Netkiller 手札系列电子书 http://www.netkiller.cn

Netkiller Architect 手札

陈景峰 著

Netkiller Architect 手札

目录

- 自述
  - 1. 写给读者
  - 2. 作者简介
  - 3. 如何获得文档
  - 4. 打赏（Donations）
  - 5. 联系方式
- I. 多维度架构
  - 1. 什么是多维度架构
    - 1. 架构与格局
    - 2. 架构师的大局观
    - 3. Architecture Overview
    - 4. CDN (Content Delivery Network)
    - 5. 微服务
    - 6. 日志
    - 7. 监控
  - 2. 多维度架构之网站HTML
    - 1. 网站的历史演变
      - 1.1. 常用软硬件组成
      - 1.2. 第一代纯静态网站
      - 1.3. 第二代纯文本文件采用分隔符做数据存储网站
      - 1.4. 第三代数据库存储网站
      - 1.5. 第四代DNS负载均衡加反向代理
      - 1.6. 第五代负载均衡集群
    - 2. 集群(Cluster)
      - 2.1. 负载均衡
        - 2.1.1. DNS负载均衡

2.1.2. 软件四层交换负载均衡
2.1.3. 应用层负载均衡
2.2. 高可用性集群
2.3. 负载均衡设备
2.4. 会话保持
2.5. 健康状态检查
3. 缓存技术
3.1. 浏览器缓存
3.1.1. Expires
3.1.2. If-Modified-Since / Last-Modified
3.1.3. ETag / If-None-Match
3.2. CDN/逆向代理缓存
3.3. 负载均衡设备
3.4. WEB服务器缓存
3.5. 应用程序缓存
3.6. 数据库缓存
4. 网站静态内容出版
4.1. 架构总览
4.2. 静态化实现手段有哪些?
4.2.1. 生成方式
4.2.2. 抓取方式
4.2.3. 伪静态化
4.2.4. 混合方式
4.2.5. 静态化中的动态内容
4.3. cdn
4.4. www 服务器
4.5. cms 服务器
4.6. img
4.7. Ajax 局部更新与缓存
5. 多媒体数据分离
5.1. 图片服务器分离

        5.2. 目录层次规划
        5.3. 多域名访问
    6. 图片尺寸优化与自动裁剪
        6.1. 背景
        6.2. 实现思路
            6.2.1. 尺寸动态变化
            6.2.2. 实时裁剪并静态化
        6.3. web或代理服务器插件实现方案
    7. 压缩数据传输
        7.1. Minify JS
            7.1.1. jsmin
            7.1.2. yuicompressor
            7.1.3. shrinksafe
    8. SSL
    9. 搜索引擎相关优化
        9.1. 搜索结果静态化
        9.2. robots.txt
        9.3. sitemaps
        9.4. Sitemap in robots.txt
        9.5. sitemap 静态内容生成工具
    10. 静态网站繁简转换
3. 多维度架构之网络损耗
    1. 中国的大网络环境
    2. 架构设计需要考虑网络损耗
        2.1. 硬件造成的网络损耗
        2.2. 云平台造成的网络损耗
        2.3. 容器中造成的网络损耗
        2.4. 微服务造成的网络损耗
    3. 总结
4. 多维度架构之超时时间
    1. 无处不在的超时时间
    2. 流量漏斗

    3. 微服务的超时时间
    4. 容器技术的超时时间
    5. 最后总结
5. 多维度架构之会话数
    1. 路由器和防火墙的会话数
    2. 负载均衡设备的会话数
    3. 服务器的会话数
    4. 应用程序的会话数
6. 多维度架构之日志
    1. 一次切割日志引发的血案
        1.1. 日志是怎么产生的
        1.2. 让我们看个究竟
            1.2.1. 第一种情况，日志文件被重命名
            1.2.2. 第二种情况，日志文件被删除
            1.2.3. 第三种情况，日志没有被删除，也没有被重命名，而是被其他程序做了修改
        1.3. 经典案例分析
        1.4. 怎样监控日志
        1.5. 总结
    2. 日志归档与数据挖掘
        2.1. 什么日志归档
        2.2. 为什么要做日志归档
        2.3. 何时做日志归档
        2.4. 归档日志放在哪里
        2.5. 谁去做日志归档
        2.6. 怎样做日志归档
            2.6.1. 系统日志
            2.6.2. 应用程序日志
            2.6.3. 日志格式转换
                2.6.3.1. 将日志放入数据库
                2.6.3.2. Apache Pipe

2.6.3.3. Log format
2.6.3.4. 日志导入到 MongoDB
2.6.4. 日志中心方案
2.6.4.1. 软件安装
2.6.4.2. 节点推送端
2.6.4.3. 日志收集端
2.6.4.4. 日志监控
7. 多维度架构之监控
1. 背景
2. 概述
3. 怎样监控
3.1. 卫星监测
3.2. 逐级诊断
3.3. 模拟人工
3.4. 数据分析
3.5. 监控与开发
4. 总结
8. 分布式计划任务
1. 什么是分布式计划任务
2. 为什么采用分布式计划任务
3. 何时使用分布式计划任务
4. 分布式计划任务的部署
5. 谁来写分布式计划任务
6. 怎么实现分布式计划任务
6.1. 分布式互斥锁
6.2. 队列
6.3. 其他
9. 多维度架构之安全
1. 植入式攻击入侵检测解决方案
1.1. 什么是植入式攻击?
1.2. 为什么骇客会在你的系统里面植入木马?

        1.3. 什么时候被挂马?
        1.4. 在那里挂马的?
        1.5. 谁会在你的系统里挂马?
        1.6. 怎样监控植入式攻击
            1.6.1. 程序与数据分离
            1.6.2. 监控文件变化
            1.6.3. 安装日志收集程序
    2. Shell 历史记录异地留痕审计与监控
        2.1. 什么是Shell历史记录异地留痕与监控
        2.2. 什么要将Shell历史记录异地留痕并监控
        2.3. 何时做历史记录异地留痕
        2.4. 在哪里做历史记录异地留痕
        2.5. 角色与权限
        2.6. 怎么实现历史记录异地留痕
            2.6.1. 节点配置
            2.6.2. 推送端
            2.6.3. 收集端
    3. 延伸阅读
10. Shell 高级编程
    1. 递归调用
    2. 实现守护进程
    3. 进程间通信
11. DevOps实施中你可能遇到的问题
    1. 什么是DevOps?
    2. 为什么会诞生DevOps?
    3. DevOps 虽好，为什么难以普及呢?
    4. 软件工程的历史与进化
    5. 为什么很多企业为什么实施 DevOps 以失败告终?
    6. CI 持续集成不是DevOps
    7. CD 持续交付不是 DevOps

    8. 自动化部署
    9. 收集各部门问题
        9.1. 自运维的需求
    10. 收缩技术栈
        10.1. 模块化思维
    11. 被遗忘的数据库
    12. 建立中心仓库
    13. 缓存
    14. 安全
12. Kubernetes & Docker 实施中你会遇到的问题
    1. 镜像会遇到的问题
        1.1. 镜像使用的OS发行版不统一
        1.2. 安装位置不统一
        1.3. Linux 系统也存在BUG
    2. 容器会遇到的问题
        2.1. 程序启动的区别
        2.2. 存储面临的问题
        2.3. 内部域名DNS
        2.4. 容器与网络
        2.5. 容器的管理
        2.6. 容器与安全
            2.6.1. 网络安全
            2.6.2. 挂马风险
            2.6.3. 隔离安全
        2.7. 容器与监控
        2.8. 容器与CI/CD
        2.9.
    3. 人员的问题
    4. 最后总结
13. 多维度架构之微服务
    1. 微服务安全吗?

        1.1. 配置中心的隐患
        1.2. 注册中心的隐患
        1.3. Eureka 客户端
        1.4. 最终总结
    2. 熔断器解决了什么问题?
    3. 微服务的性能
        3.1. 微服务的开销
    4. 多维度架构之微服务拆分
        4.1. 分布式事务之路
        4.2. 微服务拆分法则
            4.2.1. 基于工作流拆分服务
            4.2.2. 服务池的概念
        4.3. 最后总结
    5. 接口安全
        5.1. Restful 安全问提
        5.2. 第一个阶段采用 HTTP Basic Auth
        5.3. 第二阶段 HTTP Basic Auth + SSL
        5.4. 第三阶段 HTTP2 + HTTP Basic Auth + Oauth2
        5.5. 第三阶段，终极版诞生，HTTP2 + HTTP Basic Auth + Oauth2 + Jwt
14. 多维度架构之远程异地灾备
    1. 背景
        1.1. 建立容灾系统需要考虑哪些因素
        1.2. 目前容灾系统的实现方式
        1.3. 系统灾难恢复等级划分
        1.4. 做灾备你面临最大的挑战是什么?
    2. 灾备整体解决方案
        2.1. 双活互备方案
        2.2. 三机房互备方案
    3. 数据中心网络

    3.1. 单机房高可用双活互备解决方案
    3.2. 双机房互备异地灾备方案
    3.3. 三机房互备异地灾备方案
4. 服务器部署
    4.1. 网站
    4.2. 数据源
    4.3. 数据库
5. 软件开发
    5.1. 交易软件分布式交易
        5.1.1. 分布式交易解决方案一
        5.1.2. 分布式交易解决方案一
        5.1.3. 分布式交易解决方案一
    5.2. 交易终端
        5.2.1. 用户分流
        5.2.2. 会话保持
    5.3. API 应用程序接口
    5.4. 大数据的问题
        5.4.1. 第一个阶分区表
        5.4.2. 第二个阶分库分表
    5.5. 数据校对
6. 自动化运维
7. 灾备培训和演练
    7.1. 培训内容
        7.1.1. 培训对象
        7.1.2. 操作流程
    7.2. 演练环境设置
    7.3. 演练级别与方式
    7.4. 开始演练
        7.4.1. 切换前操作
        7.4.2. 切换操作

7.4.3. 切换后检查
7.5. 演练结果检查
7.6. 可能存在的风险
7.6.1. 主交易系统短期无法恢复
7.6.2. 切换灾备系统后业务的影响
7.6.3. 数据不同步产生的影响
7.7. 灾备系统备份
7.8. 系统运营维护
8. FAQ
8.1. 实现双活最大的障碍是什么?
8.2. 双活怎么解决数据冲突问题
15. 多维度架构之应用防火墙
1. 什么是应用防火墙
2. 功能需求
2.1. 计数器
2.2. 访问控制列表 ACL
2.3. 用户认证
2.4. 协议
3. 简单实现
3.1. 权限控制与实现
3.2. 演示
3.3. 增加7 Layer防火墙
16. 数据库与应用程序间通信
1. 管道通信
1.1. 背景
1.2. 解决思路
1.3. Mysql plugin
1.4. plugin 的开发与使用
1.5. 插件如何使用
1.6. 部署相关问题
2. 消息队列

        2.1. 背景
        2.2. 应用场景
        2.3. Mysql plugin
        2.4. plugin 的开发与使用
        2.5. 插件如何使用
    3. 数据库与外界文件
        3.1. 背景
        3.2. 解决思路
        3.3. 解决方案
        3.4. plugin 的开发与使用
        3.5. 在事务中使用该插件
        3.6. 通过触发器调用图片处理函数
    4. Socket 方式
        4.1. UDP
17. 多维度架构之消息队列
    1. 你是怎样使用消息队列的?
    2. 你是否真正理解了消息队列?
        2.1. 消息队列并不是实时的
        2.2. 不能替代异步执行
    3. 使用的合理吗?
    4. 是否有必要使用消息队列?
    5. 最后总结
18. 多维度架构之分库分表
    1. 分表需要从业务角度考虑
    2. 最后总结
19. 多维度架构之压力测试
    1. (What) 什么是压力测试
        1.1. 压力测试存在哪些问题
    2. (Why) 为什么做压力测试
    3. (Where) 在哪里做压力测试
    4. (When) 什么时间做压力测试

5. (Who) 压力测试过程参与人员
6. (How) 如何做压力测试

II. Database Modeling Design

20. 关系型数据库设计
1. 数据字典
2. 用户帐号表
2.1. 用户注册键盘跟踪表设计
3. 分类表设计
3.1. 树形分类表
3.2. 多对多分类
3.3. 快速检索子分类设计
3.4. 计算节点数量
3.5. Example
4. 文章表设计
4.1. 分区表设计
4.2. Title性能优化
5. 评论表
6. 记录点击率，阅读次数，及评分表
7. 产品属性表
7.1. 简单实现
7.2. 实现属性组管理
7.3. 可编辑属表
8. 商品库存表
9. 国际化语言表
10. Workflow
11. 内容版本控制
12. logging 日志表的设计
13. uuid 替代传统序列 id
14. 动态配置表
14.1. 配置表历史记录
15. 验证码

16. 手机归属地数据库表
17. 数据检查
    17.1. 身份证校验
18. 创建与修改时间
19. 在线用户表
20. HTML TO Text
21. SNS 数据库设计
    21.1. people 表
    21.2. firend 表
    21.3. 演示
    21.4. network 表
22. 数据库与缓存
    22.1. 什么是数据库缓存?
    22.2. 为什么缓存数据呢?
    22.3. 什么时候使用数据库缓存
    22.4. 涉及缓存的地方有哪些
    22.5. 谁来控制数据库缓存
    22.6. 怎么控制数据库缓存
        22.6.1. SQL_CACHE 缓存
        22.6.2. 禁止缓存 SQL_NO_CACHE
        22.6.3. 关闭缓存 set session query_cache_type=off
23. PostgreSQL 所特有数据库设计
    23.1. 国家地区表的设计
    23.2. 话题讨论表的设计
    23.3. 账户表/余额表/消费储蓄表
24. 数据库并行访问控制
    24.1. 防止并行显示
25. Sharding
    25.1. horizontal
    25.2. vertical

25.3. 新闻数据库分表案例
26. 数据库安全
26.1. 数据库结构版本控制
26.1.1. 什么是数据库结构版本控制
26.1.2. 为什么要做数据库结构本版控制
26.1.3. 何时做数据库结构本版控制
26.1.4. 在哪里做数据库结构本版控制
26.1.5. 谁来负责数据库结构本版控制
26.1.6. 怎样做数据库结构本版控制
26.1.6.1. 安装脚本
26.1.6.2. 启动脚本，停止脚本
26.1.6.3. 查看历史版本
26.2. 保护表
26.3. 保护表字段
26.4. 时间一致性
26.5. 为数据安全而分库
26.6. 内容版本控制,撰改留痕
26.7. 数据库审计表
26.8. 用户/角色认证
26.9. Token 认证
26.10. 数据加密
26.10.1. AES_ENCRYPT / AES_DECRYPT
26.10.2. 加密字段
26.11. 开发加密插件开发
26.12. 数据区块链
26.13. 状态保护
26.14. 数据归档
27. MySQL 大数据操作注意事项
27.1. 关于 delete
27.2. 关于 update
27.3. 关于创建索引

27.4. 关于 OPTIMIZE
27.5. 关于切换引擎
27.6. 确保SELECT不被受阻
21. 参考例子
1. CMS 数据库设计
2. 数据属性例子
2.1. 布尔状态
2.2. 流状态
2.3. 商品属性
2.3.1. 鞋
2.3.2. 裤子
2.3.3. 服装
2.3.4. 内衣
2.3.5. 隐形眼镜
2.3.6. 戒指
2.4. 手机号码分配
2.5. 身份证
2.6. 银行卡
22. NoSQL OOD(Object-Oriented Design)
1. MongoDB
1.1. 配置表 config
1.2. 日志表
2. Cassandra
2.1. User And Profile
2.2. Category
2.3. Article
2.4. Product and ProductAttribute
2.5. Address
2.6. 练习
23. Spring Data 最佳实践
1. MySQL
1.1. 分类表

1.2. 为字段增加索引
1.3. 复合索引
1.4. 一对多实例
1.5. ManyToMany 多对多
1.6. 外键级联删除
2. MongoDB
2.1. 枚举定义
2.2. 日志表
2.3. 地址与定位
III. 运维篇
24. IDC
1. 网络设备配置管理与版本控制
1.1. 背景
1.2. 怎样实现网络设备配置管理
1.3. 总结
2. 机房迁移
2.1. 拓扑确立
2.2. 存储规划
2.2.1. RAID Disk Group 规划
2.2.2. 文件系统规划
2.2.3. 目录规划
2.3. 设备上架
2.4. 操作系统初始化
2.5. 服务器及运行环境
2.6. 部署应用程序
2.7. 监控系统
2.8. 日志中心
2.9. 测试
3. 网线怎样连接才合理
3.1. 单个硬件防火墙方案
3.1.1. 防火墙

3.1.2. 交换机
3.2. 双防火墙方案
3.3. 网卡
3.3.1. 内外隔离
3.3.2. 负载均衡
3.3.3. 交叉互联
3.3.4. 网络适配器
3.3.4.1. 常见网络适配器品牌
3.3.4.2. 1G 千兆以太网产品
3.3.4.3. 10G 万兆以太网产品
4. 记录思科路由器/防火墙/交换机日志
4.1. 开启日志
4.2. syslogd 服务器脚本
5. 影响网络流量的因素
5.1. 带宽
5.1.1. 防火墙带宽
5.1.2. 交换机带宽
5.1.2.1. 聚合端口
5.1.3. 服务器带宽
5.2. 会话数
5.2.1. 防火墙会话数
5.2.2. 服务器会话数
5.2.3. 应用服务器会话数
5.3. IO
5.3.1. 硬盘 HDD
5.3.2. 固态硬盘 SSD
5.3.3. 分布IO
5.3.4. FC SAN
5.3.5. iSCSI / FCoE
5.3.6. InfiniBand 或 RDMA
25. Server

1. Linux 系统安全与优化配置
    1.1. Openssh 安全配置
        1.1.1. 禁止root用户登录
        1.1.2. 限制SSH验证重试次数
        1.1.3. 禁止证书登陆
        1.1.4. 使用证书替代密码认证
        1.1.5. 图形窗口客户端记忆密码的问题
        1.1.6. 关闭 GSSAPI
        1.1.7. 禁止SSH端口映射
        1.1.8. IP地址限制
        1.1.9. 禁止SSH密码穷举
    1.2. Shell 安全
        1.2.1. .history 文件
        1.2.2. sudo 安全问题
        1.2.3. 临时文件安全
        1.2.4. 执行权限
    1.3. 防火墙
        1.3.1. 策略
        1.3.2. 防止成为跳板机
        1.3.3. 端口安全
        1.3.4. 封锁特定字符串
    1.4. Linux 系统资源调配
        1.4.1. /etc/security/limits.conf
        1.4.2. 关闭写磁盘I/O功能
    1.5. PAM 插件认证加固配置
        1.5.1. pam_tally2.so
        1.5.2. pam_listfile.so
        1.5.3. pam_access.so
        1.5.4. pam_wheel.so
2. Tomcat 安全配置与性能优化
    2.1. JVM

2.1.1. 使用 Server JRE 替代JDK。
2.1.2. JAVA_OPTS
2.1.3. java.security 优化
2.2. Tomcat 优化
2.2.1. maxThreads 连接数限制
2.2.2. 虚拟主机
2.2.3. 压缩传输
2.3. Tomcat 安全配置
2.3.1. 禁用8005端口
2.3.2. 安装后初始化配置
2.3.2.1. 隐藏版本信息
2.3.2.2. 应用程序安全
2.3.2.3. JSESSIONID
2.3.3. 启动用户与端口
2.4. 如何部署应用程序
2.5. 延伸阅读
3. PHP 安全与性能优化
3.1. Apache mod_php / php-fpm
3.1.1. 用户权限
3.1.1.1. Apache
3.1.1.2. Nginx / lighttpd + fastcgi
3.1.2. web server 版本信息
3.1.3. php_flag / php_admin_flag
3.1.4. 防止URL注入
3.2. php.ini
3.2.1. Magic quotes
3.2.2. 危险PHP函数
3.2.2.1. chdir()函数安全演示
3.2.3. 隐藏PHP版本信息
3.2.4. session名字可以泄露你的服务器采用php技术
3.2.5. 隐藏PHP出错信息

3.2.6. open_basedir 防止操作web环境意外文件目录
3.3. 开发于安全
3.3.1. 彻底解决目录于文件的安全
3.3.2. 目录访问控制
3.3.3. Session / Cookie安全
3.3.4. 注入安全
3.3.4.1. 禁止输出调试信息
3.3.4.2. 预防SQL注入攻击
3.3.4.3. SHELL 命令注入
3.4. 执行效率
3.4.1. timeout
3.4.1.1. mysql
3.4.2. 浏览器上传文件尺寸控制
3.5. 服务器版本信息
4. 环境安装模板化
4.1. 云主机初始化
4.2. CentOS 7 初始化
4.3. Nginx
4.4. Tomcat
4.5. Node.js
4.6. MongoDB
5. 时间同步
6. 邮件系统
6.1. Mailing List
7. TPC
8. IOPS (Input/Output Operations Per Second, pronounced i-ops)
9. rPerf
10. 磁盘规划
10.1. 物理隔离
10.2. 硬件逻辑卷隔离

11. Distributed File System(簇文件系统)
    11.1. FC 光纤存储
    11.2. 聚合文件系统
    11.3. 全局文件系统
    11.4. 负载均衡文件系统
    11.5. 网络块设备
    11.6. Storage 存储
        11.6.1. 存储种类
            11.6.1.1. Direct Attached Storage
            11.6.1.2. Network-attached storage
            11.6.1.3. Storage area network
                11.6.1.3.1. FC SAN
                11.6.1.3.2. IP SAN
                11.6.1.3.3. FCoE (Fibre Channel over Ethernet)
        11.6.2. RAID
            11.6.2.1. 缓存服务器
            11.6.2.2. Web 服务器
            11.6.2.3. 数据库
            11.6.2.4. 数据备份
        11.6.3. File System 文件系统
            11.6.3.1. Distributed File System(DFS)
        11.6.4. 数据访问协议
        11.6.5. 数据管理
            11.6.5.1. Share 共享
            11.6.5.2. Mirror 远程镜像同步
            11.6.5.3. 压缩与重复数据消除
            11.6.5.4. Backup 备份与恢复
            11.6.5.5. 故障报告
    11.7. 磁盘快照
12. iDRAC / iLO / IMM
26. Monitor solution

