计算机研究与发展 ISSN 1000-1239/CN 11-1777/TP
Journal of Computer Research and Development 45(10): 1654-1661, 2008

基于缓冲 *Cell* 的用户迁移方法研究

周 忠 王少峰 吴 威
(虚拟现实技术与系统国家重点实验室(北京航空航天大学) 北京 100083)
(北京航空航天大学计算机科学与技术学院 北京 100083)
(zz@vrlab.buaa.edu.cn)

An Avatar Migration Mechanism Based on Cell Buffer

Zhou Zhong, Wang Shaofeng, and Wu Wei
(*State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing* 100083)
(*School of Computer Science and Technology, Beihang University, Beijing* 100083)

Abstract Avatar migration which changes the resident server of an avatar is an important issue in distributed virtual environment. Existing methods for multi-server DVE (distributed virtual environment) systems mainly conduct avatar migration according to the relationship between the avatar's movement and servers' region buffer, and the region buffers are fixed. However, regions are often regulated to balance the load in the latest multi-server DVE technologies. Then affected avatars have to migrate during the period of region regulation. This avatar migration of dynamic region buffer needs further study. An avatar migration mechanism based on cell buffer is proposed in this paper, where each region buffer is composed of a group of cells. Both cell and avatar are assigned to some status. Avatar migration is induced by two kinds, either avatar movement or region regulation. The mechanism abstracts the two kinds into a process of four critical conditions of status migration. Migration operations are devised on the status of avatar and the cell it resides on. In this way, many avatars can migrate concurrently with efficiency. The attribute update request of an avatar is only put in its region server's charge, which guarantees the status consistency. Experiments show that the mechanism can support concurrent migration efficiently, adding a low communication cost.

Key words distributed virtual environment; multi-server; avatar migration; region buffer; area of interest

摘 要 分布式虚拟环境中现有的用户迁移方法主要基于固定缓冲区,而近年研究热点的多服务器结构采用动态缓冲区,服务器区域调整中受影响的用户不能进行迁移,影响了交互性和效率.提出一种基于缓冲 *Cell* 的用户迁移方法,将缓冲区用一组 *Cell* 来表示,将用户替身运动和区域调整引起的迁移抽象为4项状态迁移条件.根据用户和所属 *Cell* 的状态变化触发用户状态迁移,多个用户可并发高效地完成迁移.实验表明,这种用户迁移方法可有效支持用户的并发迁移,并具有较低的通信附加负载.

关键词 分布式虚拟环境; 多服务器; 用户迁移; 缓冲区; 感兴趣区域

中图法分类号 TP393

收稿日期: 2007-03-19; 修回日期: 2008-07-02
基金项目: 国家"八六三"高技术研究发展计划基金项目(2006AA01Z331); 国家自然科学基金项目(60603084)

采用多个服务器来支持分布式虚拟环境系统已经成为互联网上虚拟现实、网络游戏应用的常用技术, 这些服务器一般负责真实或虚拟世界中不同的地理区域, 因此用户迁移是其中的重要问题. 在大部分应用中, 这种区域的划分是静态的, 各服务器之间不进行负载平衡, 除了最直接的所有权直接转移外, 用户迁移一般发生在用户替身在跨越区域边界运动的过程中, 其处理较为简单. 而基于多服务器(multi-server 或 server cluster)的分布式虚拟环境的原型系统约出现在 1995 年, 如 BrickNet[1], RING[2], NetEffect[3]. 近年来, 随着互联网的升级换代, 这成为一个研究热点, 出现了一些新的实验系统, 如 CittaTron[4], CyberWalk[5], ATLAS[6] 等. 虚拟环境同样划分为多个区域, 每个区域由一台服务器负责, 服务器之间以高速网络连接, 相邻的服务器可以通过调整负责区域的大小来进行负载平衡. 这种动态缓冲区的用户迁移新引入了计算开销、网络开销和用户状态一致性的问题.

在基于多服务器的分布式虚拟环境中, 用户迁移的产生主要包括两类情况, 一是由于用户替身的运动跨越了不同服务器负责的区域, 二是由于区域调整引起了用户及其邻居的所属权变化. 现有的原型系统研究大多关注于多服务器的场景划分和负载平衡技术, 对第 2 类用户迁移研究较少, 将区域调整影响的用户在调整过程中直接迁移, 更不考虑这两类迁移都存在时的处理[1-6].

已有的一些研究主要集中于第 1 类情况, 比较有代表性的有: Huang 等人在相邻区域间设置共有的边界(border), 根据用户在边界的位置变化, 通过状态图来控制用户迁移的过程[7]. 边界实现了区域缓冲的功能, 可避免反复越界带来的突发开销, 但未考虑边界内部用户的状态一致性和多个区域相邻的边界处理等. Yuan 等人提出在区域的外部设置扩展边界(extended border), 对用户的迁移进行预判, 可预先建立迁移所需连接[8]. 这种方法将扩展边界设在区域的外部, 相邻区域有重叠, 在实现多服务器的负载平衡时, 扩展边界的动态维护很困难. Cai 等人基于分布交互仿真的 HLA(high level architecture)标准进行多服务器网络游戏的构建[9], 刘晓建等人设计了 HLA/RTI 框架下的实体迁移协议[10], 其迁移机制都是通过 HLA 的所有权管理和数据分发管理完成. 基于 HLA 服务实现用户迁移可使迁移过程标准化, 但这是 HLA 相关的, 效率也有赖于 RTI 的具体实现.

