地址与接收者的对应关系这两个问题,证明了保证每个接收者能接收到感兴趣的数据源的数据并使得总的通信量以及接收者接收的不必要数据最少是 NP 完全性问题,并给出了寻找次优解的随机分配方案 Flow Based Merge(FBM)和 User Based Merge(UBM).结论是 FBM 方法在通常情况下要好于 UBM,除非数据源的个数远大于用户数. FBM 算法与 Morse 的 LOC 算法均有全局信息的收集问题,得到的分配方案的通信量并不一定比采用点对点的通信量少[10].通过最优问题求解可以获得最经济的组播地址分配,浪费最小,但需要大量的实时通信来维护动态变化.

综合以上 4 种方法,现有的组播地址分配方法各有优缺点,需要根据不同的应用特点进行选择.第 4 种方法的额外通信量大,但可达到最优的组播地址利用,其它 3 种方法的共同问题是对组播地址的大量需求.

2 组播地址分配中的兴趣约束

在大规模虚拟环境尤其是军事仿真中,很少会

出现大量实体不存在感兴趣者的现象[11],也就是说,在基于组播的虚拟环境系统中,大部分实体都需要参与分配组播地址. 大规模虚拟环境中的数据传输量巨大,主机的计算负载很重,而为每个实体进行频繁的发布订购关系匹配所需的计算量很大,这是美军历次演练都以基于数据内容进行组播地址分配的网格法为基础的重要原因.

在 STOW 97,STOW 98,AO '00 等大规模军事仿真系统中应用的 STOW-RTIs 是基于数据内容进行组播地址分配的,但和传统的网格法有所区别. STOW-RTIs 采用了一种基于多层网格的组播地址分配方法[12],对整个地形进行一层大的网格划分,根据演练部署的需要,在不同的地点覆盖上一定尺度、不同精度的网格,当某个地点被覆盖上多层网格之后,以其所占据的最外层的网格进行组播地址分配. 这种基于多层网格的组播地址分配方法避免了网格分配法对不同使用率的地点平均分配组播地址的做法,大大降低了使用的组播地址数量. 网格子集的部署和精度需要根据具体演练进行配置[11]. 但是这种方法是一种静态分配方法,不能根据实际运行中组播地址的需要进行动态分配,组播地址的分配策略和过滤效率必然受到影响,仍然可能存在地址空间的较大浪费. 而且大规模虚拟环境一般采用很大的地形,受网格的约束,那些演练涉及不到的区域还是必须分配组播地址,STOW 等大规模仿真的地理范围甚至大到几十万平方公里,这一问题的影响尤其明显.

我们在研究过程中,从兴趣所基于的真实世界映像出发来进行分类,分析兴趣的仿真特性,从现有的各种兴趣技术中归纳出兴趣的四个基本要素,包括任务驱动、感知能力、交互能力、关注(聚焦)程度,并定义了兴趣及兴趣层次等概念[13]. 分析现有的组播地址分配方法,结合兴趣的基本要素,可以发现,组播地址分配的出发点是来自于虚拟环境中兴趣分布的特点,影响组播地址需求的内在原因是兴趣的约束力,本文把这种约束力称为兴趣约束. 因此,影响组播地址需求的重要因素是兴趣的约束作用. 由于感知能力的影响,参与仿真的对象之间会动态地产生兴趣,可以从这种兴趣作用的主体或客体出发,基于发送方或接收方进行组播地址分配. 最优问题求解则是从兴趣作用的关系本身出发,从数学上寻求以最少的编号来表示对象之间动态变化的兴趣关系. 网格分配法是以枚举的方法从数据内容上将路径空间中所有可能发生兴趣的单元进行定义. 而 STOW-RTIs 则是利用了兴趣的首要要素 ——任务驱动,将演练任务和组播地址分配相结合,对预定的产生大量兴趣关系的地点进行精细的组播地址分配,大量可能发生兴趣的地点进行较粗的组播地址分配,少数地点单独进行分配. 同样,交互作用和关注程度也对这种约束作用的大小有一定影响. 概括起来就是:兴趣约束决定了组播地址的需求,从而影响了组播地址分配.

下面基于静态网格分配法结合兴趣的四个要素来分析兴趣约束. 任务驱动产生了兴趣约束的最基本也是静态的作用,可以以一些手段来在静态网格分配法中体现这种任务驱动的作用. 而感知能力、交互作用、关注程度等决定了在对象交互过程中动态变化的兴趣关系,适合于以动态的方式来指导组播地址分配. 在静态网格分配法中,应该对那些对象密集的区域进行精细的网格划分,以减少单个组播组的数据量. 事实上,可以以一个简单的指标来近似表征这种动态的对象密集程度,即发布组播组数据的对象数统计,因为组播组数据的生产者是那些发布该组播组数据的对象,对象数统计也就能在一定程度上体现数据的密集程度. 简言之,兴趣约束可以体现为静态和动态兴趣因子,静态兴趣因子是任务驱动的作用,动态兴趣因子则应是感知能力、交互作用、关注程度的综合作用,用发布组播组数据的对象数统计可以近似表征.

