可扩展的MMORPG游戏框架的设计与实现

李建微,陈 新,黄调祥,林 淮,邱贤熠
(福州大学 物理与信息工程学院,福建 福州 350002)

摘 要:MMORPG在运营过程中为了增强游戏的可玩性,增加同时在线人数,不定期要对系统进行更新,这就对整个软件框架设计提出了具有可扩展性的要求。根据MMORPG类游戏的特点,在C/S体系结构理论及先进软件设计原则与设计模式指导下设计系统的整体框架:服务器端综合采用基于Cell和基于服务两种技术,客户端采用五大系统进行划分;最后在OGRE图形引擎和RakNet网络引擎平台下开发了多人在线游戏。实践证明在互联网环境下本游戏框架具有良好的可扩展性。

关键词:大型多人在线角色扮演游戏;框架;可扩展性;设计模式
中图分类号:TP31 **文献标识码**:A **文章编号**:1673-629X(2012)02-0001-05

Design and Implementation of an Expansible Framework of MMORPG

LI Jian-wei,CHEN Xin,HUANG Zhou-xiang,LIN Huai,QIU Xian-yi
(College of Physics and Information Engineering,Fuzhou University,Fuzhou 350002,China)

Abstract:In order to enhance the playability and add the number of online user during the running of the MMORPG,the game software system should be updated irregularly,so it's important that the frame of software system should be optimized in design and be expansible easily overall. Based on the feature of MMORPG,by the system frame of client/server and under the guidance of advanced software design rule and model,realize the whole system frame,which is composed of server and client. The server is developed based on cell and serve and the client is made of five modules. In the end use an example to test and verify the expansibility of the frame.

Key words:MMORPG;framework;expansibility;design model

0 引 言

大型多人在线角色扮演游戏(MMORPG)是当前网络游戏中的主流游戏类型。由于需要支持数以万计乃至数十万甚至数百万玩家的同时在线,同时还要处理玩家的战斗、场景地图、角色升级等游戏逻辑,以及聊天组队、物品交易等玩家的互动交流,为玩家提供丰富的交互体验和流畅的游戏体验,这就对MMORPG本身的框架设计提出了很高的要求。

在设计MMORPG网游框架的过程中,需要框架具有较好的稳定性、高效性、可扩展性、可开发性[1]。其中可扩展性是衡量一个网游框架优劣的重要指标。再丰富的游戏内容也无法持续吸引玩家几年,一般的做法就是周期性地持续提供游戏更新。此外,随着注册人数的急剧增多,网游系统能否通过简单的升级从而同时增加玩家在线人数,由于网游系统这些特性,因此设计人员必须在框架设计阶段,从内容及功能两方面入手增强系统的可扩充性。

文中以现今最流行的MMORPG类游戏为设计对象,从服务器端和客户端两方面入手,在多种最新的设计模式的指导下实现系统框架、功能模块等内容的设计,最后通过实例对系统的可扩展性进行分析,实践证明该设计方案能够极大提高实现MMORPG类网游系统的可扩充性。

1 MMORPG游戏框架分析

基于C/S(Client/Server)结构的网络游戏框架是目前采用网游开发中采用较多的主流框架,即一个网游平台是由服务器端和若干个客户端所组成的。C/S模式方案中整个网游平台分布于客户端和服务器端上,客户端向服务器端发送请求,服务器端处理各种请求后反馈给客户端。C/S框架的缺点在于客户端和服

收稿日期:2011-07-05;**修回日期**:2011-10-15
基金项目:国家大学生创新性实验计划项目(101038633);福建省自然科学基金(2010J01255);福建省教育A类面上项目(JA09005);福州大学本科生科研实训计划项目(11118)
作者简介:李建微(1979-),男,助理研究员,博士研究生,研究方向为数字媒体与互动游戏设计;陈 新,硕导,教授,研究方向为虚拟现实。

务器端的高度非对称性,这导致服务器端负载过大[2]。虽然现今有人研究综合利用P2P技术和C/S技术的解决方案[3]或者是B/S技术与C/S技术的混合模式[4],但是无论从商业角度,还是技术角度,C/S结构依然是网游产业较为成熟可行的解决方案。文中将提出一种基于C/S结构的可扩展的MMORPG网游的框架方案。

