**北京理工大学现代远程教育学院**

**毕业实践报告**

**题目：游戏运维计算机应用体系设计实现**

**专业：计算机科学技术**

**班级：20140111400**

**学生学号：20140111400011**

**学生姓名：郭子朋**

**指导老师：姜增如**

**日期:2016年04月**

**学生毕业实践（报告）评语表**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 姓 名 | 郭子朋 | 学 号 | 20140111400011 | 班 级 | 20140111400 | |
| 第一阶段  （40分） | | | 第二阶段  （60分） | | 合 计  （100分） | |
|  | | |  | | |  |
| 毕业实践（报告）评语  郭子朋同学以游戏运维计算机应用体系设计实现为毕业实践，分析了游戏运维业务范围和各种游戏业务需要的环境。找到了运营维护的管理内容，包括：资产管理、配置管理和批量管理。学习了维护中工作任务，如：下载服务器、搭建集群，做资源分发、网络故障、拥塞、质量监控等。该生毕业实践符合计算机科学技术大专要求，同意毕业。  指导老师（签字）： 年 月 日 | | | | | | |
| 毕业实践（报告）成绩  指导老师组长（签字）： 年 月 日 | | | | | | |

**北京理工大学继续教育暨现代远程教育学院**

**专科学生毕业实践（报告一）**

学生姓名:郭子朋 指导教师：姜增如 学号：20140111400011号 专 业：计算机科学技术

**一、课题目的和意义**

当今互联网时代，移动设备及网络设备（如手机、电脑等）的发展可谓是日新月异。在人们休憩之余娱乐的项目，游戏也就成了非常重要的一个组成部分。如何能保证用户完美的游戏体验，正需要后台强有力的支持，才能够稳定运行。整套运维体系的良好支撑，也就变得尤为关键。所以在这里大胆设想与规划展望，畅谈游戏运维体系在计算机应用上的实现与发展。

随着互联网建设的不断深入和完善，计算机硬软件系统的运行维护已经成为了各行各业的普遍关注和质量要求的很大问题。由于这是一个随着计算机信息技术的深入应用而产生的新课题，因此如何进行有效的运维管理体系，这方面的知识积累和应用技术还刚刚起步。对这一领域的研究和探索，将具有广阔的发展前景和巨大的现实意义。

脱离每天被繁冗的工作和故障所包围的现状，利用“自动化运维管理体系”高效而清晰地管理业务系统的运行状况，立即把所有的运维问题及时解决，并能够提供追踪改善状况的评估方案，以便达成可持续的改进与发展。

中国网游潜力依然巨大、各游戏公司对游戏运维体系需求量依然会很大。运维工程师技术含量及要求会越来越高，同时也是对公司应用、架构最了解最熟悉的人、越来越得到重视。游戏运维是一个融合网络、系统、开发、安全、架构、存储等的综合性技术岗位，给大家提供一个很好的个人能力与技术广度的发展空间。

**二、本课题主要研究内容和预期目标**

**2.1研究内容：**

游戏运维在计算机上的服务器（包括网络/系统/软件等 ）应用实现；

详谈游戏框架的各方面（服务器/数据库等）运维实现；

游戏运维体系的模式、发展与规划。

**2.2体系结构：**

·监控服务体系

·Zabbix

·Nagios

·服务器虚拟化

·Autopatch

·Vmware

·VPS

·安全审计

·应用安全

·系统安全

·网络安全

·事件管理

·问题管理&知识库

·变更管理

·配置管理

图2-2 应用体系结构

**2.3预期目标**：

在计算机环境下,以虚拟化为平台,使用服务器、数据库、监控软件、管理系统及安全机制，搭建的网络游戏运维流程体系更加方便稳定。整部架构基本上实现了预期的各项功能，达到了任务书中的主要设计内容的各项任务，整个体系也在经过测试和不断地改正之后能够顺利运行。基本上实现了开题报告中的需求设计，技术完成，该体系优化的实现:

1. 方便化

可将目前主流服务器虚拟成独立的多台不等的服务器，极大地提高了服务器的部署效率，大量节约了硬件数量，节约了软性运维人力上的不可即性，高效率的解决了服务器突发的故障及应用问题。

1. 实用性

应用运维监控体系有针对性地对所有可能影响游戏运行的因素进行全方位的监控和数据采集、分析，目前能展示10万多条游戏在线人数和性能曲线，监控覆盖率达100%，有效率高达80%以上。

1. 可发展度

随着系统、技术的日益复杂，维护水平要求的逐步提高，游戏运维体系借签业界最佳实践应用管理理念，从人员、流程、技术等各方面着手，逐步提升维护能力，以有限的资源支持不断扩展的游戏需求。

**2.4进度安排：**

2015年12月20日-2015年1月15日 完成开题报告；

2016年1月16日-2016年1月30日 完成系统需求分析；

2016年2月1日-2016年2月14日 完成概要设计、详细设计；

2016年2月15日-2016年2月19日 完成部署环境等搭建架构前准备工作；

2016年2月20日-2016年3月31日 完成流程部署；

2016年4月1日-2016年4月8日 完成平台测试；

2016年4月9日-2016年4月15日 完成论文设计；

**三、游戏运维业务的发展和规划**

自动化运维：建设各运维事项的自动化平台，以及在这个平台上的自动化运维框架

自动化监控：就是对游戏业务的整条运维线进行全方位的、全实时的监控，及时发现和处理故障，提高可用性

安全管理：分析游戏运维的各个安全隐患点，并且都有相应的对策

运维支撑平台：建立和游戏运维相关的周边平台的N个统一　统一下载、统一官网、统一论坛，统一图片存储，统一内容发布

流程管理：事件如何处理，问题怎么解决，如何对外提供服务等

**北京理工大学继续教育暨现代远程教育学院**

**专科学生毕业实践（报告二）**

学生姓名:郭子朋 指导教师：姜增如 学号：20140111400011号 专 业：计算机科学技术

**摘 要**

随着互联网建设的不断深入和完善，计算机硬软件系统的运行维护已经成为了各行各业的普遍关注和质量要求的很大问题。由于这是一个随着计算机信息技术的深入应用而产生的新课题，因此如何进行有效的运维管理体系，将具有广阔的发展前景和巨大的现实意义。

脱离每天被繁冗的工作和故障所包围的现状，利用“自动化运维管理体系”高效而清晰地管理业务系统的运行状况，立即把所有的运维问题及时解决，并能够提供追踪改善状况的评估方案，以便达成可持续的改进与发展。随着自动化体系日益丰富，运维体系开始建立报警和监控系统，快速准确地发现和定位故障，提高维护稳定性。随着服务器数量的快速增长，运维体系着手建立游戏远程操作平台，使运维主见摆脱依赖人海战术的模式，提高生产效率。

本文从运维服务管理理论、方法和标准出发，结合操作实际和建设需要，遵循立足需求、保障重点、分步实施、务求实效的原则，设计并实践一套更加完善的网络游戏运维体系。

**关键词：**自动化，问题管理，运维框架, 虚拟化,Linux,数据库,Zabbix

**目 录**

[**摘 要** 4](#_Toc448745461)

[**目 录** 5](#_Toc448745462)

[**第一章 体系概述** 7](#_Toc448745463)

[1.1 概述 7](#_Toc448745464)

[1.2运维体系的背景 7](#_Toc448745465)

[1.3运维体系的发展现状 7](#_Toc448745466)

[1.4课题研究的意义 8](#_Toc448745467)

[1.5体系的实现目标 8](#_Toc448745468)

[**第二章 需求分析** 10](#_Toc448745469)

[2.1架构搭建背景分析 10](#_Toc448745470)

[2.2功能需求分析 10](#_Toc448745471)

[2.3性能需求分析 11](#_Toc448745472)

[**第三章 运维体系搭建平台** 12](#_Toc448745473)

[3.1服务器简介 12](#_Toc448745474)

[3.2数据库简介 13](#_Toc448745475)

[3.3 Zabbix监控 15](#_Toc448745476)

[3.4安全机制 20](#_Toc448745477)

[**第四章 运维体系搭建平台** 23](#_Toc448745478)

[4.1总体设计原则 23](#_Toc448745479)

[4.2架构概述 23](#_Toc448745480)

[4.3数据库设计E-R图 24](#_Toc448745481)

[4.4体系优化改进 25](#_Toc448745482)

[**第五章 运维体系详细实现** 26](#_Toc448745483)

[5.1体系搭建环境 26](#_Toc448745484)

[5.2 所需功能的实现 26](#_Toc448745485)

[5.3服务器管理的实现 27](#_Toc448745486)

