# 摘 要

Plain framework（以下简称PF）轻量级服务器模型构架文档主要从服务器整个服务器的构架和服务器之间的关系来分析其优缺点，并用简明易懂的语言来描述服务器用到的技术以及实现的相关原理。

这里所阐述的指网络游戏的服务器模型，存在一定的针对性，所以需要各位同仁不要一位生搬硬套。游戏服务器（game server）在网络游戏中起着举足轻重的地位，有的人将之称之为游戏的大脑，更有人戏谑的称之为“皇上”，不管怎样乖僻的称呼，总之是说明其在整个游戏中至关重要。

轻量级服务器主要针对小中型游戏，结合市面上常见的游戏进行设计，保证能够顺畅的应用于市面上的大部分游戏。

我们在这篇文章中将会仔细的剖析各个游戏的功能，以及其存在的关键技术应用，从实现的复杂度，与其的优点进行一定说明和分析。

# Abstract

Plain framework (hereinafter referred to as PF) lightweight server architecture document model primarily from the relationship between the server and the server for the entire server architecture to analyze their strengths and weaknesses, and use straightforward language to describe the server to use the technology and the realization of the related principle.

Set forth herein refers to the network game server model, there are some specific, so you need not a rote colleagues. Game server (game server) in the network plays a pivotal role in the game, some people will call the game's brain, but it was jokingly called "emperor", however perverse call, in short, explain its is critical throughout the game.

Lightweight server mainly for small to medium sized game that combines the common market, the game is designed to ensure that the market can be smoothly applied to most of the game.

We will carefully analyze the function of each game, and the key technology of its existence in this article, certain descriptions and analyzes the complexity of implementation, and its advantages from

# 目录

[摘 要 1](#_Toc408053508)

[Abstract 2](#_Toc408053509)

[目录 3](#_Toc408053510)

[前 言 4](#_Toc408053511)

[第一篇 构架 5](#_Toc408053512)

[1.1 构架思想 5](#_Toc408053513)

[1.2 构架模型 5](#_Toc408053514)

[1.3 性能分析 5](#_Toc408053515)

[第二章 关系与作用 6](#_Toc408053516)

[2.1 关系 6](#_Toc408053517)

[2.2 作用 6](#_Toc408053518)

[第三章 技术分析 7](#_Toc408053519)

[3.1 关键技术 7](#_Toc408053520)

[3.2 技术剖析 8](#_Toc408053521)

[总结 8](#_Toc408053522)

[致 谢 9](#_Toc408053523)

[参考文献 10](#_Toc408053524)

# 前 言

在前面已经大致说明了本文的内容，但其只限于PF的商业版开发。每个人的思想没有百分百的完美，所以文章中可能还有些不太完善的地方，希望每一位PF的核心人员能够踊跃的指正错误。

我们需要大概的想象，哪怕那件事自己认为不能实现，但是你可以分享出来。不能实现的东西，在人类的进化史中逐渐都实现了，说不定你一说出来的时候有人就能实现了。不过在紧张的开发中，我们可以将这些奇特的想法暂时记录下来，在空余的时间再来讨论。

本文档适用于PF商业版开发过程中核心的参考资料，请勿任意使用于其他商业用途或传播该资料，否则PF成员有权利追求其所带来的法律责任。

# 第一篇 构架

## 1.1 构架思想

轻量级设计的总方向是尽可能将功能系统化，并且将这些零碎的功能进行统一的管理。

## 1.2 构架模型

我们的游戏服务器分为：消费服务器（billing）、登陆服务器（login）、游戏服务器（server）、共享内存服务器（sharememory）。

## 1.3 性能分析

这样的设计本来是针对小中型的项目，因为游戏逻辑本身不是很复杂。所以服务器方面只分为了4个，但是每个服务器相应负责的功能增多了，主要体现在游戏服务器和共享内存服务器上。

所有的游戏逻辑，包括场景的管理都放到了游戏服务器，但是在多CPU的今天，我们可以使用多线程来缓解单进程的压力，只要硬件能跟上，这样的设计也同时满足大部分大型游戏的需求。

共享内存服务器，在这里同时扮演了共享内存的分配维护以及数据库的通讯的角色，也就是说所有需保存的数据全部在共享内存服务器上。这样分离了游戏逻辑与数据读写和保存，使得我们的数据有多一层的保护，好比游戏服务器突然宕机，我们的数据却丝毫不受影响。