1. 网络监控
    1.1. 流量监控
    1.2. 交换机监控
2. 集群监控
3. 操作系统监控需求
    3.1. CPU 相关监控
    3.2. 磁盘相关监控
    3.3. 内存相关监控
    3.4. 网络监控
    3.5. 权限监控
    3.6. 进程相关监控
    3.7. 时间同步监控
    3.8. 文件系统与系统日志监控
4. 服务监控
    4.1. Nginx 监控
    4.2. Redis 监控
    4.3. Rabbit 监控
    4.4. Elasticsearch 监控
    4.5. 数据库监控需求
        4.5.1. 数据库监控指标
5. 网站安全与预警机制
6. 网络监控
    6.1. DNS解析监控
    6.2. 路由监控
    6.3. 流量监控
    6.4. 会话数监控
7. 内容监控
8. DHCP
    8.1. DHCP Server
9. Routing
    9.1. 策略路由

10. routing-table
11. Example
12. 监控方法
    12.1. 人工监控
    12.2. 机器监控
27. Backup
    1. help
        1.1. Task
        1.2. Schedule
        1.3. Crontab
    2. 配置文件备份
        2.1. Firewall and Switch
        2.2. Server
28. DIY Firewall & VPN
    1. Firewall
    2. 3 Layer VPN
    3. 7 Layer VPN
    4. 替代 CentOS 7/8 中的 firewalld
        4.1. Demo
        4.2. Firewall Script

IV. Software architecture (软件架构)

29. 前端架构
    1. Javascript Framework
30. Project
    1. 开源模式
    2. 开发语言及平台
        2.1. 分层架构
            2.1.1. 中间件 Middleware
            2.1.2. 分层
        2.2. Web 2.0
        2.3. 云计算
            2.3.1. 云计算的三种服务模式
        2.4. 跨平台
        2.5. 编译语言比脚本语言安全

2.6. 封装重用
2.7. 相关的工具
2.7.1. 开发工具
31. Framework Design
1. 开发框架 Framework
1.1. HMVC
1.2. REST
1.3. SNA (Shared Nothing Architecture)
1.4. 其他
2. MVC Framework Design (设计MVC框架)
2.1. HMVC Framework
3. REST
3.1. RESTful JSON API
3.2. Ajax 与 RESTful 跨域
4. Service-oriented architecture (SOA)
4.1. SOAP实现
4.2. MQ 实现
5. Dispatcher MVC核心分发器
5.1. URL设计
5.1.1. URL 作为MVC 的Controller
5.1.2. URL 伪静态化，用于SEO优化
5.2. Dispatcher 的实现方式
6. Plugin & Hook 设计与实现
6.1. 插件管理平台
6.2. 接口定义
6.3. 插件
6.4. 测试
7. Interface
7.1. 访问接口协议
7.2. 接口性能问题
7.3. 接口安全问题
7.3.1. 访问权限

8. 模板(template)
    8.1. HTML 页面优化
9. Session/Cookie
    9.1. Session
    9.2. Session 共享
    9.3. Cookie
        9.3.1. Cookie 安全
        9.3.2. cookie-free domains
        9.3.3. P3P
10. 国际化 Locale database。
    10.1. Unicode
11. 数据库访问
    11.1. CRUD
    11.2. Active Record
    11.3. OR Mapping
12. Cache
    12.1. 页面缓存
    12.2. 局部缓存
13. Single sign-on (SSO) 单点登录
14. 搜索引擎
15. Synchronous/Asynchronous
16. Message Queuing
17. Hash
18. Sharding 垂直/水平切割
    18.1. 面向服务
    18.2. 面向数据库
19. 日志系统
20. Cache
    20.1. CDN (Content Delivery Network) 与反向代理
        20.1.1. CDN接口API
        20.1.2. 方向代理页面过期处理
        20.1.3. 内容版本化
    20.2. HTTP Cache

20.2.1. 在程序中灵活操作 Cache-Control
20.2.2. 非程序文件缓存处理
20.3. Cache 生存时间
21. i18n 国际化
21.1. 数组方式
21.2. 数据库方式
21.3. 文件文件
21.4. Gettext
21.5. 数据结构
22. RSS / ATom
22.1. Atom
23. Logging 日志
23.1. 日志的格式
23.2. 日志存贮
23.2.1. 本地存储
23.2.2. 远程存储
23.3. Log4cpp/Log4j/Log2PHP
23.4. Remote Syslog
24. debug
25. 性能优化
25.1. 尽量使用单引号
26. Design pattern (设计模式)
26.1. Singleton 单件模式
27. AOP（Aspect Oriented Programming)
28. 信息安全
28.1. CSRF（Cross-site request forgery）跨站请求伪造
28.2. Session 撰改演示
28.3. 用户注册与登录安全
28.4. 目录文件与权限
28.4.1. 读写权限

28.4.2. 访问权限
28.5. 密码安全
28.6. 注入检查
28.7. 防止恶意刷新与重复提交
28.8. 屏蔽出错信息
28.8.1. 屏蔽php出错信息
29. 序列化
V. 设计与解决方案
32. 支付平台方案
1. 方案
1.1. 商户方案
1.2. 银行方案
2. 支付接口
3. 支付派台后台
33. 电子商务网站
1. Product
2. Cart & Checkout
2.1. 物流配送插件设计
3. 促销优惠组件设计
34. 微信公众平台
1. 微信公众平台原理
2. 微信公众平台通常提供的服务模式
3. 微信公众平台开发
A. 附录
术语表

插图清单

8.1. 分时方案
8.2. HA 高可用方案
8.3. 多路心跳方案

8.4. 任务抢占方案
8.5. 任务轮循或任务轮循+抢占排队方案
14.1. 单机房高可用双活互备解决方案
14.2. 双机房异地灾备方案
14.3. 三机房互备异地灾备方案
14.4. 动态页面方案
14.5. 数据源灾备解决方案
14.6. 数据库灾备解决方案
14.7. 双向通知解决方案
14.8. 消息对列解决方案
14.9. CVS开发框架
14.10. 传统的分表方案
14.11. 推荐的分表方案
14.12. 基于功能分表方案

表格清单

20.1. 工作流模拟

范例清单

2.1. example robots.txt
16.1. 发送短信
16.2. 处理图片
16.3. 身份证号码校验
16.4. 静态化案例
16.5. 数据同步案例
20.1. identity_card 身份证归属地表
20.2. 演示 SQL_CACHE
20.3. 演示 SQL_NO_CACHE
20.4. 演示 query_cache_type=off 关闭查询缓存

20.5. 递归查询实例 city 表
20.6. 话题讨论表的设计
25.1. /etc/pam.d/sshd - pam_tally2.so
25.2. /etc/pam.d/sshd - pam_listfile.so
26.1. dhcp vlan rip
27.1. Backup program
31.1. php language package
31.2. sql table language package
31.3. Example for ECSHOP

Netkiller Architect 手札

Multi-dimension Architecture Design & Implementation / Full Stack Architect

ISBN#

Mr. Neo Chan, 陈景峯(BG7NYT)

中国广东省深圳市望海路半岛城邦三期
518067
+86 13113668890

<netkiller@msn.com>

电子书最近一次更新于 2020-11-15 03:22:51 .

版权 © 2008-2019 Copyright Editor Groups, All Rights Reserved

版权声明

转载请与作者联系，转载时请务必标明文章原始出处和作者信息及本声明。

http://www.netkiller.cn
http://netkiller.github.io
http://netkiller.sourceforge.net

微信订阅号 netkiller-ebook
(微信扫描二维码)

QQ：13721218 请注明“读者”

QQ群：128659835 请注明“读者”

知乎专栏 | 多维度架构

2020-11-15 03:22:51

本电子书采用碎片化写作方式，所以没有截止时间，写作也很随意，内容不断填充，更新，章节不断扩展和调整。每年的年底我会推出 epub, Kindle mobi 等格式的电子书。

修订历史		
修订 1.0.0	Jun 29, 2020	Neo
计划改变，启动写作计划		
修订 0.1.1	Sep 12, 2011	Neo
章节做了大调整，将文档分为五块，多维度架构，开发，运维，SQA，还有 DevOps。		
修订 0.1.0	May 15, 2010	Neo
增加解决方案一节，并填充了大量章节。同时对完成这篇文档信心大增		
修订 0.0.4	2010	Neo
这篇文档几乎没有时间和精力编辑，内容增加不多。		
修订 0.0.4	April 15, 2009	Neo
这篇文档几乎搁浅，没有时间和精力，没有编辑加入。今天做了一下布局调整，增加一些内容。		
修订 0.0.3	Sep. 17, 2008	Neo

加入关于存储的内容		
修订 0.0.1	May 24, 2008	Neo
李振韬加入编译团队		
修订 0.0.0	May 22, 2008	Neo
这是一个值得纪念的日子		

自述

《Netkiller 系列 手札》是一套免费系列电子书，netkiller 是nickname 从1999 开使用至今，“手札” 是札记，手册的

2003年之前我还是以文章形式在BBS上发表各类技术文章，后来发现文章不够系统，便尝试写长篇技术文章加上章节目录等等。随

IT知识变化非常快，而且具有时效性，这样发布非常混乱，经常有读者发现第一版例子已经过时，但他不知道我已经发布第二版。

我便有一种想法，始终维护一个文档，不断更新，使他保持较新的版本不过时。

第一部电子书是《PostgreSQL 实用实例参考》开始我使用 Microsoft Office Word 慢慢随着文档尺寸增加 Word 开始表

我看到PostgreSQL 中文手册使用SGML编写文档，便开始学习Docbook SGML。使用Docbook写的第一部电子书是《Netkille

至于“手札”一词的来历，是因为我爱好摄影，经常去一个台湾摄影网站，名字就叫“摄影家手札”。

由于硬盘损坏数据丢失 《Netkiller Postfix Integrated Solution》 的 SGML文件已经不存在；Docbook SGML存在

目前技术书籍的价格一路飙升，动则￥80，￥100，少则￥50，￥60. 技术书籍有时效性，随着技术的革新或淘汰，大批书记成为废纸垃圾。并且这些书技术内容雷同，相互抄袭，质量越来越差，甚至里面给出的例子错误百出，只能购买影印版，或者翻译的版本。

在这种背景下我便萌生了自己写书的想法，资料主要来源是我的笔记与例子。我并不想出版，只为分享，所有我制作了基于CC License 发行的系列电子书。

本书注重例子，少理论（捞干货），只要你对着例子一步一步操作，就会成功，会让你有成就感并能坚持学下去，因为很多人遇到障碍就会放弃，其实我就是这种人，只要让他看到希望，就能坚持下去。

1. 写给读者

为什么写这篇文章

有很多想法,工作中也用不到所以未能实现，所以想写出来,和大家分享.有一点写一点,写得也不好,只要能看懂就行,就当学习笔记了.

开始零零碎碎写过一些文档，也向维基百科供过稿，但维基经常被ZF封锁，后来发现sf.net可以提供主机存放文档，便做了迁移。并开始了我的写作生涯。

这篇文档是作者20年来对工作的总结,是作者一点一滴的积累起来的，有些笔记已经丢失，所以并不完整。

因为工作太忙整理比较缓慢。目前的工作涉及面比较窄所以新文档比较少。

我现在花在技术上的时间越来越少，兴趣转向摄影，无线电。也想写写摄影方面的心得体会。

写作动力:

曾经在网上看到外国开源界对中国的评价，中国人对开源索取无度，但贡献却微乎其微.这句话一直记在我心中，发誓要为中国开源事业做我仅有的一点微薄贡献

另外写文档也是知识积累，还可以增加在圈内的影响力.

人跟动物的不同,就是人类可以把自己学习的经验教给下一代人.下一代在上一代的基础上再创新,不断积累才有今天.

所以我把自己的经验写出来,可以让经验传承

没有内容的章节:

目前我自己一人维护所有文档，写作时间有限，当我发现一个好主题就会加入到文档中，待我有时间再完善章节，所以你会发现很多章节是空无内容的.

文档目前几乎是流水帐试的写作，维护量很大，先将就着看吧.

我想到哪写到哪,你会发现文章没一个中心,今天这里写点,明天跳过本章写其它的.
文中例子绝对多,对喜欢复制然后粘贴朋友很有用,不用动手写,也省时间.
理论的东西,网上大把,我这里就不写了,需要可以去网上查.
我爱写错别字,还有一些是打错的,如果发现请指正.
文中大部分试验是在Debian/Ubuntu/Redhat AS上完成.

写给读者

至读者:

我不知道什么时候，我不再更新文档或者退出IT行业去从事其他工作，我必须给这些文档找一个归宿，让他能持续更新下去。

我想捐赠给某些基金会继续运转，或者建立一个团队维护它。

我用了20年时间坚持不停地写作，持续更新，才有今天你看到的《Netkiller 手扎》系列文档，在中国能坚持20年，同时没有任何收益的技术类文档，是非常不容易的。

有很多时候想放弃，看到外国读者的支持与国内社区的影响，我坚持了下来。

中国开源事业需要各位参与，不要成为局外人，不要让外国人说：中国对开源索取无度，贡献却微乎其微。

我们参与内核的开发还比较遥远，但是进个人能力，写一些文档还是可能的。

系列文档

下面是我多年积累下来的经验总结，整理成文档供大家参考:

Netkiller Architect 手札
Netkiller Developer 手札
Netkiller PHP 手札
Netkiller Python 手札
Netkiller Testing 手札
Netkiller Cryptography 手札
Netkiller Linux 手札
Netkiller FreeBSD 手札
Netkiller Shell 手札
Netkiller Security 手札
Netkiller Web 手札
Netkiller Monitoring 手札
Netkiller Storage 手札
Netkiller Mail 手札
Netkiller Docbook 手札
Netkiller Version 手札
Netkiller Database 手札
Netkiller PostgreSQL 手札
Netkiller MySQL 手札
Netkiller NoSQL 手札
Netkiller LDAP 手札
Netkiller Network 手札
Netkiller Cisco IOS 手札
Netkiller H3C 手札
Netkiller Multimedia 手札
Netkiller Management 手札
Netkiller Spring 手札
Netkiller Perl 手札
Netkiller Amateur Radio 手札

2. 作者简介

陈景峯 (ㄔㄣ ㄐㄧㄥ ㄈㄥ)

Nickname：netkiller | English name: Neo chen | Nippon name: ちんけいほう (音訳) | Korean name: 천징봉 | Thailand name: ภูมิภาพภูเขา | Vietnam: Trần Cảnh Phong

Callsign: BG7NYT | QTH: ZONE CQ24 ITU44 ShenZhen, China

程序猿，攻城狮，挨踢民工, Full Stack Developer, UNIX like Evangelist, 业余无线电爱好者（呼号：BG7NYT）,户外运动，山地骑行以及摄影爱好者。

《Netkiller 系列 手札》的作者

成长阶段

1981年1月19日(庚申年腊月十四)出生于黑龙江省青冈县建设乡双富大队第一小队

1989年9岁随父母迁居至黑龙江省伊春市，悲剧的天朝教育，不知道那门子归定，转学必须降一级，我本应该上一年级，但体制让我上学前班，那年多都10岁了

1995年小学毕业，体制规定借读要交3000两银子(我曾想过不升初中)，亲戚单位分楼告别平房，楼里没有地方放东西，把2麻袋书送给我，无意中发现一本电脑书BASIC语言，我竟然看懂了，对于电脑知识追求一发而不可收，后面顶零花钱，压岁钱主要用来买电脑书《MSDOS 6.22》《新编Unix实用大全》《跟我学Foxbase》。。。。。。

1996年第一次接触UNIX操作系统，BSD UNIX, Microsoft Xinux(盖茨亲自写的微软Unix，知道的人不多)

1997年自学Turbo C语言，苦于没有电脑，后来学校建了微机室才第一次使用QBASIC(DOS 6.22 自带命令)，那个年代只能通过软盘拷贝转播，Trubo C编译器始终没有搞到，

1997年第一次上Internet网速只有9600Bps, 当时全国兴起各种信息港域名格式是www.xxxx.info.net, 访问的第一个网站是NASA下载了很多火星探路者拍回的照片，还有“淞沪”sohu的前身

1998~2000年在哈尔滨学习计算机，充足的上机时间，但老师让我们练打字（明伦五笔/WT）打字不超过80个/每分钟还要强化训练，不过这个给我的键盘功夫打了好底。

1999年学校的电脑终于安装了光驱，在一张工具盘上终于找到了Turbo C, Borland C++与Quick Basic编译器，当时对VGA图形编程非常感兴趣，通过INT33中断控制鼠标，使用绘图函数模仿windows界面。还有操作 UCDOS 中文字库，绘制矢量与点阵字体。

2000年沉迷于Windows NT与Back Office各种技术，神马主域控制器，DHCP，WINS，IIS，域名服务器，Exchange邮件服务器，MS Proxy, NetMeeting...以及ASP+MS SQL开发；用56K猫下载了一张LINUX。ISO镜像，安装后我兴奋的24小时没有睡觉。

职业生涯

2001 年来深圳进城打工,成为一名外来务工者. 在一个4人公司做PHP开发，当时PHP的版本是2.0, 开始使用Linux Redhat 6.2.当时很多门户网站都是用FreeBSD,但很难搞到安装盘，在网易社区认识了一个网友,从广州给我寄了一张光盘，FreeBSD 3.2

2002 年我发现不能埋头苦干,还要学会"做人".后辗转广州工作了半年，考了一个Cisco CCNA认证。回到深圳重新开始，在车公庙找到一家工作做Java开发

2003 年这年最惨,公司拖欠工资16000元,打过两次官司2005才付清.

2004 年开始加入分布式计算团队,目前成绩，工作仍然是Java开发并且开始使用PostgreSQL数据库。

2004-10月开始玩户外和摄影

2005-6月成为中国无线电运动协会会员,呼号BG7NYT,进了一部Yaesu FT-60R手台。公司的需要转回PHP与MySQL，相隔几年发现PHP进步很大。在前台展现方面无人能敌，于是便前台使用PHP，后台采用Java开发。

2006 年单身生活了这么多年,终于找到归宿. 工作更多是研究PHP各种框架原理

2007 物价上涨,金融危机，休息了4个月（其实是找不到工作）,关外很难上439.460中继，搞了一台Yaesu FT-7800.

2008 终于找到英文学习方法， 《Netkiller Developer 手札》，《Netkiller Document 手札》

2008-8-8 08:08:08 结婚,后全家迁居湖南省常德市

2009《Netkiller Database 手札》,2009-6-13学车，年底拿到C1驾照

2010 对电子打击乐产生兴趣，计划学习爵士鼓。由于我对Linux热爱，我轻松的接管了公司的运维部，然后开发运维两把抓。我印象最深刻的是公司一次上架10个机柜，我们用买服务器纸箱的钱改善

伙食。我将40多台服务器安装BOINC做压力测试，获得了中国第二的名次。

2011 平凡的一年，户外运动停止，电台很少开，中继很少上，摄影主要是拍女儿与家人，年末买了一辆山地车

2012 对油笔画产生了兴趣，活动基本是骑行银湖山绿道，

2013 开始学习民谣吉他，同时对电吉他也极有兴趣；最终都放弃了。这一年深圳开始推数字中继2013-7-6日入手Motorola MOTOTRBO XIR P8668，Netkiller 系列手札从Sourceforge向Github迁移；年底对MYSQL UDF，Engine与PHP扩展开发产生很浓的兴趣，拾起遗忘10+年的C，写了几个mysql扩展（图片处理，fifo管道与ZeroMQ），10月份入Toyota Rezi 2.5V并写了一篇《攻城狮的苦逼选车经历》

2014-9-8 在淘宝上买了一架电钢琴 Casio Privia PX-5S pro 开始陪女儿学习钢琴，由于这家钢琴是合成器电钢，里面有打击乐，我有对键盘鼓产生了兴趣。

2014-10-2号罗浮山两日游，对中国道教文化与音乐产生了兴趣，10月5号用了半天时间学会了简谱。10月8号入Canon 5D Mark III + Canon Speedlite 600EX-RT香港过关被查。

2014-12-20号对乐谱制作产生兴趣（https://github.com/SheetMusic/Piano），给女儿做了几首钢琴伴奏曲，MuseScore制谱然后生成MIDI与WAV文件。

2015-09-01 晚饭后拿起爵士鼓基础教程尝试在Casio Privia PX-5S pro演练，经过反复琢磨加上之前学钢琴的乐理知识，终于在02号晚上，打出了简单的基本节奏，迈出了第一步。

2016 对弓箭（复合弓）产生兴趣，无奈天朝法律法规不让玩。每周游泳轻松1500米无压力，年底入 xbox one s 和 Yaesu FT-2DR, 同时开始关注功放音响这块

2017 7月9号入 Yamaha RX-V581 功放一台，连接Xbox打游戏爽翻了，入Kindle电子书，计划学习蝶泳，果断放弃运维和开发知识体系转攻区块链。

2018 从溪山美地搬到半岛城邦，丢弃了多年攒下的家底。11 月开始玩 MMDVM，使用 Yaesu FT-7800 发射，连接MMDVM中继板，树莓派，覆盖深圳湾，散步骑车通联两不误。

2019 卖了常德的房子，住了5次院，哮喘反复发作，决定停止电子书更新，兴趣转到知乎，B站

2020 准备找工作

职业生涯路上继续打怪升级

3. 如何获得文档

下载 Netkiller 手札 (epub,kindle,chm,pdf)
EPUB https://github.com/netkiller/netkiller.github.io/tree/master/download/epub
MOBI https://github.com/netkiller/netkiller.github.io/tree/master/download/mobi
CHM https://github.com/netkiller/netkiller.github.io/tree/master/download/chm

通过 GIT 镜像整个网站

https://github.com/netkiller/netkiller.github.com.git

$ git clone https://github.com/netkiller/netkiller.github.com.git

镜像下载

整站下载

```
wget -m http://www.netkiller.cn/index.html
```

指定下载

```
wget -m wget -m http://www.netkiller.cn/linux/index.html
```

Yum 下载文档

获得光盘介质，RPM包，DEB包，如有特别需要，请联系我

YUM 在线安装电子书

http://netkiller.sourceforge.net/pub/repo/

```
# cat >> /etc/yum.repos.d/netkiller.repo <<EOF
[netkiller]
name=Netkiller Free Books
```

```
baseurl=http://netkiller.sourceforge.net/pub/repo/
enabled=1
gpgcheck=0
gpgkey=
EOF
```

查找包

```
# yum search netkiller

netkiller-centos.x86_64 : Netkiller centos Cookbook
netkiller-cryptography.x86_64 : Netkiller cryptography Cookbook
netkiller-docbook.x86_64 : Netkiller docbook Cookbook
netkiller-linux.x86_64 : Netkiller linux Cookbook
netkiller-mysql.x86_64 : Netkiller mysql Cookbook
netkiller-php.x86_64 : Netkiller php Cookbook
netkiller-postgresql.x86_64 : Netkiller postgresql Cookbook
netkiller-python.x86_64 : Netkiller python Cookbook
netkiller-version.x86_64 : Netkiller version Cookbook
```

安装包

```
yum install netkiller-docbook
```

4. 打赏（Donations）

If you like this documents, please make a donation to support the authors' efforts. Thank you!