下文将就兴趣约束的静态和动态兴趣因子提出一种新的组播地址分配思路,通过建立布种模型进行组播地址分配.

3　布种模型

从基于数据内容的组播地址分配方法的角度来看,进行组播地址分配也就是将一定数量的组播地址"点"按照一定的规律放置到路径空间的一些小单元中,由于组播地址"点"的数量是有限的,如果能够建立一种模型实现组播地址"点"的动态重组和检索,将能够很好地用于组播地址的动态分配.

3.1　空间"第三维"的特点

基于地理坐标系的路径空间定义是最常用的,仿真中存在多种坐标系统,但是高度(或加上某个常量,如地球半径)是各种坐标系所共同定义的,此处称为第三维,z 轴. 二维平面和三维空间所区别的这个第三维和前两维有着根本的区别,从仿真上来说,由于在高度上遮挡物少,受地球曲率的影响小,z 轴

上的变化不像前两维那么明显,更适宜划分为确定的几个层次,如美军在新一代的"空间战"战略中根据高度进行的作战空间划分.而且外空间一般采用的是短波通信,受距离的影响小,高层空间可以接收低层空间的信息.因此普通应用可以为 z 轴按层次分配组播地址.由于三维路径空间对组播地址的需求是随空间大小成立方级增长的,这样可以把对组播地址的需求降低为平方级关系.也因为这些原因,美军的大规模仿真采用的都是二维路径空间的分配方法.

本文遵循简单实用的原则,将路径空间的前两维作为主过滤维,进行组播地址分配,高维则由数据分发管理进行接收方的区域匹配.

3.2 基本思想

在描述基于兴趣约束的组播地址分配时,布种模型是把路径空间作为土地,种子就是在土地里静态部署的组播地址,可进行分配的组播地址集合是种子的全集,组播地址分配也就成为布种并自发生长的过程.组播地址"点"包括种子和生长后的各结点的集合.布种模型以网格划分为基础,其基本思想是:

(1)由于地质有好坏,布种是不均匀的,在那些规划中(即存在静态兴趣因子)的网格单元中才进行布种,其它网格单元与规划任务的关联小,可以统一处理.

(2)类似于种子有了阳光、水和养分等可以发芽生长,网格单元中的动态兴趣因子积累多了,种子将生长成树,树的结点是动态分配的组播地址,这样,每粒种子都可能成为一棵树.

(3)随着仿真的推进,布种模型演变成为记录组播地址的森林,组播地址的动态分配就成为对森林的生长/剪枝问题,查找组播地址就成为对森林的叶结点的检索.

布种模型主要包括以下两个方面的内容:种子的部署和种子的生长.

3.2.1 种子的部署

静态兴趣因子是任务驱动的作用产生的.大规模分布式虚拟环境由于大的地理跨度、时间的限制、任务或想定的设定,一般会分布在有限的一些地点进行交互,现行的网络游戏也具有类似的特点,交互一般集中在一些小镇或野外,道路也进行限定,限制玩家的运动范围.这种静态兴趣因子在仿真之初就可以确定,因此,可以寻求一些手段进行任务的编辑或设定.

本文在传统网格分配法网格定义的基础上,提供编辑工具将演练地图直观地进行网格划分,并通过两种手段来体现静态兴趣因子在不同地点的差异:

(1) x, y 维的划分不是均匀的,可以根据实际的需要进行调节;

(2)任务之外或不可能涉及到的地点,不部署种子.

在那些未被反选的网格单元,按照先 x 轴后 y 轴的顺序分配种子,被反选的地点则不予以分配,也就不参与统一的组播地址分配.

如图 1 所示,以三国鼎立的二维地形为例,在三方战端频繁的地点,可以将网格线的距离减小,增加网格的密度,其它地点可以适当放宽网格线的距离;在边荒地区、远海等地点,可以设置为反选状态,不部署种子.在图中,一共 400 个网格单元,实际分配了 146 个网格单元,所需组播地址个数大大降低,地址占用率降低到 36.5 %.

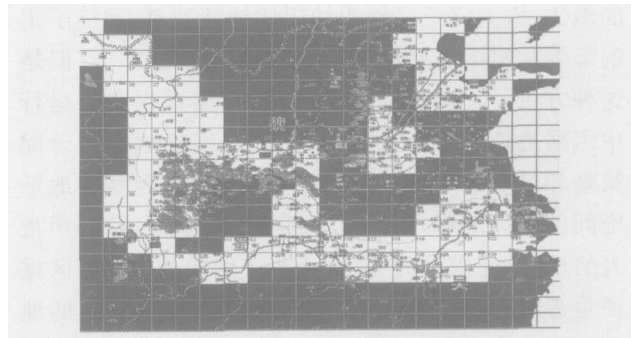


图 1 基于兴趣约束的地形网格编辑

不参与组播地址分配的那些地点,可以有两种方式:(1)作为不可能或不应该进入的区域,不分配组播地址;(2)考虑可能的少数进入情况,将所有的反选网格单元设置为同一个组播地址.在本文的具体实现中,采用了第 2 种方法.

