# 三层结构的网络游戏服务器设计及其性能分析

王华峰, 张新家
WANG Hua-feng, ZHANG Xin-jia

西北工业大学 自动化学院, 西安 710072
College of Automation, Northwestern Polytechnical University, Xi'an 710072, China

WANG Hua-feng, ZHANG Xin-jia. Design of three-tiered gaming server and its performance and analysis. Computer Engineering and Applications, 2007, 43(2): 125-127.

Abstract: This paper proposes an extended multiplexing structure for three-tired networking application servers, which has the feature of one thread simultaneously servicing multiple clients, so that it is very responsive to users. Based on this model, a kind of gaming application server is designed, then its performance is in details analyzed. The experiments indicate that it can be efficient, scaleable and extendable.

Key words: application server; multiplexing; three-tiered structure

**摘 要**: 鉴于现有的网络游戏服务器端一个线程对应一个客户端的情况, 提出了一个扩展的多路复用模式的服务器模型, 基于该思想设计了一种游戏应用服务器。在此基础上对该模型的应用服务器进行了性能分析。游戏应用服务器运行测试表明这种服务器模型具有可用性和高效性。

**关键词**: 服务器; 多路复用; 三层结构
**文章编号**: 1002-8331(2007)02-0125-03 **文献标识码**: A **中图分类号**: TP393

http://www.ixueshu.com

## 1 引言

以 Windows 作为服务器端的网络游戏 I/O 模型有: WSAAsyncSelect, WSAEventSelect, Overlapped, IOCP, Multiplexing。其中 WSAAsyncSelect 模型是 Windows 最早的 I/O 模型, 使用一个窗口接收消息。WSAEventSelect 模型用等待多重事件的方式来响应网络事件。Overlapped 模型由系统为应用进程接收数据。完成端口模型(IOCP)唤醒被挂起的线程进行 I/O 处理。多路复用(Multiplexing)是应用最广泛的模型。

在以上几种模型中, 除了多路复用(Multiplexing)模型以外全都采用一个线程服务一个客户端机制 (IOCP 同样也是如此), 这样不可避免要开辟数量很大的线程, 结果不但会占用极大资源, 而且也不易管理。本文在多路复用模型的基础上进行改进, 开辟线程池, 使之既满足一个线程服务多个客户端, 又保证了服务器的效率, 还可以自由地在 Windows 与 UNIX 之间移植。并在该方案的基础上开发了一个棋牌游戏并对服务器性能作一些分析。

## 2 服务器框架结构

网络游戏服务器主要包括连接管理模块、消息处理模块、数据库管理模块等。连接管理模块负责接入外部连接请求; 消息处理模块负责对客户端的消息进行分析, 处理; 数据库管理模块负责进行数据库交互, 存取客户端数据。对于网络游戏服务器还应集成引擎线程, 负责驱动整个游戏的运转。因此整个网络游戏服务器构架如图 1 所示。

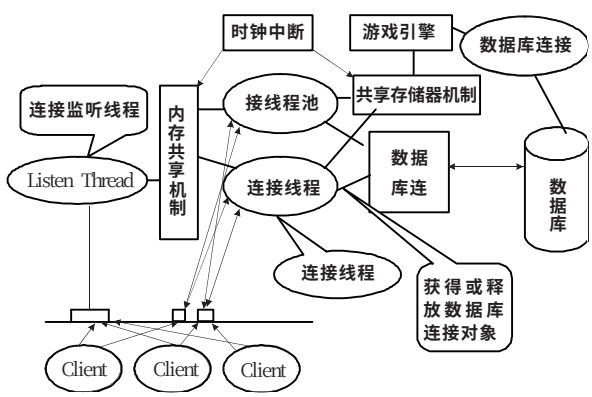


图 1 游戏服务器构架图

在图 1 所示的服务器构架图中, 监听进程, 连接线程池, 以及它们之间的共享存储区构成了连接管理模块; 数据库连接池承载了数据库管理模块; 消息处理模块存在于连接线程池与游戏引擎中, 游戏引擎通过专用的数据库连接来实时更新数据并且与连接线程池通过共享存储交换数据。

为了提高服务性能, 应用服务器的设计采用如下设计技术:

(1)针对客户端的连接请求, 采用具有自动伸缩能力的连接线程池技术, 根据负载情况进行线程数调节。每个线程管理多个用户连接, 最大连接线程数可配置。

(2)事务处理采用数据库连接线程池技术, 也应具有伸缩能力, 每个数据库连接线程可为所有用户服务, 但用户获得一个数据库管理线程必须按一个完整的事务进行调度。

(3)采用线程寻槽方法给线程池中的线程分配套接字。监

作者简介: 王华峰(1981-), 男, 硕士研究生, 主要研究方向: 网络应用服务器; 张新家(1961-), 男, 副教授, 主要研究方向: 控制, 网络通信等。