[5.4 数据库管理的实现 28](#_Toc448745487)

[5.5 监控机制的实现 31](#_Toc448745488)

[5.6 安全机制的实现 32](#_Toc448745489)

[5.7 流程管理应用的实现 35](#_Toc448745490)

[**第六章 总结** 36](#_Toc448745491)

[**第七章 自我鉴定** 37](#_Toc448745492)

[**致谢** 38](#_Toc448745493)

[**参考文献** 39](#_Toc448745494)

1. **体系概述**

**1.1 概述**

一直以来，中国网游在世界上都处于举足轻重的地位。从最早的端游到页游再到手游，不仅市场用户爆发式增长，近15年来网游技术的发展也是无比迅速。随着互联网建设的不断深入和完善，计算机硬软件系统的运行维护已经成为了各行各业的普遍关注和质量要求的很大问题。由于这是一个随着计算机信息技术的深入应用而产生的新课题，因此如何进行有效的运维管理体系，将具有广阔的发展前景和巨大的现实意义。和单机游戏不同，网游除了游戏制作本身以外还牵涉到[服务器](http://product.it168.com/list/b/0402_1.shtml)端，再好的游戏如果出现链接、延迟等问题时也会造成巨大损失，这时候游戏运维便发挥了举足轻重的作用。

## 1.2运维体系的背景

在运维人力有限、游戏数量巨大、变化频繁、复杂情况层出不穷、运营紧迫度只增不减的背景下，已知的传统运维平台的建设思路是无力应对的，而游戏的运营质量效率必须得到提高。同时运维人员的热情也会被日渐琐碎、枯燥、疲惫、低价值、高事故率的工作磨灭掉。游戏的支撑体系建设团队，一方面需要帮助公司更出色的完成业绩目标、提升运营的质量和效率，另一方面需要帮助运维人员实现转型和增值、同时为游戏运营提供更大的助力，所以目前现状需要急迫改变, 在游戏运维领域崭露头角，并得到了技术同行们的很多关注和学习的热情。

## 1.3运维体系的发展现状

目前的游戏运维，基本上就机器上线，安装开发环境，做基础的安全优化，部署批量管理软件，添加cacti和nagios监控，再有安装下LNMP环境（不用负载均衡），数据备份，数据分析，内部机器搭建点基础服务；平时发个运维报表，处理点临时需求，比如开发或者测试要些LOG， 写几条shell就给过滤出来了；还有导点数据，写脚本丢给QC部门。过去一年里，通过标准化工具，在发布、开区、扩容，缩容、自定义等场景中共执行2W+次任务，剔除了等待、中途未完成的任务，相当于700天+工作量，这使得我们的运维有更多时间边喝咖啡边创造更多有意义的工作。基础的运维工作是被动的。服务化的运维是主动的运维，我们自己发现问题，并且通过我们的能力解决问题。

在运维体系方面，游戏市场处于探索期，其实运维也处于探索期。端游时代每个新服都要经历上架、装系统，装服务的过程，一般一到两周可以上线一个区服，对于端游高粘性低流动的特性来说可能还好，但是当页游出现时，转变给运维带来的冲击无法估量。页游时代1天开100多个新服的概念，是传统端游运维所不能理解的。当时的运维认为页游就是把所有服务器实现自动部署服务，同时搭配运维自动部服工具就可以了。但事实上如果在开服时一组一组的使用物理服务器，开服速度根本跟不上，资源浪费还非常巨大，两周后用户留存率仅剩5%-7%。成本巨大亏空，急需技术转型，这个时间点上出现了两种概念影响了以后的手游以及云的发展。

## 1.4课题研究的意义

21世纪什么最重要?信息最重要!运维所需信息要涉及：机房、物理服务器、虚拟机、[交换机](http://product.pcpop.com/Exchange/10734_1.html)、网络、承载业务、业务配置、承载服务进程、端口等信息。不管是自己采购还是购买云服务，物理服务器和虚拟服务器都做为资产存在，在采购后录入相关的资产管理，给它打上[标签](http://product.pcpop.com/bq/10734_1.html)，属于哪个游戏，哪个平台，这样不同游戏平台间就不能混用服务器了。然后，是再给不同的服务器标识它承担的业务角色，比如它是MongoDB，我们需要打上的标签会是大掌门-[APP](http://product.it168.com/list/b/1432_1.shtml)STORE-MongoDB-主库-90000端口-第一组服务。这样一个基础信息录入就完成了。

　这样的信息只要是用来将来批量化部署、管理服务器使用，以及当出现故障时，运维可以很方便的查询相当的服务器以及服务信息。发生高并发、服务器压力激增的情况时，平时运行正常的服务器异常概率大幅增加，日常可能的性能瓶颈点会被成倍放大，这就需要实时定位和解决性能瓶颈点，和提前进行预防改善。

一般来说，传统日志收集方式耗时耗力，效果非常不好，大掌门用了听云Server后，可以进行1分钟级定位能迅速有效发现瓶颈点。同时还结合了听云Network的压测功能，能够在服务器上线前提前发现到高压力下的瓶颈点，提前预防，避免由于高并发出现的服务器瓶颈,达到更高效率的应用运维。

## 1.5体系的实现目标

脱离每天被繁冗的工作和故障所包围的现状，利用“自动化运维管理体系”高效而清晰地管理业务系统的运行状况，立即把所有的运维问题及时解决，并能够提供追踪改善状况的评估方案，以便达成可持续的改进与发展。

掌握如何高效安全的管理游戏在计算机的应用运维体系，使得应用更加稳定向上，方便快捷。为了更好的支持快速发展的业务，运维工作必须朝着规范化、标准化、可管理的方向发展，近年来游戏行业也逐步引入管理系统、安全标准等理念，打造更可靠模式的运维体系。

随着自动化体系日益丰富，运维体系开始建立报警和监控系统，快速准确地发现和定位故障，提高维护稳定性。随着服务器数量的快速增长，运维体系着手建立游戏远程操作平台，使运维主见摆脱依赖人海战术的模式，提高生产效率。

1. **需求分析**

## 2.1架构搭建背景分析

　游戏的运维工种：IDC运维、系统型运维、网络运维、业务型运维、运维值班等。各个工种分工各有侧重。

　　IDC运维：装机、换配件、扛着2U的服务器全国各个机房来回跑。

　　系统运维：安装各种软件，调试各种不兼容的软件，在各种版本的Linux、Windows上。

　　网络运维：二层交换、三层交换、四层交换，还要区分华为、思科。

　　业务运维：24点维护，零晨2点维护，零晨5点维护，早上7点维护……

　　运维值班：0点盯着屏幕打电话，1点盯着屏幕打电话，2点盯着屏幕打电话……

运维开发：写着各种的逻辑，因为业务、网络环境、BUG、刚刚帮忙扛完服务器。

　　游戏运维业务范围：在端游时代，大部分游戏公司都是自主做各种业务环境，做各种游戏业务需要的各种环境。

　　资产管理：服务器、[交换机](http://product.it168.com/list/b/0412_1.shtml)、各种服务分布位置，端口等。

　　下载服务器：搭建BT集群，做种子、分发，供玩家下载游戏客户端使用。

　　静态缓存服务器：squid + apache|nginx

　　邮件服务器：postfix +sasl +ssl 收发服务、反垃圾邮件服务。

　　网络质量监控：somkingping各个机房的交换，各个安放点服务器。

　　配置管理：nginx 、apache、lighttpd、MySQL。

　　批量管理：ssh公钥/私钥 。

监控管理：nagios、catcai 然后是c|perl|python|shell +rrdtool各种业务监控图。

游戏游戏服务器架构：一般来讲都是以一组服务器集群为一个区服单位，单机上的进程提供不同的服务。

## 2.2功能需求分析

批量管理的难点在于：

　命令的并发执行，要控制各点的超时时间

　　执行过程中，不同功能的不同权限要求

　 数据通信安全的保证，以及能够正常解析数据指令

　 人员账号权限管理，权限分发及回收

　 物理服务器、云服务器统一化安装及老项目改造

　 网络质量不可靠的情况下，执行不完整的情况下业务功能回滚

## 2.3性能需求分析

　　每天都会对服务器进行上线，升级等操作，每款游戏在一个平台的集群数在几十个到几百个不等(根据平台大小)。因此每天维护和升级服务器压力极大，服务器异常或响应慢等问题的发生会给用户体验带来伤害。 这样的隐患在于一旦发生游戏关服之后就必须对玩家进行游戏中货币和元宝的赔偿，平均每个玩家补偿的元宝至少在5元以上，游戏币和各类游戏道具若干，以此类推由于服务器故障造成的损失可想而知。