针对这种动态缓冲区的用户迁移问题, 本文提出一种基于缓冲 *Cell* 的用户迁移方法, 将缓冲区用一组 *Cell* 来表示, 将用户运动和区域调整引起的两类迁移抽象为 4 项状态迁移的临界条件. 根据用户和所属 *Cell* 的状态变化, 触发相应的用户迁移消息, 这样多个用户可并发高效地完成迁移.

1　多服务器的 DVE 系统结构

基于用户感知范围的局部性, 将整个虚拟场景分割成多个区域, 每个区域由一台服务器来负责, 将整个虚拟环境的负载分配到多台服务器, 以提高系统的整体负载能力, 典型的基于多服务器的分布式虚拟环境系统结构如图 1 所示. 整个虚拟环境分割成多个 *Region*, *Region* 由最小的规则单元(*Cell*)组成, 每一 *Region* 由一个服务器 RS(region server)负责, 主控服务器(master server)对 RS 进行协调控制, 数据库(database)记录一些静态信息.

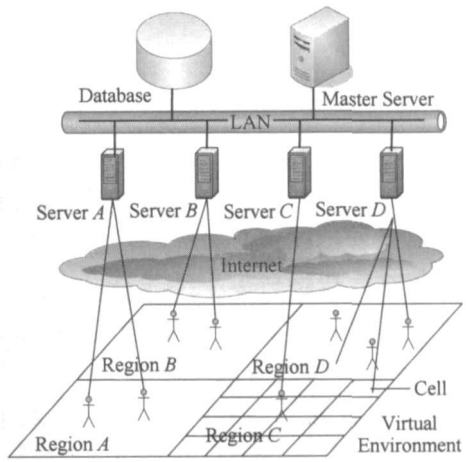


Fig. 1　Multi-server DVE system.

图 1　基于多服务器的分布式虚拟环境

基于上述的多服务器系统结构, 下面给出用户迁移涉及到的一些关键对象定义.

Cell: 整个虚拟环境在二维上划分为多个相邻的规则正方形小单元, 每个单元称为 *Cell*. *Cell* 是虚拟环境中最小的划分单元, 不同的 *Cell* 以下标 m, n …进行区分. *Cell* 划分是静态的, 在系统运行过程中不会发生变化.

Region: Region 是一组相邻 *Cell* 的集合, 整个虚拟环境进行初始划分后, 各个 *Region* 大小是相等的, 但在系统运行过程中 *Region* 的大小可能发生变化, 即管理的 *Cell* 数目可能改变. 每个区域由一个区域服务器 RS 负责, 不同的 RS 以下标 i, j, t, v …

进行区分.

感兴趣区域*AOI*: 用户*AOI*代表用户当前的感知范围, 一般以用户替身位置为中心的圆形、正方形或六边形等形状的区域表示. 与HLA相一致, 本文采用以用户替身位置为中心的正方形区域来表示, 如图2所示:

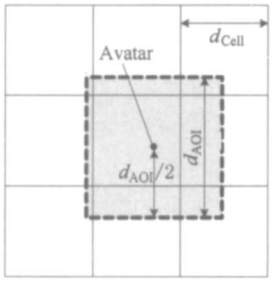


Fig. 2 Avatar's *AOI*.

图2 用户的兴趣区域*AOI*

图中*AOI*的边长为d_{AOI}, *Cell*的边长为d_{Cell}, 满足$d_{Cell} < d_{AOI} < 2 \times d_{Cell}$, 这样用户的*AOI*位于替身所在*Cell*及其相邻8个*Cell*内.

2 基于缓冲*Cell*的用户迁移方法

我们进行用户迁移处理的主要思想是, 在用户替身进入一个RS的缓冲区时进行预处理, 将整个迁移过程分配到用户所在缓冲*Cell*的移动中, 根据*Cell*状态和用户位置的变化, 触发相关用户迁移消息, 接收方RS进行响应处理, 协同完成迁移过程.

2.1 定 义

根据*Cell*在用户迁移过程中的状态, 我们将*Cell*分为3种类型, 并给出相关定义.

1) *Cell*类型. RS_i内$Cell_m$的类型$Type(Cell_m)$, 定义*Cell*有3种类型: ①缓冲*Cell*, 指RS_i边界上的*Cell*, 它们和其他区域相邻, 记为*BUFFERCELL*; ②内部*Cell*, 指除缓冲*Cell*外RS_i负责的*Cell*, 记为*INTERCELL*; ③外部*Cell*, 指RS_i外的非缓冲*Cell*, 记为*EXTERCELL*.