听进程与连接线程之间采用共享存储器技术进行通信。

## 3 服务器设计与实现

为了更清楚从功能上划分，应用服务器具有如下功能：用户连接管理、消息处理和数据库连接管理。用户连接管理负责给连接线程池分配客户端的套接字；消息处理机制负责解析客户端的消息，根据消息的动作信息来执行不同的消息处理模块；数据库连接管理负责客户端的登录验证，客户端信息的及时存取，维护等待队列等。

下面从以下几个方面介绍应用服务器的设计思路。

### 3.1 用户连接管理

要使连接线程池高效率地服务客户端，监听进程必须采用高效的连接管理机制，使用共享存储技术将用户连接分配给负载最少的连接线程进行处理。共享存储区是一个数组，数组中的每个元素如表 1 所示的结构，数组的大小为线程池的最大线程数。

表 1 共享存储区结构

| |
|---|
| 套接字滞留时间片(ConnTime) |
| 线程报告状态时间片(RunningTime) |
| 套接字号(NewConnID) |
| 套接字状态(wConnStatus) |
| 套接字是否可被清除(CanDestroyed) |
| 线程当前服务套接字数(TreadConnNo) |
| 线程状态(ThreadStatus) |

连接管理必须考虑负载不平衡性，即任务请求处理时间的不同导致连接线程利用率的倾斜。本方案采用了动态负载均衡算法，该算法考虑每一个连接线程的实时负载和响应能力，不断调整任务分配的比例，来避免线程超负荷运行，从而提高整体吞吐率。

服务器启动时分配了共享存储区。图 1 中的每个连接线程在图 2 共享存储区中都对应着一个槽位，并且每个连接线程只能从它对应的槽位上取套接字。监听进程寻槽时采用了循环加模机制。循环加模的基本原理可描述为：

假设连接线程池有一组线程 $T=\{T_0, T_1, \cdots, T_{n-1}\}$，$C(N_i)$ 表示线程 $T_i(i=0,1,\cdots,n-1)$ 当前所服务的套接字数。$\sum_{0}^{n-1} C(N_i)$ 表示当前服务器的连接总数。n 表示当前连接线程池的线程数。则：

$AVG=\frac{\sum_{0}^{n-1} C(N_i)}{n}$ 表示连接线程服务的平均套接字数。

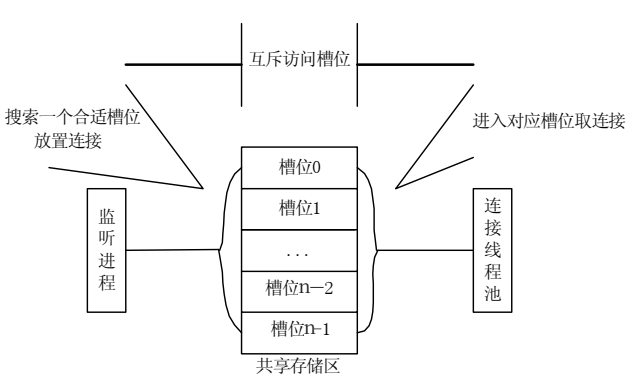


图 2 连接管理示意图

首先判断 AVG 是否大于设定的阀值 $\tau$（$\tau$ 是线程所服务套接字数的上限，超过了这个值就会影响服务质量）。如果大于，就新开辟一个连接线程，给新线程一个空槽位，把新连接放到该槽位。反之，首先采用缩减分配将新连接分配给相应的工作线程：即找一个当前服务套接字数大于 m（$m<\tau$，m 值根据系统配置设定）但小于 $\tau$，并且对应的槽位上连接为空的线程，接着把连接放到对应的槽位，然后把可分配槽位的位置指向下一个相邻槽位，当前连接是槽位 n-1 时，利用取模的方法使下一个槽位指向槽位 0。随着客户端的不断加入和退出，在某一时间段会出现线程的客户端连接为零的情况，如果这样的线程太多会占用很大服务器资源，因此在缩减分配中要及时关掉连接数为零的连接线程。

如果缩减分配不成功，进行正常分配：如果当前指向槽位中没有连接，直接把连接放到该槽位，否则指向下一个位置，直到找到一个满足要求的槽位。随后连接线程池里的连接线程到对应的槽位取走连接。