3.2.2 种子的生长

在分布式虚拟环境中,大量的对象处于运动状态,由于兴趣约束,每个地点的动态兴趣因子是不同的,当一个地点聚集了过多的对象,动态兴趣因子累加到一定程度,这个网格单元应以更多的组播地址来进行通信,以减少对象数据交互量,当对象渐渐散开,过多的组播地址又不再必要.因此,由于组播地址预分配不能为每个地点提供足够多的组播地址,动态组播地址分配可以充分利用组播资源,减少组播资源的浪费.

在不同地点根据实际仿真要求需要使用不同精度的网格分配法,PR 四分树(Point-Region Quadtree,即点-区域四分树)是一种空间数据结构,非常适合

定义此类二维平面上的无规律点分布模型,并已在一些相关的应用中得以实现[14~16]. 如果一个区域需要定义更多的组播地址,那么将这个区域四等分,相应的 PR 四分树就包含一个内部结点和四个子树(或叶结点). 仿真过程中某区域兴趣因子的动态改变导致对组播地址的需求发生变化,可以通过 PR 四分树进行生长或剪枝来提供动态的组播地址分配,当该网格单元的内部对象增多,动态兴趣因子增大,该单元对应的结点进行生长,继续四分;当动态兴趣因子减小,组播地址使用率低时,进行剪枝,将小的区域进行合并.

4　仿真推进中布种模型的推演

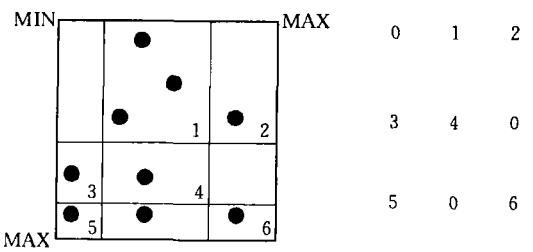
随着仿真过程的推进,布种模型需要进行推演,以满足动态的组播地址分配要求. 核心问题包括空间数据结构定义、自适应生长/剪枝算法、检索算法.

4.1　空间数据结构定义

多维范围查询是一个空间应用(spatial application)的重要特性. 要高效率地实现空间应用程序,需要使用空间数据结构(spatial data structure). 空间数据结构存储根据位置组织的数据对象,它是在地理信息系统、计算机图形学、机器人学和许多其它应用中使用的一类重要数据结构[17]. 布种模型是一种空间应用的数学模型,其关键的空间数据结构包括二维点阵和 PR 四分树.

4.1.1　二维点阵

如图 2 所示,整个路径空间被划分为 3 ×3 的不均匀网格,不考虑种子的位置,种子 1～6 以点阵的方式在空间中分布,数据结构需要定义每维上的坐标以及种子的点阵分布. 图中的点表示仿真中活动在该区域的一定数量的对象.



(a) 种子部署图示例　　(b) 二维点阵示意

图 2　种子部署及二维点阵

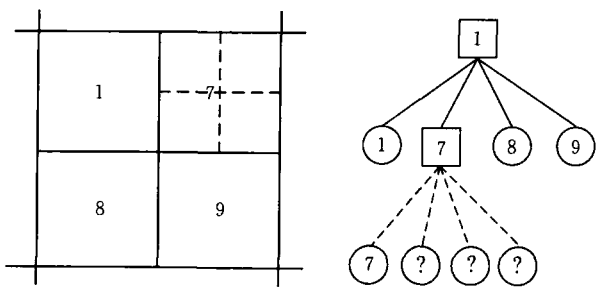
布种模型中用于种子部署的二维点阵 Grid 定义为

```
// m, n:int
// dimension X 上的划分:
        // value of X[m]:int (已归一化)
// dimension Y 上的划分:
        // value of Y[n]:int (已归一化)
// 组播地址分配数组:
        // handle mapping[m-1][n-1]:int
typedef struct{
  // X 轴点的个数
  int num of X;
  // Y 轴点的个数
  int num of Y;
  // X 轴点的值, int[num of X]
  U32 * value of X;
  // Y 轴点的值, int[num of Y]
  U32 * value of Y;
  // 二维网格数组,记录句柄编号. int[num of X-1]
        //[num of Y-1]
  int ** handle mapping;
} Grid;
```

上述定义中的 Grid 数据结构记录了每维上划分的坐标,并以二维点阵结构定义了种子在网格中的部署.