2) 缓冲*Cell*集合. RS_i的缓冲*Cell*集合$BufferCell(RS_i)$指RS_i内的缓冲*Cell*集合, 简称为RS_i的缓冲区; 除缓冲区外RS_i负责的*Cell*集合为RS_i的内部*Cell*集合, 简称RS_i的内部区; RS_i的外部*Cell*集合简称为RS_i的外部区.

当一个用户进入RS的缓冲区时, 如果用户的*AOI*覆盖相邻区域内的用户, 会引起RS间的网络通信. 同时, 用户可能会由于运动从一个区域迁移到另一个区域. 如图3所示, 用户*A*, *B*分别位于RS_i和RS_j的缓冲区, *A*和*B*可见会引起RS_i和RS_j间的通信, 同时用户*A*可能会运动迁移到$Region_j$.

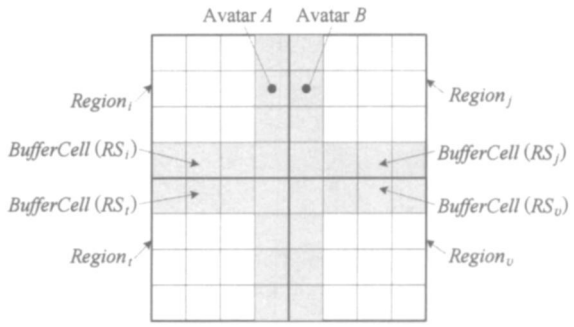


Fig. 3 Region's BufferCell.

图3 区域的缓冲*Cell*集合

3) 关联RS集合$RelativeRS(Cell_m)$. 获取一个$Cell_m$直接相邻的*Cell*列表(包括对角相邻), 其中*Cell*所属的RS集合为$Cell_m$的关联RS集合, 记为$RelativeRS(Cell_m)$. 位于$Cell_m$的用户可能与$RelativeRS(Cell_m)$内的RS产生交互. 如图3所示, 设用户*A*位于$Cell_m$, 则$RelativeRS(Cell_m)$包含RS_j, *A*可能与RS_j内的用户产生交互.

4) 实体对象和影子对象. 一个用户在*Region*间迁移时, 会和多个RS发生关联, 多个RS都会维护该用户的状态信息. 为了与RS_i维护的用户对象进行区别, 称RS_i维护实体对象*A*, RS_j维护影子对象A'. 此处影子对象就是指各RS上维护的远程对象, RS只能被动地接收其状态更新.

2.2 迁移消息定义

用户在RS间迁移, 需要RS和用户、RS之间通过网络消息协调完成, 表1给出了用户迁移相关的消息定义:

Table 1 Messages of Avatar Migration

表1 用户迁移网络消息定义

direction	message	description
$RS \rightarrow avatar$	*MS_CONNECT*	Connects to its RS.
	MS_CLOSE	Disconnect with its RS.
	MS_RS_CHG	Avatar's RS changes.
$RS_i \rightarrow RS_j$	*MS_BUF_AUPD*	Send avatar's update to relative RS.
	MS_BUF_AM	Avatar's record migrates from old RS to new RS.
$avatar \rightarrow RS$	*MC_CONNECT*	Connects to relative RS
	MC_CLOSE	Disconnect with former relative RS.

表1主要包括RS给用户、用户给RS, RS之间3类消息. 在用户迁移过程中根据*Cell*类型和用户

替身当前的位置信息, 触发相应消息的发送, 接收方进行响应, 协调完成用户迁移的过程. RS 间的网络通信负载主要来自 *MS_BUF_A UPD*, 此消息主要将用户的位置等属性信息更新给关联服务器, 其余信息由用户所属的 RS 进行维护.

2.3　迁移临界条件

对于区域服务器 RS_i, 用户迁移的临界条件在 *Cell* 类型或用户替身所在 $Cell_m$ 发生变化时触发. 根据用户 *A* 位置更新前后的 *Cell* 类型, 建立迁移相关的 4 个临界条件, 设计了每个条件下对迁移消息的触发及处理. 缓冲区的动态变化可归结到这 4 个临界条件, 触发相应的用户迁移处理.

1) C1: $Type(Cell_m) = INTERCELL \;\&\&\; Type(Cell_n) = BUFFERCELL$,
表示 *A* 从区域服务器 RS_i 的内部 $Cell_m$ 移动到了一个缓冲 $Cell_n$, 则 *A* 可能与 $Cell_n$ 关联的 RS 集合 $RelativeRS(Cell_n)$ 内的 RS 产生交互. 如图 4 所示:

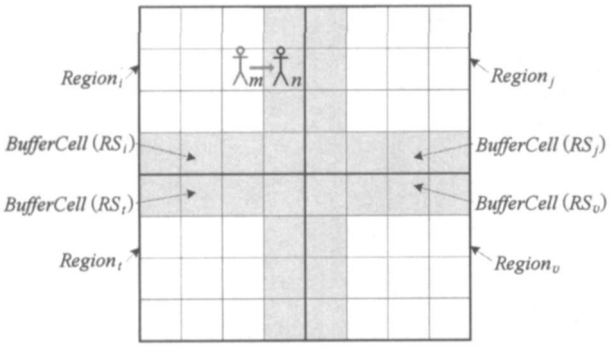


Fig. 4　Migrate from inside to buffer *Cell*.
图 4　用户从内部 *Cell* 迁移到缓冲 *Cell*

处理: 对于 $RelativeRS(Cell_n)$ 内的每个 RS_k, RS_i 给用户 *A* 发送 *MS_CONNECT*, *A* 接收到后获得 RS_k 信息, 给 RS_k 发送 *MC_CONNECT* 与 RS_k 建立连接. RS_k 维护影子对象 A', 用户 *A* 将自己的状态更新请求只发送给其所属的 RS_j, RS_j 处理后通过 *MS_BUF_A UPD* 将必要的状态更新发送给 RS_k, 用户同时从所有建立连接的 RS 接收更新.