这里要指出一点，不论是缩减分配还是正常分配，都采用信号量机制控制分配流程：监听进程访问共享存储区时，把相应的槽位锁定后才能进行槽位的数据操作，此时其它的线程不能访问该槽位，保证数据一致性。

如果某个连接线程出了问题，则需销毁该线程。这里采用线程主动报告机制：服务器设置一个定时器，负责对每个槽位的 ConnTime 和 RunningTime 计数，ConnTime 对应着套接字没有被连接线程取走的时间，RunningTime 对应着连接线程没有报告工作状态的时间。定时器每中断一次，如果槽位不存在连接，槽位的 RunningTime 递增，如果槽位存在连接：ConnTime 和 RunningTime 都要递增。一旦连接线程进入槽位取连接，无论有没有取到，都把 RunningTime 置 0，如果连接线程取到了连接，把 ConnTime 也置 0。当这两个值到达设定的上限时，服务器就认为槽位对应的线程有问题，清理该线程，关闭被该线程服务的所有连接，同时对应槽位上的新连接重新分配槽位。

在连接线程池一方，连接线程首先通过信号量机制获取对应槽位的访问权，随即进入槽位读取监听进程分配的连接，如果取到连接，连接线程就把套接字放进自身的一个套接字数组中（该数组存放当前连接线程所服务的客户端套接字），进而修改槽位的一系列数据，然后放弃槽位访问权，执行消息处理模块。

### 3.2 消息处理与数据库连接管理

客户端发送的消息有长有短，客户端的消息格式如图 3 所示。

| | | | |
|---|---|---|---|
| 动作类型/Type | 长度/MsgLength | 消息类型/Commnd | 匹配码/Identifier |
| 分片标识/SegMark | 偏移量/Offset | 数据/Data | |

图 3 消息格式

SegMark 是消息完整性标识，如果是完整消息，SegMark 被置为-1，服务器直接把后面的 Data 数据取出来，传送给消息处理模块，如果是消息片断，服务器要把该消息片断暂时保存起来，直到接收最后一个分片（SegMark 为-1），随后根据每个分片的 Offset 和 Data 进行消息的重组。

服务器取到客户端的完整消息后，随即开始消息处理流程。消息分为消息头和消息体两部分，消息头在图 3 中对应着 Data 域之前的部分，Data 是一个消息的消息体部分，用 XML 的格式存放。根据消息头的 Type 得到动作类型后，接着就调用与该消息对应的消息处理函数，在消息处理函数中对消息体的 XML 消息进行解析，提取消息传达的信息。

http://www.ixueshu.com

消息处理函数是消息处理流程的核心。由图1可知，消息处理分为两种方式：第一，直接读取数据库，如处理客户端登录消息；第二，读取连接线程池与游戏引擎之间的共享存储区，如处理客户端请求游戏桌信息。第二种方式中的共享存储区中存放着当前开设的所有游戏桌，连接线程池访问该存储区进行客户端动作的提交和游戏桌状态的读取，游戏引擎访问该存储区进行游戏状态的判断和调整。连接线程池和游戏引擎对于每张游戏桌都是互斥访问，以此来保证游戏数据的完整性。

由上面的分析可知数据库事务处理的效率关系到游戏的响应时间，利用ADO.NET连接池可以满足要求：ADO.NET连接池允许服务器线程从该连接池中获得一个已建立的连接使用，而不需要重新建立一个数据库连接。服务器连接线程使用完连接后，该连接被归还给ADO.NET连接池。使用连接池的最主要的优点是不需要为请求重新连接与认证，提高了访问效率。

## 4 性能分析

由上文可知，服务器快速稳定的处理能力以及应对高负荷的能力是服务器的基本要求，对其分析和优化与套接字连接管理，消息在系统中的停留时间，连接线程数和线程服务套接字数等指标密切相关，这里只分析服务器在单一处理器前提下的运行情况。

(1) 套接字连接管理

服务器端监听进程在任一时刻最多只能处理一个连接请求，监听进程在[0，t]时间段内处理的连接数是服从泊松分布的，连接线程池的平均处理速度u服从指数分布。

设监听进程的平均寻槽速度为$\lambda$，当服务强度$\rho=\frac{\lambda}{\mu}<1$时，系统能处于相应稳定的状态，否则要增加服务线程。但是单靠增加服务线程不能从根本上解决客户端连接数与系统资源之间的矛盾。连接线程数目很大的情况下，不但每个线程的运行时间要增长，而且连接线程池中的线程切换也在占用系统资源，有效的方法是增加服务器，组成服务器集群来分担负载。