您可以通过微信，支付宝，贝宝给作者打赏。

银行(Bank)

招商银行(China Merchants Bank)

开户名：陈景峰

账号：9555500000007459

微信（Wechat）

微信支付

支付宝（Alipay）

PayPal Donations

https://www.paypal.me/netkiller

5. 联系方式

主站 http://www.netkiller.cn/

备用 http://netkiller.github.io/

繁体网站 http://netkiller.sourceforge.net/

联系作者

Mobile: +86 13113668890

Email: netkiller@msn.com

QQ群: 128659835 请注明“读者”

QQ: 13721218

ICQ: 101888222

注：请不要问我安装问题！

博客 Blogger

知乎专栏 https://zhuanlan.zhihu.com/netkiller

LinkedIn: http://cn.linkedin.com/in/netkiller

OSChina: http://my.oschina.net/neochen/

Facebook: https://www.facebook.com/bg7nyt

Flickr: http://www.flickr.com/photos/bg7nyt/

Disqus: http://disqus.com/netkiller/

solidot: http://solidot.org/~netkiller/

SegmentFault: https://segmentfault.com/u/netkiller

Reddit: https://www.reddit.com/user/netkiller/

Digg: http://www.digg.com/netkiller

Twitter: http://twitter.com/bg7nyt

weibo: http://weibo.com/bg7nyt

Xbox club

我的 xbox 上的ID是 netkiller xbox，我创建了一个俱乐部 netkiller 欢迎加入。

Radio

CQ CQ CQ DE BG7NYT:

如果这篇文章对你有所帮助,请寄给我一张QSL卡片, qrz.cn or qrz.com or hamcall.net

Personal Amateur Radiostations of P.R.China

ZONE CQ24 ITU44 ShenZhen, China

Best Regards, VY 73! OP. BG7NYT

守听频率 DMR 438.460 -8 Color 12 Slot 2 Group 46001

守听频率 C4FM 439.360 -5 DN/VW

MMDVM Hotspot:

Callsign: BG7NYT QTH: Shenzhen, China

YSF: YSF80337 - CN China 1 - W24166/TG46001

DMR: BM_China_46001 - DMR Radio ID: 4600441

部分 I. 多维度架构

Multi-dimension Architecture

第 1 章 什么是多维度架构

1. 架构与格局

俗话说：烙饼再大，也大不过烙饼的锅。一个人的成就再大，也大不过他的格局。

格局就是具备高视点，宽视野，深洞察，能够跨越时间和空间去看事物，同时不受思维定势的限制。

思维定势又称“习惯性思维”，是指人们按习惯的、比较固定的思路去考虑问题、分析问题，表现为在解决问题过程中作特定方式的加工准备。它阻碍了思维开放性和灵活性，造成思维的僵化和呆板。这使得人们不能灵活运用知识，创造性思维的发展受到阻碍。

人的思维定势会被经历，职业，圈子，财富，出身等很多因素所困扰。

2. 架构师的大局观

我们从小的教育就是如何拆分问题、解决问题，这样做显然会使复杂的问题变得更容易些。但是这带来一个新问题，我们丧失了如何从宏观角度看问题，分析问题，解决问题，对更大的整体的内在领悟能力。这导致了我们对现有问题提出的解决方案，但无法预计实施该方案后产生的各种后果，为此我们付出了巨大代价。

而我们试图考虑大局的时候，总要在脑子里重新排序，组合哪些拆分出来问题，给它们编组列单。习惯性认为解决了所有微观领域的问题，那么宏观上问题就得到了解决。然而，这种做法是徒劳无益的，就好比试图通过重新拼起来的碎镜子来观察真实的影像。

所以在一段时间后，我们便干脆完全放弃了对整体的关注。

当今的社会，几乎所有的企业情况都是岗位职责清晰，分工明确，员工是企业机器上的一颗螺丝钉，我们在招聘下属的时候也仅仅是用他的一技之长。项目一旦立项，我们就根据项目需求针对性性的招聘，短短半年团队就会膨胀数倍，但效率并不是成正比增长。另一个问题是这个庞大的团队合作起来并不尽人意。结果是 80% 协调的时间，20% 实际工作时间。

很多技术人员埋头在自己的专业领域，更擅长解决自己专业范畴的问题，这样是不对的，技术人员应该培养广泛的兴趣，不仅仅要成为技术全栈，还要涉略艺术等领域。

3. Architecture Overview

Overall structure

4. CDN (Content Delivery Network)

I analyzed CDN realization principle, look at the picture below.

5. 微服务

Public Cloud
HTML5
APP
IoT
OSS
Images
CSS
Javascript
Audio
Video
CDN
text/html
text/plain
text/css
javascript
Cloud Firewall
IPS
DDOS
Nginx
HTTP2
Rewrite
Exapires
Compression
Microservice core
Config Server
Service Registry
Gateway
Load Banlancing
Route
Auth
Microservice Application (Kubernetes Container)
Microservice 1
Microservice 2
Microservice 3
Microservice 4
Private Container
Message Queue
Kafka
RabbitMQ
Monitor
Logging
System
Network
Hardware
Could Database
Redis
MySQL
MongoDB
Blockchina
TSDB
Elasticsearch
http://www.netkiller.cn
Cloud Disk Storage
Design by netkiller
Data
Backup
Snapshot

6. 日志

Application
Java
PHP
Ruby
Python
DotNet
Asynchronous （API）
TCP
Design by neo
netkiller@msn.com
UDP
http://netkiller.sourceforge.net/
http://netkiller.github.com/
Pipeline
Logging daemon (server side)
Fork process
Socket
Threading
accept
accept
accept
accept
Buffer
Pipeline
Buffer
Log
Queue
Log Filter
Accept
Drop
Alert
Event
Email
SMS
Storage engine
DB Driver
SQL
LDAP
NOSQL
mdb/bdb
File Driver
gzip
Rotate logs
txt
csv
xml
TCP
Console (client side)
Synchronous Monitor
Asynchronous Analysis

VT100 Terminal
Schedule
Data mining
HTML Report
PDF Report

7. 监控

IDC MGR
command
user
add
del
lock
unlock
permission
ntpdate
clear
logs
cache
tmp
os
start
stop
restart
app
apache
start
stop
restart
mysql
start
stop
restart
monitor
network
bandwidth
traffic
ping
operating system
hardware
power
raid
fan
application
user
login
register
profile
shopping
cart
pay
order
cms
comment
click
manager
batch
single
group
all
slb
pxy
www
cch
app
db
alert
level
warning
critical
sound
sms
email
protolcol
ipmi
snmp
agent

IT Infrastructure Monitoring Framework
Outside Monitoring
Port scan
DNS
Ping
Traceroute
Workflow Monitoring
Page
User Login
Checkout
Search
Inside Monitoring
Application Server
Application
Interface
Module
API/SDK
Agent Client
Shell
Command
/proc
dmesg
Operating system
Monitoring Daemon
Address
Firewall
Route
Switch
Server
Group
User
Host
Service
Service
HTTP
IGMP
SSH
MYSQL
ORACLE
HTTPS
SMTP
POP
IMAP
Auth Plugin
OpenLDAP
Shadow
NIS
MySQL
User Profile
Username
Email
Cellphone
Jabber / IM
Diagnosis
SNMP
IPMI
Script
Alert
Email
SMS
Jabber
Sound
Logging
Alert Queue
Graph
RRD
Gnuplot
Agent Server (data collection)
SNMP
Network
Interface
Traffic
System Infomation
CPU
Memory
HDD
Load Average
IO
IPMI
System Event Log
FAN
Temperature
Power Supply
Voltage
Ampere
RAID
Volume Group
HDD
RAID Levels

第 2 章 多维度架构之网站HTML

```
                                     .---> media [mp3, wma, wmv, rmvb, asf, divx]-\
                                    /                                              +------------+
                                   .-----> photo [gif, jpg, png, swf] ----> | Raid Array | <--.
    /------------------- <---------\/                                              +------------+
user -> dns -> load balancing -> squid -> [cache] <----[html]----\                  /
                  \ \ \______<______/\                      +-------------+         /
                   \ \                \-----> web app ----> | html        |--------^
                    \ \_____________________________/\       | php,jsp,cgi |
                     \                                \      +-------------+
                      \                                `-----> memcached [node1, node2, node(n)]
                       \_____________________________________________/\
                                                                      `------> Database cluster
    +--------------------------------------+                                        \
    | Author: neo chen <openunix#163.com>  |                                         `-------------
    | Nickname: netkiller                  |
    | Homepage: http://netkiller.github.io |
    +--------------------------------------+
```

1. 网站的历史演变

1.1. 常用软硬件组成

一个网站由下面几部分组成:

网站组成

1. 硬件 hardware
2. 操作系统 operation system
3. 应用软件 application software
4. 网站程序以及开发语言 web program and script

硬件包括

1. 网络硬件（路由器 route, 交换机 switch)
2. 服务器 server
3. KVM over IP
4. 其他包括，机柜等

操作系统包括

1. Windows Server
2. Linux
3. FreeBSD

4. Other(Sun,Novell,Sco...)

其中应用软件按平台分类

Windows 应用软件包括

1. dns (dns)
2. web (IIS)
3. ftp (IIS)
4. mail (Exchange)
5. database (SQL Server)
6. ldap (Active Directory)

Unix like 应用软件包括

1. dns (bind)
2. web (apache, lighttpd, tomcat)
3. ftp (proftpd, pureftpd, wu-ftp, vsftpd)
4. mail (sendmail,postfix, qmail)
5. database (PostgreSQL, MySQL)
6. ldap (OpenLDAP)

其他应用软件包括

1. cache (squid, nginx, memcached...)
2. web (jboss, weblogic...)
3. database (Oracle, DB2...)

网站程序以及开发语言

1. php
2. java (jsp)
3. .net (aspx)
4. fastcgi (python,perl,rubby,c/c++ ...)

怎样定义多大的网站叫大型网站呢？我也不知道，但凡大型网站都具备本文所提的几点。

门户网站的需求

1. 海量用户访问
2. 海量用户存储

3. 国内外互通及南北互通

4. 快速响应

5. 7×24不间断运行

6. 易于维护

门户网站的几个技术要点

1. 智能域名服务器 Smart DNS

2. 集群负载均衡 Cluster

3. 缓存技术 Cache

4. 静态化

5. 图片服务器分离

6. 压缩数据传输

7. 时间同步

8. 数据存储

1.2. 第一代纯静态网站

ftp/rsync 同步

1.3. 第二代纯文本文件采用分隔符做数据存储网站

这是出现了cgi/isapi/nsapi技术,perl 成为主流

1.4. 第三代数据库存储网站

出现动态脚本语言如php,asp,coldfusion，这个时期 php成为主流

出现所见即所得网页工具

256色gif 动画

1.5. 第四代DNS负载均衡加反向代理

开发语言则百花齐放php,asp.net,java,三大主流三足鼎立。perl慢慢没落。

flash 动画

1.6. 第五代负载均衡集群

ajax, css+div, xhtml

2. 集群(Cluster)

集群有很多实现方法，分为硬件和软件，集群可以在不同网络层面上实现

1. 实现IP轮循（Bind DNS）
2. 硬件四层交换（硬件负载均衡设备 F5 BIG IP）
3. 软件四层交换（linux virtual server）
4. 应用层上实现（tomcat）

越是低层性能越好，越是上层功能更强

集群的分类

1. 高可用性集群
2. 负载均衡集群
3. 超级计算集群

网站一般用到两种集群分别是高可用性集群和负载均衡集群

2.1. 负载均衡

2.1.1. DNS负载均衡

这是早期出现的负载均衡技术，直到现在，很多网站仍然使用DNS负载均衡。

你可通过ping命令观看它是如何工作的，例如你可反复ping个网域名。

```
C:\>ping www.163.com

Pinging www.cache.split.netease.com [220.181.28.52] with 32 bytes of data:

Reply from 220.181.28.52: bytes=32 time=226ms TTL=53
Reply from 220.181.28.52: bytes=32 time=225ms TTL=53
Reply from 220.181.28.52: bytes=32 time=226ms TTL=53
```

```
Reply from 220.181.28.52: bytes=32 time=226ms TTL=53

Ping statistics for 220.181.28.52:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 225ms, Maximum = 226ms, Average = 225ms

C:\>ping www.163.com

Pinging www.cache.split.netease.com [220.181.28.53] with 32 bytes of data:

Reply from 220.181.28.53: bytes=32 time=52ms TTL=52
Reply from 220.181.28.53: bytes=32 time=53ms TTL=52
Reply from 220.181.28.53: bytes=32 time=52ms TTL=52
Reply from 220.181.28.53: bytes=32 time=52ms TTL=52

Ping statistics for 220.181.28.53:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 52ms, Maximum = 53ms, Average = 52ms

C:\>ping www.163.com

Pinging www.cache.split.netease.com [220.181.28.50] with 32 bytes of data:

Reply from 220.181.28.50: bytes=32 time=51ms TTL=53
Reply from 220.181.28.50: bytes=32 time=52ms TTL=53
Reply from 220.181.28.50: bytes=32 time=52ms TTL=53
Reply from 220.181.28.50: bytes=32 time=51ms TTL=53

Ping statistics for 220.181.28.50:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 51ms, Maximum = 52ms, Average = 51ms

C:\>
```

DNS负载均衡主要优点

1. 技术简单，容易实现，灵活，方便，成本低
2. Web服务器可以位于互联网的任意位置上，无地理限制。
3. DNS的主从结构非常稳定
4. 可以有效的分散DDOS攻击。

5. 你甚至可以在DNS服务商那里实现，自己不需要添加设备。而且没有带宽开销。

DNS负载均衡主要缺点

1. DNS负载均衡采用的是简单的轮循负载算法，不能够按照服务器节点的处理能力分配负载。

2. 不支持故障转移(failover)和自动恢复failback ,如果某台服务器拓机，DNS仍会将用户解析到这台故障服务器上，导致不能响应客户端。

3. 如果添加节点或撤出节点，不能即时更新到省市级DNS,可导致部分地区不能访问。

4. 占用大量静态IP。

2.1.2. 软件四层交换负载均衡

软件四层交换负载均衡为我们解决了几个问题

1. 能够按照服务器节点的处理能力分配负载。

2. 支持故障转移(failover)和自动恢复failback ,如果某节点拓机，调度器自动将它剔除，不响应客户端访问，当节点故障排除调度器立即恢复节点。

3. 可以随时添加节点或撤出节点，即时生效,方便网站扩容。

软件四层交换负载均衡优点

1. 仅仅需要一个静态IP。

2. 节点位于私有网络上与WAN隔离，用户面对的只是调度器。

3. 可以随时添加节点或撤出节点。

4. 通过端口可以组建多个集群。

2.1.3. 应用层负载均衡

Tomcat balancer

mod_proxy_balancer.so ,tomcat mod_jk.so

MySQL proxy / MySQL-LB

2.2. 高可用性集群

俗称：双机热备份

关键词：心跳线

两部服务器，或多部服务器，形成一个集群，当主服务器崩溃是，立即切换到其它节点上。

两部服务器要做到，内容实时同步，保持数据一直。

一般用 heartbeat + DRBD 实现。heartbeat负责切换服务器，DRBD用于同步数据。

2.3. 负载均衡设备

负载均衡成熟产品

1. F5 Big IP
2. Array

这些设备可提供3,4,7层负载均衡HA，硬件已经压缩，HTTP头改写，URL改写...

其中3层交换部分多采用硬件实现。

更多关于F5 与 Array 资料点击进入

2.4. 会话保持

2.5. 健康状态检查

3. 缓存技术

首先要说明，很多缓存技术依赖静态化。下面展示了缓存可能出现的位置。

用户user -> 浏览器缓存 IE/Firefox Cache -> 逆向代理缓存 Reverse proxy Cache -> WEB服务器缓存 Apache cache -> 应用程序缓存 php cache -> 数据库缓存 database cache

当然交换机，网络适配器，硬盘上也有Cache 但这不是我们要讨论的范围。

缓存存储方式主要是内存和文件两种，后者是存于硬盘中。

网站上使用的缓存主要包括五种：

1. 浏览器 缓存
2. 逆向代理/CDN缓存
3. WEB服务器缓存
4. 应用程序缓存
5. 数据库缓存

将上面的缓存合理地，有选择性的使用可大大提高网站的访问能力。

总之，想让你的网站更快，更多并发，答案是cache,cache 再 cache

3.1. 浏览器缓存

3.1.1. Expires

只要向浏览器输出过期时间HTTP协议头，不论是html还是动态脚本，都能被缓存。

HTML META

```
<meta http-equive="Expires" content=" Mon, 10 Jan 2000 00:00:00 GMT"/>
<meta http-equive="Cache-Control" content="max-age=300"/>
<meta http-equive="Cache-Control" content="no-cache"/>
```

动态脚本

```
Expires: Mon, 10 Jan 2000 00:00:00 GMT
Cache-Control: max-age=300
Cache-Control: no-cache

header("Expires: " .gmdate ("D, d M Y H:i:s", time() + 3600 * 24 * 7). " GMT");
header("Cache-Control: max-age=300");
header("Cache-Control: no-cache");
```

很多web server都提供 Expires 模块

提示

有些浏览器可能不支持。

3.1.2. If-Modified-Since / Last-Modified

If-Modified-Since 小于 Last-Modified 返回 200

```
neo@neo-OptiPlex-780:/tmp$ curl -I http://www.163.com/
HTTP/1.1 200 OK
Server: nginx
Content-Type: text/html; charset=GBK
Transfer-Encoding: chunked
Vary: Accept-Encoding
Expires: Mon, 16 May 2011 08:12:05 GMT
Cache-Control: max-age=80
Vary: User-Agent
Vary: Accept
Age: 38
X-Via: 1.1 ls100:8106 (Cdn Cache Server V2.0), 1.1 lydx156:8106 (Cdn Cache Server V2.0)
Connection: keep-alive
Date: Mon, 16 May 2011 08:11:23 GMT
```

If-Modified-Since 大于 Last-Modified 返回 304

```
neo@neo-OptiPlex-780:/tmp$ curl -H "If-Modified-Since: Fri, 12 May 2012 18:53:33 GMT" -I http://w
HTTP/1.0 304 Not Modified
Content-Type: text/html; charset=GBK
Cache-Control: max-age=80
Age: 41
X-Via: 1.0 ls119:80 (Cdn Cache Server V2.0), 1.0 lydx154:8106 (Cdn Cache Server V2.0)
Connection: keep-alive
Date: Mon, 16 May 2011 08:11:14 GMT
Expires: Mon, 16 May 2011 08:11:14 GMT
```

3.1.3. ETag / If-None-Match

```
neo@neo-OptiPlex-780:/tmp$ curl -I http://images.example.com/test/test.html
HTTP/1.1 200 OK
Cache-Control: s-maxage=7200, max-age=900
Expires: Mon, 16 May 2011 09:48:45 GMT
Content-Type: text/html
Accept-Ranges: bytes
ETag: "1984705864"
Last-Modified: Mon, 16 May 2011 09:01:07 GMT
Content-Length: 22
Date: Mon, 16 May 2011 09:33:45 GMT
Server: lighttpd/1.4.26
```

```
neo@neo-OptiPlex-780:/tmp$ curl -H 'If-None-Match: "1984705864"' -I http://images.example.com/tes
```

```
HTTP/1.1 304 Not Modified
Cache-Control: s-maxage=7200, max-age=900
Expires: Mon, 16 May 2011 09:48:32 GMT
Content-Type: text/html
Accept-Ranges: bytes
ETag: "1984705864"
Last-Modified: Mon, 16 May 2011 09:01:07 GMT
Date: Mon, 16 May 2011 09:33:32 GMT
Server: lighttpd/1.4.26
```

3.2. CDN/逆向代理缓存

具有代表性的逆向代理服务器：

1. Squid
2. Nginx
3. Varnish
4. Apache cache module

其它逆向代理服务器

1. 一些提供cache的硬件设备
2. 最近几年出现了的 China Cache 服务商，也称CDN

很多CDN厂商使用Squid 二次开发做为CDN节点，通过全球负载均衡使用分发

这些CDN厂商主要做了一下二次开发

1. logs 日志集中
2. 流量限制
3. push,pull操作
4. url 刷新

s-maxage 与 max-age用法类似，s-maxage针对代理服务器缓存。同样适用于CDN

s-maxage 与 max-age 组合使用可以提高CDN性能

3.3. 负载均衡设备

F5 Big-IP, Array 等设备都提供硬件加速，其原理与squid, apache提供的功能大同小异

其中Array 页面压缩采用硬件压缩卡实现，SSL加速也采用硬件实现

3.4. WEB服务器缓存

例如，通过配置apache实现自身 cache

3.5. 应用程序缓存

在这个领域百花齐放，相信你一定能找到适合你的。这些cache会为你提供一些api，来访问它。

代表性的 memcached 据我所是sina广泛使用，腾讯也曾经使用过后来开发了TC(Tencent Cache)，台湾雅虎则使用APC Cache。

另外模板引擎也有自己的缓存系统

3.6. 数据库缓存

数据库本身就有这个配置选项，如果需要你仍然可以在数据库前面加一道Cache。

例如PostgreSQL, MySQL 都提供参数可以将memcached编译到它内部

4. 网站静态内容出版

4.1. 架构总览

www 负责静态文件浏览, 台数不定, 可以采用零成本的DNS轮询, 或者4层LVS, 或者7层HAProxy, 还可以用F5, Array 等负载均衡设备.

cms 负责静态文件生成. 生成后的文件要同步到www中, 或者采用网络共享, 再者使用分布式文件系统, 总之将生成的文件交给www服务器, 根据你压力横向扩展即可

img 负责图片文件浏览. 通过给图片加版本号, 结局图片更新问题, 这样更新网站不用频繁刷新 CDN

这里不谈论负载均衡, 以及存储方案, 有情绪可以延伸阅读: http://netkiller.github.com/architect/index.html

你掌握了这个方案, 你可以很容易实现 向"京东商城", "VANCL凡客诚品", "走秀网" 这样的网站

这些网站的特点是: 浏览量大, 数据存储主要是图片, 登录/注册与购物车,用户中心 访问量只占到 5% 使用负责均衡很容易解决.