2) C2: $Type(Cell_m) = BUFFERCELL \;\&\&\; Type(Cell_n) = BUFFERCELL$,
表示 *A* 在 RS_i 的缓冲 *Cell* 间移动, 因为 $RelativeRS(Cell_m)$ 和 $RelativeRS(Cell_n)$ 可能不同, 所以 *A* 关联的 RS 可能发生变化. 如图 5 所示.

处理: 计算 $RelativeRS(Cell_m)$ 和 $RelativeRS(Cell_n)$, 对于任何一个 RS_s 属于 $RelativeRS(Cell_n)$ 且不属于 $RelativeRS(Cell_m)$, RS_i 给用户 *A* 发送

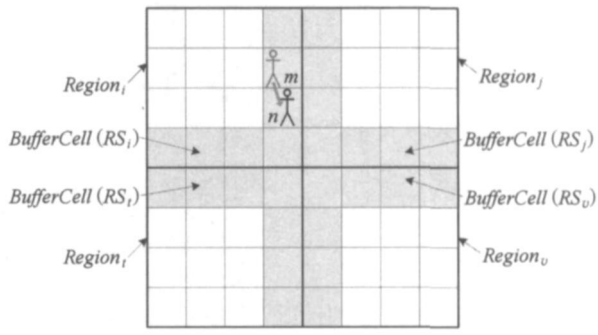


Fig. 5　Move in buffer Cells.
图 5　用户在缓冲 *Cell* 间移动

MS_CONNECT 让其与 RS_s 建立连接; 对于任何一个 RS_t 属于 $RelativeRS(Cell_m)$ 且不属于 $RelativeRS(Cell_n)$, RS_i 给 *A* 发送 *MS_CLOSE* 让其与 RS_t 断开连接. *A* 收到相应消息后通过发送 *MC_CONNECT* 或 *MC_CLOSE* 进行响应. 当 RS 接收到 *A* 发送的 *MC_CONNECT* 后, 建立影子对象 A', 接收到 *MC_CLOSE* 后, 销毁影子对象 A'.

同时, 当 RS_i 将用户 *A* 的状态更新请求处理后, 发送 *MS_BUF_A UPD* 将必要的信息更新给 *A* 关联的 RS.

3) C3: $Type(Cell_m) = BUFFERCELL \;\&\&\; Type(Cell_n) = EXTERCELL$.

设 $Cell_n$ 属于 RS_j, 则表示 *A* 从 RS_i 的缓冲 $Cell_m$ 进入到了 RS_i 的外部 $Cell_n$, 即 RS_j 的缓冲区 $Cell_n$. 即 *A* 所属的 RS 由 RS_i 改变为 RS_j.

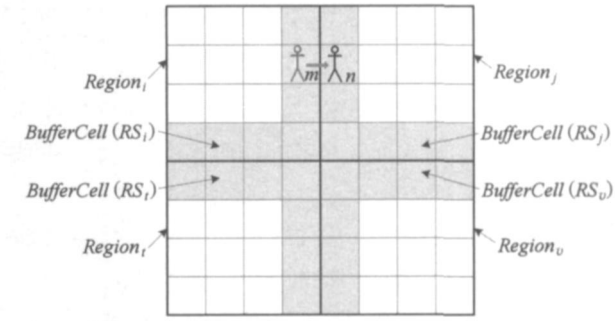


Fig. 6　Migrate from buffer *Cell* to outside.
图 6　用户从缓冲 *Cell* 迁移到外部 *Cell*

处理: RS_i 给用户 *A* 发送 *MS_RS_CHG* 消息告知其所属的 RS 改为 RS_j, *A* 收到后更新其所属的 RS; RS_i 发送 *MS_BUF_AM* 将 *A* 的详细对象信息迁移给 RS_j, RS_j 收到消息后, 将维护实体对象 *A*, 而 RS_i 将维护影子对象 A', 此过程中网络连接不需要改变.

4) C4: $Type(Cell_m) = BUFFERCELL \&\& Type(Cell_n) = INTERCELL$,
表示用户 A 从 RS_i 的缓冲 $Cell$ 进入到内部 $Cell$, 如图 7 所示, 有两种可能.

C4.1: A 原来属于 RS_i 的内部 $Cell$, 进入到 RS_i 的缓冲 $Cell$ 后又回到 RS_i 的内部 $Cell$, 没有发生迁移;

C4.2: A 从相邻的 RS_j 进入到 RS_i 的缓冲 $Cell$, 然后进入 RS_i 的内部 $Cell$, 发生了迁移.