2 可扩展的MMORPG网游框架设计

2.1 系统整体框架设计

多层C/S体系结构是目前网络游戏框架设计中比较成熟稳定的体系框架,文中将这种框架分为数据层、功能层、通信层、表示层(见图1)。

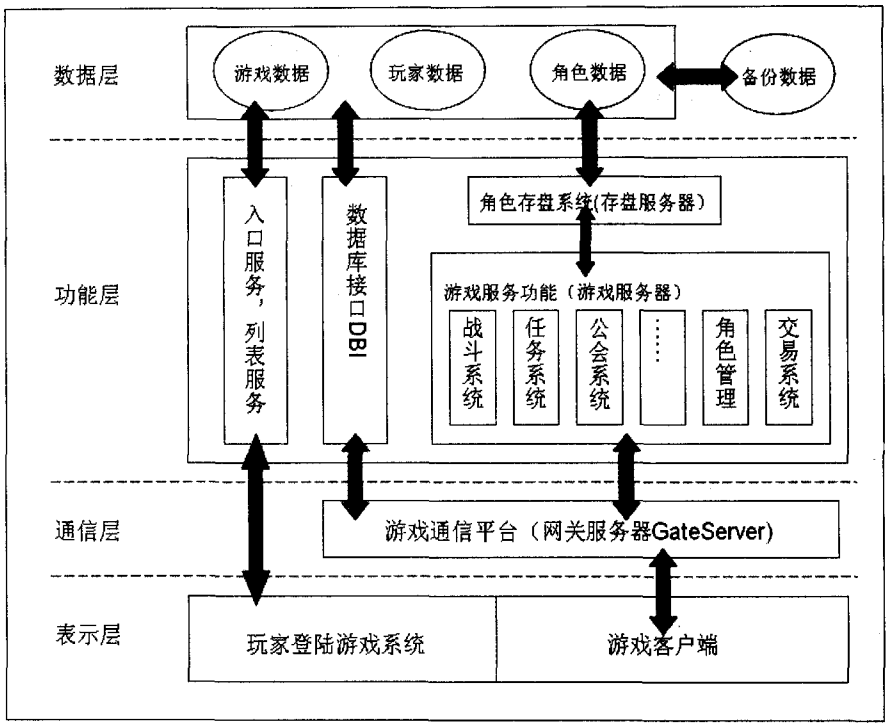

图1 基于C/S结构的MMORPG系统框架结构

(1)表示层。

负责玩家与网游平台的对话功能,是平台的对外用户接口部分。表示层接收玩家从键盘或鼠标上输入的数据与请求,接收应用服务输出的数据。游戏客户端就是在这一层。玩家输入的数据通过通信层传入功能层。

(2)通信层。

即通信服务器(GameServer),它负责功能层与表示层的数据交换工作,管理所有玩家的客户端与游戏服务器端的连接,并且负责客户端的登陆登出、计费校验工作。

(3)功能层。

负责网络游戏平台的所有业务逻辑处理,在功能层中包含有确认用户对应用和数据库存取权限的功能,以及记录系统处理日志的功能。

功能层可分为数据库和游戏服务模块。游戏服务模块被设计为一个个单独的子系统,用来处理游戏世界中的业务。数据库服务模块是网络游戏连接数据库读取玩家信息和服务器群组信息、存取玩家档案的接口。游戏服务模块实现具体的应用逻辑。平台的通信服务器受到玩家的操作请求后,转换成对数据库或游戏服务器的请求,调用服务器处理请求并将结构返回给通信服务器[5]。

(4)数据层。

即DBMS,负责管理对数据库的读写。中心数据库系统存放并处理所有玩家资料及其相应的数据,采用通用的大型关系数据库系统。

2.2 服务器端的设计

2.2.1 服务器端整体框架设计方案

服务器端分布在分层框架中的功能层,主要负责网游平台的所有游戏业务逻辑的处理。服务器端通常采用服务器集群技术,集群技术一般有两种方案:

(1)基于cell的框架。

这个框架中,每一个cell都在不同的物理服务器上运行完全一样的游戏,但是每个cell都负责承载不同的游戏场景区域的游戏逻辑,这种技术主要用于实现游戏世界的无缝连接[6]。