静态化网站可不避免的使用ajax做局部更新, ajax请求也要考虑缓存问题

静态化另一个目的是改善SEO

首次访问服务器

1. 访问www服务器
2. nginx 判断文件是否存在,如果存在将文件显示出来
3. 如果文件不存在,去cms服务器上查找, 如果存在便返回给www服务器,并显示出来
4. 如果cms上文件不存在,cms服务器便使用rewrite生成该文件, 同时将内容返回给www服务器,www将内容缓存在自己的服务器上,并将内容显示出来

第二次访问

1. 访问www服务器
2. nginx 判断文件是否存在,如果存在将文件显示出来
3. 如果文件不存在,去cms服务器上查找, 如果存在便返回给www服务器,并显示出来
4. 如果cms上文件不存在,cms服务器便使用rewrite生成该文件, 同时将内容返回给www服务器,www将内容缓存在自己的服务器上,并将内容显示出来

4.2. 静态化实现手段有哪些?

静态化方法包括:

1. 生成方式

2. 抓取方式

3. 伪静态化

4. 混合方式

4.2.1. 生成方式

主要由程序实现

例如

```
content = "<html><title>my static</title><body>hello world</body></html>"
file = open( your static file)
file.write(content)
file.close()
```

4.2.2. 抓取方式

主要由程序实现

程序中抓取

```
content = get_url('http://netkiller.8800.org/index.php')
file = open( index.html)
file.write(content)
file.close()
```

使用软件抓取，不仅限于wget。

```
wget http://netkiller.8800.org/index.php -O index.html
```

这时只给出简单例子，使用复杂参数实现更复杂的拾取，然后将脚本加入crontab中可。

4.2.3. 伪静态化

伪静态化是主要是通过在URL上做一些手脚，使你看去是静态的，实质上它是动态脚本。

伪静态化实现主要包括两种方法：

1. Rewrite rule

2. path_info

下面是一个PATH_INFO例子

http://netkiller.8800.org/zh-cn/photography/browse/2009.html

根本就不存在这个目录'zh-cn/photography/browse/'和文件'2009.html'

下面是一个Rewrite例子

http://example.org/bbs/thread-1003872-1-1.html

4.2.4. 混合方式

其实目前网站使用的基本上都是上面几种方法混合方式。

例如首先将动态url(example.org/news.php?cid=1&id=1) 通过rewrite转换为 (example.org/new_1_1.html)

接下来就比较容易解决了，一种方法是使用wget example.org/new_1_1.html，另一种方法你无需静态化，直接使用squid规则配置让他永不过期

4.2.5. 静态化中的动态内容

在静态化页面中有一些内容是无法实现静态的。像登录信息，用户评论等等

我们用三种方法实现静态中嵌入动态内容:

1. iframe - 灵活性差
2. SSI - 消耗web服务器资源
3. Ajax - 依赖浏览器，稳定性差

4.3. cdn

如何使用 cdn 来缓存你的网站内容

让你的网页缓存在 cdn 节点上的方式有下面几种

1. 让cdn的客服帮你配置缓存的规则, 他们很喜欢一刀切, 例如所有html都缓存2小时
2. 在他们管理后台自行使用正则配置缓存的时间, 这个他们一般不会提供, 某些公司的CDN会提供这个功能. 非常方便.
3. 通过HTTP头自行控制缓存时间, 一般是使用 max-age / s-maxage / Last-Modified 判断缓存时间

我比较喜欢最后一种, 通常我们使用max-age 与 s-maxage 同时使用, 这样我可以按照我的意向来决定文件的缓存时间 这里有更详细的解释说明.

4.4. www 服务器

下面给出一个精简后的配置例子

如果文件不存在就会连接后端cms服务器生成文件,并且显示出来,同时加上缓存. 生成的文件会从cms中同步到www服务器上.

你可以采用

1. rsync同步方案
2. nfs/samba/gluster 共享方案
3. iscsi 共享存储方案
4. 分布式文件系统方案

参考阅读: [分布式文件系统](), [Netkiller Linux Storage 手札]()

```
upstream  cms.mydomain.com  {
  server   192.168.2.11         weight=5        max_fails=3  fail_timeout=30s;
  server   192.168.2.21     weight=5    max_fails=3  fail_timeout=30s;
  server   192.168.2.23 backup;
  server   192.168.2.23 down;
}

server {
    listen       80;
    server_name  www.mydomain.com;

    charset utf-8;
    access_log  /var/log/nginx/www.mydomain.com.access.log  main;

    location / {
        root   /www/mydomain.com/www.mydomain.com;
        index  index.html index.htm;

                if ($request_uri ~* "\.(ico|css|js|gif|jpe?g|png|html)$") {
            expires 1d;
        }
        if ($request_uri ~* "\.(xml|json)$") {
            expires 1m;
        }

                valid_referers none blocked *.mydomain.com;
                if ($invalid_referer) {
                        #rewrite ^(.*)$  http://www.mydomain.com/cn/$1;
                        return 403;
                }

        proxy_intercept_errors  on;
        if (!-f $request_filename) {
          proxy_pass http://cms.mydomain.com;
          break;
        }
    }

    #error_page  404              /404.html;

    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }

    # deny access to .htaccess files, if Apache's document root
    # concurs with nginx's one
    #
    #location ~ /\.ht {
    #    deny  all;
    #}
}
```

4.5. cms 服务器

CMS 内容管理系统的主要功能

1. 内容分类管理
2. 内容模板管理
3. 内容编辑与发布
4. 内容生成

服务应该实现

1. 当发现目录中文件不存, 通过rewrite生成html, 这样可能根据需要生成html页面
2. 当页面更新的时候,应该通过api 刷新cdn的缓存, 图片的版本好应该加一
3. 将页面分成多个模块, 通过SSI拼装页面, 避免有重大改版时, 整站生成HTML.
4. 避免使用seesion技术, 这样在负载均衡的时候可以使用最小连接数算法

例如:

```
rewrite ^/product/(phone|notebook)/(\d+).html /product/$1.php?id=$2 last;
```

URL 唯一, url设计要考虑唯一性, 不要出现同一个url处理两个任务, 例如下面的例子, 每个用户的profile一个URL, 当被访问的时候便可以缓存在CDN或者用户浏览器上.

```
http://www.mydomain.com/profile/neo.html
http://www.mydomain.com/profile/jam.html
```

```
server {
    listen       80;
    server_name  www.mydomain.com;

    #charset koi8-r;
    access_log  /var/log/nginx/www.mydomain.com.access.log  main;

    location / {
        root   /www/mydomain.com/www.mydomain.com;
        index  index.html;
    }
}

server {
    listen       80;
    server_name  cms.mydomain.com;
```

```
    charset utf-8;
    access_log  /var/log/nginx/cms.mydomain.com.access.log  main;

    location / {
        root   /www/mydomain.com/cms.mydomain.com;
        index  index.html index.php;

    }

    location ~ ^/(cn|tw)/(comment|affiche)/.*\.html {
                root /www/mydomain.com/www.mydomain.com;
                if (!-f $request_filename) {
                        rewrite ^/(cn|tw)/(comment|affiche)/(\d+).html /publish/$2.php?id=$3&lang
                }
    }
    location /xml/ {
                root /www/mydomain.com/www.mydomain.com/xml;
    }
    location ~ ^/(config|include|crontab)/ {
                deny all;
                break;
    }

    #error_page  404              /404.html;

    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }

    # proxy the PHP scripts to Apache listening on 127.0.0.1:80
    #
    #location ~ \.php$ {
    #    proxy_pass   http://127.0.0.1;
    #}

    # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
    #
    location ~ \.php$ {
        root            /www/mydomain.com/cms.mydomain.com;
        fastcgi_pass   127.0.0.1:9000;
        fastcgi_index  index.php;
        fastcgi_param  SCRIPT_FILENAME  /www/mydomain.com/cms.mydomain.com$fastcgi_script_name;
        include        fastcgi_params;
                fastcgi_param  DOCUMENT_ROOT /www/mydomain.com/cms.mydomain.com;
                fastcgi_param  HOSTNAME cms.mydomain.com;
    }

    # deny access to .htaccess files, if Apache's document root
    # concurs with nginx's one
    #
    location ~ /\.ht {
        deny  all;
    }
}
```

4.6. img

img.mydomain.com

```
server {
    listen       80;
    server_name  img.mydomain.com;
```

```
    charset utf-8;
    access_log  /var/log/nginx/img.mydomain.com.access.log  main;

    location ~ .*\.(gif|jpg|jpeg|png|bmp|swf|ico)$
    {
                expires      7d;
    }
    location ~ .*\.(js|css)$
    {
                expires      1d;
    }
    location ~ .*\.(html|htm)$
    {
                expires      15m;
    }

    location / {
        root   /img/mydomain.com/img.mydomain.com;
        index  index.html;

        rewrite  "/theme/([0-9] {4})([0-9] {2})([0-9] {2})/(.+)\.(.+)\.(.+)" /theme/$1/$2/$3/$4.$
        rewrite  "/news/([0-9] {4})([0-9] {2})([0-9] {2})/(.+)\.(.+)\.(.+)" /news/$1/$2/$3/$4.$6;
        rewrite  "/product/([0-9] {4})([0-9] {2})([0-9] {2})/(.+)\.(.+)\.(.+)" /product/$1/$2/$3/
    }

}
```

/theme/2012/08/15/images.1.jpg 实际上就是 /theme/2012/08/15/images.jpg 文件

/theme/2012/08/15/images.2.jpg 也是 /theme/2012/08/15/images.jpg

/theme/2012/08/15/images.3.jpg 也是 /theme/2012/08/15/images.jpg

但CDN与你的浏览器会每次下载新的文件, 这样只要更新CDN中的html页面即可, 不用去理睬图片, 你的浏览器会用新的地址下载图片. 这样就解决了烦琐的刷新工作.

4.7. Ajax 局部更新与缓存

例如我的新闻评论页面,需要使用ajax技术, 将用户回复的品论显示来, ajax 载入json数据然后局部更新, 我对他做了1分钟的缓存

```
if ($request_uri ~* "\.(xml|json)$") {
        expires 1m;
}
```

如果有新的提交我们可以为json增加版本是控制例如: http://api.mydomain.com/news.json?1.0

5. 多媒体数据分离

5.1. 图片服务器分离

为什么要将图片服务器分离出来?

1. 图片通常比较大，下载需要更长的时间，而web容器并发数也是相当宝贵的仅次于数据库。

2. 传统浏览器一个窗口只占用一个链接数，目前主流浏览器都支持多线程下载，下载HTML页面同时，采用多线程下载其它多媒体数据。

我们举一个例子，你的服务器并发能力只用1000，早期浏览器不支持多线程，所以同一时刻，你的服务器可以承受1000个人同时访问。 但现在不同了，基本所有的浏览器都支持多线程，假如你的页面中有9张小图片,同一时刻你的服务器仅仅能应付1000/10 = 100个用户。

所以我们要将图片和其他多媒体文件分离出来，单独使用一台服务器处理请求。

提示

图片服务器建议使用lighttpd与squid缓存配合使用效果更好或购买CDN的服务。

5.2. 目录层次规划

日期有利于归档

```
/www/images
```

```
/www/images/2008
/www/images/2008/01
/www/images/2008/01/01
```

分类不同用途的文件

```
/www/images
/www/images/theme/2009

# article id 000001
/www/images/article/2009/01/000001

# product id 00001
/www/images/product/2009/01/01/00001

# member name neo
/www/images/member/2009/01/01/neo
```

根据你的数据量，创建目录深度,并且目录深度有规律可循。

虽然64bit 文件系统不限制文件数量与目录深度，但是我还是建议按我的方式规划目录。

这样规划目录便于缓存控制，如:

images/2008/* 永久缓存

images/2009/* 缓存一个月

images/2010/* 缓存一小时

images/2010/06/* 缓存5分钟

5.3. 多域名访问

部分浏览器（IE）相同域名只能创建2个线程，在页面中使用多个域名可以解决这个限制

```
img1.example.com IN CNAME images.example.com.
img2.example.com IN CNAME images.example.com.
img3.example.com IN CNAME images.example.com.
...
imgN.example.com IN CNAME images.example.com.
```

6. 图片尺寸优化与自动裁剪

http://netkiller.github.io/journal/image.html

6.1. 背景

某天我的前同事给我打电话，说他们的负载很高，经查发现网站首页有20M，原因是首页直接引用高清图片，没有安装分辨率生成缩图。于是我便想出了下面的方案。

我认为方案需求有如下几个要素：

1. 图片压缩
2. 尺寸修改
3. 图片缓存
4. 带宽因素

例如用户使用手机访问网站，手机屏幕尺寸非常多样化，常见的有QVGA(320×240)、HGVA(480×320)、WVGA(800×480)、QCIF(176×144)、SVGA(640x480)、WXGA(1280×800)。如果一个用户的手机屏幕是320×240，打开网站后显示1027*768图片很不切合实际。同时用户也多出不少带宽开销。

我们需要给用户更好的体验，就要多从用户的角度去考虑，如根据用户网速，带宽，分辨率，为用户提供更适合他终端的多媒体资源。

6.2. 实现思路

6.2.1. 尺寸动态变化

B/S结构应用程序无法获取客户端的分辨率等信息，我们将采用Javascript取出参数，然后告知服务器端。

有下面几种实现方式：

1. 通过cookie
2. post传递给服务器，然后存储在session中
3. get 传递给服务器，然后存储在session中

仅举一个例子

```
<script type="text/javascript">
$(function(){
    var width=window.screen.height;
    var height=window.screen.width;
    $.post('http://www.example.com/screen/resize.html',{w:width,h:height});
});
</script>
```

HTML页面中的图片的引用路径

```
<img src="http://img.example.com/sample.jpg" />
```

图片服务器rewrite处理

```
http://img.example.com/sample.jpg => http://img.example.com/index.php/sample.jpg
```

index.php会首先载入sample.jpg文件，然后综合网速，带宽，分辨率等因素，重新压缩图片，修改尺寸，发送mime头，输出正文。

6.2.2. 实时裁剪并静态化

为了防止图片地址冲突，我们首先需要URL唯一化，这样每访问一次会生成一张符合你需求尺寸的图片。

http://img.example.com/sample_(width)x(height)_(quality).jpg

```
<img src="http://img.example.com/sample_1980x1080_100.jpg" />
<img src="http://img.example.com/sample_800x600_80.jpg" />
<img src="http://img.example.com/sample_640x480_50.jpg" />
```

配置nginx通过try_files配置项可以实现检查静态文件是否存在，如果不存在边调用index.php生成图片，当再次访问时会直接读取静态文件，不会再重新生成。

```
server {
    listen       80;
    server_name  inf.example.com;

    charset utf-8;
    access_log  /var/log/nginx/inf.example.com.access.log  main;
    error_log  /var/log/nginx/inf.example.com.error.log;

    location / {
        root   /www/example.com/inf.example.com/images;
        index  index.html;
                try_files $uri $uri/ /index.php?_url=$request_uri;
    }

    #error_page  404              /404.html;

    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }

    # proxy the PHP scripts to Apache listening on 127.0.0.1:80
    #
    #location ~ \.php$ {
    #    proxy_pass   http://127.0.0.1;
```

```
    #}

    # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
    #
    location ~ /index\.php$ {
        root           html;
        fastcgi_pass   127.0.0.1:9000;
        fastcgi_index  index.php;
        fastcgi_param  SCRIPT_FILENAME  /www/example.com/inf.example.com/frontend/public$fastcgi_
        include        fastcgi_params;
    }

    # deny access to .htaccess files, if Apache's document root
    # concurs with nginx's one
    #
    location ~ /\.ht {
        deny  all;
    }
}
```

通过这种方法还可以实现更复杂的需求，例如调整亮度，对比度，饱和度，色阶，图层叠加等等.....

6.3. web或代理服务器插件实现方案

首先我们要将分辨率参数放到cookie中，因为web服务器是可以跟踪cookie数据的

通过 web 扩展实现，例如我们开发一个apache插件，编译后是".so"文件，配置httpd.conf载入插件，插件具体功能是综合网速，带宽，分辨率等因素，重新压缩图片，修改尺寸，最后展现图片。

反向代理与web服务器实现原理相同

7. 压缩数据传输

服务器将html或脚本输出压缩，用户从服务器取得数据后由浏览器解压

压缩数据传输实现方法：

1. apache mod_deflate
2. lighttpd compress module

7.1. Minify JS

最小化js文件

7.1.1. jsmin

http://crockford.com/javascript/jsmin

```
jsmin <fulljslint.js >jslint.js
```

7.1.2. yuicompressor

http://developer.yahoo.com/yui/compressor/

```
Usage: java -jar yuicompressor-x.y.z.jar [options] [input file]
```

7.1.3. shrinksafe

http://dojotoolkit.org/docs/shrinksafe

8. SSL

SSL 加密传输，为电子商务提供交易安全保护，什么时候该使用 SSL呢：

使用SSL

1. 用户登录
2. 购物流程
3. 支付

什么时候不使用SSL? 经过SSL加密后，你就失去了很多功能，你不能在对页面做Cache/CDN，SSL加密与解密需要耗费你的服务器CPU与内存资源，能不使用尽量不使用。

对于SSL消耗你服务器资源这方面有两个方案解决

1. 将SSL证书安装到CDN上，目前蓝讯，网宿等等CDN厂商都提供SSL服务。我与上两家技术人员沟通过，也安装了证书实际测试一下，你可以放心是使用。
2. 将SSL证书安装到负载均衡设备，这些设备都采用专用硬件处理SSL请求，我测试过F5，Array，Banggoo

采用上面两种方案，无需改变你目前的服务器配置，他们的原理是

```
user (https://www.example.com) --> CDN or SLB (SSL) --> http://www.example.com
```

用户访问https,到达CDN或者负载均衡，CDN/SLB 通过http://请求源站，然后将内容SSL加密，返回给用户，这样用户得到的是加密内容。

用户提交数据，交给CDN/SLB，CDN/SLB将SSL加密数据卸载证书，然后将解密后数据发回源站。

CDN与SLB加载卸载证书原理很简单，不难理解。

我来教你DIY一个，你可以使用Squid，Nginx，Apache等等反向代理服务器，将证书安装在反向代理上，请求源站仍然采用http。

SSL注意事项

你如果认为把SSL挂载到网站前端就，大功告成，完事了，那你错了。

幸运的话你会成功，但有时的时候你发现你的证书不被信任。如果你是个细心的人，你会发现单个图片，或者你创建换成测试文件 echo helloworld > index.html 证书都是OK的。

这个问题出在你的html页面中，安装有SSL证书的网站，不能有外链js,flash等等不安全内容。

9. 搜索引擎相关优化

9.1. 搜索结果静态化

每个搜索关键字都应该有一个惟一的URL，例如

```
https://www.google.com.hk/search?sourceid=chrome&ie=UTF-8&q=netkiller&sei=9v-QT_q1L6SZiAel2bGnBA&
https://www.google.com.hk/search?aq=f&sourceid=chrome&ie=UTF-8&q=neo
https://www.google.com.hk/search?sourceid=chrome&ie=UTF-8&q=bg7nyt
```