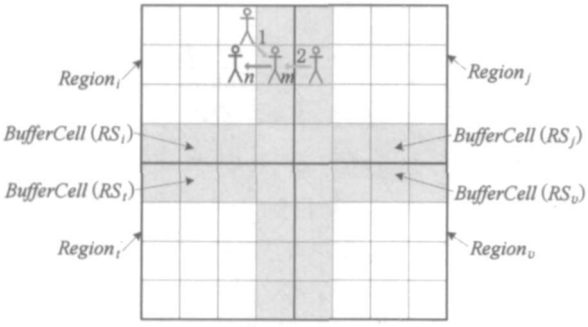


Fig. 7 Migrate from buffer $Cell$ to inside.

图 7 用户从缓冲 $Cell$ 迁移到内部 $Cell$

处理: 对于 $RelativeRS(Cell_m)$ 内的每个 RS_k, RS_i 给用户 A 发送 MS_CLOSE 告知其与 RS_k 断开连接, 然后 A 发送 MC_CLOSE 给 RS_k 并且断开连接, RS_k 接收到 MC_CLOSE 后销毁影子对象 A'.

在多服务器进行负载平衡时, 一些缓冲区会发生变化, 体现为缓冲 $Cell$、内部 $Cell$、外部 $Cell$ 会发生类型转换, 一些缓冲 $Cell$ 发生迁移, 引起用户迁移. 根据受影响 $Cell$ 类型的变化, 可将其归结于上述的 4 个条件, 由关联的 RS 分别进行处理. 以图 8 为例, RS_i 将 $Cell_m$ 转移给 RS_j, 导致 RS_i 和 RS_j 的缓冲区发生变化, 除 $Cell_m$ 外, $Cell_m$ 相邻的 $Cell$ 可能会受到影响.

图 8(a) 为 $Cell_m$ 迁移前 RS_i 和 RS_j 的缓冲区状态, $Cell_q$ 属于 RS_i 的内部区, $Cell_n$ 属于 RS_j 的缓冲区; 图 8(b) 为 $Cell_m$ 迁移后 RS_i 和 RS_j 的缓冲区状态, $Cell_q$ 属于 RS_i 的缓冲区, $Cell_n$ 属于 RS_j 的内部区, 因此, $Cell_m$ 迁移导致前后类型发生变化的有 $Cell_m$, $Cell_n$ 和 $Cell_q$. 对于 RS_i 来说, $Cell_m$ 和 $Cell_q$ 前后类型的变化分别为 $Type(Cell_m)_{old} = BUFFERCELL$, $Type(Cell_m)_{new} = EXTERCELL$, $Type(Cell_q)_{old} = INTERCELL$, $Type(Cell_q)_{new} = BUFFERCELL$, 分别满足上述的临界条件 C3 和 C1, 按照相应条件进行迁移处理. RS_j 应进行类似的处理.

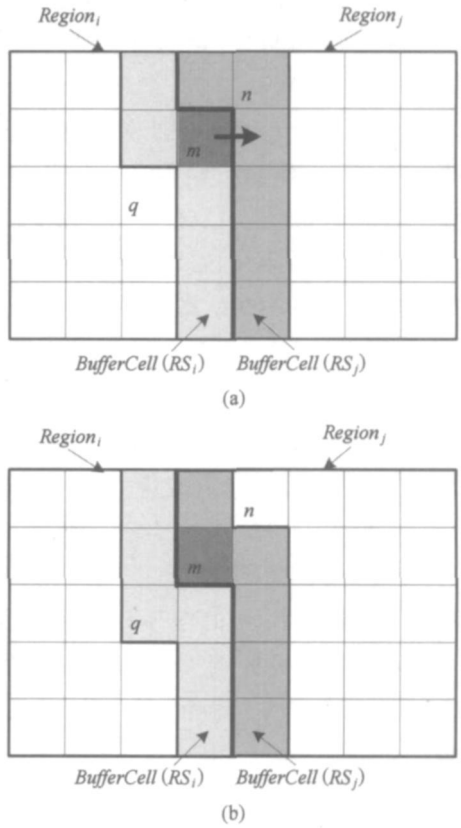


Fig. 8 Migrate during $Cell$ migration. (a) Before migration of $Cell_m$ and (b) After migration of $Cell_m$.

图 8 缓冲区变化时用户迁移处理示意图. (a) $Cell_m$ 迁移前; (b) $Cell_m$ 迁移后

2.4 迁移处理流程

基于上面的 4 项用户迁移临界条件, 设用户 A 从区域服务器 RS_i 迁移到区域服务器 RS_j, RS_i 和 RS_j 的迁移处理流程分别如图 9 所示.