4.1.2　PR 四分树

将种子 1 所在的网格单元进行四等分,由于结点 7 所在单元的对象密集,动态兴趣因子增加,需要再次生长. 图 3(a) 是种子 1 生长一层后,子结点 7 再次进行生长的示意图. PR 四分树是一个典型的空间数据结构,其结点要么正好有 4 个子女,要么是 1 个叶结点,图 3(b) 是种子 1 生长后的 PR 四分树结构.



(a) 数据"点"分布　　(b) (a)的 PR 四分树

图 3　一个种子生长的例子

PR 四分树是完全四叉树结构,内部结点记录了 4 个子结点的索引,递归次数 *level* 不宜过大,应作一定的限制. 它的实现采用了一种"子结点表"表示法(list of children),每个分支结点均存储其子结点按序排列的索引表,如图 4 所示.

用"子结点表"表示法来实现 PR 四分树,如果以地址句柄作为索引,PR 四分树的内部结点也需

索引	值	层数	子结点索引			
0	1	0	1	2	3	4
1	1	1				
2	7	1	5	6	7	8
3	8	1				
4	9	1				
5	7	2				
6	?	2				
7	?	2				
8	?	2				

4 以"子结点表"表示法实现的图 3 中的 PR 四分树

要分配组播地址,这种动态分配也存在一定的地址浪费,如图 3 中的 1,7 地址句柄不能重复使用. 不考虑内部结点的第一子结点的二次生长,可以将内部结点的第一个子女退化到内部结点中,这样,PR 四分树就可以从完全四叉树退化到完全三叉树. 如图 5 所示,退化的 PR 四分树使每个结点与已用的组播地址句柄一一对应,由于组播地址句柄是唯一标识的,可以作为索引将整个森林用一个子结点表来表示,便于进行地址的组织、高效检索、分配和回收.

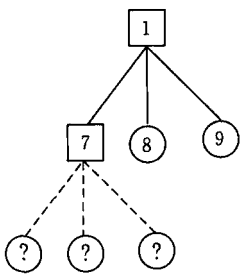


图 5 图 3(a)退化的 PR 四分树

本文在实现中对布种模型进行了简化,组播句柄和索引相一致,使用的是退化的 PR 四分树. 其结点的数据结构定义如下:

```
//退化的 PR 四分树结点定义
typedef struct{
  //结点生长的层数(0 表示未用或种子)
  int level;
  //01,10,11 三个子结点索引, 0 表示空,
      //3 = 2 ** 2 - 1.
  int split node[3];
  //是否叶结点
  bool is leaf;
  //组播地址句柄
  int is groupHandle;
} tree node;
```

4.2 自适应生长/剪枝算法

组播地址的动态分配对实时性的要求不高,但不能出现组播分配森林不一致的现象,由集中的管理者基于布种模型根据动态兴趣因子和用户定义的递归层数进行分配,可以较容易地维护数据的一致性. 管理者进行一段时间内各仿真节点的发布对象数统计(动态兴趣因子),根据某种子或叶结点对应的对象数统计,进行自适应的生长/剪枝,并发布给各仿真节点.

4.2.1 生长算法

当某个网格子单元的对象数达到了预定的上限,且它所属的原始划分的网格单元进行的四分次数尚未达到用户限定的层次数时,需要将该网格子单元对应的叶结点(或种子)进行生长,对应的子单元再次进行四分. 相应的逻辑描述如下式所示:

IF (动态兴趣因子 > 网格单元的对象数上限 && level < 限定的层次数)
  生长一层.

生长算法 GROW 对布种模型的森林中的种子或退化的 PR 四分树的叶结点进行生长,增加新的子结点到树上,其过程分为 6 步:

1. 从组播句柄获得结点的索引;

2. 检查该结点是否已是树的内部结点,因为内部结点不能再次生长,是则分支退出,否表示是叶结点或未分配结点,继续;

3. 检查该结点是否是未分配结点,因为未在某棵树上的结点不能生长,否则分支退出. 已分配结点包括森林中未生长的种子和各退化的 PR 四分树上的结点;

4. 调用 findNullNode 找出 3 个未用的组播句柄;

5. 根据新的组播句柄分别创建子结点并分配到新的位置;

6. 将原结点设置为内部结点,以新创建的结点为子结点,这样,该结点实现了生长.

在查找新的组播句柄时,采用递增分配的方法,已经释放的组播句柄不立即参与分配,只有当整个句柄空间的最后部分被分配之后,才会从头顺次进行检索,以提高查找未用组播句柄的效率. findNullNode 算法分为 3 步:

1. 对当前空组播句柄索引进行合法性检查,不正确则设置重置;

2. 根据当前空组播句柄索引查找空组播句柄;

3. 如地址已分配完一遍,从头重新开始寻找.