(2)基于服务的框架。

这种方案将整个游戏作为一个单一的场景来处理,但可以根据处理数据、逻辑的相关性来划分集群组内各个服务器的工作任务,如专门负责玩家聊天交流的聊天服务器、专门负责游戏战斗的战斗服务器、专门负责游戏道具的相关逻辑的道具服务器。这样的划分使得逻辑处理的并发有了可能,并为将来游戏的升级、增加新的游戏模块提供了方便。

文中采用综合以上两种设计方案的优点进行服务器框架设计(见图2)。步骤如下:先分割游戏场景,在总体上基于cell框架,而每一个场景采用基于服务的框架,有专门的服务器集群来处理该客户端的请求。最后通过综合运用服务器集群和负载平衡技术来实现每个单独场景内服务器的框架。

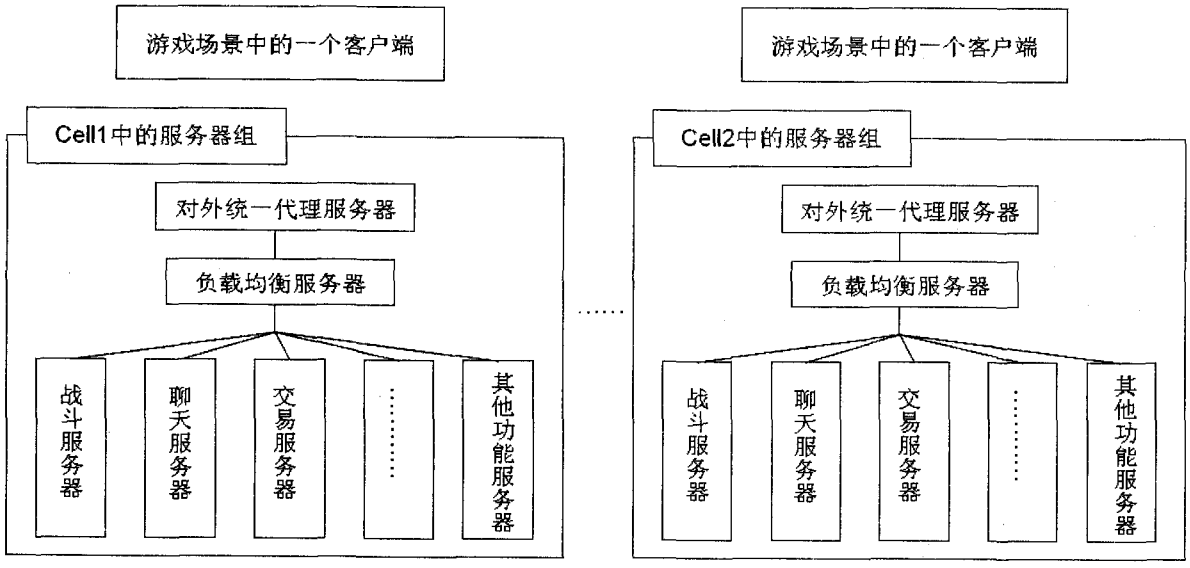

图2　综合运用基于cell和基于服务的服务器端的体系结构

2.2.2　服务器功能模块的设计

服务器端的服务器都采用IOCP线程池技术来解决游戏中的同步问题。本框架中的所有服务器程序基本可以分解为三个功能模块[7](见图3):

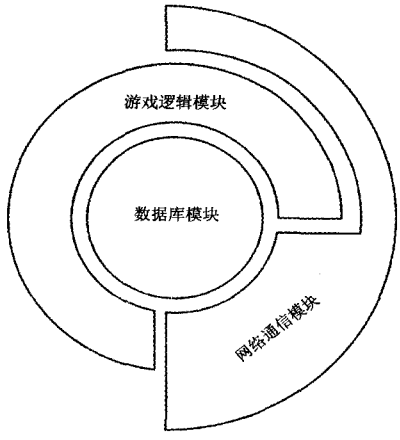

图3　服务器端功能模块

(1)游戏逻辑模块。

负责维护和管理服务器内部的诸多对象,处理这些对象之间的逻辑关系,响应客户端和其他服务器发过来的各种请求。在网络模块和数据库模块的协同下完成服务器网络数据包收发和数据库的访问之外的几乎所有事物。包括游戏主要逻辑的处理,如聊天服务模块、战斗服务模块、交易服务模块、公会服务模块等。