在图 9(a) 中, 用户 A 初始状态在 RS_i 的内部区域, 满足条件 C1 时, 将进入 RS_i 的缓冲 $Cell$, 与其关联的 RS_j 会建立连接, RS_j 会维护影子对象 A', 用户 A 在 RS_i 的缓冲区内移动时, 根据 A 移动的位置判断触发的临界条件: 当满足 C2 时, 用户继续在缓冲区移动; 当满足 C4.1 时, 用户从 RS_i 的缓冲区回到 RS_i 的内部区, 没有发生迁移; 当满足 C3 时, 用户 A 离开 RS_i, 进入 RS_j 负责区域, 此时 RS_i 将维护影子对象 A', 而 RS_j 将维护实体对象 A, RS_i 会给用户 A 发送 MS_RS_CHG 消息通知用户 A 所属的 RS 发生变化, 将由 RS_j 负责处理用户 A 发出的状态更新请求.

图 9(b) 表示了用户 A 进入到 RS_j 的缓冲区后的处理, 随着用户 A 在 RS_j 的缓冲区内的移动, 当满足 C4.2 条件时, A 将进入 RS_j 的内部区域, 完成

迁移, RS_j 将通知 A 断开和 RS_i 的连接; 当满足 C3 条件时, A 将离开 RS_j 的缓冲区, 进入到别的区域服务器的缓冲区, 继续进行图 9(b)相同的处理.

```
while(true) {
  get avatar A's new position;
  get avatar A's Cell_n;
  if(Type(Cell_m)= INTERCELL && Type(Cell_n)=
  BUFFERCELL)
  {//C1 is satisfied
    RS_i asks A connect to RS_j;
    RS_j build shadow object A'; }
  else if(Type(Cell_m)= BUFFERCELL && Type(Cell_n)=
  BUFFERCELL)
  {//C2 is satisfied
    A sends update to RS_i;
    A recv updates from RS_i and RS_j;}
  else if(Type(Cell_m)= BUFFERCELL && Type(Cell_n)=
  EXTERCELL)
  {//C3 is satisfied
    RS_i notice A its changed RS;
    RS_j is now in charge of A's update request;
    break; }
  //C4.1 is satisfied
  else if(Type(Cell_m)= BUFFERCELL && Type(Cell_n)=
  INTERCELL)
    RS_i asks A to disconnect with RS_j;
  else// A is inside RS_i
    RS_i is in charge of A;
}
```

(a)

```
while(true) {
  get avatar A's new position
  get A's Cell_n;
  if(Type(Cell_m)= BUFFERCELL && Type(Cell_n)=
  BUFFERCELL)
  {// C2 is satisifed
    A send update request to RS_j;
    A receives updates from RS_i and RS_j; }
  else if(Type(Cell_m)= BUFFERCELL && Type(Cell_n)=
  INTERCELL)
  {// C4.2 is satisifed. A from RS_i to RS_j
    RS_j asks A vdisconnect with RS_i; }
  else if(Type(Cell_m)= BUFFERCELL && Type(Cell_n)=
  EXTERCELL)
  {// C3 is satisifed. A returns from RS_j to RS_i
    RS_j notices A changed RS;
    RS_i is now in charge of A's update requests;
    break; }
  else // migration completed and A is inside RS_j
    RS_j is in charge of A;
}
```

(b)

Fig. 9　Process of avatar migration. (a) RS_i process on A migration and (b) RS_j process on A migration.

图 9　用户迁移处理流程. (a) RS_i 对用户 A 的迁移处理; (b) RS_j 对用户 A 的迁移处理

在实现用户迁移方法时有几个实现方面的具体考虑: 1) 当用户进入缓冲区时, 预先建立连接, 相关服务器维护影子实体, 由 RS 根据 AOI 的匹配情况进行必要数据的转发, 使整个迁移过程对用户透明; 2) 用户的属性更新请求只发送给其所属的 RS, 由 RS 处理后, 更新给用户关联的其他 RS, 保证用户状态一致性, 防止了用户更新发送给多个 RS 进行处理发生不一致的情况; 3) 用户在从一个区域迁移到另一个区域时, 不发生网络连接的变化, 仅切换所属的 RS, 连接都是预先建立的, 可以尽量减少连接的建立和断开带来的影响.

3　实验及结果分析

我们实现了支持用户迁移机制的区域服务器程序 RegionServer 和用户迁移测试工具 MigrationTest. MigrationTest 控制在虚拟环境中加入不同数量的用户, 触发用户迁移. 实验将用户在虚拟环境中的某个初始区域进行随机分布, 同时触发用户进行 RS 间的迁移.

3.1　实验方案及环境设置

实验方案将整个虚拟环境划分为两个 *Region*, 相应由两台 RS 负责, 每个 *Region* 划分为 25 × 25 个 *Cell*, 如图 10 所示. 在区域 A 中标识的用户初始化区域内, 随机加入一定数量的用户然后同时向区域 B 进行迁移, 直至所有用户进入区域 B 的内部区域完成整个迁移.

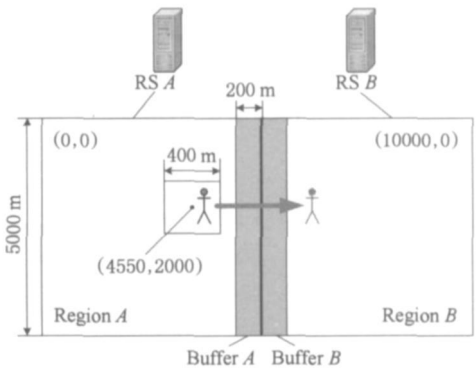


Fig. 10　Experiment settings.