　　有的游戏使用了听云Server，能够对服务器响应慢和不可用进行定位，查看慢应用追踪和Web应用过程功能，能够实时定位消耗资源最大的代码和语句，这样就能帮助实时进行有针对性的调整和优化，并且可以快速定位问题时间，最快能到分钟级别。

发生高并发、服务器压力激增的情况时，平时运行正常的服务器异常概率大幅增加，日常可能的性能瓶颈点会被成倍放大，这就需要实时定位和解决性能瓶颈点，和提前进行预防改善。一般来说，传统日志收集方式耗时耗力，效果非常不好，大掌门用了听云Server后，可以进行1分钟级定位能迅速有效发现瓶颈点。同时还结合了听云Network的压测功能，能够在服务器上线前提前发现到高压力下的瓶颈点，提前预防，避免由于高并发出现的服务器瓶颈。

还有一种性能情况需要提前预防，游戏公司盈利在于玩家的充值，对于官网上从登陆到充值全流程的成功率业务部门极其关注，玩家点击跳转的失败会直接导致充值付费用户的转化率。对此，有的通过听云Network的事务流程功能能够实时对事物流程进行警报，帮助业务部门提升用户充值的转化率。

运维体系提供方便稳定的管理，使得将来易于与当前游戏实现互连互通，为用户提供全方位、高质量和高效率服务。在了解运维体系背景的前提下就可以进行可行性分析，目的是为了避免盲目投资，减少不必要的损失。可行性研究，就是用最小的代价在尽可能短的时间内确定问题是否能够解决。它需要进一步分析和澄清问题的定义，在问题定义阶段初步确定的规模和目标，如果是正确的就进一步加以肯定，如果有错误就应该及时修正。

1. **运维体系搭建平台**

## 3.1服务器简介

通常我们可以将服务器看做一台配置功能强大的超级电脑，它也有自己独立的操作系统，其中核心系统以Linux系统为主的服务器，我们都可以称为Linux服务器，在服务器选择时时，我们通常可以根据我们的喜好和操作习惯来选择服务器的操作系统。Linux服务器是被设计来处理的业务应用，如网络和系统管理，数据库管理和Web服务等，是具备高性能和开源性的一种服务器。Linux 内核源代码可以免费下载。大多数 Linux 发布版本，包括 GNU/Linux 的发行版本和商业的发行版本几乎都提供免费下载服务。免费意味着零试用成本，也不需要为安装在第二台机器上付费。如果你在大学机房里勤工俭学，从朋友那里听到一个新兴的操作系统，还是免费的。你有什么理由不下载下来试一试呢？如果你使用之后觉得比一些老掉牙的 Unix 系统好用，为什么不给安装到机房的其他机器上呢？Linux系统是众所周知最具稳定性的系统;事实上，实时上，以Linux系统的美国服务器用户在出现系统崩溃的反馈上比例非常少。这对于用户来说，特别是小型和中小型企业的用户特别有价值，因为服务器系统一旦崩溃对企业的业务会造成很大的影响。与windows相比，Linux处理不运行的进程的能力远高于windows，因为正是这些进程影响了windows的稳定性的主要因素。再有就是重新启动的方面上来说。Windows配置更改通常需要重新启动-造成不可避免的停机时间。而Linux操作系统一般不需要重新启动，几乎所有的Linux配置变化可以在系统运行时进行，并不会影响相关的服务。同样的，而Windows服务器必须经常频繁地进行碎片整理，而在Linux服务器上就很少需要进行这一步处理。Linux是天生就比Windows更具安全性，无论是从服务器系统，桌面上或在嵌入式环境Linux系统都更具安全性。因为Linux是基于Unix所发展的，从一开始就设计成一个多用户操作系统。只有管理员或root用户，具有管理权限，和更少的用户和应用程序有权限访问内核。这样的设计使得Linux无论在管理上还是在安全上都更加规范。  
　　当然，Linux受到攻击较少受到病毒和恶意软件的频率更加小，如果存在漏洞时，往往会更加容易被发现。在Linux的世界中，没有出现过已经潜伏了超过六年的内核漏洞。在内部，同时，Windows系统的用户有时会隐藏在系统管理员的文件。而在Linux服务器上上，系统管理员总能对所有的文件可见，并且这些文件始终处于受控状态。而Windows服务器通常需要频繁的硬件升级，以容纳不断增加的资源需求，Linux服务器更灵活性和可扩展性，无论处理器或机器架构都能很好地进行资源上的容纳。如果企业有需要时，也很容易对Linux服务器进行重新配置，从而进一步减少了内存的需求，从而提高性能并保持事情就更简单了。因为软件本身是免费，因而Linux服务器在应用开发上更能节约成本。即使有企业购买了专业的版本，但是在性价比上它也比window或者其他系统更加高因为它通常包括基于用户的许可和一系列昂贵的附加组件，尤其是在服务器的安全机制上。

Linux服务器更具自由性，无论用户选择什么样的业务，都可以用Linux系统对业务进行更加自由的开发。Linux服务器就是采用Linux系统的网络服务器，同时也有采用windows的服务器，作用是类似的。而Linux并不是一个特定的系统，而是使用Linux内核的系统， 现在发行版很多，具体拿CentOs 6.0为例。

CentOS是Linux发行版之一，它是来自于Red Hat Enterprise Linux依照[开放源代码](http://baike.baidu.com/view/1708.htm)规定释出的源代码所编译而成。由于出自同样的[源代码](http://baike.baidu.com/view/60376.htm)，因此有些要求高度稳定性的[服务器](http://baike.baidu.com/view/899.htm)以CentOS替代商业版的[Red Hat](http://baike.baidu.com/view/1139590.htm) Enterprise Linux使用。两者的不同，在于CentOS并不包含封闭源代码软件。CentOS 是一个基于Red Hat Linux 提供的可自由使用[源代码](http://baike.baidu.com/view/60376.htm)的企业级[Linux](http://baike.baidu.com/view/1634.htm)发行版本。每个版本的 CentOS都会获得十年的支持（通过安全更新方式）。新版本的 CentOS 大约每两年发行一次，而每个版本的 CentOS 会定期（大概每六个月）更新一次，以便支持新的硬件。这样，建立一个安全、低维护、稳定、高预测性、高重复性的 Linux 环境。CentOS是[源代码](http://baike.baidu.com/view/60376.htm)再编译的产物，而且在RHEL的基础上修正了不少已知的 Bug ，相对于其他 Linux 发行版，其稳定性值得信赖。

## 3.2数据库简介

数据库是按照[数据结构](http://baike.baidu.com/view/9900.htm)来组织、[存储](http://baike.baidu.com/view/87682.htm)和管理数据的仓库，它产生于距今六十多年前，随着[信息技术](http://baike.baidu.com/view/3226.htm)和市场的发展，特别是二十世纪九十年代以后，[数据管理](http://baike.baidu.com/view/14717.htm)不再仅仅是存储和管理数据，而转变成用户所需要的各种数据管理的方式。数据库有很多种[类型](http://baike.baidu.com/subview/738155/8050031.htm)，从最简单的存储有各种数据的[表格](http://baike.baidu.com/view/899068.htm)到能够进行海量[数据存储](http://baike.baidu.com/view/551712.htm)的大型[数据库系统](http://baike.baidu.com/view/7809.htm)都在各个方面得到了广泛的应用。在信息化社会，充分有效地管理和利用各类信息资源，是进行科学研究和决策管理的前提条件。数据库技术是管理信息系统、办公自动化系统、决策支持系统等各类信息系统的核心部分，是进行科学研究和决策管理的重要技术手段。

数据库，简单来说是本身可视为[电子化](http://baike.baidu.com/view/766789.htm)的[文件柜](http://baike.baidu.com/view/497625.htm)——存储电子[文件](http://baike.baidu.com/view/345685.htm)的处所，用户可以对文件中的数据进行新增、截取、更新、删除等操作。数据库指的是以一定方式储存在一起、能为多个用户共享、具有尽可能小的[冗余度](http://baike.baidu.com/view/1030522.htm)的特点、是与应用程序彼此独立的数据[集合](http://baike.baidu.com/view/15216.htm)。在[经济管理](http://baike.baidu.com/view/616869.htm)的日常工作中，常常需要把某些相关的数据放进这样的“仓库”，并根据管理的需要进行相应的处理。数据库是依照某种数据模型组织起来并存放二级存储器中的数据集合。这种数据集合具有如下特点：尽可能不重复，以最优方式为某个特定组织的多种应用服务，其数据结构独立于使用它的应用程序，对数据的增、删、改、查由统一软件进行管理和控制。从发展的历史看，数据库是数据管理的高级阶段，它是由文件管理系统发展起来的。