其伪代码如图 6 所示.

```
//查找空组播句柄,其中空组播句柄索引 MinNullIndex 用以
//指示开始寻找空组播句柄的起始位置
INT findNullNode( ){
   INT temp = MinNullIndex;
   IF ((MinNullIndex belong to 静态组播句柄空间) ‖
       (MinNullIndex beyond 总组播句柄空间))
     RESET MinNullIndex AS 动态组播句柄空间下界;
   IF (结点 MinNullIndex 为闲置结点) {
      MinNullIndex = findNextNullNode(MinNullIndex);
      RETURN temp;
   }
   // MinNullIndex 无效,从头开始重新找被释放的地址空间
   temp = findNextNullNode(动态组播句柄空间下界);
   //再找下一个并记录到空组播句柄索引
   MinNullIndex = findNextNullNode(temp);
   RETURN temp;
}
```

图 6　算法 findNullNode 伪代码

4.2.2　剪枝算法

当某个内部结点对应的网格单元的对象数达到了预定的下限,对该内部结点进行递归剪枝,合并其下属的所有子结点. 相应的逻辑描述如下式所示:

找到其父结点,计算动态兴趣因子;

IF(动态兴趣因子 < 网格单元的对象数下限)

　　对父结点剪枝.

剪枝算法 PRUNE 是一个递归的过程,它对布种模型的森林中退化的 PR 四分树的内部结点进行剪枝,递归遍历下属的子结点并进行合并,其过程分为 4 步:

1. 由组播句柄得到结点的索引,并进行合法性检查.

2. 如该结点是叶结点,销毁该结点,释放组播句柄并退出.

3. 递归删除第一子结点、第二子结点、第三子结点.

4. 重新设置原内部结点为叶结点(或还原为种子).

4.3　检索算法

在分布式虚拟环境中,对象的位置多变,引起其区域频繁更新,因此检索效率很为重要. 对布种模型进行检索就是从森林中找出该空间位置所属的结点,检索的关键码是空间属性. 检索算法需要先在二维点阵中检索出种子结点,再检索该位置的 PR 四分树,具体包括两步. 图 7 是检索算法的伪代码.

对检索算法的关键步骤解释如下(其中 Grid 的定义同 4.1 节):

(1) 检索二维点阵,找出种子结点. 根据空间属性在二维点阵中确定横纵坐标,并在该种子结点所属的网格中确定相对位置,以进行(2)中树的叶结点的检索.

(2) 若种子已经生长为树,以该种子结点为树的根结点,在这棵退化的 PR 四分树中检索所需的叶结点.

涉及到的关键算法介绍如下:

(1) 在二维点阵中查找种子以二分法分别在横纵坐标上进行检索,横纵轴分别是路径空间的第一维和第二维,其坐标的划分分别按照从小到大的顺序记录在 Grid.value X[ ]与 Grid.value of Y[ ]中,适合进行二分法查找. 图 8 是计算横坐标的伪代码,纵坐标的计算与此类似.

```
//检索算法
INT SEARCH(U32 x, U32 y){
   INT i = COLUMN(y);
   INT j = ROW(x);
   IF (Grid.handle mapping[i][j] = =0)
      //森林中此处未设置种子
      RETURN 0;
   FLOAT fx = (x - Grid.value of X[j - 1]) *1.0/
              (Grid.value of X[j] - Grid.value of X[j - 1]);
   FLOAT fy = (y - Grid.value of Y[i - 1]) *1.0/
              (Grid.value of Y[i] - Grid.value of Y[i - 1]);
   RETURN SearchInTree(fx,fy,mapping grid.handle mapping[i][j]);
}
```

图 7　检索算法 SEARCH 伪代码

```
//根据 Grid 计算横坐标值
INT COLUMN(U32 x){
   //起点
   INT i = 0;
   //终点
   INT j = Grid.num_of_X - 1;
   //中点
   INT m = 0;
   //边界情况
   IF (x < = Grid.value_of_X[1])
      RETURN i;
   IF (x > = Grid.value_of_X[j - 1])
      RETURN j - 1;
   //二分法搜索
   j - - ;
   WHILE(i < = j){
      m = (i + j)/2;
      IF(x < mapping grid.value of X[m])
        j = m - 1;
      ELSE
        i = m + 1;
   }
   RETURN i;
}
```