(2) 消息在系统中的停留时间

消息的停留时间是消息在线程池中的等待时间与实际接受服务时间之和。

设$\bar{t}$为消息的平均服务时间，t为消息已获得的服务时间，则系统的服务强度为

$$\rho=\lambda\bar{t}$$

当系统满足服务强度$\rho<1$时，每个消息的停留时间T(t)可作如下分解：

$$T(t)=W(t)+S(t)$$

其中W(t)表示消息的等待时间，S(t)表示消息的接受服务时间。它们均为t的函数。

服务器在单一处理器条件下，由排队论可知，该线程池服务过程属于M/M/1模型，服务器CPU对线程池采取循环调度，线程池中每个线程的优先级相同。

消息的平均等待时间W可以由M/M/1系统的性质可知，只要满足$\rho<1$，并且$\bar{t}<\infty$，则：

$$W=\frac{\rho\bar{t}}{1-\rho}$$

单独对于某个消息来说，其等待时间为：

$$W(t)=\frac{\rho t}{1-\rho}$$

由此可以看出消息的服务时间与等待时间之间存在一个比值，即如果消息需要使用的CPU时间越长，则其等待时间也要线性增长。

所以消息在系统中的平均停留时间为：

$$T(\bar{t})=\frac{\rho\bar{t}}{1-\rho}+\bar{t}$$

由此可知，本方案对于服务时间要求较短的消息的处理能达到很好的效果，如果某些消息的服务时间相对较长，则要有相应较长的等待时间。这就要求：在游戏设计中，如果有些功能可以由客户端来处理就尽量在客户端，同时要优化游戏引擎，使消息的处理时间尽量缩短。

(3) 连接线程数与线程服务套接字数

当服务器在长时间的高负荷状态下运行时，要求系统在某时刻达到平衡状态的条件是$\bar{t}\sum_{1}^{n_1}n_2<T_r$，$n_1$表示连接线程数，$n_2$表示每个线程服务的套接字个数，$\bar{t}$是服务每个套接字对应消息的平均时间，$T_r$表示可以忍耐的消息服务时间。可以把$\bar{t}$和$T_r$看成定值，$\sum_{1}^{n_1}n_2$则应该小于一个常量，该常量是服务器所能服务的最大客户端连接数。每个线程服务的客户端连接数$n_2$是动态变化的，连接线程数$n_1$也是随着网络游戏的进行而动态变化的，即在服务器繁忙的时$n_1$的值相应就要增加，而在服务器任务少的情况下$n_1$的值也相应地减少。最理想的情况是：连接线程池中每个线程所服务的客户端套接字数都相等，否则会造成连接线程池负载不均衡。但现实网络游戏中考虑到服务器的实际处理能力以及运营成本等指标，每个服务器的最大客户端连接数是综合各方面要求而得出的。

## 5 总结

本文讨论了服务器在多路复用模式下多线程处理多客户端的情况，给出了框架思想和实现算法，并对服务器性能作了分析。结果表明该模式是可行的，可以节省系统资源并有很好的反应性，适应了网络环境下构建游戏平台的要求。

(收稿日期：2006年3月)

## 参考文献：

[1] Stevens W R.TCP/IP Illustracted Volume1：the protocals[M].范建华，译.北京：机械工业出版社，2000.

[2] Comer D E，Stevens D L.Internetworking with TCP/IP VolIII：client-server programming and application[M].赵刚，林瑶，蒋慧，译.北京：电子工业出版社，2001.

[3] 段雪峰，张新家，戴冠中.中间件服务性能建模与分析[J].计算机应用，2004(1)：106-107.

[4] 盛友招.排队论及其在计算机通信中的应用[M].北京：北京邮电大学出版社，1998.

[5] 王自果，田铮.随机过程[M].西安：西北工业大学出版社，1990.

[6] 孙容桓，李建平.排队论基础[M].北京：科学出版社，2002.

http://www.ixueshu.com

论文降重，论文修改，论文代写加微信:18086619247或QQ:516639237

论文免费查重，论文格式一键规范，参考文献规范扫二维码：

相关推荐：

一种新型直流接触器设计特点的分析

LJ276M型摩托车发动机性能与结构分析

三层结构的网络游戏服务器设计及其性能分析

城轨车辆前窗玻璃的设计与分析

基于多连接自适应技术的三层服务器设计及性能分析

武汉轨道交通1号线车辆前窗玻璃破裂问题分析

Servlet Container的分析与设计

浅谈不同型号卷扬机在施工现场的应用

MAN B&W公司TCR新系列涡轮增压器的开发和初步试验结果

建筑用聚氯乙烯绝缘尼龙护套电线初探