[数据库](http://baike.baidu.com/view/1088.htm)是一个[单位](http://baike.baidu.com/view/142835.htm)或是一个应用领域的通用[数据处理系统](http://baike.baidu.com/view/451490.htm)，它存储的是属于企业和事业部门、[团体](http://baike.baidu.com/view/635653.htm)和个人的有关数据的[集合](http://baike.baidu.com/view/15216.htm)。数据库中的数据是从全局观点出发建立的，按一定的[数据模型](http://baike.baidu.com/view/72142.htm)进行组织、描述和存储。其结构基于数据间的自然联系，从而可提供一切必要的存取[路径](http://baike.baidu.com/view/59642.htm)，且数据不再针对某一应用，而是面向全组织，具有整体的结构化特征。数据库中的数据是为众多用户所[共享](http://baike.baidu.com/view/191006.htm)其信息而建立的，已经摆脱了具体[程序](http://baike.baidu.com/view/17674.htm)的限制和制约。不同的用户可以按各自的用法使用数据库中的数据；多个用户可以同时共享数据库中的数据资源，即不同的用户可以同时存取数据库中的同一个数据。数据共享性不仅满足了各用户对信息内容的要求，同时也满足了各用户之间信息通信的要求。

数据库的优点：[数据共享](http://baike.baidu.com/view/125982.htm)包含所有用户可同时存取数据库中的数据，也包括用户可以用各种方式通过接口使用数据库，并提供数据共享。同文件系统相比，由于数据库实现了数据共享，从而避免了用户各自建立应用文件。减少了大量重复数据，减少了数据冗余，维护了数据的一致性。数据的独立性包括逻辑独立性（数据库中数据库的[逻辑结构](http://baike.baidu.com/view/540423.htm)和[应用程序](http://baike.baidu.com/view/330120.htm)相互独立）和物理独立性（数据物理结构的变化不影响数据的逻辑结构）。[文件管理方式](http://baike.baidu.com/view/2961936.htm)中，数据处于一种分散的状态，不同的用户或同一用户在不同处理中其文件之间毫无关系。利用数据库可对数据进行集中控制和管理，并通过[数据模型](http://baike.baidu.com/view/72142.htm)表示各种数据的组织以及数据间的联系。[数据一致性](http://baike.baidu.com/view/4138648.htm)和可维护性，以确保数据的安全性和可靠性

安全性控制：以防止数据丢失、错误更新和越权使用；[完整性控制](http://baike.baidu.com/view/3253924.htm)：保证数据的正确性、有效性和相容性；[并发控制](http://baike.baidu.com/view/1672732.htm)：使在同一时间[周期](http://baike.baidu.com/view/120466.htm)内，允许对数据实现多路存取，又能防止用户之间的不正常交互作用。故障恢复：由[数据库管理系统](http://baike.baidu.com/view/68446.htm)提供一套方法，可及时发现故障和修复故障，从而防止数据被破坏。[数据库系统](http://baike.baidu.com/view/7809.htm)能尽快恢复数据库系统运行时出现的故障，可能是物理上或是逻辑上的错误。比如对系统的误操作造成的数据错误等。

MySQL是一个开放源代码的关系数据库管理系统，开发者为瑞典MySQL AB公司，现为Sun公司的一部分。2009年Oracle宣布收购Sun公司，因此MySQL成为Oracle旗下产品，如图所示。



图3-2 mysql数据库图形标志

MySQL由于性能高、成本低、可靠性好，已经成为最流行的开源数据库，被广泛地应用在Intenet上的中小型网站中。随着MySQL的不断成熟，它也逐渐用于更多大规模网站和应用，比如Google和Facebook。

## 3.3 Zabbix监控

Zabbix是一个企业级的开源分布式监控解决方案，由一个国外的团队持续维护更新，软件可以自由下载使用，运作团队靠提供收费的技术支持赢利。Zabbix通过C/S模式采集数据，通过B/S模式在web端展示和配置。

Zabbix功能:具备常见的商业监控软件所具备的功能（主机的性能监控、网络设备性能监控、数据库性能监控、FTP等通用协议监控、多种告警方式、详细的报表图表绘制）支持自动发现网络设备和服务器；支持分布式，能集中展示、管理分布式的监控点；扩展性强，server提供通用接口，可以自己开发完善各类监控。分别有几类: CPU负荷、内存使用、磁盘使用、网络状况、端口监视、日志监视等。

优点：开源，无软件成本投入；Server对设备性能要求低（实际测试环境：虚拟机CentOS5，2GCPU 1G内存，监控5台设备，CPU使用率基本保持在10%以下，内存剩余400M以上）；支持设备多；支持分布式集中管理；开放式接口，扩展性强。

Zabbix运行条件：Server-Zabbix Server需运行在LAMP（Linux+Apache+Mysql+PHP）环境下，对硬件要求低；Agent-目前已有的agent基本支持市面常见的OS，包含Linux、HPUX、Solaris、Sun、windows；SNMP-支持各类常见的网络设备。

Server：指安装zabbix服务的服务器（以下简称服务器端），是最重要的部份，主要安装在linux系统上（支持多种操作系统），采用mysql存储监控数据并使用apache+php的方式呈现。

　　Agent：指安装在被监控设备上的zabbix代理（以下简称代理），被监控设备上的数据由代理收集后统一上传到服务器端由服务器端收集、整理并呈现。

　　SNMP：也是agent的一种，指支持SNMP协议的设备（也可以是服务器），通过设定SNMP的参数将相关监控数据传送至服务器端（大部份的交换机、防火墙等网络设备都支持SNMP协议）。

　　IPMI：Agent的另一种方式，主要应用于设备的物理性能监控，例如设备的温度、风扇的转速等。

Zabbix支持多种网络方式下的监控，可通过分布式的方式部署和安装监控代理，整个架构如图所示。

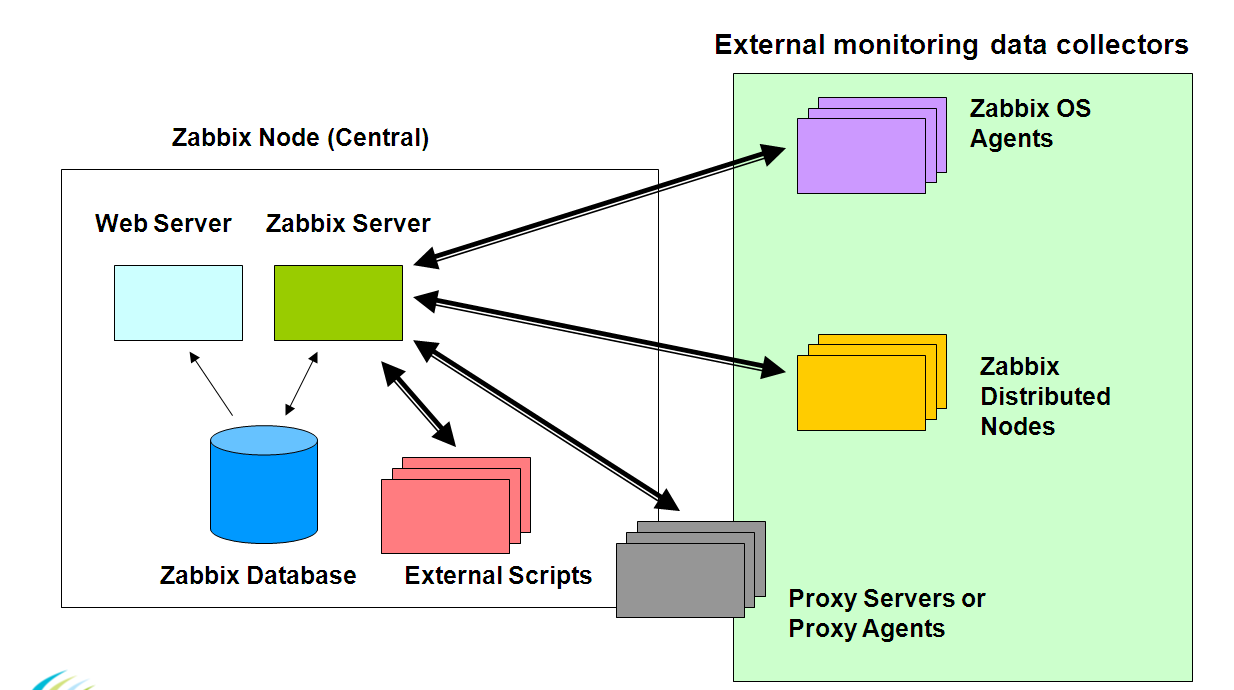
[](http://img1.51cto.com/attachment/201208/130335905.png)