(2)网络通信模块 。

主要是网关服务器,一方面处理客户端的连接,管理客户端的会话;另一方面管理与服务器端的连接,保证数据交换的正常进行,控制玩家的登入登出,注册验证,并为玩家分配合适的游戏逻辑服务器等。它承担着建立连接和收发数据的任务,是服务器对外交流的通道。

(3)数据库模块。

数据库模块负责服务器和数据库的交互,包括访问和修改数据库。数据库模块处理的数据主要有三种,包括用户个人信息、用户的游戏信息、游戏资源[8]。

2.2.3　服务器游戏逻辑系统设计

为了使游戏更有竞争力,更能满足玩家的期望,从而到达吸引更多同时在线玩家,游戏需要不断地更新,这样就使得游戏的逻辑不断变化。而玩家的需求往往是庞大复杂的,这就使得游戏逻辑模块的设计不仅要充分反映用户的需求,还要易于修改和添加功能。这些问题在一定程度上给设计带来了复杂度,所以需要通过一些设计原则和模式来完成这些工作。

(1)问题分离原则。

为了确定用合理的工作量就可以将变更引入到系统中,变更应该局部化,需求中的一个变更应该只导致系统的某一个位置发生变革。对系统的变更进行分类后可以发现对游戏逻辑的变更和对技术的变更有不同的原因和约束条件[9]。应用中被分解的子问题都可以独立演变,这样可以降低整个系统的复杂度,单独考虑子问题。如果技术基础设施是各自独立于功能的,那么这个基础设施就可以在不同的几个项目中重用。命令模式、入口模式等设计模式是实现问题分离的几种设计模式。

(2)接口隔离原则。

游戏子系统是整个游戏的一个有机组成部分。因此,在实现上不存在A子系统与B子系统之间的严格无关性,A,B两个子系统之间的最低限度也会通过整个游戏系统产生联系[10]。为了达到游戏子系统模块化的目标,可以根据游戏子系统的需要产生子系统的

参与者接口，即实现游戏子系统参与者的接口化，这个接口可通过一个适配器模式进行转接而成，从而实现逻辑上游戏子系统之间的无关性。

每一个游戏逻辑子系统中发布供上层使用的接口，可以将容易变化的游戏逻辑部分和上层控制有效地分离开，使得它们直接只通过接口来耦合，这样使得它们可以独立变化，耦合度降低，优化了设计。

(3)依赖倒转原则。

子系统应针对或依赖抽象的接口进行编程，而不是依赖具体的对象进行编程。这个接口是通过适配器转接而成的。这个原则是从子系统内部来看问题，而接口隔离原则是从子系统外部来看问题。

(4)里氏代换原则。

如果子系统有可能需要进一步演化成多种形态的子系统，以适应玩家对游戏不断增长变化的需求，就应该将子系统的中介者设计成抽象类，以便演化出多种类型的中介者。如对于参与者类来说，它参与切磋模式与混战模式都是向战斗子系统通知它的行为，而切磋子系统的中介者与混战子系统收到通知后，会做出完全不同的行为逻辑，因此，将中介者设计为抽象类成为必要[11]。

(5)管理器原则。

在游戏中能够同时存在的子系统对象是有限的，因为子系统对象存在需要占用系统内存，创建与销毁也会耗费大量CPU资源。为了提高效率，可将子系统创建的中介者对象用管理器管理起来，每创建一个中介者对象，就加入到管理器对象池中，中介者对象池大小应该是动态增长的，本例中采用的是IOCP线程池技术。另外管理器的实现应该采用单件模式[12]。