图 8　二分法计算种子横坐标的伪代码

(2) 通过空间属性与根结点的相对坐标(fx, fy)来比较确定进一步要搜索的子结点,对原空间属性乘 2 求余以确定在子结点中新的空间属性,进行深度搜索,直至查找到叶结点,其中 tree node 的定义如前面所示. 对树的检索算法的伪代码如图 9 所示.

```
//在树中进行结点的搜索
INT SearchInTree(FLOAT x, FLOAT y, INT Index){
  //叶结点,检索成功
  IF(索引 Index 是叶结点)
    RETURN Index;
  //内部结点,需要深度检索,直至查到叶结点
  INT num = 0;
  //第一子结点已退化
  IF((x > 0.5) &&(y <= 0.5))
    num = 0;
  ELSEIF((x <= 0.5) &&(y > 0.5))
    num = 1;
  ELSEIF((x > 0.5) &&(y > 0.5))
    num = 2;
  //2x,2y 对 1 进行求余运算,确定在子结点中的空间属性
  x *= 2;
  y *= 2;
  IF(x >= 1)
    x = x - 1;
  IF(y >= 1)
    y = y - 1;
  //深度搜索子结点
  tree node = Index 对应的内部结点;
  RETURN SearchInTree(x, y, tree node.split node[num]);
}
```

图 9 退化的 PR 四分树检索算法的伪代码

5 算法分析

本小节进行算法复杂性分析,由于仿真运行过程中组播地址的使用和查询都可能进行密集的检索,检索算法的效率尤为重要.

5.1 生长/剪枝算法

生长算法是对退化的 PR 四分树上某结点进行生长的一组操作,其代价主要是查找 3 个未使用的组播句柄.由于是按顺序分配,在组播句柄空间没有被分配一遍之前,每一次查找就是加 1 过程,再开始新一轮查找时,查找次数和释放的组播句柄位置有关系,对于 n 次随机的句柄释放,每次检索的平均情况代价是 $n/2$,在最差情况下,空的组播句柄空间在最后的位置,检索的代价是 n.

剪枝算法是对退化的 PR 四分树的某个内部结点遍历的过程,这是个递归过程,设该结点被分配了 m 个组播句柄,则遍历的代价是 m.

5.2 检索算法

检索的第一步是二维点阵检索,采用的是二分法,时间复杂度为 $O(\log_2 n)$,因为每维上的坐标是顺序的,很适合于二分法.第二步是对退化的 PR 四分树的检索,最差情况是检索位于最高层的结点,这时代价为该结点所生长的层数,一般不会很大,根据具体应用中用户的限制有所不同决定.

6 性能实验

下面对生长/剪枝速度和检索速度进行测试,因为进行一次生长/剪枝或检索的时间是微秒级,此处采用每毫秒执行的次数来表示速度.主机环境:P4 2.8GHz,512MB 内存,Win 2000 Professional.

6.1 生长/剪枝速度

将路径空间进行 20×20 的网格划分,测试不同种子数量的情况下全部生长一遍和剪枝一遍的时间,计算出每毫秒的平均生长/剪枝速度.

定义:

平均生长速度 = 种子数量 / 全部生长一遍的时间(ms),

平均剪枝速度 = 种子数量 / 全部剪枝一遍的时间(ms).

测试的种子数量是 50,100,150,200,250,300,350,400.

图 10 是生长/剪枝速度的测试结果,从图中可见,每毫秒可以进行约 1800 次生长或 3000 次剪枝,能够满足仿真的需要.

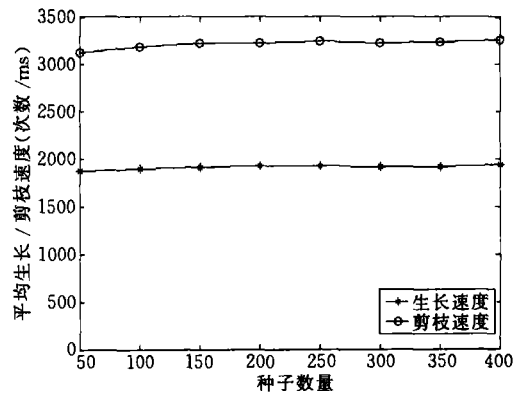


图 10 生长/剪枝速度测试结果

6.2 检索速度

测试了对未生长的结点和生长一层的结点的检索速度,对未生长结点的检索包括的是二维点阵的二分法检索,对生长一层的结点则包含二维点阵的检索和退化的 PR 四分树的检索.

定义:

平均检索速度 = 检索次数/检索总时间(ms).

检索点:(*MAX RANGE EXTENT*/20×2, *MAX RANGE EXTENT*/20×1.5),测试的种子数量是 50,100,150,200,250,300,350,400.

图 11 是对检索速度的测试结果,从图中可以得出,第 1 步的检索时间约为 0.4μs,第 2 步的检索时

间约为 0.58μs - 0.4μs = 0.18μs,每毫秒可以检索生长一层的结点 2500 次左右,可以满足大规模的仿真中进行组播地址查询的需要.