针对大部分的运维环境中，这样设计的缺点就是压力的集中化，因为一个服务器负责的功能太多。如果这个服务器不小心过度劳累，那么它所负责的功能全部瘫痪。

# 第二章 关系与作用

## 2.1 关系

在一个完整的游戏服务器中，其实这些服务器的功能几乎都是需要用的，只不过设计不同而已。那么这些服务器之间是否有关联？就像我们在日常生活里的人际关系一样？

针对上述的两个问题，答案是肯定的，它们之间肯定是存在一些紧密的关联。正如日常生活中，人们之间的关系一样，每个人都有自己的生存技能，我们之间很多时候都是彼此互利的。

它们的关系大致是这样的：登陆服务器接收到客户端登陆请求消息，将验证账号的消息发送到消费服务器，消费服务器将结果返回给登陆服务器，登陆服务器将登陆结果返回给客户端。

进入游戏流程：客户端向登陆服务器发送进入游戏场景的消息，登陆服务器收到消息后将此消息推送到游戏服务器，游戏服务器返回进入游戏世界的结果到登陆服务器，登陆服务器将结果发送到客户端（如果是成功进入，则会发送目标服务器的IP和端口）。成功的前提下，客户端断开与登陆服务器的连接，并利用得到的IP和端口连接游戏服务器。

充值流程：消费服务器收到平台或WEB服务的消息，将充值结果返回到对应的服务器，如成功则派送消息到登陆服务器，登陆服务器找到对应的游戏服务器并将充值的数据同步（同重量级一样，这种方式打算放弃）。新的模式，采用玩家自主查询点数，以及自主兑换游戏币的方式。

## 2.2 作用

消费服务器：一切和金钱有关的数据处理。

登陆服务器：起到网关的作用，处理玩家与游戏服务器之间的连接，处理消费服务器与游戏服务器之间的连接。同时登陆服务器还有简单的数据读写的功能，如创建角色、删除角色、角色查询等功能。

共享内存：提供共享内存服务，并处理存档的功能（定时、下线）。

游戏服务器：游戏所有逻辑的处理，一切复杂的游戏逻辑都放在这里处理，主要有模块系统的处理、场景的管理等。

# 第三章 技术分析

## 3.1 关键技术

在这里我们所用到的关键技术主要有：网络通讯的处理、数据存储技术、事件触发机制、脚本技术。

网络通讯：网络通讯（net）在网络游戏中是必不可少的部分，在这里分为两个部分处理，就如构架图里的关系一样，服务器与客户端、服务器与服务器都有一根线紧密连接在一起。

这根线的建立以及维护，我们在这里就使用到了通讯的技术，不过有的不是net通讯，如共享内存和服务器之间，使用的通讯技术是内存共享。而网络通讯中，我们主要采取tcp通讯技术，使用系统给出的套接字（socket）接口，前期战士不使用udp技术，后期可能会加入日志服务器概念。

数据存储技术：除了内存的存储（memory）以外，我们需要将一些重要的数据存档到数据库，所以我们不得不用到数据库相关的技术。PF框架中，提供了一个名为ODBC的接口，可以方便的连接到各种数据库。我们针对数据库的操作，也利用了ODBC作为桥接的方式进行。

事件触发：完整的架构中有着许多的事件（event），就好比接收到网络包，我们对其进行逻辑处理一样，它也可以看做一个事件，只是这个事件比较单一。而在游戏世界中，分为全局的事件、模块的事件、玩家的事件等等。

有的时候我们有必要对复杂的事件进行统一的管理，这就需要用到PF框架里的事件模块。

脚本技术：利用脚本（script）处理一些简单的逻辑，让开发和维护成本大大降低。我们在这里优先选用PF框架提供的lua脚本接口技术。

## 3.2 技术剖析

消费服务器：数据库存储（DB）、网络通讯（NET）

登陆服务器：数据库存储（DB）、网络通讯（NET）

共享内存服务器：数据库存储（DB）、共享内存（MEMORY）

游戏服务器：共享内存（MEMORY）、网络通讯（NET）、脚本处理（SCRIPT）、事件处理（EVENT）

# 总结

暂时没想到

# 致 谢

暂时没想到。

# 参考文献