图3-3 Zabbix配置架构图

　 Zabbix安装后服务端默认的配置文件存在安装文件夹中的/etc中，假设编译的参数是“—prefix=/usr/local/zabbix”，那么服务端的配置文件则在参数比较多的Zabbix\_server.conf中，设定值的跨度也很大，这里仅列出常用的一些参数。如图三所示。Zabbix\_server.conf参数除了保证服务正常运行外还涉及该服务器的性能，如果参数设定不合理可能会导致zabbix添加主机不正常、代理端数据无法正常收集或是zabbix服务器性能严重下降，经常报告CPU占用过高或是IO占用过高等问题。

Zabbix安装

zabbix WEB环境搭建  
zabbix的安装需要LAMP或者LNMP环境。需要其它的软件包。

yum install mysql-dev gcc net-snmp-devel curl-devel perl-DBI php-gd php-mysql php-bcmath php-mbstring php-xm

zabbix 数据库设置

zabbix数据库可以和zabbix服务器分离，采用用专门的mysql服务器存储数据，此时要给zabbix数据库受相应的权限。

grant all privileges on zabbix.\* to zabbix\_user@'ip' identified by '123456';

注：ip为zabbix服务器的IP地址。

关于数据库的安装，可以查看Mysql安装，我习惯使用二进制包。

启动数据库

/usr/local/mysql/bin/mysqld\_safe  --user=mysql &

登录数据库，创建帐号和设置权限：

mysql> use mysql;

mysql>create database zabbix character set utf8;

mysql>grant all privileges on zabbix.\* to zabbix\_user@'192.168.10.197' identified by '123456';

安装zabbix服务

增加zabbix用户和组

#groupadd zabbix  
#useradd -g zabbix -m zabbix

官网下载解压软件包。

下载地址：

[http://www.zabbix.com/download.php](http://www.zabbix.com/download.php%E5%8F%AF%E4%B8%8B%E8%BD%BD)

#wget http://fossies.org/unix/misc/zabbix-2.0.3.tar.gz  
# tar -zxvf  zabbix-2.0.3.tar.gz  
#cd zabbix-2.203  
导入数据库表

#cd  zabbix-2.0.3/database/mysql  
#mysql -uroot -pmysql zabbix < mysql.sql  
#mysql -uroot -pmysql zabbix < images.sql   
#mysql -uroot -pmysql zabbix < schema.sql

编译安装zabbix  
./configure --prefix=/usr/local/zabbix --with-mysql --with-net-snmp --with-libcurl --enable-server --enable-agent --enable-proxy  
编译报错checking for main in -lmysqlclient ... no  
yum 安装mysql-devel即可  
#make  
#make install  
添加服务端口  
vim /etc/services

zabbix-agent 10050/tcp # Zabbix Agent

zabbix-agent 10050/udp # Zabbix Agent

zabbix-trapper 10051/tcp # Zabbix Trapper

zabbix-trapper 10051/udp # Zabbix Trapper

添加配置文件  
# mkdir -p /etc/zabbix  
# cp -r zabbix-2.203/conf/\* /etc/zabbix/  
# chown -R zabbix:zabbix /etc/zabbix  
修改server配置文件，添加zabbix数据库密码  
vim /etc/zabbix/zabbix\_server.conf

LogFile=/tmp/zabbix\_server.log

PidFile=/tmp/zabbix\_server.pid

DBName=zabbix

DBUser=zabbix\_user

DBPassword=123456     #指定zabbix数据库密码

ListenIP=192.168.10.197    #服务器IP地址

修改Agentd配置文件，更改HOSTNAME为本机的hostname  
vim /etc/zabbix/zabbix\_agentd.conf

PidFile=/tmp/zabbix\_agentd.pid #进程PID

LogFile=/tmp/zabbix\_agentd.log #日志保存位置

EnableRemoteCommands=1 #允许执行远程命令

Server=192.168.10.197 #agent端的ip

Hostname=client1 #必须与zabbix创建的host name相同

如图设置hostname

添加web前段php文件  
# cd zabbix-2.203/frontends/  
# cp -rf php /home/httpd/zabbix   #虚拟主机目录  
# chown -R zabbix:zabbix zabbix  
web前端安装配置  
修改PHP相关参数  
vim php.ini

max\_execution\_time = 300

max\_input\_time = 300

memory\_limit = 128M

post\_max\_size = 32M

date.timezone = Asia/Shanghai

mbstring.func\_overload=2

PHP还必须支持一下模块，在php源码包直接编译安装。详细模块需要在安装是会提示。

bcmath.so、gettext.so

在客户端浏览器上面访问zabbix，开始WEB的前端配置，http://ZabbixIP/zabbix，按提示点击下一步

Step1：下一步。

Step2：如果全部OK的话才能进行下一步的安装，如果有错误请返回到server端检查相关的软件包是否安装。

Step3：需要输入mysql数据库帐号密码,如果数据库不在zabbix服务器上面，在Host里面添加数据库服务器的地址，并且要用grant命令给数据库授权。

Step4：输入服务器端 host name or host IP addres；

最后会自动写入配置文件：zabbix.conf.php，配置完成后出现登陆界面，默认的用户名为：admin，密码为：zabbix。

启动zabbix服务  
在zabbix安装目录下面可以直接启动  
#/usr/local/zabbix/sbin/zabbix\_server start  
tcp        0      0 0.0.0.0:10050               0.0.0.0:\*                   LISTEN      7140/zabbix\_agentd

设置开启自动启动  
vim /etc/rc.d/rc.local  最后添加下面两行

/usr/local/zabbix/sbin/zabbix\_server start

/usr/local/zabbix/sbin/zabbix\_agentd start

至此，zabbix server端的安装完毕，我们可以通过软件来监控CPU负荷、内存使用、磁盘使用、网络状况、端口监视、日志监视等。在游戏运维体系里尤其重要，为最不可或缺的功能实现。

**3.4安全机制**

以防火墙为例来述说安全机制。这里说的是iptables 。iptables 是与最新的 3.5 版本 Linux [内核](http://baike.baidu.com/view/1366.htm)集成的 IP 信息包过滤系统。如果 Linux 系统连接到因特网或 LAN、服务器或连接 LAN 和因特网的[代理服务器](http://baike.baidu.com/view/751.htm)， 则该系统有利于在 Linux 系统上更好地控制 IP 信息包过滤和[防火墙](http://baike.baidu.com/view/3067.htm)配置。防火墙在做信息包过滤决定时，有一套遵循和组成的规则，这些规则存储在专用的信 息包过滤表中，而这些表集成在 Linux 内核中。在信息包过滤表中，规则被分组放在我们所谓的链（chain）中。而且只有一条netfilter/iptables IP 信息包过滤系统是一款功能强大的工具，可用于添加、编辑和移除规则。虽然 netfilter/iptables IP 信息包过滤系统被称为单个实体，但它实际上由两个组件netfilter 和 iptables 组成。netfilter 组件也称为内核空间（kernelspace），是内核的一部分，由一些信息包过滤表组成，这些表包含内核用来控制信息包过滤处理的规则集。iptables 组件是一种工具，也称为用户空（userspace），它使插入、修改和除去信息包过滤表中的规则变得容易。

netfilter/iptables 的最大优点是它可以配置有状态的[防火墙](http://baike.baidu.com/view/3067.htm)，这是 ipfwadm 和 ipchains 等以前的工具都无法提供的一种重要功能。有状态的防火墙能够指定并记住为发送或接收[信息包](http://baike.baidu.com/view/1654236.htm)所建立的连接的状态。防火墙可以从信息包的连接跟踪状态获得该信息。在决定新的信息包过滤时，防火墙所使用的这些状态信息可以增加其效率和速度。这里有四种有效状态，名称分别为 ESTABLISHED 、 INVALID 、 NEW 和 RELATED。状态 ESTABLISHED 指出该信息包属于已建立的连接，该连接一直用于发送和接收信息包并且完全有效。INVALID 状态指出该[信息包](http://baike.baidu.com/view/1654236.htm)与任何已知的流或连接都不相关。

规则（rules）其实就是网络管理员预定义的条件，规则一般的定义为“如果数据包头符合这样的条件，就这样处理这个数据包”。规则存储在内核空间的信息 包过滤表中，这些规则分别指定了源地址、目的地址、传输协议（如TCP、UDP、ICMP）和服务类型（如HTTP、FTP和SMTP）等。当数据包与规 则匹配时，iptables就根据规则所定义的方法来处理这些数据包，如放行（accept）、拒绝（reject）和丢弃（drop）等。配置防火墙的 主要工作就是添加、修改和删除这些规则。