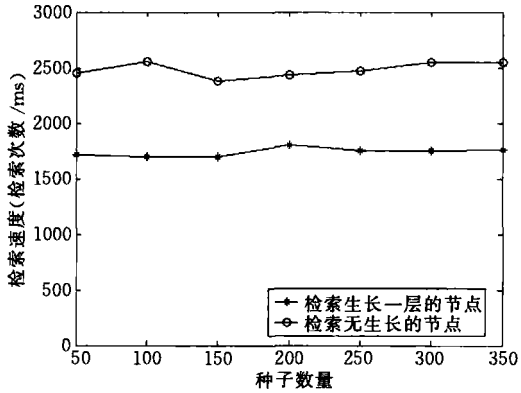


图 11　检索速度测试结果

7　本文方法在 BH RTI 中的实现

本文方法已在北京航空航天大学虚拟现实重点实验室开发的分布式仿真运行平台 BH RTI 2 中实现,其中 BH RTI 的网络抽象层 NAL 管理着组播地址资源、本地的组播加入/退出、网络上的发布/更新等底层操作,将这些资源进行统一的封装,向上层提供资源号,核心服务模块 KernelService 的内核 Kernel 基于布种模型进行具体组播地址的分配. 为了方便用户完成布种模型的初始化,BH RTI 提供了地形网格编辑器 Map Edit Ex,如图 12 所示,用以直观地在特定的应用背景下进行仿真应用. Map Edit Ex 生成了.gin 文件,其格式如图 13 所示,组播地址分配所需的 Raw Grid 从该初始化文件生成,布种模型初始化的数据结构如下所示:

```
typedef struct{
  // X 轴点的个数
  int num of X;
  // Y 轴点的个数
  int num of Y;
  // X 轴点的值, int[num of X], index[0,num of
       // X - 1]
  double * value of X;
  // Y 轴点的值, int[num of Y], index[0,num of
       // Y - 1]
  double * value of Y;
  //二维网格数组,记录句柄编号
  //int[num of X - 1][num of Y - 1], index[0,
       // num of X - 2][0,num of Y - 2]
  int ** handle mapping;
} Raw Grid;    //用于组播地址分配初始化
```

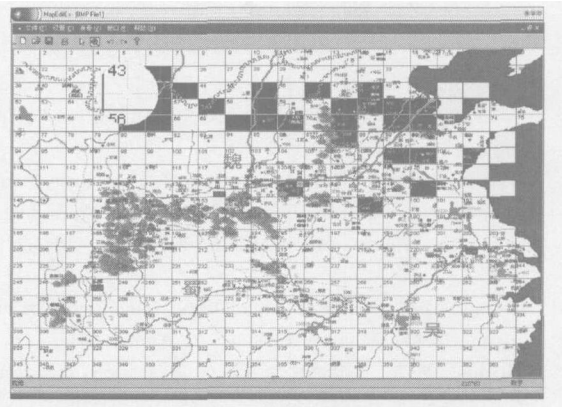


图 12　地形网格编辑器 Map Edit Ex

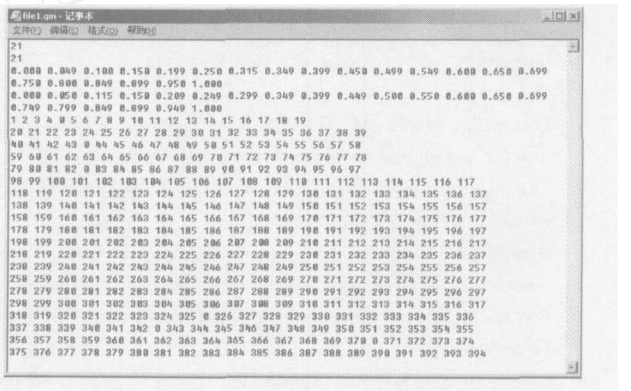


图 13　布种模型初始化文件.gin 示例

8　结束语

本文从一个全新的角度研究了分布式虚拟环境中的组播地址分配问题,指出影响组播地址分配的重要因素是兴趣的约束作用. 就兴趣约束的静态和动态兴趣因子提出一种新的组播地址分配思路——通过布种模型进行组播地址分配. 该组播地址分配方法能较好地消除由于组播地址有限性所带来的难题,通过组播地址的动态分配提高了组播的过滤效率. 算法分析和性能测试表明,生长/剪枝和检索算法效率高,可满足大规模分布式虚拟环境中的组播地址分配和检索.

本文提出的方法同样适用于网络游戏,尤其是大规模的网络游戏服务器机群之间的组播通信. 另外,本文从应用需求出发,对布种模型中的 PR 四分树进行了一定的简化,如有需要,可考虑更完整的实现. 在具体实现中还需要考虑服务方式和网络通信,根据实现体系的不同,可能会涉及到"子结点表"的数据压缩算法和补丁发布机制等以及加入/退出组播组等实现技巧.