图 10　用户迁移测试环境

实验环境设置如下: *Cell* 的半边长为 100 m, 每个区域的边长为 5000 m; 用户移动速度 5 m/s; 用户初始化区域中心坐标为(4550, 2000), 半边长为 200 m; 用户数量选择 16, 32, 64 个. RS 配置: P4 2.8 GHz CPU, 512 MB 内存, Windows XP; 网络环境: 100 Mbps 局域网.

3.2 测试指标

基于迁移过程对用户的影响、用户及区域服务器引入的通信负载等方面来衡量用户迁移机制的有效性,设计如下的性能测试指标:

1) RS 对迁移用户的平均响应时间(migration response time, MRT). 在用户迁移过程中,区域服务器对所有迁移用户的平均回路(round-trip)响应时间,每秒统计一次. *MRT* 的变化越小说明用户迁移对客户端的影响越小,迁移过程越平滑.

2) 迁移用户平均接收速率(user receiving speed, URS). 迁移用户从关联的 RS 上接收数据的平均速率(KB/s),每秒统计一次. *URS* 越高,说明用户迁移引入的通信负载越大.

3) RS 平均通信速率(server communication speed, SCS). 在用户迁移过程中,关联的 RS 间网络通信平均速率(KB/s),每秒统计一次. *SCS* 越小,说明迁移引入的 RS 间网络通信量越小.

3.3 结果分析

在上述的实验环境下,对 16, 32 和 64 个用户的迁移过程进行测试,获得 RS 对迁移用户的平均响应时间 *MRT*(ms),如图 11 所示. 从图中可以看出,对于小规模用户(16 个)的同时迁移, *MRT* 在 44~48 ms,中规模用户的 *MRT* 在 54~64 ms,较大规模的用户(64 个)的 *MRT* 略高为 70~80 ms. 在整个迁移过程中, *MRT* 没有很大的波动,整个迁移过程较为平滑.

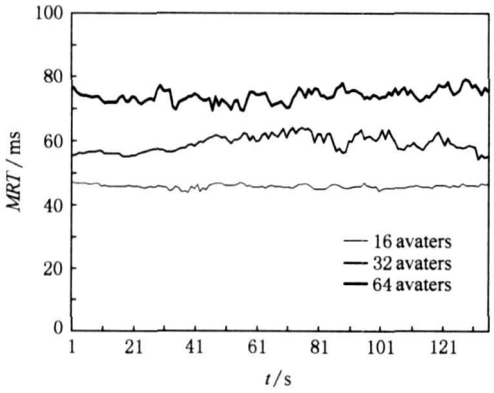


Fig. 11 *MRT* experiment results.

图 11 RS 对迁移用户的平均响应时间

对于多个用户的同时迁移,迁移用户平均接收速率 *URS* 与未迁移时比较,迁移过程中用户的 *URS* 由两个关联的 RS 同时负担. 图 12 是 64 个用户迁移过程中的 *URS*, 在迁移过程中,迁移用户平均接收速率 *URS* 峰值约为 23 KB/s,迁移对用户引入的通信负载较小.

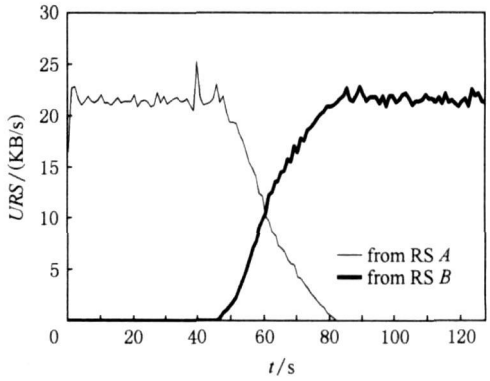


Fig. 12 *URS* experiment results (64 avatars).

图 12 用户平均接收速率(64 个用户迁移)

用户在从 RS *A* 迁移到 RS *B* 的过程中, RS 需要通过网络通信进行协调,以完成用户迁移. 图 13 给出了 64 个用户迁移过程中 200 s 内 RS 间平均通信速率的情况, RS 平均通信速率 *SCS* 的峰值约为 34 KB/s.

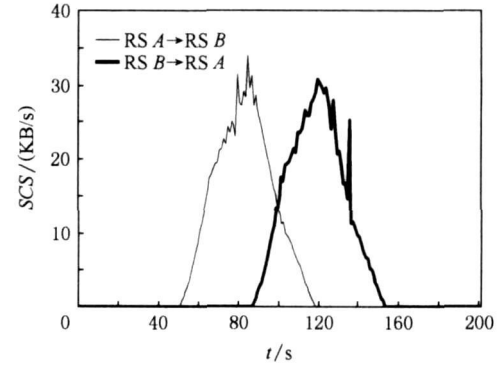


Fig. 13 *SCS* experiment results (64 avatars).

