|  |  |
| --- | --- |
| 文档号 |  |
| 密级 |  |

**电脑彩票系统（信息显示）**

**技术可行性分析报告**

|  |  |  |
| --- | --- | --- |
| **动作** | **签名** | **日期** |
| 创建人 |  |  |
| 审核人 |  |  |

文档变更历史

|  |  |  |  |
| --- | --- | --- | --- |
| **修订人** | **版本号** | **日期** | **修改内容与原因** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**目 录**

[1 系统概要叙述 1](#_Toc11349)

[1.1 系统方案 1](#_Toc7967)

[1.2 主要技术 1](#_Toc19052)

[2 关键技术分析 1](#_Toc21305)

[2.1 关键技术1 1](#_Toc6057)

[2.1.1 技术说明 1](#_Toc28930)

[2.1.2 技术难点 1](#_Toc26470)

[2.1.3 性能指标分析 1](#_Toc22940)

[2.1.4 解决方案 1](#_Toc11462)

[2.1.5 风险分析 1](#_Toc30743)

[3 结论 1](#_Toc16678)

# 系统概要叙述

## 系统方案

服务端主要由网络通信模块和后台数据库模块组成，后台数据库查询进程与网络通信模块之间，采用消息队列的方式进行进程间通信。其中，后台数据库查询进程执行轮询数据库的操作，在整个服务端体系中属于消息生产者角色，网络通信进程在整个服务端体系中属于消息消费者角色。网络通信进程不断接收后台数据库进程的指令消息，并将指令消息转发给所有连接开奖客户端。

客户端主要由资源管理和加载、后台指令通信模块、指令动画模块三个主要模块组成。 资源管理和加载模块，主要是合理组织客户端渲染显示需要的各种图片、脚本、模型动画等资源，并采用预加载和后台加载的方法，提高客户端的用户体验效果。

后台指令通信模块，后台指令通信模块主要采用事件机制触发机制，接收到指令信息后就执行相关的指令动画。

指令动画模块，指令动画模块采用可扩展的方式，可以基于指令动画模块的扩展方法，增加或者修改指令动画内容。接收到一个指令动画就执行一个指令动画内容。在没有更新指令前，窗口动画停留在上个指令动画结束画面。

## 主要技术

电脑彩票显示系统主要由两个部分组成，服务端主要由使用以下技术：

1. PHPSocket通信技术
2. 多进程守护技术
3. 高并发连接技术
4. 数据库技术

客户端主要需要使用以下技术：

1. 三维渲染显示技术
2. 资源后台加载技术
3. 动态资源网络下载技术
4. 流媒体渲染技术
5. WebSocket网络通信技术
6. 版本动态更新技术

# 关键技术分析

## 多进程守护技术

### 技术说明

服务端的服务需要保证稳定和可靠，需要保证后台服务进程不能被中断，或者进程崩溃后，然后自动重启新的进程来提供服务。

### 技术难点

建立后台守护进程，进程崩溃后保存进程状态信息，以及进程崩溃后根据进程状态恢复进程服务是主要难点。

### 可靠性分析

经过测试，目前提供的方案可以满足后台服务进程持续提供的要求。

### 解决方案

目前后台服务进程充分利用Linux服务端守护进程的方式实现服务高可靠性。

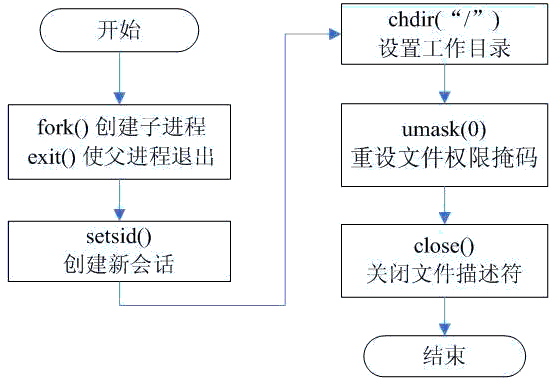


图 1 后台守护进程

## 高并发技术

### 技术说明

采用高并发技术，可以尽可能利用单机的硬件能力，提高更多客户端TCP/IP连接数量。从而减少硬件投入和系统开发设计难度。

### 技术难点

为了便于服务端开发，采用PHP实现Socket相关开发，PHP在线程方面的支持能力比较差，因此需要采用多进程方案，来支持更多的客户端连接，并提高连接客户端的相应能力。