规则表：

1.filter表——三个链：INPUT、FORWARD、OUTPUT  
作用：过滤数据包  内核模块：iptables\_filter.  
2.Nat表——三个链：PREROUTING、POSTROUTING、OUTPUT  
作用：用于网络地址转换（IP、端口） 内核模块：iptable\_nat  
3.Mangle表——五个链：PREROUTING、POSTROUTING、INPUT、OUTPUT、FORWARD  
作用：修改数据包的服务类型、TTL、并且可以配置路由实现QOS内核模块：iptable\_mangle(别看这个表这么麻烦，咱们设置策略时几乎都不会用到它)  
4.Raw表——两个链：OUTPUT、PREROUTING  
作用：决定数据包是否被状态跟踪机制处理

规则链：  
1.INPUT——进来的数据包应用此规则链中的策略  
2.OUTPUT——外出的数据包应用此规则链中的策略  
3.FORWARD——转发数据包时应用此规则链中的策略  
4.PREROUTING——对数据包作路由选择前应用此链中的规则  
5.POSTROUTING——对数据包作路由选择后应用此链中的规则  
规则配置：

#DEFINE OUTPUT CHAINS  
iptables -A OUTPUT -p icmp --icmp any -j ACCEPT  
iptables -A OUTPUT -s localhost -d localhost -j ACCEPT  
iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

iptables -A OUTPUT -p tcp -m state --state NEW --dport $HTTP\_PORT  -j ACCEPT  
iptables -A OUTPUT -p tcp --dport $SECURE\_HTTP\_PORT -j ACCEPT  
iptables -A OUTPUT -p udp --dport $DNS\_PORT -j ACCEPT  
iptables -A OUTPUT -p tcp --dport $SSH\_PORT -j ACCEPT

到这里一个简单的iptables就可以使用了。iptables的基本实现，就是要知道什么样的需求，什么程序不重要，重要的是需要用什么端口，对端口写相应的写出INPUT和OUTPUT的规则就可以了。

1. **运维体系搭建平台**

## 4.1总体设计原则

1. 可靠式管理

简单明了,模式同质；业务特色,分区分类；本地中控,集中管理；安全控制,低自动化；推陈出新,更新频繁。

1. 实用性原则

游戏运维体系的设计要考虑其实用性，运维应用的目的是为了实现业务处理自动化、规范化，提高工作效率，减轻工作人员的劳动强度，减少开支，规范管理。

## 4.2架构概述

一、服务器划分原则

在现有的网络游戏服务器端架构中，多是以功能和场景来划分服务器结构的。负载均衡和集群暂且不在本文中讨论（bigworld、atlas）。服务器划分可以基于以下原则：

分离游戏中占用系统资源（cpu，内存，IO等）较多的功能，独立成服务器。

以多线程或多进程的编程方式适应多核处理器。

在同一个服务器架构下，应尽可能的复用某些服务器（进程级别的复用，比如场景服务器）。

运行时玩家数据的保存、修改及数据流向应该是设计的焦点，它同时也决定了服务器应该如何划分。

服务器的划分应该适度，在保证清晰的数据流向的前提下，根据游戏的类型和规模尽量减少服务器或服务器进程的个数，以减少服务器之间过多的复制数据、锁冲突（使用共享内存进行通讯时）。

主要按照场景划分进程，若需按功能划分，必须保持整个逻辑足够简单，一目了然，易于理解。

二、功能框架

Apache Resin

Web服务 Lighttpd 提供jsp Tomcat

Nginx Jboss

Nginx Memcached

负载均衡 LVS 缓存 Redis

Varvish

Oracle Keepalived

DB Mysql 高可用 Heartbeat

Mongodb

图4-2 服务器功能框架图

## 4.3数据库设计E-R图

以单区单服为例，说明游戏数据库应用列表。

角色

gdb01 状态

充值

任务

等

游戏应用整体架构图，如下。

Game服务器 管理服务器

管理系统 MasterDB SlaveDB

SVN 缓存服务器 Tlog服务器

图4-3 游戏应用架构图

## 4.4体系优化改进

存在问题:业务需求增长迅速，人手不足；新项目中的数据库设计不规范；除了传统游戏数据库应用，出现了其他新型应用；早期基础运维体系建设存在缺陷导致了后期运维规范化、自动化的难度骤增。

体系特点：疲于应对暴增的业务需求；急迫需要提高自动化水平；依靠规范化制度应对变化；不断优化数据库应对压力。

改进措施：自动化优化强制规范化；根据每个项目的特点，选择更加合适的架构、设计、引擎；蚂蚁搬家式推进基础运维体系的规范化；不断提高自身运维自动化水平。

发展阶段-展望：管理系统平台；事件驱动响应；审核审计流程；在线知识积累；在线分析统计。

1. **运维体系详细实现**

## 5.1体系搭建环境

服务器端需求：

服务器使用腾迅开放平台(http://open.qq.com/)云服务.

一组服务器表示一个独立的游戏世界, 分别是游戏服务器和数据库服务器，Tlog服务器。idipProxy服务器，经分服务器各一台。

游戏服务器 第一代机型VB3。4核CPU，16G内存，100G SAS raid1结构双盘硬盘

数据库服务器 CDB服务。

Tlog服务器 第一代机型VC2。4核CPU，3.5G内存，200G SATA raid0结构单盘硬盘。

idipProxy服务器 第一代机型VC2。

经分服务器 第一代机型VC2。

操作系统选择：

系统使用CentOS 6.5位.

系统分区：

系统使用腾迅cvm默认分区.

/swap：2G

/：5G

/usr/local：1G

/data：VB3--85G VC2--188G

## 5.2 所需功能的实现

主要进行版本的发布、维护：

非典型数据的统计需求；周边工具的进一步优化和完善；故障和突发的响应及处理；所有版本的测试配合和支持；运营需求的变更（合区、扩容等）；效率提升（包含发布、突发故障处理、统计等等运营支撑的各方面的效率提升）。

## 5.3服务器管理的实现

游戏服安装配置

基本环境部署：

ssh key配置**，**开平系统使用ssh2作ssh服务器。默认端口是36000。ssh key、配置放置在 /root/.ssh2/目录。由测试服向其他服务器同步，实现root帐号免密码登录，端口可以不指定。

java环境配置：

安装jre-6u33-linux-x64-rpm.bin

crontab计划任务配置：

定时执行脚本，进行日志备份清理。定时向Tlog服务器发送日志。

开平服务器crontab root帐号计划任务配置文件可能为以下两个：

/var/spool/cron/root /var/spool/cron/tabs/root

修改计划任务建议重启cron服务 /etc/init.d/cron restart

脚本：

更新游戏包，数据库表结构等。 各包在newserver/pkg下。如果为最新，可以不执行此步。部署基本环境。全新的游戏服执行。

游戏目录:

Server 服务目录

bin 脚本目录

conf 配置目录

新服要修改conf下配置

conf/env.properties 修改以下几项：

worldServiceId=服务器id

gameServer1=.服务器ip,域名,443,3125

loginServerIpAddr=服务器ip

chatServerIpAddr=服务器ip,域名,8001

worldServerIpAddr=服务器ip

platformServerIpAddr=服务器ip

dbServerIpAddr=服务器ip

jdbcUrl=jdbc:mysql://数据库ip:端口?useUnicode=true&characterEncoding=utf8

conf/zoneWorldIds.properties

zoneWorldIds=域名中的数字，服务器id

bin目录下为启服脚本：

startserver.sh / stopserver.sh 分别为启服和停服

手动启服务顺序：

dbServer-->worldServer-->platformServer-->gameServer-->chatServer-->loginServer

停服顺序与启服相反。

日志目录：

日志目录在/data/logs/

对应服务在对应目录。

XyChatServer XyDbServer XyGameServer XyLoginServer XyPlatformServer XyWorldServer

YLog目录是向Tlog传的日志。

DBmerge是合服工具日志。

备份策略：

各服务日志每天压缩保存。压缩后日志保存30天。另保留7天非压缩的原始log文件

YLog日志每天压缩，永久保存。另保留7天非压缩的原始log文件

创建数据库：

db\_gfxy\_gdb\_$id 为库名。$id对应服务器id

1.create database db\_gfxy\_gdb\_$id default charset utf8\_general\_ci;

2.grant select on db\_gfxy\_gdb\_$id.\* to 'sgf'@'10.18.41.15' identified by 'gfxy123’;