2.3 客户端框架设计方案

客户端的功能模块见图4，主要有五大系统。

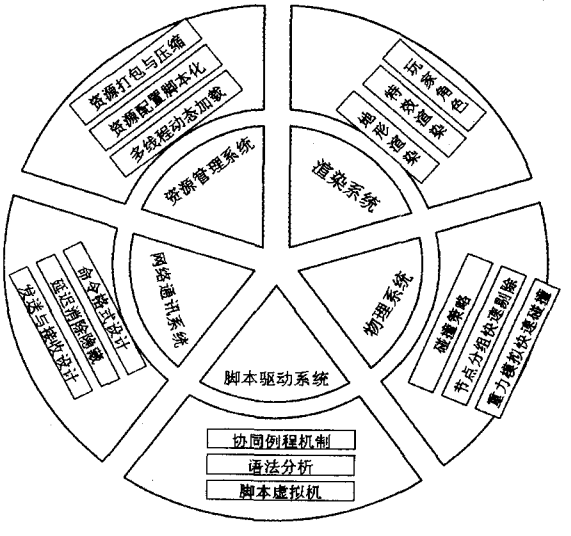

图4 客户端功能模块

(1)资源管理系统。

资源管理系统管理的资源主要包括硬盘存取、内存占用、CPU时间、带宽、显卡时间、显存这六大部分。它的工作包括资源打包与压缩、资源配置脚本化、动态多线程加载资源等。资源管理系统性能的好坏关系到整个网游系统运行的流畅和稳定。一个不好的资源管理系统会让游戏程序占用海量的内存资源，会让游戏的FPS(每秒绘制图像的帧数)降到无法忍受的程度，导致资源误用、程序崩溃等等。所以在构造客户端的时候，首先应该先设计好它的资源管理系统。

(2)渲染系统。

渲染系统提供最基本的渲染管道过程加速，最底层是3D加速卡硬件，其上是3D加速卡驱动程序，如Directx和OpenGL都对硬件和它们的驱动程序进行了抽象，提供API。另外再对Directx和OpenGL进一步抽象，以实现平台的无关性。一般来说，渲染系统包括四个子系统，它们是游戏界面系统、地形渲染、特效渲染、玩家角色。

(3)物理系统。

物理系统是最耗CPU资源的系统，它主要包括碰撞检测。由于3D网络游戏的在线玩家数量非常巨大，设计物理系统时必须力求简单快速。目前通过特殊的模型简化、节点式渲染管理、部分的三角形检测等技术手段来保证整个系统的高效运行。

(4)网络通信系统。

网络通信系统是玩家与玩家之间连接虚拟世界的纽带，负责客户端与服务器端的通信。它要保证通信过程的快速稳定。

(5)脚本驱动系统。

脚本驱动系统是为了方便开发和编辑一些纯逻辑的部分和较为简单的模块，加速制作和调试过程，让更多不懂C++的人员如策划参与进来。其核心就是一个协同例程体制，即先运行脚本模块，再转而执行C++内核模块，然后再跳转脚本模块。

3 整体方案的可扩展性分析

本例中的MMORPG游戏框架方案设计的目标是使得游戏系统平台具有一定的产品和技术特性和可扩展性。

系统从以下几个方面，实现本游戏的可扩展性：

(1)服务器端的每个组中的服务器在存储能力、扩展能力上预留巨大空间，整个系统的扩展空间基本能够满足将来一段时间业务的海量增长需求，同时能够方便地通过添加服务器组来实现承受更高的同时在线人数压力。

(2)系统采用基于服务的框架，使得系统可以通

过增加新的游戏逻辑服务器来增加新的游戏模块,增加每个服务器组中不同功能的服务器数量。实现游戏内容的更新升级。

(3) 服务器的逻辑系统模块的设计是基于问题分离、接口隔离原则等设计原则与设计模式出发,做到每个逻辑模块都是相对独立的。这使得在对某一个逻辑模块的改动更新时,实现对其他逻辑模块影响最小化,降低系统的耦合度,从而增强系统的可扩展性。另外设计模式的运用在一定程度上也优化了代码的质量,增强了代码的可扩展性。

(4) 系统采用基于标准的技术和产品,如系统框架、网络协议、硬件产品、游戏软件等,这使得系统将来的升级也十分方便。

4 应用结果展示

本系统在设计的同步机制基础上,在校园网环境下进行实验,开发环境:图形引擎 VC+OGRE,网络引擎 RakNet;运行环境:服务器端系统运行软件平台为 Windows2008 系统,硬件配置:CPU Intel E5620 2.4GHz,RAM 4G。图5为本游戏在30台游戏客户端运行测试截图。