每搜索一次新的关键字就会产生一条唯一的URL，这样就可以实现反向代理缓存，甚者通过HTTP头，实现浏览器段的缓存。

9.2. robots.txt

```
<meta name="robots" content="noarchive">
```

例 2.1. example robots.txt

http://www.google.com/robots.txt

```
User-agent: *
Disallow: /search
Disallow: /groups
Disallow: /images
Disallow: /catalogs
Disallow: /catalogues
Disallow: /news
Allow: /news/directory
Disallow: /nwshp
Disallow: /setnewsprefs?
Disallow: /index.html?
Disallow: /?
Disallow: /addurl/image?
Disallow: /pagead/
Disallow: /relpage/
Disallow: /relcontent
Disallow: /imgres
Disallow: /imglanding
Disallow: /keyword/
Disallow: /u/
Disallow: /univ/
Disallow: /cobrand
Disallow: /custom
Disallow: /advanced_group_search
Disallow: /googlesite
Disallow: /preferencessection
Disallow: /setprefs
Disallow: /swr
Disallow: /url
Disallow: /default
Disallow: /m?
Disallow: /m/?
Disallow: /m/blogs?
```

```
Disallow: /m/ig
Disallow: /m/images?
Disallow: /m/local?
Disallow: /m/movies?
Disallow: /m/news?
Disallow: /m/news/i?
Disallow: /m/place?
Disallow: /m/setnewsprefs?
Disallow: /m/search?
Disallow: /m/swmloptin?
Disallow: /m/trends
Disallow: /wml?
Disallow: /wml/?
Disallow: /wml/search?
Disallow: /xhtml?
Disallow: /xhtml/?
Disallow: /xhtml/search?
Disallow: /xml?
Disallow: /imode?
Disallow: /imode/?
Disallow: /imode/search?
Disallow: /jsky?
Disallow: /jsky/?
Disallow: /jsky/search?
Disallow: /pda?
Disallow: /pda/?
Disallow: /pda/search?
Disallow: /sprint_xhtml
Disallow: /sprint_wml
Disallow: /pqa
Disallow: /palm
Disallow: /gwt/
Disallow: /purchases
Disallow: /hws
Disallow: /bsd?
Disallow: /linux?
Disallow: /mac?
Disallow: /microsoft?
Disallow: /unclesam?
Disallow: /answers/search?q=
Disallow: /local?
Disallow: /local_url
Disallow: /froogle?
Disallow: /products?
Disallow: /products/
Disallow: /froogle_
Disallow: /product_
Disallow: /products_
Disallow: /print
Disallow: /books
Disallow: /bkshp?q=
Allow: /booksrightsholders
Disallow: /patents?
Disallow: /patents/
Allow: /patents/about
Disallow: /scholar
Disallow: /complete
Disallow: /sponsoredlinks
Disallow: /videosearch?
Disallow: /videopreview?
Disallow: /videoprograminfo?
Disallow: /maps?
Disallow: /mapstt?
Disallow: /mapslt?
Disallow: /maps/stk/
Disallow: /maps/br?
Disallow: /mapabcpoi?
Disallow: /maphp?
Disallow: /places/
Disallow: /maps/place
Disallow: /help/maps/streetview/partners/welcome/
Disallow: /lochp?
```

```
Disallow: /center
Disallow: /ie?
Disallow: /sms/demo?
Disallow: /katrina?
Disallow: /blogsearch?
Disallow: /blogsearch/
Disallow: /blogsearch_feeds
Disallow: /advanced_blog_search
Disallow: /reader/
Allow: /reader/play
Disallow: /uds/
Disallow: /chart?
Disallow: /transit?
Disallow: /mbd?
Disallow: /extern_js/
Disallow: /calendar/feeds/
Disallow: /calendar/ical/
Disallow: /cl2/feeds/
Disallow: /cl2/ical/
Disallow: /coop/directory
Disallow: /coop/manage
Disallow: /trends?
Disallow: /trends/music?
Disallow: /trends/hottrends?
Disallow: /trends/viz?
Disallow: /notebook/search?
Disallow: /musica
Disallow: /musicad
Disallow: /musicas
Disallow: /musicl
Disallow: /musics
Disallow: /musicsearch
Disallow: /musicsp
Disallow: /musiclp
Disallow: /browsersync
Disallow: /call
Disallow: /archivesearch?
Disallow: /archivesearch/url
Disallow: /archivesearch/advanced_search
Disallow: /base/search?
Disallow: /base/reportbadoffer
Disallow: /base/s2
Disallow: /urchin_test/
Disallow: /movies?
Disallow: /codesearch?
Disallow: /codesearch/feeds/search?
Disallow: /wapsearch?
Disallow: /safebrowsing
Allow: /safebrowsing/diagnostic
Allow: /safebrowsing/report_error/
Allow: /safebrowsing/report_phish/
Disallow: /reviews/search?
Disallow: /orkut/albums
Disallow: /jsapi
Disallow: /views?
Disallow: /c/
Disallow: /cbk
Disallow: /recharge/dashboard/car
Disallow: /recharge/dashboard/static/
Disallow: /translate_a/
Disallow: /translate_c
Disallow: /translate_f
Disallow: /translate_static/
Disallow: /translate_suggestion
Disallow: /profiles/me
Allow: /profiles
Disallow: /s2/profiles/me
Allow: /s2/profiles
Allow: /s2/photos
Allow: /s2/static
Disallow: /s2
Disallow: /transconsole/portal/
```

```
Disallow: /gcc/
Disallow: /aclk
Disallow: /cse?
Disallow: /cse/panel
Disallow: /cse/manage
Disallow: /tbproxy/
Disallow: /comparisonads/
Disallow: /imesync/
Disallow: /shenghuo/search?
Disallow: /support/forum/search?
Disallow: /reviews/polls/
Disallow: /hosted/images/
Disallow: /hosted/life/
Disallow: /ppob/?
Disallow: /ppob?
Disallow: /ig/add?
Disallow: /adwordsresellers
Disallow: /accounts/o8
Allow: /accounts/o8/id
Disallow: /topicsearch?q=
Disallow: /xfx7/
Disallow: /squared/api
Disallow: /squared/search
Disallow: /squared/table
Disallow: /toolkit/
Allow: /toolkit/*.html
Disallow: /qnasearch?
Disallow: /errors/
Disallow: /app/updates
Disallow: /sidewiki/entry/
Disallow: /quality_form?
Disallow: /labs/popgadget/search
Disallow: /buzz/post
Sitemap: http://www.gstatic.com/s2/sitemaps/profiles-sitemap.xml
Sitemap: http://www.google.com/hostednews/sitemap_index.xml
Sitemap: http://www.google.com/ventures/sitemap_ventures.xml
Sitemap: http://www.google.com/sitemaps_webmasters.xml
Sitemap: http://www.gstatic.com/trends/websites/sitemaps/sitemapindex.xml
Sitemap: http://www.gstatic.com/dictionary/static/sitemaps/sitemap_index.xml
```

9.3. sitemaps

http://www.sitemaps.org/

sitemap.xml

9.4. Sitemap in robots.txt

```
User-agent: *
Allow: *
Disallow: /management/
Sitemap: http://netkiller.sourceforge.net/sitemaps.xml.gz
```

9.5. sitemap 静态内容生成工具

```
#!/bin/bash
DOMAIN="http://www.netkiller.cn"
PUBLIC_HTML=~/public_html
if [ ! -z $1 ]; then
        DOMAIN=$1
```

```
fi
lastmod=`date "+%Y-%m-%d"`

echo '<?xml version="1.0" encoding="UTF-8"?>'
echo '<?xml-stylesheet type="text/xsl" href="gss.xsl"?>'
echo '<urlset xmlns="http://www.google.com/schemas/sitemap/0.84" xmlns:xsi="http://www.w3.org/200
#echo '<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">'

for htmlfile in $(find $PUBLIC_HTML/ -type f -name "*.html")
do
        url=`echo $htmlfile | sed -e "s:$PUBLIC_HTML/::"`
        echo '   <url>'
        echo '      <loc>'${DOMAIN}'/'${url}'</loc>'
        echo '      <lastmod>'${lastmod}'</lastmod>'
        echo '      <changefreq>daily</changefreq>'
        echo '      <priority>0.5</priority>'
        echo '   </url>'
done

for htmlfile in $(find $PUBLIC_HTML/ -type f -name "*.epub")
do
        url=`echo $htmlfile | sed -e "s:$PUBLIC_HTML/::"`
        echo '   <url>'
        echo '      <loc>'${DOMAIN}'/'${url}'</loc>'
        echo '      <lastmod>'${lastmod}'</lastmod>'
        echo '      <changefreq>daily</changefreq>'
        echo '      <priority>0.5</priority>'
        echo '   </url>'
done

for htmlfile in $(find $PUBLIC_HTML/ -type f -name "*.mobi")
do
        url=`echo $htmlfile | sed -e "s:$PUBLIC_HTML/::"`
        echo '   <url>'
        echo '      <loc>'${DOMAIN}'/'${url}'</loc>'
        echo '      <lastmod>'${lastmod}'</lastmod>'
        echo '      <changefreq>daily</changefreq>'
        echo '      <priority>0.5</priority>'
        echo '   </url>'
done

for htmlfile in $(find $PUBLIC_HTML/ -type f -name "*.chm")
do
        url=`echo $htmlfile | sed -e "s:$PUBLIC_HTML/::"`
        echo '   <url>'
        echo '      <loc>'${DOMAIN}'/'${url}'</loc>'
        echo '      <lastmod>'${lastmod}'</lastmod>'
        echo '      <changefreq>daily</changefreq>'
        echo '      <priority>0.5</priority>'
        echo '   </url>'
done

for htmlfile in $(find $PUBLIC_HTML/ -type f -name *.pdf)
do
        url=`echo $htmlfile | sed -e "s:$PUBLIC_HTML/::"`
        echo '   <url>'
        echo '      <loc>'${DOMAIN}'/'${url}'</loc>'
        echo '      <lastmod>'${lastmod}'</lastmod>'
        echo '      <changefreq>daily</changefreq>'
        echo '      <priority>0.5</priority>'
        echo '   </url>'
done

echo "</urlset>"
```

10. 静态网站繁简转换

方案1: User -> Squid ->Web Server

修改squid源码，加入iconv(big5,gb2312,html page)

```
e.g.1 user (gb.example.org) -> Squid (big5->gb2312) -> web server
e.g.2 user (big5.example.org) -> Squid (gb2312->big5) -> web server
```

第 3 章 多维度架构之网络损耗

1. 中国的大网络环境

你出过国吗？或旅游，或出差，或长期工作。你没发现在外国上网跟国内上网的体验完全不同吗？

给你几分钟，你现在回忆一下，在外国上网有什么不同？

你是否发现在外国上网，在浏览器地址栏输入域名，敲下回车键的那一刻，网站立即展现出来？而在中国你的浏览器总是卡那么一下，转一圈（约0.2~1秒的时间）才出来。

这是为什么？

这是因为我们从自己的电脑到达服务器并返回数据经过的路径太长，节点太多造成了。

中国的网络环境是相当复杂

- 南北互通问题
- 带宽容量的问题
- 层层NAT转发问题
- 架构设计不合理的问题
- GFW过滤的问题
- 等等

访问过程中每经过一个节点都会造成一定延迟，当我们在浏览器中输入域名，中国DNS解析压力绝对比外国的服务器压力大，这是人口数量决定的。接着由于中国IP资源有限，我们需要层层NAT转发，然后还要被GFW拆包解包判断你的方案是否合规合法，最终到达我们的服

务器，现在的云环境，微服务架构等等，大量应用七层负载均衡和代理。

最终使我们无法体验到真正的互联网速度。

2. 架构设计需要考虑网络损耗

2.1. 硬件造成的网络损耗

网上有大量的架构设计文章，但几乎没有人探讨过，架构设计中的网络损耗问题。

现在的架构设计很多是相互借鉴，堆技术，架构师也多是技术控。不能从产品和用户体验角度思考。

最理想情况是服务器托管在电信机房，服务器网卡直接设置公网IP地址，网关直接指向电信的核心路由器，通过操作系统携带的防火墙控制访问策略。这是我们2005年之前的做法，那时还没有实力购买硬件防火墙。也正是因为这样使用部署过，对比后面架构才意识到我们是不是走偏了。

2005-2010 年我们开始使用 Cisco ASA 系列防火墙，机房分配的IP地址全部绑定在防火墙上，防火墙分为 Inside 和 Outside 两个策略，然后通过ACL将公网IP地址指向到局域网中的服务器上。使用硬件防火墙的确提高了运维效率，同时也提高了安全性。例如 xxx.xxx.xxx.xxx:80 => 172.16.0.10 将某公网IP的80端口指向局域网中的WEB服务器。

与之前相比，之前是从机房核心交换机上直接拉网线插到每台服务上，所有服务器网关指向机房出口路由器。服务器在机房的交换机上交换，出口走机房核心路由器，压力被转嫁。单台服务器故障，不影响整体。

现在的情况，机房一条网线拉过来，插到防火墙，我们的防火墙承担了所有服务器的流量和会话数。如果防火墙挂掉，全玩完，还好思科的设备没有掉过链子。

回到损耗话题，每次进入机房维护服务器，都会看看其他机柜，发现放多公司更多是使用路由器接入，不用防火墙。防火墙放在路由器后面给部分服务器做安全防护。

路由器与防火墙是有差异的。路由器侧重路由，携带了完善的路由协议，而防火墙侧重转发，只有RIP，其他路由协议被阉割（只有RIP，没有OSPF，IS-IS，BGP等等）

我们测试过物理服务器直接设置公网IP的延迟是比经过防火墙要低的，经过路由器的延迟可以忽略不计。

到了2010之后，云计算兴起，少量服务器不再托管，直接购买云服务。所以几乎没有人在托管一两台服务器，服务器网卡直接设置公网IP的做法了。此时的云服务器简称VPS，使用 Xen，OpenVZ 等技术实现，KVM还在孕育中。Xen,OpenVZ 等等都是半虚拟化，也不太成熟，当时体验VPS的整体印象，卡顿，卡顿，还是卡顿。所以我们还是以托管服务器为主。

F5，Array，A10…… 慢慢都用到了项目当中。还有开源的LVS, haproxy, nginx 用的一个爽，用的一个High。我心里清楚，从用户到服务器，中间每经过一个节点，都会造成网络延迟，但是这些技术能不用吗？不能！所以我视而不见。

我发现每经过一个网络设备可能会造成0.2-0.5秒的延迟。

2.2. 云平台造成的网络损耗

在云平台中，大平台接入层使用的是万兆旗舰网络设备，价格在百万到千万之间。那些小的云平台不太可能使用这种网络设备。

进入云平台后，云平台的网络是SDN（软件定义网络），通过操作系统中的虚拟网络设备，例如网桥，实现平台的网络管理。SDN 能整合

所有服务器的网络，有点类似交换机中的生成树VLAN 技术，随意在一组服务器上划分私有网络，映射公网IP地址和端口转发。

配置过linux 系统的小伙伴很容易理解，就是sysctl + iptables +tc 等等工具实现的。

所以云平台的损耗是非常高的，大家在一个共用的SDN下。

很多架构中，无法避免使用 haproxy, nginx 等等再做一层转发。

2.3. 容器中造成的网络损耗

最近几年容器技术很火，云平台和容器都降低了运维难度和对运维人员的技术能力要求。

容器的网络也是虚拟网络设备技术，大量使用iptables 做IP映射和端口转发。

Kubernetes 中Ingress 是 nginx 实现，使用Istio 实现 sidecar 模式，可谓疯狂，为每个 pod 都做一个 proxy。

这么做有错吗？ Google 当然没错，他们会使用旗舰机的网络设备和服务器。

你的公司 40GB的光纤交换机可能都用不起。

2.4. 微服务造成的网络损耗

现在不懂微服务，都不好意思说自己是架构师。

我以 Spring Cloud 为例，用户经过前文所说的那些网络设备，最终达到微服务网关，这个网关实质上就7层负载均衡，然后转发到Openfeign 服务，Openfeng 到注册中心获取服务节点，服务节点去配

置中心获取配置，完成业务逻辑运算，返回结果给 Openfeign，最后由经网关，再经过一堆网络设备，展现结果给用户。

不知道我说没说明白，我说明白了，你听没听明白。对比上文服务器直接绑定公网IP的做法，你又想到了什么?

微服务的拆分和治理，又引出了一个问题「分布式任务」后面会谈到这个问题。

3. 总结

用户访问系统，没经过一个节点都要建立TCP握手。无状态协议使用完后会逐一关闭TCP连接。这就是延迟的主要来源。

而对于长连接影响要小一些，例如视频播放，只是加载的时候需要依次握手，之后便是视频流传输。

除了网络延迟损耗，每增加一个节点就会多一个故障节点，如果超时时间设置不合理，就会出现雪崩效应。导致这个链路节点后面所有服务瘫痪。

我曾经写过一篇文章叫《警惕IT黑洞》有兴趣可以看看，在网上可以搜索到，现在我认为应该警惕架构黑洞。

第 4 章 多维度架构之超时时间

超时时间俗称 Timeout 它是引起应用程序无响应或者网络服务雪崩灾难的罪魁祸首。

超时时间设置非常讲究，太长不行，太短也不行。

超时时间由哪些

1. 网络超时
2. 文件系统超时
3. 执行时间超时

1. 无处不在的超时时间

早期架构相对简单，拓扑成线性，例如：

```
用户 —> WEB服务器 —> 应用服务器 —> 缓存 —> 数据库
```

这是最典型的应用了，我们就用这个来举例，所有例子都是我职业生涯中遇到过的，绝对真实。

很早的时候，那时还没有分布式文件系统，不过我们的系统IO压力不大，仅仅是为了高可用。我们使用 NFS 共享 WEB服务器上的HTML文件和图片。然后通过DNS轮询做负载均衡，这是 2005年之前常用的做法。

发生了什么呢，NFS 发神经，用一段时间后出现卡顿，读不出数据，Apache httpd 的超时时间设置为 60s 秒，此时WEB服务器进来一个用

户启动一个进程（那时 httpd 还不支持多线程），读取NFS共享的HTML，httpd 一直读不出来文件内容，直到60秒后 httpd 才会返回 500 错误给用户，用户始终超时等待。就这样，来一个用户，启动一个进程，用不了多久 httpd 最大连接数将被尽。

如果我们将超时时间30秒，可以加速进程的释放时间，可能会缓解 NFS 问题。设置成10秒呢？ 答案是不行，有些请求过程中会出现网络不稳定的情况，设置成10秒钟，httpd 就会终止用户连接。这种情况在大文件下载，上传的过程中较为明显。

从WEB服务器到应用服务器，可能是 cgi-bin, fastcgi, servlet 等等，连接方式有反向代理，也有特别的协议（好久不用忘记Tomcat 插件,好像叫 mod-jk）也有超时设置。设置不当会出现什么问题呢?

开发中我们要计算一个请求所花费的时间，尽量在5秒之内完成执行并返回结果，大于五秒就会产生用户流失，用于没有耐心等待页面完成加载，就会跳到其他网站。所以超时时间设置 60 秒基本上没有什么意义，只有下载和上传服务可能会用到。正常控制在 20-30秒可以应对大多数需求。同时 WEB 服务器 与 应用服务器的设置需要匹配，例如WEB服务器设置为 30秒，应用服务器设置为 60秒的后果是，程序还没有执行完成，WEB服务器就切断了与后面应用服务器的联系，并返回500错误。所以说后面应用服务器的超时时间设置，不能大于前面WEB服务器的超时时间设置。WEB切断与应用服务器通信后，应用服务器仍会继续执行，如果是 fastcgi 就可能看到非常多的进程在运行，并处于等待状态。

从应用服务器到缓存服务器，我遇到过这样一个事故，一名初级开发人员首次使用 memcached （那时还没有 Redis）没有经验，正确应该将一个数据结构序列化后存储到 memcache 中，他将这组数据，一条一条的写入memcache ，使用的时候使用循环遍历所有的 key。导致执行时间超过一分钟。 这样程序始终无法在规定的超时时间执行完成。

上线后立即崩溃，虽然也做了压力测试，但是有很多代码在测试环境是无法展现的。压力测不是万能的。

最后是数据库超时时间，数据库超时时间的设置，执行超时时间比网络超时时间更重要。所谓执行超时时间，就是控制执行SQL语句的时间，在规定时间没有完成查询就直接返回超时。这样做的目的是为了保护数据库，否则数据库很容易就崩溃了。跟前面的例子一样，如果将数据库执行超时时间设置为60秒，有一条SQL执行很慢，运行时间超过60秒，查询就会堆积，直到数据库连接数被占用完为止。所以我们要经常审计慢查询日志。

再说说网络设备造成的延迟，前面谈到的各种服务器应用出现问题都比较好排查，基本都是可见。网络设备造成的故障就比较难，理由水晶头氧化了，网络偶尔出现丢包卡顿。重新插拔一次水晶头故障就消失了。

2. 流量漏斗

流量是呈现漏斗状的，每经过一个节点，流量会被分流，例如 1000 个用户访问你的网站，60% 是 HTML，被WEB服务器分流了（当然还有CDN）。剩下流量进入了应用服务器， 可以不需要访问数据库，被分流了 10%，剩下访问需要查询缓存，被分流 20%，最后 10%的流量需要查询数据库。那么这样设置超时时间合理吗?

```
用户（30秒） —> WEB服务器（30秒） —> 应用服务器（30秒） —> 缓存（30秒） —> 数据库（30秒）
```

答案是不合理，你应该参考漏斗模型去设置，前大后小的原则。

3. 微服务的超时时间

前面我以2005年网络架构为例是因为简单好理解，2005-2015 年的这段时间架构发生了天翻地覆的变化，这里一笔带过。最近五年最流行的是「微服务」。 我看到很多谈及架构文章中，大量的使用网关，层层代理，才触动了我写该文章，少有文章会谈到网络损耗和超时时间。

由于微服务提出了熔断器的概念，可以防止服务雪崩，微服务中会使用这种熔断技术。这相当于被蛇咬了，断臂自救，以现在的医疗技术是可以治愈的，但是却选择了截止。而我们要做的是不让这种事情出现。

4. 容器技术的超时时间

Kubernetes 有种 sidecar 技术，为每个 pod 设置一个代理，这种方案优势是让容器更好管理。但是增加了超时的设置的规划难度。

5. 最后总结

目前几乎所有公司的架构设计首先服务于交付，交付后线上出现很多问题，然后再救火，修补，总结。交付后再做优化的结果，就是你在网上看到那些《XXXXX 踩过的坑》

第 5 章 多维度架构之会话数

上几期我们谈了，多维度架构中的网络损耗和超时时间，今天我们谈谈另一个在多维度架构中非常重要的技术点「会话数」。会话数的英文是 Session，请不要与HTTP服务中的SESSION混淆。

在无状态服务应用中会话数是不断释放的，例如 HTTP，SMTP 协议，当WEB服务器完成请求后就会关闭 TCP 端口，这是会话数就被释放。但是基于 Socket 长连接的服务器会一直占用会话数。

1. 路由器和防火墙的会话数

之前从没想过会话数会被用光的情况，第一次遇到会话数不够的情况是公司办公室网络出口，公司的出口是一台 Juniper 入门级网络设备，估计会话数2048 左右，适合100人左右的公司。我们遇到的问题就是局域网上网慢，网页加载总是一半或部分资源加载不出来。经过排查发现是会话数不够用了，升级了 Juniper 企业级的设备后，就立即解决了，直到公司员工数量暴涨到800多人，仍能提供服务。

后来我们又在IDC也遇到会话数用尽的问题，当时使用 Cisco PIX 防火墙，我们提早发现了可能随着业务的发展，后面会话数会成为瓶颈，提前升级了防火墙 Cisco ASA 5550。

网络设备中，交换机的背板带宽决定了数据的转发能力，带宽不够，就会延迟。路由器和防火墙的会话数决定了你能同时建立多少个 TCP 连接，会话数不够，即使你的服务器CPU没有负载，内存没有用尽，磁盘IO闲置，用户仍然进不来，无法建立TCP连接。

而有状态的 Socket 服务是持久占用会话数，比较好统计。无状态协议（如HTTP）是动态会话数，会有峰值访问，难以预测。会话数达到 80% 的时候，就要预警了。所以监控系统要将会话数也监控起来。

2. 负载均衡设备的会话数

除了路由器和防火墙还有负载均衡设备，例如F5，Array，A10 一类的设备，也有会话数的概念。购买的时候要询问好。

如何解决网络设备会话数不够的问题，答案是用Active，Active 双活方案，业务分流，互为主备。即多台并行使用。

3. 服务器的会话数

我曾经写过一篇文章《压力测试中存在的问题》文中谈及，很多压力测试人员，未对Linux系统做优化，仅仅部署应用后，就开始测试。至今还有很多人在犯这种错误。

Linux 系统如果不做优化，即使你是16核心，128G内存，是无法提供大并发访问的，系统各种资源都是被限制的。所以必须对服务器做出各种内核参数的调整，系统才能充分使用全部的硬件资源。

Linux 中影响服务器的会话数主要是 sysctl, ulimit 两项的配置。包括了文件打开数量，进程数量，内存限额，端口范围等等……

4. 应用程序的会话数

应用程序的会话数就是我们常常说的并发连接数，或叫最大连接数。这是软件开发的配置问题，这里就不深入讨论了。

影响应用软件的最大连接数的是系统资源。

第 6 章 多维度架构之日志

传统做法是文件型日志，分布在各个服务器上。在大规模部署服务器后代来很多不便，增加很多管理成本，所有我们需要集中管理服务器产生的所有日志，我们叫他日志中心服务器

1. 一次切割日志引发的血案

很多应用程序会产生日志，有些程序已经实现了日志切割，一般是每天一个文件。但有时这个切割并不能满足我们的需求，例如我们需要颗粒度更细的切割。

切割日志的目的是什么？

- 日志尺寸过大
- 便于分析

- 切割后归档，或者导入日志平台

切割日志基本两种方法:

- 手工
- shell
- 工具，例如logrotate，传统的cronolog

日志切割方案网上有很多，很多运维也是参考这些方案进行配置，网上的例子不完全都是对的，可能你用了很多年配置方案是错误的。 没有出现故障是侥幸，因为笔者15年前就在此处栽过，由于日志太大，我便清空了日志，以为程序仍然会继续写入，最后直到服务器崩溃。 最近发现很多新手再谈cronolog，我便想起当前发生的故障，有必要跟大家分享。

首先日志是可以切割的，网上的例子理论上也是可行，但我们不能不求甚解，稀里糊涂的用下去。

我们首先了解一下日志是怎么产生的，那种日志可以切割，那些日志不能切割，为什么不能切割，如果需要切割日志怎么处理?