## 5.4 数据库管理的实现

数据库管理的主要内容有：数据库的建立、数据库的调整、数据库的重组、数据库的重构、数据库的安全控制、数据的完整性控制和对用户提供技术支持。  
数据库的建立：数据库的设计只是提供了数据的类型、逻辑结构、联系、约束和存储结构等有关数据的描述。这些描述称为数据模式。要建立可运行的数据库，还需进行下列工作：  
(1)选定数据库的各种参数，例如最大的数据存储空间、缓冲决的数量、并发度等。这些参数可以由用户设置，也可以由系统按默认值设置。  
(2)定义数据库，利用数据库管理系统(DBMS)所提供的数据定义语言和命令，定义数据库名、数据模式、索引等。  
(3)准备和装入数据，定义数据库仅仅建立了数据库的框架，要建成数据库还必须装入大量的数据，这是一项浩繁的工作。在数据的准备和录入过程中，必须在技术和制度上采取措施，保证装入数据的正确性。计算机系统中原已积累的数据，要充分利用，尽可能转换成数据库的数据。

以下为实现基础命令：

mysql> select database();  
MySQL中SELECT命令类似于其他编程语言里的print或者write，你可以用它来显示一个字符串、数字、数学表达式的结果等等。  
1.显示MYSQL的版本  
mysql> select version();   
+-----------------------+   
| version()             |   
+-----------------------+   
| 6.0.4-alpha-community |   
+-----------------------+   
1 row in set (0.02 sec)   
2. 显示当前时间  
mysql> select now();   
+---------------------+   
| now()               |   
+---------------------+   
| 2009-09-15 22:35:32 |   
+---------------------+   
1 row in set (0.04 sec)   
3. 显示年月日  
SELECT DAYOFMONTH(CURRENT\_DATE);   
+--------------------------+   
| DAYOFMONTH(CURRENT\_DATE) |   
+--------------------------+   
|                       15 |   
+--------------------------+   
1 row in set (0.01 sec)   
4. 显示字符串  
mysql> SELECT "welecome to my blog!";   
+----------------------+   
| welecome to my blog! |   
+----------------------+   
| welecome to my blog! |   
+----------------------+   
1 row in set (0.00 sec)   
5. 串接字符串  
select CONCAT(f\_name, "", l\_name)   
AS Name   
from employee\_data   
where title = 'Marketing Executive';   
+---------------+   
| Name          |   
+---------------+   
| Monica Sehgal |   
| Hal Simlai    |   
| Joseph Irvine |   
+---------------+   
3 rows in set (0.00 sec)   
注意：这里用到CONCAT()函数，用来把字符串串接起来。另外，我们还用到以前学到的AS给结果列'CONCAT(f\_name, "", l\_name)'起了个假名。

命令：create table <表名> ( <字段名1><类型1> [,..<字段名n><类型n>]);  
例如，建立一个简单的名为MyClass的表。如下

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名 | 数字类型 | 数据宽度 | 是否为空 | 是否主键 | 自动增加 | 默认值 |
| id | int | 4 | 否 | primary key | auto\_increment |  |
| name | char | 20 | 否 |  |  |  |
| sex | int | 4 | 否 |  |  | 0 |
| degree | double | 16 | 是 |  |  |  |

图 5-4 数据表实现简易图

数据表简易实现执行下面的命令：  
mysql> create table MyClass(  
> id int(4) not null primary key auto\_increment,  
> name char(20) not null,  
> sex int(4) not null default '0',  
> degree double(16,2));

**5.5 监控机制的实现**

信息部所使用的的zabbix版本为2.0；数据据版本mysql5.1；zabbix客户端代理安装在c:\zabbix目录下，该文件夹大小占用600k~1M；成功安装后在系统中生成服务“Zabbix Agent”,服务启动方式为自动（如图一所示）；监控状态下windows监控代理每天（24小时）的数据传输量是14.0625K，对系统性能的影响小到可以可以忽略不计。当监控参数约为100个时（其中基本参数66个，IIS或SQL参数16到27个不等），每分钟取一个性能值（100\*60\*24=14400Byte）。

菜单栏介绍

　 　一级菜单栏 “监控”Monitoring项（图三）提供监控设备的详细数据，共十项二级菜单，包括：“仪表盘”项Dashboard、“总览”项Overview、WEB项、“最新数据”项Latest data、“触发器”项Triggers、“事件”项Events、“图形”项Graphs、“多图显示”项Screen、“拓扑图”项Maps、IT Services项；一级菜单栏Inventory“资产”项（图四）主要记录被监控设备的配置清单（较少使用）。二级菜单包括资产总览及主机项；一级菜单栏Reports“报告”项（图五）主要提供被监控设备的性能报告（较少使用）。二级菜单包括“可用性报告”Avaliablity report、“前100个触发数据”Triggers top 100、“柱形图”Bar reports。

所需参数：

Free disk space on c:                                        C盘可用空间  
Free disk space on d:                                        D盘可用空间  
Free memory                                                  可用内存空间  
Memory Commit Limit (RAM plus Page File)                     内存可用数（包括虚拟内存）  
Percent Free disk space on c:                            C盘可用空间百分比  
Percent Free disk space on d:                            D盘可用空间百分比  
Total disk space on c:                                       C盘容量  
Total disk space on d:                                       D盘容量  
Total memory                                                 内存大小

Combined CPU Utilization %                                   CPU处理非闲置进程的百分比  
CPU Count                                                    CPU内核数量（个数\*核\*线程数）  
CPU Utillization                                             CPU利用率  
Processor load                                               CPU负载（一分钟）  
Processor load15                                             CPU负载（十五分钟）  
Processor load5                                              CPU负载（五分钟）

Average Read Transfer Time to Disk \_Total                             磁盘平均读响应时间  
Average Write Transfer Time to Disk \_Total                            磁盘平均写响应时间  
Avg. Combined(Read/Write) Transfer Operation to Disk                  磁盘读写响应时间  
Physical Disk Avg. Queue Length                                       磁盘平均数据队列响应  
Physical Disk Percent Idle Time                                       磁盘空闲率

Host uptime (in sec)                                             主机已运行时间  
Paging File % Usage                                               虚拟内存利用率  
Number of threads                                                主机上运行的线程数量

**5.6 安全机制的实现**

安全性和效率以及流程的复杂程度往往是对立的，越是安全的系统，流程越是复杂，效率也越低。我们在游戏的开发过程中需要做出折中的选择，达到安全性和效率的平衡。

游戏服务器在设计和实现上的安全性问题主要可以分为以下几个大类：

协议安全  
协议安全所指的是网络数据包的安全，通常意义上讲就是指网络数据包的加密和解密，由于网络数据包传输频繁，如果使用复杂的加解密技术将会带来巨大的CPU开销，另外由于客户端程序内存有数据包加密和解密的完整代码，因此无论多复杂的加解密算法都不能阻止破解数据包事件的发生。建议对于游戏中的一般网络包不需要进行复杂的加解密过程，只对数据包进行简单编码或者不加密也是可以接受的。使用不对称的加密算法，客户端的代码中不能包含解密的算法和密钥。保证客户端每次加密后的数据是动态的，以防止网络数据包被它人录制以后可以再次成功使用。例如，客户端有很复杂的加密过程，并且没有解密算法，但是如果加密过程没有动态性，由于每次输入相同的用户名加密码生成的数据包是完全相同的，那么如果被别人录制了登录的数据包，就可以伪造成功的登录。在不考虑使用密保等具有动态特性的组件以外，我们也可以考虑使用验证码的机制，新建立连接以后，服务器会发送当前有效的一段特征码，客户端加密数据需要使用到这一段特征码。这样即使登录数据被别人录制以后也无法直接使用，因为新建立的连接特征码是不同的。

另外提一下关于服务器加入部分反外挂特性的功能。一般游戏服务器内部主要能够集成的反外挂方式是防止加速外挂。目前常用的反加速外挂主要有两种：

一种方式是在客户端连接服务器后，服务器会把当前的时间发送到客户端，以后客户端发送的 每个数据包都需要包含自己计算出的服务器当前时间信息，服务器如果检测到数据中描述的服务器时间 超过了实际的时间，就认为玩家有可能使用了加速外挂。这种方式相对服务器负担不大，也不容易出错，但是只能防止变速齿轮等通用外挂，对于能够自己构造协议的外挂没有效果。

另外一种方式是服务器（一般是网关）上根据玩家发送的数据包类型进行分类，同类的数据包相互之间间隔必须满足一定条件，如果玩家连续多次出现同类数据包间隔过短的情况，就会被判定为使用加速器的玩家。这种模式虽然相对比较耗费服务器资源且需要关注到游戏逻辑，但是判断比较准确，基本上能防止各种加速方式。