5 结束语

实践证明游戏框架具有良好的可扩展性,但系统模块过于复杂,在许多细节方面考虑并不是很周全,下阶段的任务将进一步完善各个模块,增强系统的稳定性。

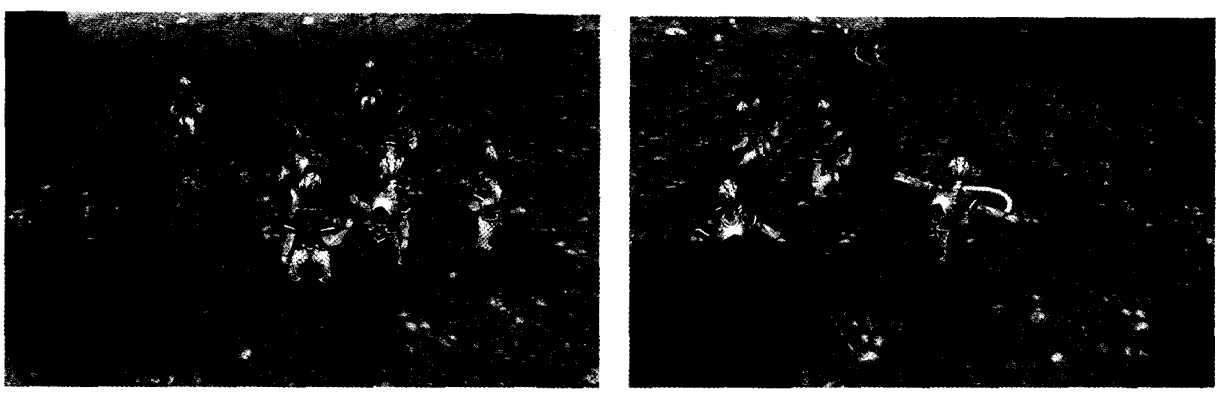

图5　网络游戏运行截图

参考文献:

[1] 刘娟娟,陶加祥.一种可扩展的MMORPG网游分布式服务器架构方案[J].软件导刊,2010,9(4):121-122.

[2] Wu Jiyi, Gong Xiangguo, Zheng Jiwen. A Model for Massively Multiplayer Role-playing Games System Performance[C]// 16th International Conf on Artificial Reality and Telexistence--Workshops, ICAT '06. [s. l.]:[s. n.],2006:252-255.

[3] Liu Hney-Ing, Tang Bingrong. DACA: Dynamic Anti-Cheating Architecture for MMOGs[C]//International Conf on Advanced Information Networking and Applications, AINA '09. [s. l.]:[s. n.],2009.

[4] Ferrara E, Fiumara G, Pagano F. Living City, a Collaborative Browser-based Massively Multiplayer Online Game[C]//Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques. [s. l.]:[s. n.],2010:1-8.

[5] Grzonkowski S, Corcoran P M. A secure and efficient micropayment solution for online gaming[C]//Games Innovations Conference, ICE-GIC 2009. [s. l.]:International IEEE Consumer Electronics Society,2009.

[6] Bezerra C E B, Geyer C F R. A load balancing scheme for massively multiplayer online games[J]. Multimedia Tools and Applications,2009,45(1-3):263-289.

[7] Quax P, Cornelissen B, Dierckx J, et al. ALVIC-NG: state management and immersive communication for massively multiplayer online games and communities[J]. Multimedia Tools and Applications,2009,45(1-3):109-131.

[8] Nae V, Iosup A, Podlipnig S, et al. Efficient Management of Data Center Resources for Massively Multiplayer Online Games[C]//International Conf on High Performance Computing, Networking, Storage and Analysis. [s. l.]:[s. n.], 2008.

[9] Kambayashi Y, Ledgard H F. The separation principle: a programming paradigm[J]. Software,2004,21(2):78-87.

[10] Shaikh A, Sahu S, Rosu M, et al. Implementation of a service platform for online games[C]//Proc of 3rd ACM SIGCOMM Workshop on Network and System Support for Games. [s. l.]: [s. n.],2004:106-110.

[11] Zhao Liping. Patterns, symmetry and symmetry breaking[J]. Communications of the ACM,2008,51(3):41-44.

[12] 金　迪,陈晓玲,林和平.基于设计模式的.NET通用数据访问层研究[J].吉林大学学报(信息科学版),2010,28(2):198-200.