参 考 文 献

1 Singh *et al*.. BrickNet: A software toolkit for networks-based virtual worlds. Presence: Teleoperators and Virtual Environments, 1994, 3(1): 19～34

2 Hyett M., Wuerfel R.. Implementation of the data distribution management services in the RTI-NG. In: Proceedings of the Spring Simulation Interoperability Workshop, Orlando FL, 2002

3 Defense Modeling and Simulation Office. High Level Architecture interface specification version 1.3, April 1998

4 Macedonia M. R.. NPSNET: A network software architecture for large scale virtual environment [Ph. D. dissertation]. Naval Postgraduate School, Monterey, 1995

5 Christer C., Hafsand O.. DIVE —A multi-user virtual reality system. In: Proceedings of the IEEE Virtual Reality Annual International Symposium, Seattle, Washington, 1993, 394～400

6 Fr écon E., Stenius M.. DIVE: A scaleable network architecture for distributed virtual environments. Distributed Systems Engineering Journal (Special Issue on Distributed Virtual Environments), 1998, 5(3): 91～100

7 Greenhalgh C., Benford S.. Supporting rich and dynamic communication in large-scale collaborative virtual environments. Presence: Teleoperators and Virtual Environments, 1999, 8(1): 14～35

8 Boukerche A., Roy A., Thomas N.. Dynamic grid-based multicast group assignment in data distribution management. In: Proceedings of the 4th IEEE International Workshop on Distributed Simulation and Real-Time Applications (DS-RT '00), San Francisco, California, 2000, 47～54

9 Morse K. L.. An adaptive, distributed algorithm for interest management [Ph. D. dissertation]. University of California, Irvine, 2000

10 Adlery M., Ge Z., Kurose J., Towsley D., Zabele S.. Channelization problem in large scale data dissemination. In: Proceedings of the 9th International Conference on Network Protocols (ICNP '01), Washington, DC, 2001, 100～109

11 Helfinstine B., Wilbert D., Torpey M., Civinskas W.. Experiences with data distribution management in large-scale federations. In: Proceedings of the Fall Simulation Interoperability Workshop, Orlando FL, 2001

12 Berrached A., Beheshti M., Sirisaengtaksin O.. Approaches to multicast group allocation in HLA data distribution management. In: Proceedings of the Spring Simulation Interoperability Workshop, Orlando FL, 1998, 1063～1068

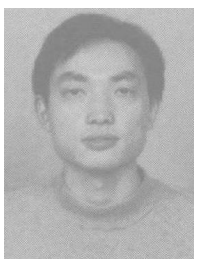
13 Zhou Zhong. Research on the layer of interest oriented to simulation high level architecture [Ph. D. dissertation]. Beihang University, Beijing, 2004 (in Chinese)
(周 忠. 面向仿真高层体系结构的兴趣层次的研究[博士学位论文]. 北京航空航天大学, 北京, 2004)

14 Christian Böhm, Gerald Klump, Hans-Peter Kriegel, XZ-Ordering. A space-filling curve for objects with spatial extension. In: Proceedings of the 6th International Symposium on Advances in Spatial Databases (SSD '99), Hong Kong, 1999, 75～90

15 Sevcik K. C., Koudas N.. Filter trees for managing spatial data over a range of size granularities. In: Proceedings of the 22nd International Conference on Very Large Data Bases, Bombay, India, 1996, 16～27

16 Lee K., Lee D.. A scalable dynamic load distribution scheme for multi-server distributed virtual environment systems with highly-skewed user distribution. In: Proceedings of the 10th ACM Symposium on Virtual Reality Software and Technology (VRST 2003), Osaka, Japan, 2003, 160～168

17 Shaffer C. A.. A Practical Introduction to Data Structures and Algorithm Analysis. Second Edition. Upper Saddle River, NJ: Prentice Hall, 2001



ZHOU Zhong, born in 1978, Ph.D., lecturer. His research interests include distributed simulation and virtual reality.

ZHAO Qin-Ping, born in 1948, Ph.D., professor, Ph.D. supervisor. His research interests include virtual reality, visualization and AI.

Background

In view of the key problems in large-scale distributed virtual environment including rapid application development and efficient run-time services for large-scale applications, the National Basic Research Program of China (973 Program) under grant No. 2002CB312105 and other High-Tech Programs setup Project "The Application Development and Run-Time Platform of Distributed Interactive Simulation — BH HLA/RTI" as the emphasis of support. BH RTI is deemed to enhance the scalability and persistency and provide efficient run-time services in large-scale distributed simulation and virtual environment. The team started work in BH RTI from 2001 after a sum-up of their related research and applications and has achieved many advancements in several aspects such as LoI extension to HLA, RTI congestion control etc. Presently BH RTI can support run-time services for thousands of federates and tens of thousands of objects, which is a big step in the scalability of distributed simulation systems.

BH RTI is based on multicast technologies and the contribution of this paper is in order to improve the multicast address usage and solve the multicast address finiteness problem in distributed virtual environment.