图 13 RS 间平均通信速率(64 个用户)

通过用户平均接收速率 *URS* 可以得出,用户数为 64 时,迁移过程中 RS 对用户的总通信速率约为 64 × 23 = 1.47 MB/s. 因此,迁移引入的服务器间通信负载占此 RS 总通信负载的 34/(34 + 1470) = 2.26%, 对 RS 的通信影响较小.

4 结束语

针对基于多服务器的分布式虚拟环境系统中存在的动态缓冲区问题,本文提出一种基于缓冲 *Cell* 的用户迁移方法,将缓冲区用一组 *Cell* 来表示,为每个 *Cell* 和用户赋予相应的状态,将用户替身运动和区域调整引起的两类迁移定义为 4 项状态迁移的临界条件. 根据用户和所属 *Cell* 的状态变化,触发相应的用户迁移消息,这样多个用户可并发高效地

完成迁移.在迁移过程中用户的属性更新请求只由其所属的区域服务器处理,保证了迁移用户状态的一致性.实验表明,这种用户迁移方法可有效支持用户的并发迁移,并具有较低的通信附加负载,引入的服务器间通信负载只占服务器通信总量的很少比例.

参 考 文 献

[1] Singh G, Serra L, Png W, *et al*. BrickNet: Sharing object behaviors on the Net [C] //Proc of IEEE VRAIS'95. Piscataway, NJ: IEEE, 1995: 19-25

[2] Funkhouser T A. RING: A client-server system for multi-user virtual environments [C] //Proc of ACM SIGGRAPH 1995 Symp on Interactive 3D Graphics. New York: ACM, 1995: 85-92

[3] Das T, Singh G, Mitchell A, *et al*. NetEffect: A network architecture for large-scale multi-user virtual worlds [C] //Proc of ACM VRST. New York: ACM, 1997: 157-163

[4] Hori M, Iseri T, Fujikawa K, *et al*. Scalability issues of dynamic space management for multiple-server networked virtual environments [C] //Proc of IEEE PACRIM. Piscataway, NJ: IEEE, 2001: 200-203

[5] Ng B, Si A, Lau R W H, *et al*. A multi-server architecture for distributed virtual walkthrough [C] //Proc of ACM VRST. New York: ACM, 2002: 163-170

[6] Lee D, Lim M, Han S, *et al*. ATLAS: A scalable network framework for distributed virtual environments [J]. PRESENCE: Teleoperators and Virtual Environments, 2007, 16(2): 125-156

[7] Huang J Y, Du Y C, Wang C M. Design of the server cluster to support avatar migration [C] //Proc of IEEE Virtual Reality. Piscataway, NJ: IEEE, 2003: 7-14

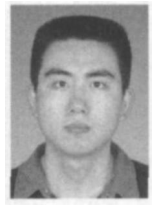
[8] Yuan Q, Lu D. A latency-adaptive communication architecture for inter-networked virtual environments [C] //IEEE ICSMC. Piscataway, NJ: IEEE, 2004: 6296-6301

[9] Cai W, Xavier P, Turner S J, *et al*. A scalable architecture for supporting interactive games on the Internet [C] //Proc of the 16th PADS. Piscataway, NJ: IEEE, 2002: 54-61

[10] Liu Xiaojian, Zhong Hairong, Jin Shiyao. Migrating entities and their time synchronization issue in large-scale federated simulation (Part One): Protocol and implementation for migrating entities [J]. Journal of Computer Research and Development, 2005, 42(4): 711-715 (in Chinese)
(刘晓建,钟海荣,金士尧.大规模联邦仿真中实体迁移及其时间同步研究(一)实体迁移协议与实现[J].计算机研究与发展,2005,42(4):711-715)

Zhou Zhong, born in 1978. Ph. D. and associate professor. His main research interests include distributed virtual reality and computer networks.

周　忠,1978 年生,博士,副教授,主要研究方向为分布式虚拟现实与计算机网络等.



Wang Shaofeng, born in 1981. Master. His main research interests include distributed virtual reality.

王少峰,1981 年生,硕士,主要研究方向为分布式虚拟现实应用技术.

Wu Wei, born in 1961. Ph. D., professor and Ph. D. supervisor. His main research interests include distributed virtual reality, computer networks and distributed system.

吴　威,1961 年生,博士,教授,博士生导师,主要研究方向为分布式虚拟现实、计算机网络、分布式系统等.

Research Background

Efficient run-time services are the focus of distributed virtual environment (DVE) and distributed interactive simulation (DIS). From 1999, the team of distributed virtual environment in the key laboratory of virtual reality, Beihang University, started the work on DVE technologies and run-time infrastructure (RTI) implementation. BH RTI① is deemed to enhance the scalability and persistency and provide efficient run-time services in large-scale distributed simulation and virtual environment. Much advancement has been achieved in several aspects such as LoI extension to HLA, RTI congestion control, multicast, time synchronization, *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. The contribution of this paper is aimed to improve the efficiency of concurrent avatar migration in multi-server DVE systems. This work is supported by the National High-Tech 863 Program of China under No. 2006AA01Z331 and the National Natural Science Foundation of China under No. 60603084.

① http://www.hlarti.com