### 性能指标分析

客户端接收指令消息的延时都在500ms内，能够满足客户端显示系统的需求。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 客户端连接数 | 1000 | 2000 | 3000 | 4000 | 5000 |
| 相应时间 | <100ms | <100ms | <150ms | <200ms | <250ms |
| 服务端资源情况 | CPU 10%  内存 <200M | CPU 15%  内存 <400M | CPU 25%  内存 <600M | CPU 40%  内存 <900M | <CPU 50%  内存 <1000M |

### 可靠性分析

系统采用了心跳检测技术，来及时剔除下线客户端，从而保证实效客户端的资源得到释放。在单机连接不超过5000个并发连接时，经过测试后台服务可以提供稳定服务的能力。

### 解决方案

采用libevent的高性能网络库核心，支持单机高并发连接，目前支持单机5000左右的并发连接。

## 渲染数据预加载技术

### 技术说明

将渲染数据进行预加载可以避免程序运行过程中的卡顿和黑屏问题。Unity默认在切换场景的过程中，如果需要加载大场景数据就会需要等待较长时间，给用户不好体验。根据目前的项目需求，本系统是重复销售、止售、开奖、报表的循环过程，因此将共用的并且循环使用的资源进行预加载，可以有效避免运行过程中卡顿、黑屏等问题。

### 技术难点

资源预加载技术主要需要解决资源的分类标准，按照不同的资源对场景进行分类。

### 可靠性分析

按照提供的方案可以实现场景资源预加载，解决场景实现过程中的卡顿和黑屏问题，可以满足显示系统项目的需求。

### 解决方案

目前后台服务进程充分利用Linux服务端守护进程的方式实现服务高可靠性。

## 渲染数据后台加载技术

### 技术说明

对于非预加载的渲染数据，为了避免影响主线程的运行，需要实现渲染数据的后台加载技术。将渲染数据通过后台资源加载线程，将资源从磁盘加载到内存，然后再加载到显存中。这种技术通常也称为资源异步加载技术。

### 技术难点

默认渲染线程是在主线程，将渲染数据在后台加载，线程间同步以及将内存数据交换到显存。

### 性能指标分析

采用后台资源加载，能够避免资源IO过程中的导致渲染主线程的卡顿。

|  |  |  |
| --- | --- | --- |
| 资源加载 | 前台加载20M模型数据 | 后台加载20M模型数据 |
| 加载前渲染帧率 | 60FPS | 60FPS |
| 加载过程渲染帧率 | 1~10FPS | 55~60FPS |

### 解决方案

后台资源的采用单独的线程处理资源加载。将每个资源加载封装成一个请求，请求包括准备、执行、结束三个动作。根据请求操作与主线程进行资源同步。

## 动态资源加载技术

### 技术说明

动态更新数据比如从网络下载的数据，需要加载到场景中进行显示，那么就需要使用使用动态资源加载技术。比如销售阶段的宣传广告资料，这类资料经常需要更新，从服务器下载完宣传广告资料，就需要使用动态资源加载技术，同时结合渲染数据后台加载技术，可以实现平滑的数据加载效果。

### 技术难点

重点解决资源网络下载并按照Unity资源组织方式存储，然后使用渲染数据后台加载方式将动态资源进行加载。

### 性能指标分析

由于资源后台下载与带宽积服务端的状态也有关系。一般情况下采用数据后台下载完成后，在执行资源更新操作，从而保证渲染窗口的画面连续性。目前测试一般1M以内数据都在1s内下载完成，在下一个开奖周期内，就能正常显示。

### 解决方案

Unity自身提供了WWW类用来发送HTTP请求获取Web数据，但是效率不高。这里采用UniRx库下载Web数据，UniRx库下载数据采用的异步下载方案。

# 结论

经过主要关键技术的验证，根据目前技术方案和系统设计，可以按照预期达到显示系统项目的需求。