1.1. 日志是怎么产生的

日志生命周期，创建/打开日志文件，追加日志记录，关闭日志文件。请看下面伪代码。

```
main (){
        f = open(/tmp/prog.log) ...
        ...
        f.append('DEBUG .............')
        ...
        f.append('INFO .............')
        ...
        f.append('WARN .............')
        f.close()
}
```

这个程序是顺序运行，每次运行都会经历，打开日志文件，追加日志记录，关闭日志文件，一个日志生命周期结束。 在完成日志生命周后，你就可以切割日志了。因为f.close()后日志文件已经被释放。

再看下面的程序

```
main (){
        f = open(/tmp/prog.log)
        loop{
                ... ...
                f.append('DEBUG .............')
                ...
                f.append('INFO .............')
                ...
                f.append('WARN .............')
                if(quit){
                        break
                }
        }
        f.close()
}
```

这个程序就不同了，程序运行，打开日志文件，然后进入无穷循环，期间不断写入日志，接到退出命令才会关闭日志。 那么这个程序你就不能随便切割日志。你一旦修改了日志文件，程序将不能在写入日志到文件中。 这个程序切割日志的过程是这样的

```
split loop { prog run prog quit && mv /tmp/prog.log /tmp/prog.2016-05-05.log }
```

再看下面的程序

```
main (){
        loop{
                f = open(/tmp/prog.log)
                loop{
                        ...
                        ...
                        f.append('DEBUG .............')
                        ...
                        f.append('INFO .............')
                        ...
                        f.append('WARN .............')
                        if(reload){
                                break
                        }
                }
                f.close()
        }
}
```

这个程序多了一层循环，并加入了重载功能。这个程序怎样切割日志呢：

```
split loop {
        prog run
        mv /tmp/prog.log /tmp/prog.YYYY-MM-DD.log
        prog reload
}
```

```
main (){
        loop{
                f = open(/tmp/prog.YYYY-MM-DD.log)
                loop{
                        ...
                        ...
                        f.append('DEBUG .............')
                        ...
                        f.append('INFO .............')
                        ...
                        f.append('WARN .............')
                        if(reload){
                                break
                        }
                }
                f.close()
        }
}
```

如果你是程序猿，这个程序可以优化一下，日志文件名自动产生日期 /tmp/prog.YYYY-MM-DD.log 在reload时候重新创建或打开日志。

最操蛋写法，很多初学者会这么干，

```
log(type, msg){
        f = open(/tmp/prog.YYYY-MM-DD.log)
        f.append(type, msg)
        f.close()
}

main(){ ...
        ...
        log('INFO','..............') ...
        ...
        log('DEBUG','..............') ...
        ...
}
```

这种代码的适应性非常强，写一个日志函数，但牺牲了IO性能，如果频繁打开/关闭文件同时进行写IO操作，这样的程序很难实现高并发。 所以很多高并发的程序，只会打开一次日志文件（追加模式），不会再运行期间关闭日志文件，直到进程发出退出信号。

1.2. 让我们看个究竟

我们手工模拟一次日志分割的过程，首先开启三个Shell终端。

1.2.1. 第一种情况，日志文件被重命名

```
终端一，模拟打开日志文件

[root@www.netkiller.cn ~]# cat > /tmp/test.log
终端二，重命名文件

[root@www.netkiller.cn ~]# mv /tmp/test.log /tmp/test.2016.05.05.log
终端一，输入一些内容然后按下Ctrl+D 保存文件

[root@www.netkiller.cn ~]# cat > /tmp/test.log
Helloworld
Ctrl + D[root@www.netkiller.cn ~]# cat /tmp/test.log
cat: /tmp/test.log: No such file or directory
```

1.2.2. 第二种情况，日志文件被删除

```
终端一，模拟打开日志文件

[root@www.netkiller.cn ~]# cat > /tmp/test.log
终端二，使用lsof查看文件的打开情况
```

```
[root@www.netkiller.cn ~]# lsof | grep /tmp/test.log
cat       20032           root    1w      REG              253,1          0     288466 /tmp/test.
```
终端三，删除日志文件

```
[root@www.netkiller.cn ~]# rm -rf /tmp/test.log
```
终端二，查看日志的状态，你能看到 deleted

```
[root@www.netkiller.cn ~]# lsof | grep /tmp/test.log
cat        5269           root    1w      REG              253,1          0     277445 /tmp/test.
```
终端一，回到终端一种，继续写入一些内容并保存，然后查看日志文件是否有日志记录被写入

```
[root@www.netkiller.cn ~]# cat > /tmp/test.log
Helloworld
^D[root@www.netkiller.cn ~]# cat /tmp/test.log
cat: /tmp/test.log: No such file or directory
```
经过上面两个实验，你应该明白了在日志打开期间对日志文件做重命名或者删除都会造成日志记录的写入失败。

1.2.3. 第三种情况，日志没有被删除，也没有被重命名，而是被其他程序做了修改

第一步，终端窗口一中创建一个文件，文件写入一些字符串，这里写入 “one”，然后查看是否成功写入。

```
[root@www.netkiller.cn ~]# echo one > /tmp/test.log[root@www.netkiller.cn ~]# cat /tmp/test.log
one
```
上面我们可以看到/tmp/test.log文件成功写入一个字符串”one”

第二步，开始追加一些字符串

```
[root@www.netkiller.cn ~]# cat > /tmp/test.log
two
```
先不要保存（不要发出^D）

第三部，在终端二窗口中清空这个文件

```
[root@www.netkiller.cn ~]# > /tmp/test.log [root@www.netkiller.cn ~]# cat /tmp/test.log
```
通过cat查看/tmp/test.log文件，什么也没也表示操作成功。

第四步，完成字符串追加，使用Ctrl+D保存文件，最后使用cat /tmp/test.log 查看内容。

```
[root@www.netkiller.cn ~]# cat > /tmp/test.log
two[root@www.netkiller.cn ~]# cat /tmp/test.log
```
你会发现/tmp/test.log文件中没有写入任何内容。这表示在日志的访问期间，如果其他程序修改了该日志文件，原来的程序将无

让我们再来一次，看个究竟

终端一，创建并追加字符串到日志文件中

```
# echo one > /tmp/test.log# cat /tmp/test.logone# cat >> /tmp/test.logtwo
```
记得不要保存

终端二，使用lsof查看文件的打开情况

```
# lsof | grep /tmp/test.logcat       22631           root    1w      REG              253,1
```
终端三，开启另一个程序追加字符串到日志文件中

```
# cat >> /tmp/test.log three
```
先不要保存（不要发出^D）

终端二，查看文件的打开情况

```
# lsof | grep /tmp/test.logcat       22631           root    1w      REG              253,1
cat       23350           root    1w      REG              253,1          0     277445 /tmp/test.
```
终端三，保存three字符串

```
# cat >> /tmp/test.log three
^D# cat /tmp/test.log three
```
回到终端一，继续保存内容

```
# cat > /tmp/test.logtwo
^D# cat /tmp/test.logtwo
e
```
出现新的行情况了，two报道最上面去了，这是因为打开文件默认文件指针是页首，它不知道最后一次文件写入的位置。

你可以反复实验，结果相同。

```
# cat /tmp/test.logtwo
e
four
five
```
我为什么没有使用 echo “five” » /tmp/test.log 这种方式追加呢？因为 cat 重定向后只要不发出^D就不会保存文件，而e

1.3. 经典案例分析

我们以 Nginx 为例

```
[root@www.netkiller.cn ~]# cat /etc/logrotate.d/nginx
/var/log/nginx/*.log {        daily
        missingok
        rotate 52
        compress
        delaycompress
        notifempty
        create 640 nginx adm
        sharedscripts
        postrotate                [ -f /var/run/nginx.pid ] && kill -USR1 `cat /var/run/nginx.pid
```

nginx 日志切割后会运行 kill -USR1 这个让nginx 重新创建日志文件或者夺回日志文件的操作权。

1.4. 怎样监控日志

那么怎样监控日志被删除，写入权被其他程序夺取？对于程序猿一定很关注这个问题。下面我们讲解怎么监控日志。

Linux 系统可以使用 inotify 开发包来监控文件的状态变化，包括开打，写入，修改权限等等。你需要启动一个进程或者线程监控日志文件的变化，以便随时reload 你的主程序。

```
prog { sign = null
        logfile = /var/log/your.log

        thread monitor {
                inotify logfile { sign = reload }
        }
        thread worker {
                loop{
                        f = open(logfile)
                        loop{
                                f.append(....)
                                if(sign == reload) { break }
```

```
                        }
                        f.close()
                }
        }
        main(){
                monitor.start()
                worker.start()
        }
}
```

不知你是否看懂，简单的说就是两个并行运行的程序，一个负责日志监控，一个负责干活，一旦日志发生变化就通知主程序 reload。 至于使用进程还是线程去实现，取决于你熟悉那种语言或者你擅长的技术。

1.5. 总结

小小的日志文件有如此大的学问，目前很多应用程序写的比较健壮，能够判断出当前日志被删除，改写。程序运行中能够在创建丢失的日志文件，当日志被其他程序改写后，能够夺回写入权。 但这样的程序会影响程序并发性能，鱼和熊掌不能兼得。看了这篇文章我想你应该对日志有了全面了解，也会在接下来的工作中谨慎处理日志。

2. 日志归档与数据挖掘

2.1. 什么日志归档

归档，是指将日志整理完毕且有保存价值的文件，经系统整理交日志服务器保存的过程。

2.2. 为什么要做日志归档

- 随时调出历史日志查询。
- 通过日志做数据挖掘，挖掘有价值的数据。
- 查看应用程序的工作状态

2.3. 何时做日志归档

日志归档应该是企业规定的一项制度(“归档制度”)，系统建设之初就应该考虑到日志归档问题。如果你的企业没有这项工作或制度，在看完本文后建议你立即实施。

2.4. 归档日志放在哪里

简单的可以采用单节点服务器加备份方案。

随着日志规模扩大，未来必须采用分布式文件系统，甚至涉及到远程异地容灾。

2.5. 谁去做日志归档

我的答案是日志归档自动化，人工检查或抽检。

2.6. 怎样做日志归档

将所有服务器的日志都汇总到一处，有几种方法

日志归档常用方法：

- ftp 定是下载， 这种做法适合小文件且日志量不大，定是下载到指定服务器，缺点是重复传输，实时性差。
- rsyslog 一类的程序，比较通用，但扩展不便。
- rsync 定是同步，适合打文件同步，好于FTP，实时性差。
- logstash, filebates, flume 等等。

2.6.1. 系统日志

`rsyslog/syslog-ng` 实现日志集中管理

2.6.2. 应用程序日志

应用程序中没有比较大量记录日志，当开启debug模式时才记录大量日志。

但是很多国内开发太过于依赖日志，导致日记非常臃肿

程序日志解决方案，请看软件架构相关章节

2.6.3. 日志格式转换

首先我来介绍一种简单的方案

我用D语言写了一个程序将 WEB 日志正则分解然后通过管道传递给数据库处理程序

2.6.3.1. 将日志放入数据库

将WEB服务器日志通过管道处理然后写入数据库

处理程序源码

```
$ vim match.d
import std.regex;
import std.stdio;
import std.string;
import std.array;

void main()
{
    // nginx
        //auto r = regex(`^(\S+) (\S+) (\S+) \[(.+)\] "([^"]+)" ([0-9]{3}) ([0-9]+) "([^"]+)" "([

        // apache2
        auto r = regex(`^(\S+) (\S+) (\S+) \[(.+)\] "([^"]+)" ([0-9]{3}) ([0-9]+) "([^"]+)" "([^"

        foreach(line; stdin.byLine)
        {

                foreach(m; match(line, r)){
                        //writeln(m.hit);
                        auto c = m.captures;
                        c.popFront();
                        //writeln(c);
                        auto value = join(c, "\",\"");
                        auto sql = format("insert into log(remote_addr,unknow,remote_user,time_lo
                        writeln(sql);
                }
        }
}
```

编译

```
$ dmd match.d
$ strip match

$ ls
match  match.d  match.o
```

简单用法

```
$ cat access.log | ./match
```

高级用法

```
$ cat access.log | match | mysql -hlocalhost -ulog -p123456 logging
```

实时处理日志，首先创建一个管道，寻该日志文件写入管道中。

```
cat  管道名 | match | mysql -hlocalhost -ulog -p123456 logging
```

这样就可以实现实时日志插入。

提示

上面程序稍加修改即可实现Hbase, Hypertable 本版

2.6.3.2. Apache Pipe

Apache 日志管道过滤 CustomLog "| /srv/match >> /tmp/access.log" combined

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost

        #DocumentRoot /var/www
        DocumentRoot /www
        <Directory />
                Options FollowSymLinks
                AllowOverride None
        </Directory>
        #<Directory /var/www/>
        <Directory /www/>
                Options Indexes FollowSymLinks MultiViews
                AllowOverride None
                Order allow,deny
                allow from all
        </Directory>

        ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
        <Directory "/usr/lib/cgi-bin">
                AllowOverride None
                Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
                Order allow,deny
                Allow from all
        </Directory>

        ErrorLog ${APACHE_LOG_DIR}/error.log

        # Possible values include: debug, info, notice, warn, error, crit,
        # alert, emerg.
        LogLevel warn

        #CustomLog ${APACHE_LOG_DIR}/access.log combined
        CustomLog "| /srv/match >> /tmp/access.log" combined

    Alias /doc/ "/usr/share/doc/"
    <Directory "/usr/share/doc/">
        Options Indexes MultiViews FollowSymLinks
        AllowOverride None
        Order deny,allow
        Deny from all
        Allow from 127.0.0.0/255.0.0.0 ::1/128
    </Directory>
```

```
</VirtualHost>
```

经过管道转换过的日志效果

```
$ tail /tmp/access.log
insert into log(remote_addr,unknow,remote_user,time_local,request,status,body_bytes_sent,http_ref
insert into log(remote_addr,unknow,remote_user,time_local,request,status,body_bytes_sent,http_ref
insert into log(remote_addr,unknow,remote_user,time_local,request,status,body_bytes_sent,http_ref
```

2.6.3.3. Log format

通过定义LogFormat可以直接输出SQL形式的日志

Apache

```
LogFormat "%v:%p %h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" vhost_combined
LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %O" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
```

Nginx

```
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';
```

但对于系统管理员使用grep,awk,sed,sort,uniq分析时造成一定的麻烦。所以我建议仍然采用正则分解

产生有规则日志格式，Apache：

```
LogFormat \
        "\"%h\",%{%Y%m%d%H%M%S}t,%>s,\"%b\",\"%{Content-Type}o\",  \
        \"%U\",\"%{Referer}i\",\"%{User-Agent}i\""
```

将access.log文件导入到mysql中

```
LOAD DATA INFILE '/local/access_log' INTO TABLE tbl_name
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' ESCAPED BY '\\'
```

2.6.3.4. 日志导入到 MongoDB

```
# rpm -Uvh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
# yum install mongodb
```

D语言日志处理程序

```
import std.regex;
//import std.range;
import std.stdio;
import std.string;
import std.array;

void main()
{
        // nginx
        auto r = regex(`^(\S+) (\S+) (\S+) \[(.+)\] "([^"]+)" ([0-9]{3}) ([0-9]+) "([^"]+)" "([^"
        // apache2
        //auto r = regex(`^(\S+) (\S+) (\S+) \[(.+)\] "([^"]+)" ([0-9]{3}) ([0-9]+) "([^"]+)" "([
        foreach(line; stdin.byLine)
        {
                //writeln(line);
                //auto m = match(line, r);
                foreach(m; match(line, r)){
                        //writeln(m.hit);
                        auto c = m.captures;
                        c.popFront();
                        //writeln(c);
                        /*
                        SQL
                        auto value = join(c, "\",\"");
                        auto sql = format("insert into log(remote_addr,unknow,remote_user,time_lo
                        writeln(sql);
                        */
                        // MongoDB
                        string bson = format("db.logging.access.save({
                                                'remote_addr': '%s',
                                                'remote_user': '%s',
                                                'time_local': '%s',
                                                'request': '%s',
                                                'status': '%s',
                                                'body_bytes_sent':'%s',
                                                'http_referer': '%s',
                                                'http_user_agent': '%s',
                                                'http_x_forwarded_for': '%s'
                                                })",
                                                c[0],c[2],c[3],c[4],c[5],c[6],c[7],c[8],c[9]
                                                );
                        writeln(bson);

                }
        }

}
```

编译日志处理程序

```
dmd mlog.d
```

用法

```
cat /var/log/nginx/access.log | mlog | mongo 192.169.0.5/logging -uxxx -pxxx
```

处理压错过的日志

```
# zcat /var/log/nginx/*.access.log-*.gz | /srv/mlog | mongo 192.168.6.1/logging -uneo -pchen
```

实时采集日志

```
tail -f /var/log/nginx/access.log | mlog | mongo 192.169.0.5/logging -uxxx -pxxx
```

2.6.4. 日志中心方案

上面的方案虽然简单，但太依赖系统管理员，需要配置很多服务器，每种应用软件产生的日志都不同，所以很复杂。如果中途出现故障，将会丢失一部日志。

于是我又回到了起点，所有日志存放在自己的服务器上，定时将他们同步到日志服务器，这样解决了日志归档。远程收集日志，通过UDP协议推送汇总到日志中心，这样解决了日志实时监控、抓取等等对实时性要求较高的需求。

为此我用了两三天写了一个软件，下载地址：https://github.com/netkiller/logging

这种方案并不是最佳的，只是比较适合我的场景，而且我仅用了两三天就完成了软件的开发。后面我会进一步扩展，增加消息队列传送日志的功能。

2.6.4.1. 软件安装

```
$ git clone https://github.com/netkiller/logging.git
$ cd logging
$ python3 setup.py sdist
$ python3 setup.py install
```

2.6.4.2. 节点推送端

安装启动脚本

CentOS

```
# cp logging/init.d/ulog /etc/init.d
```

Ubuntu

```
$ sudo cp init.d/ulog /etc/init.d/

$ service ulog
Usage: /etc/init.d/ulog {start|stop|status|restart}
```

配置脚本，打开 /etc/init.d/ulog 文件

配置日志中心的IP地址

```
HOST=xxx.xxx.xxx.xxx
```

然后配置端口与采集那些日志

```
        done << EOF
1213 /var/log/nginx/access.log
1214 /tmp/test.log
1215 /tmp/$(date +"%Y-%m-%d.%H:%M:%S").log
EOF
```

格式为

```
Port | Logfile
------------------------------
1213 /var/log/nginx/access.log
1214 /tmp/test.log
1215 /tmp/$(date +"%Y-%m-%d.%H:%M:%S").log
```

1213 目的端口号（日志中心端口）后面是你需要监控的日志，如果日志每日产生一个文件写法类似 /tmp/$(date +"%Y-%m-%d.%H:%M:%S").log

提示

每日产生一个新日志文件需要定时重启 ulog 方法是 /etc/init.d/ulog restart

配置完成后启动推送程序

```
# service ulog start
```

查看状态

```
$ service ulog status
13865 pts/16   S      0:00 /usr/bin/python3 /usr/local/bin/rlog -d -H 127.0.0.1 -p 1213 /var/log/
```

停止推送

```
# service ulog stop
```

2.6.4.3. 日志收集端

```
# cp logging/init.d/ucollection /etc/init.d

# /etc/init.d/ucollection
Usage: /etc/init.d/ucollection {start|stop|status|restart}
```

配置接收端口与保存文件，打开 /etc/init.d/ucollection 文件，看到下面段落