在开发协议函数时，需要对所有的协议参数都进行合法性检查，服务器能够通过上下文获取的数据都不应该由客户端来提供。

流程安全

玩家游戏中的流程安全是相当重要的一个部分。对于一般的游戏，建议最好的方式是尽量维持使用同一条经过验证的Socket(Tcp)连接进行网络通信，因为相对来说想替换别人的网络数据是一件比较困难的事，除非玩家自己的设备中了木马。说简单一点就是尽量使用同一条TCP连接，先做登录验证，然后进行游戏，中间尽量不做切换连接的流程。但是对于实际的游戏而言并不一定能完全实现游戏过程中不切换连接。有些简单的游戏就没有网关的设计，如果切换线路服务器，必然会重新进行连接。还有部分游戏由于游戏过程是基于UDP或者Http短连接的通信模型，这些模式都需要重新验证玩家的有效性。

同一个SessionID(或者称为accessToken)最好只能通过一次验证。如果游戏类型是切换游戏服务器、线路服务器或者是进入副本服务器时需要重新验证的类型，建议每次验证的SessionID都是重新生成的。如果条件允许的话，SessionID最好还包含有效期限制，但是这个需要针对不同类型的游戏有所不同。遇到重复登录时候，一定要先确定新用户是否合法，然后再处理老的用户。否则就会出现没有验证新用户的有效性，直接就把老用户踢掉了的设计，这样会造成整个系统的玩家都可以被别人随便踢下线。

数据存储安全

游戏最重要的部分是玩家游戏数据的安全性，必须要保证用户数据能够实时安全的保存。玩家切换服务器时，必须保证切换前数据已经被成功保存之后，才能被新的服务器读取。有部分游戏设计中，在切换服务器时，不进行实际存盘动作，只是将数据缓存在DBServer上，这个模式也可以接受，但是需要注意的是如果过程中发生切换服务器失败，玩家掉线等错误状态，DBServer要能够合理的将数据保存。出现重复登录的情况，在踢出老用户的同时需要注意保存用户数据。建议重复登录的同时，最好也拒绝掉本次新的登录，确保用户数据的成功保存、以及老用户临时数据的清理。玩家交易等过程一定要有事务的特性，两个玩家一定要完成物品的交换之后才可能发生写入数据库的操作，这个对于单线程逻辑程序一般很少有问题，但是对于多线程逻辑的程序必须注意。

如果处理数据库的读取和写入操作采用连接池的操作方式，需要注意单个用户数据的事务性和顺序性。可以通过一个用户数据只会在一个数据库连接上处理的方式来进行限制。如果数据库发生故障或者玩家数据异常时，需要通过玩家的操作日志让玩家的数据能回滚到一个正常的值。提供合理的数据库备份机制，目前一般的做法是利用数据库的主从备份机制，每天凌晨对从库进行全量备份，在数据库需要回滚到某个时间点t1的时候，可以每天的全量备份数据为基础，再执行凌晨到t1的binlog。不过，现在很多云服务商都提供了这样的服务，操作起来非常便利。比如阿里云的RDS服务。

安全审计-内部用户登录,外部权限管理,运维操作行为

应用安全:Web漏洞扫描、Webshell监控、渗透性压力测试、代码管理

系统安全：补丁分发管理、主机访问控制、病毒扫描、漏洞跟踪

网络安全：ARP攻击测试、DDOS攻击防御、网络访问控制、网络流量分析、IDC节点优化

## 5.7 流程管理应用的实现

随着系统技术的日益复杂，维护水平要求的提高，运维体系从人员、服务技术等流程方面提升管理能力，以现有的技术水平支撑更庞大的游戏体系并加以优化改进，使运维水平更前进了一大步。

·调整组织结构，以适应业务和流程的需求，并实现量化考核

·以服务导向取代技术较难的理解化

·通过合适的技术，固化脚本，以及提升自动化程度

·参考业界较好实践，建设规范化流程

借鉴ITIL等技术保障中心开发实施了事件管理、问题管理、变更管理、配置管理等流程，通过一系列平台的推广使用，配合相应管理制度，流程的推行，使得所提供的服务器日趋稳定和成熟。配合建立的知识库，提供了经验技能的沉淀模式，更大大有利于运维水平的提高，安全稳定便捷。

自动化运维：客户端补丁自动更新、服务器应用程序更新和版本管理、系统软件自动安装、服务器操作远程登录、操作系统及驱动补丁自动安装、服务器虚拟化系统搭建简单实现。

自动化监控：网络质量监控、IDC机房监控、设备异常监控、游戏服务器监控、系统日志收集及分析、游戏程序监控。

流程管理：以服务导向为核心，建设了游戏稳定保障平台。

1. **总结**

在这次的实验中我有了很深刻的体会，无论是在知识技术方面还是个人自身素质的培养等方面都有 了更新的认识。这次的实验过程中有悲有喜，虽然在此之前做过了课程设计，熟悉了系游戏运维的流程，但在这次的实验中还是碰到了很多的问题，更丰富了自己运维技术的经验，这种学习实难可贵。

下面谈一下经验分享：

制定标准化上线需求

（业务类型、硬件要求、读写比特点、并发预计）

根据需求适当选型

（硬件、操作系统、数据库、引擎、架构）

上架及初始化一步到位

（PXE集成初始化设置、数据库安装自动化）

监控、语境、提前解决问题

（仅有一种监控还不够，需要补充监控；预警&监控敏感度很重要）

1. **自我鉴定**

时光匆匆而过，转眼的时间我在北京理工大学继续教育学院计算机科学技术专业的学习生涯就快结束了，二年半的学习生涯和社会实践生活让我更充实了自己，为此，我特别珍惜在校园短暂的学习时光，也更提高自己自己的求知欲望，为以后的人生打下坚定的基础。

通过这次实践报告，我不仅加强理解了这二年半所学的专业理论知识，并且将所学的知识应用于实践工作中，我觉得自己的收获是巨大的.不仅提高了自己独立思考、独立研究和探索能力，也进一步提高了技能培养和解决实际问题的能力，为此我非常庆幸自己能有机会在学校学习的这短短二年半的时光，也非常感谢老师们的辛苦培育！

这两年半的夜大生活和社会实践，让我不断的挑战自我，充实自己，紧张的学习，丰富的活动使我为实现人生的价值打下了坚实的基础。同时，我始终以提高自身的综合素质为目标，以自我的全面发展为努力方向，树立了正确的人生观，价值观和世界观。

通过学习使自己又了很多更深刻的了解和认知。专业技术上不断的学习与掌握，让我在工作上更加游刃有余。学历上的提升与性格的自我修炼，让我在生活及人生的道路上一路披荆斩棘，勇往直前，又积累了相当重要的生命财富。诚然这段时间有成功有挫折，更有苦恼与释然。夜大的学习不是我们追求的新起点。我将不断迎接挑战，更好的完善自己的人生，更塑辉煌，博得精彩。

**致谢**

经过将近四个月的努力，终于完成了我的毕业设计——在大学的最后一份作业。毕业设计的完成说明我的大学生活已经结束了，接下来就是进入社会，证明自己的时候了。在这两年半里，我不仅仅学到了计算机专业的知识，还懂得了一些为人处事的道理，了解了社会中一些对人对事的技巧，为以后进入社会打下了坚实的基础，在此我要感谢在我大学生活中给予过我帮助的老师、同学和朋友，谢谢你们！我要特别感谢我的毕业设计指导老师姜增如老师，您从毕业设计的开题报告一直到论文的定稿都认真的指导我，及时指出我论文中存在的问题，并指引我论文的不足，在此向所有学校老师及学校致敬并加以最深的感谢！ **参考文献**

[1]Linux运维之道 丁明一编著.工业电子出版社.2014-01

[2]高性能MySQL Baron 王小东.电子工业出版社.2010-02

[3]游戏运营管理 陈洪.清华大学出版社.2009-11

[4]鸟哥的Linux私房菜 鸟哥.人民邮电出版社.2010

[5]服务器架构案例 武新华等.人民大学出版社.2013-10

[6]游戏架构与设计 罗林斯、付毅.红旗出版社.2009-03

[7]游戏架构设计与策划基础 黄石等.清华大学出版社.2010-01

[8]游戏编程入门 King.译者虹夜.人民邮电出版社.2008-09

[9]服务器搭建实战详解 张栋.西安电子科技大学出版社.2013

[10]高性能Linux服务器构建实战 高俊峰编著.机械工业出版社.2012