```
done << EOF
1213 /tmp/nginx/access.log
1214 /tmp/test/test.log
1215 /tmp/app/$(date +"%Y-%m-%d.%H:%M:%S").log
1216 /tmp/db/$(date +"%Y-%m-%d")/mysql.log
```

```
1217 /tmp/cache/$(date +"%Y")/$(date +"%m")/$(date +"%d")/cache.log
EOF
```

格式如下，表示接收来自1213端口的数据，并保存到/tmp/nginx/access.log文件中。

```
Port | Logfile
1213 /tmp/nginx/access.log
```

如果需要分割日志配置如下

```
1217 /tmp/cache/$(date +"%Y")/$(date +"%m")/$(date +"%d")/cache.log
```

上面配置日志文件将会产生在下面的目录中

```
$ find /tmp/cache/
/tmp/cache/
/tmp/cache/2014
/tmp/cache/2014/12
/tmp/cache/2014/12/16
/tmp/cache/2014/12/16/cache.log
```

提示

同样，如果分割日志需要重启收集端程序。

启动收集端

```
# service ulog start
```

停止程序

```
# service ulog stop
```

查看状态

```
$ init.d/ucollection status
12429 pts/16   S      0:00 /usr/bin/python3 /usr/local/bin/collection -d -p 1213 -l /tmp/nginx/ac
12432 pts/16   S      0:00 /usr/bin/python3 /usr/local/bin/collection -d -p 1214 -l /tmp/test/tes
12435 pts/16   S      0:00 /usr/bin/python3 /usr/local/bin/collection -d -p 1215 -l /tmp/app/2014
12438 pts/16   S      0:00 /usr/bin/python3 /usr/local/bin/collection -d -p 1216 -l /tmp/db/2014-
12441 pts/16   S      0:00 /usr/bin/python3 /usr/local/bin/collection -d -p 1217 -l /tmp/cache/20
```

2.6.4.4. 日志监控

监控来自1217宽口的数据

```
$ collection -p 1213

192.168.6.20 - - [16/Dec/2014:15:06:23 +0800] "GET /journal/log.html HTTP/1.1" 304 0 "-" "Mozilla
```

```
192.168.6.20 - - [16/Dec/2014:15:06:23 +0800] "GET /journal/docbook.css HTTP/1.1" 304 0 "http://1
192.168.6.20 - - [16/Dec/2014:15:06:23 +0800] "GET /journal/journal.css HTTP/1.1" 304 0 "http://1
192.168.6.20 - - [16/Dec/2014:15:06:23 +0800] "GET /images/by-nc-sa.png HTTP/1.1" 304 0 "http://1
192.168.6.20 - - [16/Dec/2014:15:06:23 +0800] "GET /js/q.js HTTP/1.1" 304 0 "http://192.168.6.2/j
```

启动后实时将最新日志传送过来

第 7 章 多维度架构之监控

1. 背景

每个企业都意识到监控工作的重要性，但80%企业的监控工作仍然处在监控的初级阶段。

什么是初级阶段呢?

1. 被动监控，故障发生运维人员永远不是第一个发现故障的人
2. 监控IP地址与TCP端口，很多时候HTTP 80端口正常接受请求，但WEB服务器不能正常工作。
3. 人肉监控（人肉运维），采用人海战术，桌面摆放很多显示器，甚至投影仪，要求监控者盯着各种仪表板界面，制定各种工作流程以及KPI考核监控人员。
4. 人肉测试，要求监控人员每间隔几分钟人工操作一次，以确认系统正常工作，例如（没15分钟登陆一次，下一笔顶单，做一次支付等等）。
5. 万能的重启，定其重启所有的服务器。

什么是中级阶段呢?

1. 报警：手机短信更靠谱，因为手机随身携带（邮件不算，邮件到达速度慢，各种因素不稳定）

2. 监控服务：探测服务的可用性，而不是仅仅监控端口，注意我是指私有协议的监控（HTTP，SMTP,FTP,MySQL 不算在内）
3. 故障分析：通过日志与调试工具分析软件BUG，指导开发人员改善软件质量，使其故障不会再次发生，达到不用restart重启方式解决故障
4. 半自动化测试

什么是高级阶段呢?

1. 我认为高级阶段是监控与灾备系统打通融合一体。
2. 除此之外监控与开发密切相关，在开发阶段需要为监控数据采集做铺垫，每开发一个新功能就要想到未来这个功能是否需要监控，怎样监控。
3. 数据前期采集与数据挖掘非常重要，监控不仅能做软件与硬件的性能分析，还能提供决策支持，这里又涉及了BI。
4. 除了监控，另一个息息相关的是自动故障转移，有兴趣可以看看我的其他文章 http://netkiller.github.io/journal/

监控从初级向中继再到高级，是转被动到主动，从人工到自动化。

监控不应该局限在硬件与服务，还应该延伸到业务领域。

2. 概述

你在百度上搜索监控多半是一些开源或商业软件的安装配置指南。这些文章中会告诉你怎样监控CPU、内存、硬盘空间以及网络IP地址与端口号码。

开源软件无非是 Nagios, Cacti, Mrtg, Zibbix ..... 这些软件在我的电子出书 《Netkiller Monitoring 手札》 中都有详细说明安装与配置方法。

商业软件也有很多如 SolarWinds， Whit's Up，PRTG .....

所有的服务器，网络设备，监控你都做了，那么按照我上面的监控分级，你处于监控的那个阶段?

3. 怎样监控

监控都有哪些手段跟方式呢?

3.1. 卫星监测

中心卫星站为中心站点向外放射，通常是通过IP地址访问远程主机，实施监控，常用方法是SNMP,SSH,以及各种Agent(代理)，方式是请求然后接收返回结果，通过结果判断主机状态。

```
      Monitor Server
            |
--------------------------------
  |         |            |
[Web]    [Mail]    [Database]
```

以监控服务器为中心，星型散射连接其他监控节点，没有什么优点，缺点是Web跟Mail节点的通信没有监控

3.2. 逐级诊断

一级一级的向下探测，寻找故障点，需要在各个节点埋探针。

```
      Monitor Server
               |
--------------------------------
  |            |              |
  V            V              V
  |            |              |
[Web] ---> [Cache] ---> [Database]
  \                          ^
   `-------------------------|
```

首先监控服务器跟星型拓扑一样监控，再让Web节点去访问Cache节点然后返回监控结果，以此类推，让Cache节点访问Database, 让Web访问Database节点。

将所有业务逻辑都逐一模拟一次，任何一个环节出现问题，立即发出警告。

3.3. 模拟人工

这里主要监控服务是否可用，可以检查软件的工作情况，涉及测试环节。

通过自动化测试工具辅助监控，例如模拟鼠标点击，键盘输入，可以监控图形界面程序与网页程序。

Windows 监控可以通过 Windows Automation API实现，通过程序控制，能够模拟人工操作软件，实现操作匹配返回结果实现自动化监控

Web页面监控的方案就太多了，比较经典的是Webdriver衍生出的各种工具Selenium - Web Browser Automation最为出名。我通过这个工具模拟用户操作，例如用户注册，登陆，发帖，下单等等，然后匹配返回结果实现自动化监控与报警

3.4. 数据分析

通过数据分析，将故障消灭在故障发生前。举一个例子，开发人员忘记设置redis 时间，虽然程序一直完好工作，但redis内存不断增长，总一天会出现故障。

我们通过采集redis状态信息，分析一段时间内数据变化发现了这个问题。

3.5. 监控与开发

谈到监控很多人认为这是运维的事情，实则不然，不懂运维的测试不是好开发。

开发过程中需要考虑到监控，例如Nginx的status模块， MySQL的show status命令, Redis的info命令，都是为监控预留的。那么你开发的程序是否考虑到了监控这块呢?

你可以通过日志形式或者管道，再或者Socket将程序的运行状态提供给监控采集程序。

4. 总结

好的监控的能让你对系统了如指掌，做到心里有数。有数据才好说话。

第 8 章 分布式计划任务

本章主要通过分布式计划任务设计与实现。

1. 什么是分布式计划任务

首先我们解释一下计划任务，计划任务是指有计划的定时运行或者周期性运行的程序，我们最常见的就是Linux “crontab”与Windows “计划任务程序”，我们也常常借助他们实现我们的计划任务，因它们的时间调度程序非常成熟，无需我们再开发一套。

2. 为什么采用分布式计划任务

起初，我们也跟大多数人一样采用crontab调度程序，但随着项目越来越大，系统越来越复杂，就抱漏出许多问题。

首先是高可用HA需求，当运行计划任务的服务器一旦出现故障，所有的计划任务将停止工作。

其次是性能问题，越来越多的大型计划任务程序出现，对CPU/IO密集操作，单个节点已经不能满足我们的需求。

让计划任务7*24*365不间断运行，必需有一套行之有效的方案才行，我意识到必须开发一个全新的分布式计划任务框架，这样开发人员无需关注怎样实现分布式运行，集中写任务即可。

我首先提出这个框架必需具备几个特性：

分布式计划任务需具备以下特性

1. 故障转移，我们至少使用两个节点，当一个节点出现问题，通过健康状态检查程序，另一个节点会自动接管任务。

2. 分布式运行，一个任务可以运行在多个节点之上，能够同时运行，能够调整运行的前后顺序，能够并发互斥控制。

3. 节点可动态调整，最少两个节点，可以随时新增节点，卸载节点。

4. 状态共享，任务可能会涉及的通信，例如状态同步等等。

3. 何时使用分布式计划任务

何时使用分布式计划任务

1. 遇到性能问题，遇到性能问题你可能首先想到的是分服务器，但很多应用不具备跨服务器运行。

2. 高可用，一个节点出现故障，另一个节点将接管并继续运行。

3. 灾备，你可以将两个或两个以上的计划任务节点分别部署在两个以上的机房，通过HA特性任何一个机房出现故障，其他机房仍会继续运行。

4. 分布式计划任务的部署

两个节点部署

两个节点可以实现“主”、“备”方案，队列（排队）运行方案与并行方案，其中并行方案又分为不同运行于异步运行，还涉及到互斥运行。

两个以上节点部署

多节点建议采用队列运行方案，并行方案，但不建议使用互斥并行方案（浪费资源）

5. 谁来写分布式计划任务

当我们的分布式计划任务框架一旦完成，任务的编写部分非常轻松，只需继承框架程序便具备分布式运行的特性。

6. 怎么实现分布式计划任务

计划任务是一个相当复杂的一块，有操作系统计划任务，有运用程序计划任务，有基于TCP/IP的访问的，有基于命令行访问的，有定时执行的，有周期运行的，还有基于某些条件触发运行的。总之解决计划任务灾备，要比web，cache, database 复杂的多。

图 8.1. 分时方案

严格划分时间片，交替运行计划任务,当主系统宕机后，备用系统仍然工作，只不过处理周期拉长了。缺点：周期延长了

图 8.2. HA 高可用方案

正常情况下主系统工作，备用系统守候，心跳检测发现主系统出现故障，备用传统启动。缺点：单一系统，不能负载均衡，只能垂直扩展（硬件升级），无法水平扩展

图 8.3. 多路心跳方案

上面的HA是三层的基于VIP技术实现，下面这个方案我采用多路心跳，做服务级，进程级，IP与端口级别的心跳检测，做正常情况下主系统工作，备用系统守候，心跳检测发现主系统出现故障，备用传统启动，当再次检测到主系统工作，将执行权交回主系统.缺点：开发复杂，程序健壮性要求高

图 8.4. 任务抢占方案

A,B 两台服务器同时工作，启动需要一前一后，谁先启动谁率先加锁，其他服务器只能等待，他们同时对互斥锁进行监控，一旦发现锁被释放，其他服务谁先抢到谁运行，运行前首先加排他锁。 优点：可以进一步优化实现多服务器横向扩展。 缺点：开发复杂，程序健壮性要求高，有时会出现不释放锁的问题。

图 8.5. 任务轮循或任务轮循+抢占排队方案

任务轮循或任务轮循+抢占排队方案

1. 每个服务器首次启动时加入队列。
2. 每次任务运行首先判断自己是否是当前可运行任务，如果是便运行。
3. 否则检查自己是否在队列中，如果在，便推出，如果不在队列中，便加入队列。

6.1. 分布式互斥锁

互斥锁也叫排它锁，用于并发时管理多进程或多线程同一时刻只能有一个进程或者线程操作一个功能。如果你理解什么是互斥锁，便很容易理解分布式锁。

我们将进程，线程中的锁延伸到互联网上，实现对一个节点运行的进程或线程加锁，解锁操作。这样便能控制节点上进程或线程的并发。

```
+------------------+                            +------------------+
| Server A         |                            | Server B         |
+------------------+      +---------------+     +------------------+
| Thread-1         |      | Cluster Mutex |     | Thread-1         |
| Thread-2         |----> +---------------+ <---| Thread-2         |
| Thread-3         |      | A Thread-2    |     | Thread-3         |
+------------------+      +---------------+     +------------------+
                                  |
                                  V
                          +---------------+
                          | Cluster Mutex |
                          +---------------+
```

```
                          | A Thread-2    |
                          +---------------+
```

上图中有两台服务器上运行任务，其中Server A 的 Thread-2 做了加锁操作，其他程序必须等待它释放锁才能运行。

你会问如果 Server A 宕机怎么办，是否会一直处于被锁状态？我的答案是每个锁都有一个超时阀值，一旦超时便自动解锁。

另外我们还要考虑“域”的问题，你也可以叫它命令空间，主要是防止锁出现同名被覆盖。

6.2. 队列

排队运行

```
+------------------+                            +------------------+
| Server A         |                            | Server B         |
+------------------+      +---------------+     +------------------+
| Thread-1         |      | Task Queue A  |     | Thread-1         |
| Thread-2         |----> +---------------+ <----| Thread-2         |
| Thread-3         |      | A Thread-2    |     | Thread-3         |
+------------------+      | B Thread-1    |     +------------------+
                          | B Thread-3    |
                          | A Thread-3    |
                          +---------------+
                                  |
                                  | <sync>
                                  V
                          +---------------+
                          | Task Queue B  |
                          +---------------+
                          | A Thread-2    |
                          | B Thread-1    |
                          | B Thread-3    |
                          | A Thread-3    |
                          +---------------+
```

从上图中我可以看到Task Queue中排队情况，运行是自上而下的。

注意Task Queue 需要两个节点，它们是主从结构，A 节点实时向 B 节点同步sh状态。如果 A 节点出现故障， B 节点立即取代 A 节点。

6.3. 其他

计划任务可以分布式运行了，但并不能保证万无一失，配套其他服务器也要做调整。例如数据库，缓存等等。

第 9 章 多维度架构之安全

1. 植入式攻击入侵检测解决方案

1.1. 什么是植入式攻击?

什么是植入式攻击，通俗的说就是挂马，通过各种手段将木马上传到你的系统，修改原有程序，或者伪装程序是你很难发现，常住系统等等。

1.2. 为什么骇客会在你的系统里面植入木马?

通常挂马攻击骇客都是有目的的很少会破坏你的系统，而是利用你的系统。

例如，使用你的网络作DDOS攻击，下载你的数据资料卖钱等等

1.3. 什么时候被挂马?

有时你到一家新公司，接手一堆烂摊子，俗称“擦屁股”。这是中国是离职，中国式裁员，中国式工作交接.....的结果，各种奇葩等着你。

你接手第一项工作就是工作交接，最重要的工作可能就是检查系统后门。通常工作交接少有积极配合的，全要靠你自己。

1.4. 在那里挂马的?

在我多年的工作中遇到过很多种形式挂马，有基于Linux的rootkit，有PHP脚本挂马，Java挂马，ASP挂马。通常骇客会植入数据库浏览工具，文件目录管理工具，压缩解压工具等等。

1.5. 谁会在你的系统里挂马?

98%是骇客入侵，1%是内人干的，1%是开后门仅仅为了工作方便。

本文对现有的系统无能为力，只能监控新的入侵植入

1.6. 怎样监控植入式攻击

1.6.1. 程序与数据分离

程序包括脚本，变异文件等等，通常是只读权限

数据是指由程序生成的文件，例如日志

将程序与数据分离，存放在不同目录，设置不同权限,请关注“延伸阅读”中的文章，里面有详细介绍，这里略过。

我们这里关注一旦运行的程序被撰改怎么办，包括入侵进入与合法进入。总之我们要能快速知道那些程序文件被修改。前提是我们要将程序与数据分离，才能更好地监控程序目录。

1.6.2. 监控文件变化

我使用 Incron 监控文件变化

```
# yum install -y incron
# systemctl enable incrond
# systemctl start incrond
```

安装日志推送程序

```
$ git clone https://github.com/netkiller/logging.git
$ cd logging
$ python3 setup.py sdist
$ python3 setup.py install
```

配置触发事件

```
# incrontab -e
/etc IN_MODIFY /srv/bin/monitor.sh $@/$#
/www IN_MODIFY /srv/bin/monitor.sh $@/$#

# incrontab -l
/etc IN_MODIFY /srv/bin/monitor.sh $@/$#
/www IN_MODIFY /srv/bin/monitor.sh $@/$#
```

/srv/bin/monitor.sh 脚本

```
# cat /srv/bin/monitor.sh
#!/bin/bash
echo $@ | /usr/local/bin/rlog -d -H 172.16.0.10 -p 1220 --stdin
```

/etc 与 /www 目录中的任何文件被修改都回运行/srv/bin/monitor.sh脚本，/srv/bin/monitor.sh脚本通过/usr/local/bin/rlog程序将文件路径数据发给远程主机172.16.0.10。

1.6.3. 安装日志收集程序

```
$ git clone https://github.com/netkiller/logging.git
$ cd logging
$ python3 setup.py sdist
$ python3 setup.py install
```

配置收集端端口，编辑文件logging/init.d/ucollection

```
done << EOF
1220 /backup/172.16.0.10/incron.log
1221 /backup/172.16.0.11/incron.log
1222 /backup/172.16.0.12/incron.log
EOF
```

然后根据incron.log给相关管理人员发送邮件或短信警报等等，关于怎么发邮件与短信不再本文谈论范围，有兴趣留意我的官网。

2. Shell 历史记录异地留痕审计与监控

2.1. 什么是Shell历史记录异地留痕与监控

首先谈谈什么是“历史记录异地留痕”，历史记录就是~/.bash_history文件，不同Shell名字可能不同，它会记录每次用户在键盘上敲下的命令，我们可以通过下面命令查询历史记录。

```
$ history | head
 1009  ls /www
 1010  vim Makefile
 1011  cat Makefile
 1012  make index.html
 1013  vim Makefile
 1014  make index.html
 1015  vim Makefile
 1016  make index.html
 1017  vim Makefile
 1018  make index.html

 $ history | tail
 2000  find /tmp/var/
 2001  ll
 2002  cd workspace/Journal/
 2003  s
 2004  ls
 2005  make shell.html
 2006  cat ~/.bash_history
 2007  history
 2008  history | head
 2009  history | tail

 $ cat ~/.bash_history | head -n 100
 cat /etc/issue
cat /etc/resolv.conf
ifconfig
cat /etc/resolv.conf
dmd
df
df -T
cat /etc/fstab
```

```
cat /etc/issue
uname -a
ps ax
cd /srv/
ls
cd workspace/
ls
df
df -T
df
ls
cd ..
ls
```

由于篇幅的限制，我是用了head,tail 命令限制显示长度。

现在我在看看“监控”，监控就是过滤 ~/.bash_history 文件内字符串，达到匹配标准，做出报警操作等等。例如我们发现adduser命令应立即报警，通知相关人员检查。

2.2. 什么要将Shell历史记录异地留痕并监控

首先我们将要用户操作留痕，以方便随时调阅，我们要知道系统管理员做了那些操作，还可用于审计工作。例如我们开发工作中有一个环节就是Code Review (代码审查)，可以帮助我们提前发现BUG，以及不合理做法，甚至是人为恶意植入后门等等。

历史记录异地留痕就是运维工作的 sysop review(运维审查)。

其次是监控，注意这里的~/.bash_history监控并非实时监控，因为只有用户推出shell后才能保存~/.bash_history文件。所以监控是滞后的，但也足够能帮助我们更早的知道系统发生了那些变化。

2.3. 何时做历史记录异地留痕

这个系统可以实时部署，对现有的业务不会影响。

2.4. 在哪里做历史记录异地留痕

历史记录异地留痕分为两个部分，第一个部分是节点，第二部分是收集端，收集段同时还负责监控与报警。节点将收集的数据发送给收集端，然后收集端归档日志。

2.5. 角色与权限

最高权限着负责部署即可

2.6. 怎么实现历史记录异地留痕

2.6.1. 节点配置

首先修改history格式，默认只有行号，我需要记录每一个命令的输入时间点。

```
cat >> /etc/bashrc <<EOF
export HISTTIMEFORMAT="%Y-%m-%d-%H:%M:%S "
EOF
```

此时输入history命令你可以看到时间点

```
# history
  741  2014-12-24-10:06:26 ll
  742  2014-12-24-10:06:40 ls
  743  2014-12-24-10:06:44 ll
  744  2014-12-24-10:06:47 ls
  745  2014-12-24-10:58:13 history
```

2.6.2. 推送端

```
$ git clone https://github.com/netkiller/logging.git
$ cd logging
$ python3 setup.py sdist
$ python3 setup.py install
```

配置启动脚本，打开文件logging/init.d/uhistory

```
HOST=127.0.0.1 #此处为收集端的IP地址

# Port | User
# --------------------
# 配置端口号与用户
done << EOF
1220 neo
1221 jam
1222 sam
EOF
```

2.6.3. 收集端

```
$ git clone https://github.com/netkiller/logging.git
$ cd logging
$ python3 setup.py sdist
$ python3 setup.py install
```

配置收集端端口，编辑文件logging/init.d/ucollection

```
done << EOF
1220 /backup/neo/.bash_history
1221 /backup/jam/.bash_history
1222 /backup/sam/.bash_history
EOF
```

3. 延伸阅读

《日志归档与数据挖掘》

数据库记录安全解决方案

Tomcat 安全配置与性能优化

Linux 系统安全与优化配置

网站防刷方案

PHP 安全与性能

http://netkiller.github.io/storage/incron.html

第 10 章 Shell 高级编程

1. 递归调用

不懂递归不算是合格的程序员

递归调用是一种特殊的嵌套调用，是一个函数在它的函数体内调用它自身称为递归调用。这种函数称为递归函数。

```
#!/bin/bash
#######################################
# Author: Neo <netiller@msn.com>
# Home : http://netkiler.github.io
# Project: https://github.com/oscm/shell
#######################################
domain=$1
#######################################
function include(){
        txt=$1
        for host in $(echo $txt | egrep -o "include:(.+) ")
        do
                txt=$(dig $(echo $host | cut -d":" -f2) txt | grep "v=spf1")
                echo $txt;
                if [ "$(echo $txt | grep "include")" ]; then
                        include "$txt"
                fi
        done
}
function main(){
        spf=$(dig ${domain} txt | grep "v=spf1")
        echo $spf

        if [ "$(echo $spf | grep "include")" ]; then
                include "$spf"
        fi
}

main $domain
```

运行上面的程序

```
$ bash spf.sh 163.com
163.com. 6878 IN TXT "v=spf1 include:spf.163.com -all"
spf.163.com. 16991 IN TXT "v=spf1 include:a.spf.163.com include:b.spf.163.com include:c.spf.163.c
a.spf.163.com. 8001 IN TXT "v=spf1 ip4:220.181.12.0/22 ip4:220.181.31.0/24 ip4:123.125.50.0/24 ip
b.spf.163.com. 10131 IN TXT "v=spf1 ip4:176.34.21.58 ip4:176.34.53.178 ip4:121.195.178.48/28 ip4:
c.spf.163.com. 17199 IN TXT "v=spf1 ip4:223.252.206.0/24 ip4:43.230.90.0/27 -all"
d.spf.163.com. 17615 IN TXT "v=spf1 ip4:123.126.65.0/24 ip4:106.2.88.0/24 ip4:220.181.97.0/24 ip4
```

2. 实现守护进程

无论是C语言还是php/python/perl 通过fork命令实现守护进程，让当前程序进入后台运行，这种手段常常用于服务器软件。

启用 shell 解决重复运行问题，记录PID以便可以停止Shell运维

```
#!/bin/bash
##############################################
# $Id$
# Author: Neo <netiller@msn.com>
# Home : http://netkiler.github.io
# Project: https://github.com/oscm/shell
##############################################
NAME=info
BASEDIR='/www'
PROG=$BASEDIR/bin/$(basename $0)
LOGFILE=/var/tmp/$NAME.log
PIDFILE=/var/tmp/$NAME.pid
##############################################
PHP=/usr/local/webserver/php/bin/php
##############################################
#echo $$
#echo $BASHPID
function start(){
        if [ -f "$PIDFILE" ]; then
                echo $PIDFILE
                exit 2
        fi

        for (( ; ; ))
        do
                cd $BASEDIR/crontab/
                $PHP readfile.php > $LOGFILE
                $PHP chart_gold_silver_xml.php > /dev/null
                sleep 60
        done &
        echo $! > $PIDFILE
}
function stop(){
        [ -f $PIDFILE ] && kill `cat $PIDFILE` && rm -rf $PIDFILE
}
```

```
case "$1" in
  start)
        start
        ;;
  stop)
        stop
        ;;
  status)
        ps ax | grep chart.xml | grep -v grep | grep -v status
        ;;
  restart)
        stop
        start
        ;;
  *)
        echo $"Usage: $0 {start|stop|status|restart}"
        exit 2
esac

exit $?
```

3. 进程间通信

进程间通信就是在不同进程之间传播或交换信息。

脚本具有黑白名单功能，一个进程专门负责采集数据，另一个进程专门负责处理由第一个进程发送过来的数据。

```
#!/bin/bash
########################################
# Homepage: http://netkiller.github.io
# Author: neo <netkiller@msn.com>
########################################
BLACKLIST=/tmp/BLACKLIST.lst
PIPE=/tmp/pipe
pidfile=/tmp/firewall.pid
KEYWORD=XXDD0S
ACCESSLOG=/www/logs/www.example.com/access.$(date +'%Y-%m-%d').log
########################################
if [ -z $1 ]; then
    echo "$0 clear|fw|collect|process|close"
fi

if [ "$1" == "clear" ]; then
    rm -rf $BLACKLIST
    rm -rf $PIPE
    echo "Clear OK!!!"
fi

if [ "$1" == "close" ]; then
        kill `cat $pidfile`
    echo > $pidfile
fi

if [ ! -f $BLACKLIST ]; then
    touch $BLACKLIST
fi

if [ ! -e $PIPE ]; then
    mkfifo $PIPE
fi

if [ "$1" == 'fw' ]; then
    iptables -A OUTPUT -p tcp --dport 2049 -j REJECT
    iptables -A OUTPUT -p tcp -m multiport --dports 22,21 -j REJECT
fi
```

```
if [ "$1" == "collect" ]; then
    killall tail
    for (( ; ; ))
    do
        tail -f $ACCESSLOG | grep $KEYWORD | cut -d ' ' -f1 > $PIPE
    done &
    echo $! > $pidfile
fi

if [ "$1" == "process" ]; then
for (( ; ; ))
do
    while read line
    do
        grep $line ${BLACKLIST}
        if [ $? -eq 1 ] ; then
            echo $line >> ${BLACKLIST}
            iptables -I INPUT -p tcp --dport 80 -s $line -j DROP
        fi
    done < $PIPE
done &
echo $! >> $pidfile
fi
```

首先启动第一个进程，准备接收数据

```
# ipfw process
```

然后启动第二个进程，发送采集数据

```
# ipfw collect
```

这个程序使用管道作为进程见通信手段，所以只能在一个系统下运行，如果改为Socket通信就可以实现跨服务器数据处理

第 11 章 DevOps实施中你可能遇到的问题

1. 什么是DevOps?

首先DevOps 不是一个产品，其次软件工程方法论也不准确。

在 DevOps 模式下，产品，设计，开发，测试和运维团队更紧密地结合在一起，贯穿应用程序的整个生命周期。通过自动化工具替代手工操作，实现快速，高效，安全的测试，构建，部署项目。

1. 可用的软件胜过完备的文档
2. 团队合作胜过需求文档
3. 响应变化胜过遵循流程与计划

2. 为什么会诞生DevOps?

传统软件企业，以软件开发为主，开发部是最大的部门，根据项目分组，下设许多项目组，需求，开发，测试等等组别。运维显得不那么重要。这个模式已经不适合互联网企业。互联网企业通常是设置开发部，测试部，运维部，产品部，运营部，客服等等部门，但这样的组织架构带来了新的问题。

产品部关注用户体验，不考虑性能与开发合理性。开发部门的驱动力通常是“频繁交付新特性”，完成产品部提出的需求，测试部关注的是开发部是否按照需求完成所有的功能，运维部更关注7*24小时无故障运行。从产品->开发->测试->运维过程看似完美，但他们目标不匹配，就在这些部门之间造成了鸿沟，从而减慢了交付业务的速度。

随着管理学的不断完善，例如工商管理，被分成很多纵深领域，行政管理，人事管理，财务管理，营销管理，项目管理……等等。

而软件管理又被细分为：时间管理，范围管理，需求管理，质量管理，风险管理，成本管理.....

由于组织架构的需要，又把人分成很多岗位，每个岗位上紧紧需要一种知识体系。企业按照自身的需要招聘某个领域的人才。

同时我们学校也按照知识体系划分院系，本科教育程专科趋势，不重视通识教育，最终学生紧紧掌握了微观的知识。

如果说哲学是科学的科学，那么 DevOps 就是管理的管理。所以我认为 DevOps 是多维度宏观管理学。

3. DevOps 虽好，为什么难以普及呢?

实施DevOps 第一个遇到的问题就是人才，DevOps 需要经验丰富的跨界人才。第二个问题就是没有案例可循，无法借鉴和参考。

实施DevOps需要具备管理，开发，测试，运维等等背景的人才。每个领域至少也需要三年的积累，至少需要 3+3+3+3 = 12 年工作经验，多少公司员工都比较年轻，普遍在 3~5年。 一般员工工作10年以上，遍开始转向管理岗位，或者寻找其他出路。即使转管理岗的员工紧紧负责开发管理或者测试管理…… 不太可能10年的开发，转运维部重头开始。

我上面说过我们教育模式有问题，本科教育应该培养 “T” 型人才，专科教育培养 “I” 型人才，本科教育呈专科化。学校只教会学生一项技能（如Java 开发）,而没有教会学生如何学习。

在中国企业的年龄歧视 “T” 型人才流失严重。“I”人才只能掌握一项技能解决一个领域的问题，无法完成DevOps 的实施。

DevOps 不是产品，是一种管理思想，每个企业根据自身特点，制定自己的DevOps规范，所以第二个难点就是，没有案例可循，无法参考。

4. 软件工程的历史与进化

传统软件工程学出现的年代互联网还不普及，主要是单机运行的软件，或者C/S结构的软件，其特点是开发周期长，迭代慢，每半年或者一年交付一次。流程主张：

```
需求->设计->开发->测试->交付
```

进入互联网时代，已B/S为主的软件，交付周期缩短到一个月，在传统软件工程做了改进，放弃了瀑布开发模式，提出了快速迭代，螺旋上升，管理上也逐步完善。出现了软件项目管理，CMM5软件开发成熟度模型。

互联网快速发展，使传统软件企业面临挑战，理论上互联网应用程序没有稳定版，新的特性源源不断加入，如果出现稳定版就意味着企业停滞。

互联网企业面临的问题是

1. 需求频繁变更，一天一个想法，需求尚未成熟就开始投入开发软件生命周期短，以各种活动为例，很多功能是一次性的，软件生命周期可能是几周，几个月。
2. 频繁交付新特性，不能像传统软件一样几个月甚至几年升级一次，我们需要应对互联网快速变化，可能需要每周升级一次，甚至每天升级数次。
3. 随时可能回撤，随时做好回撤准备，要支持版本的任意切换
4. 多项目并行开发，并行开发还会产生耦合依赖，升级顺序限制，集成测试更复杂。

随着Web 2.0 和 云计算思想的提出，软件也在发生变化，软件运行不在限于一台物理机，而是多台服务器的集群中，传统的模块或原件，被独立部署在世界各地。

软件的开发面临前所未有的挑战：

1. 异构平台，软件不限于那种操作系统，例如 Unix,Linux,As/400,windows,Mac
2. 语言混合开发，不在紧紧使用一种语言开发软件。每一种语言都有他所在领域的优势。
3. 分布式，软件再也不是只运行在 一个CPU下，软件被分成很多模块被分布式部署到多台服务上。

这时便出现了极限编程，敏捷开发….等等，同时诞生了新的岗位“产品”，新的思想不断提出，但是仍然无法解决面临的问题。

1. 产品部：需如雪片飞来，需求堆积如山
2. 开发部：版本延期，质量问题频发，疲于奔命修复 BUG，刚修复了一个， 又出现新的 Bug
3. 测试部：手工测试，升级后问题爆发，测试环境通过，到生产环境就出问题
4. 运维部：环境不统一，每次部署都是一场灾难，配置易出错，回撤时间长，
5. 团队现状：加班严重，效率地下，每天奔波救火

测试环境无法重现，开发人员直接在线上修改代码，跳过测试直接将代码交给运维升级

运维事故严重影响运营和广告投放

人员流动导致代码丢失

问题的原因在于，他们紧紧从各自部门的角度解决问题，同时 KPI 考核也不合理：

1. 产品从产品的角度解决产品遇到的问题。
2. 开发从开发的角度解决开发遇到的问题。
3. 测试从测试的角度解决测试遇到的问题。
4. 运维从运维的角度解决运维遇到的问题。

实际上现在的软件已经不是当年交付后一个网管就能搞定剩下的工作。同时软件开发交付周期缩的更短，一周甚至每天升级数次，遇到突发事件要做好随时准备升级。

总结这个时期实际上是： 软件项目管理 加 ITSM (IT Service Management) IT服务管理

所以聚焦微观管理解决宏观管理问题的做法是错误的，于是诞生了 DevOps。 DevOps 是多维度宏观管理学，是管理的管理。

5. 为什么很多企业为什么实施 DevOps 以失败告终?

很多企业实施DevOps 紧紧是软件堆砌，根本没有深入理解 DevOps，吧devops 相关的软件全部安装上，然后做系统集成。

这时各部门一片抱怨声:

1. 管理层说：项目管理工具不好用。
2. 产品说：我不用那个Wiki写需求
3. 设计说：版本控制对于PSD这种大文件兼容不好
4. 开发说：我们不用 Docker，而我们也不用 maven
5. 测试说：怎么随意部署环境，我们还没有测试完，就清空数据了。
6. 运维说：我不敢用你的自动发布

可能用户需要打开浏览器数个窗口，频繁切换才能完成具体工作。

时不时就能听到有人在公司的QQ群、微信群、钉钉上有人喊，XXXX 环境又挂了。

改变现有的工作方式是非常痛苦的，任何不合理的流程和工作方式已经使用了多年，习惯已经根深蒂固。

实施devops 需要各部门收集意见，对各个部门培训，等各部门理解了 DevOps 原理和流程后，才能实施。

6. CI 持续集成不是DevOps

Jenkins 不是 DevOps

持续集成是一种软件开发实践，即团队开发成员经常集成他们的工作，通常每个成员每天至少集成一次，也就意味着每天可能会发生多次集成。每次集成都通过自动化的构建（包括编译，发布，自动化测试)来验证，从而尽快地发现集成错误。

持续集成只是 DevOps 中的一个小小的环节，并不是最主要的核心工作。

持续集成可以解决什么问题

1. 能验证代码是否可以正常编译
2. 验证组建或模块是否能够集成
3. 验证自动化测试用例是否正常运行
4. 测试环境的部署

持续集成不能解决什么问题

1. 生产环境的发布
2. 部署失败后回撤
3. 不能构建环境，虽然支持 Docker

持续集成智能单向操作,例如

```
代码->构建->测试->部署  等等
```

持续集成中我们遇到很多问题

例如就是通过 git hook 触发 Jenkins 实现持续集成，自动构建项目。问题来了，任何提交都会触发一次 pipeline 脚本，当项目频繁提交时，第一个构建过程还未运行完毕，第二个进程便启动。导致构建排队，阻塞，同时 pipeline 可能会争夺资源（多个进程读写同一个文件），产生冲突，轻则稍等片刻，重则测试环境崩溃。

另外通过CI 持续集成部署代码也不靠谱，会出现和上面相同问题，例如第一个进程用 scp 复制 jar 包到远程主机，还未传输完成，第二个进程便做同样的操作。

还有 第一个进程重启 tomcat ，tomcat 还未停止退出，第二个请求便发出。最终导致 tomcat 崩溃。

以上的特性，你敢在生产环境上使用吗？一旦发布失败，或者需要回撤，持续集成并没有很好的解决方案。

我认为，持续集成尚不完善，测试环境玩玩可以，生产环境还是不要了。

7. CD 持续交付不是 DevOps

持续集成、持续交付、持续部署是一系列的软件工程实践方法，使用自动化手段达到完成软件。

持续交付(Continuous Delivery)和持续部署(Continuous Deployment)的区别

1. 持续集成(Continuous Integration) 通过将每一次改动都提交到一个模拟产品环境中，使用严格的自动化测试，确保业务应用和服务能符合预期，最终产生构建产物。
2. 持续交付(Continuous Delivery) 通过持续集成产生构建物，确保让软件产品能够快速、安全的部署到产品环境中。持续交付并不是指软件每一个改动都要尽快的部署到产品环境中。它指的是任何的修改都已证明构建物可以在任何时候实施部署。
3. 持续部署(Continuous deployment) 是持续交付的更高阶段即生产阶段，就是将最终的产品发布到线上生产环境，给用户使用。所以当业务开发完成时，经过持续集成，持续交付后，你有信心只需要按一次按钮就能将应用快速并安全的部署到产品环境中。

请不要再混淆持续交付与持续部署了。

8. 自动化部署

我习惯将持续部署(Continuous deployment)称为自动化部署

本章节重点谈自动化部署，每个人对自动化部署都有自己的理解，每个企业对自动化部署的需求也不同。

目前很多云平台开始推出一些列 DevOps 工具，体验了一下，仍然处在初级阶段，也不十分成熟。严格的说他们实现的 CD (持续交付)。

前面讲过持续集成不是 DevOps，这里我要说持续部署也不是 DevOps。自动化部署是从CI/CD中分离出来的，将部署单独提炼出来。

自动化部署远比 CD(Continuous Delivery) 持续交付要复杂，涉及包括

1. 网络层：网络设备管理，负载均衡切换，路由表管理
2. 系统层：基础设施，操作系统，软件运行环境，
3. 软件层：软件部署
4. 缓存层：缓存的刷新
5. 搜索层：重建全文索引
6. 数据层：数据库结构管理，数据库数据管理
7. 日志层：谁，什么时间，做了什么操作，结果怎样
8. 除此之外，管理上还需要提案和审批流程等等

所以 CD （持续交付）解决不了企业的生产环境自动化部署需求，CD紧紧是CI （持续集成）运行完成后，将构建物部署到指定的运行环境中。通常CD并不提供回撤功能，所以极少由企业使用 CD 部署生产环境。

```
Git -> 编译 -> 测试 -> 打包 -> 构建物 -> 部署 -> 运行
```

CI/CD 的流水线作业只能部署单一项目，对于大型网站就无能为例

例如很多大型网站

1. 构建过程非常复杂，不仅仅是一个项目打包， 而是需要多个模块，处理复杂的配置过程。
2. 一次部署多台服务器，每个服务器可能有多个实例，实例间相互依赖关系
3. 需要遵守严格的部署和启动顺序
4. 记录部署日志，文件的新增，覆盖，删除
5. 部署时间点
6. 升级不仅仅是代码，还有数据库，缓存……
7. 需要改变负载均衡设备节点，设置防火墙策略
8. 需要有完备的回撤方案
9. 除此之外好虚考虑增量部署和差异部署，例如部署100mb 以上的大文件，甚至GB尺寸的文件

很多 DevOps 方案注重 Docker，K8s解决方案。但实际情况 Docker 并不适用于所有场景，更多是物理服务器，虚拟机，云主机，刀片服务器…

使用 Docker 的前提是，Docker必须部署在宿主主机上，在云主机中部署 Docker 意义不大。

很多企业大量使用云主机，对 Docker 并无强烈的需求。

运维需要怎样的自动化部署工具

1. 项目管理：升级提案，工作流转，工作审批
2. 备份管理：任何生产环境部署前都需要备份，必须实现增量备份和差异备份。
3. 环境管理：环境部署，基础设施管理
4. 阶段管理：开发，测试，生产
5. 仓库管理：分支切换，分支保护（例如只允许合并不允许提交）
6. 配置管理：每个阶段拥有自己的配置
7. 文件过滤：排除过滤，包含获取，替换过滤（替换指定文件中的内容，用户不同阶段的差异区分）
8. 覆盖删除：覆盖指定文件，删除指定文件
9. 内容优化：Grup, Webpack 优化，压缩js, css,html5, 图片雪碧图…..
10. 自动构建：编译，测试，测试报告，打包
11. 部署管理：节点管理，增量部署，差异部署，md5sum 校验检查
12. 部署脚本：部署前脚本（停止），部署后脚本（启动）或者环境初始化，解决部署依赖
13. 时间线：谁，什么时间，做了部署，可以指定时间点随时回撤到指定版本。
14. 部署日志：谁，什么时间，做了什么操作，产生什么结果
15. 部署报告：生产 Issue或Ticker 报告

持续集成与持续交付和持续部署的关系：

```
持续集成(CI) -> 构建物 --> 持续交付(CD) --> 交付验收环境 （Alpha）--> 验收成功
               \                                                      |
                \...................................................../
                                          |
                                          v
                                     持续部署 ----> 生产环境 (Beta/Preview/Release) ----> 生产环境验
```

9. 收集各部门问题

实施DevOps前需要收集各部门问题

问题如下

1. 产品线都多少条?
2. 同时进行并行开发的多少条?
3. 怎么进行项目管理?
4. 产品团队的情况：怎样管理需求文档，多个产品人员怎样协作
5. 设计团队的情况：都使用什么设计软件，一般文件尺寸多大
6. 开发团队的情况：使用什么语言，什么框架，开发人员数量，采用哪种版本控制，急需解决的问题?
7. 测试团队的情况：测试工具，测试的方法，测试用例怎样管理，人员数量，急需解决的问题?
8. 运维团队的清况：服务器数量，云的使用情况，docker使用情况，运维工具，运维人员，急需解决的问题?
9. 目前最迫切解决的问题是什么?
10. 你的企业目前还面临哪些问题（非技术）?

有了这些数据，在DevOps工具选型是，你才能判断是否符合你的需求。例如很多商用工具的 License 是按照用户数收费的。有些则按照部署节点收费。

9.1. 自运维的需求

例如下面是来自运维的需求

运维团队需要什么呢

1. 合同管理

2. 成本管理
3. 续费管理
4. 问题管理
5. 突发事件管理
6. 环境配置
7. 设备管理
8. 配置管理
9. 自动化部署
10. 监控和报警
11. 备份和恢复

上面大部需求以用Issue/Ticket 凑合，但是有几个功能例如，环境配置，自动化部署，监控/报警，备份/恢复，这些就凑合不了，实打实的硬性需求。如果不能实现这些功能，就不能称为 DevOps。

我们就先从监控说起把，你很发现很多 DevOps 的文章中，不会涉及到监控，但是这是运维的重中之重。

10. 收缩技术栈

技术部门常常会陷入技术思维，恨不得将所有主流技术都使用上，却忽略了他们兼容性，以及对该技术的掌握程度。

当团队没有100%掌握某项技术是，风险是巨大的，我们常常会看到网上有这种文章《XXX踩过的坑》，无疑是拿生产环境练手，为自己的职业生涯打怪升级。

大炮打蚊子，很多需求根本无需使用复杂的技术，最终变成庞然大物。

尽量使用一种技术解决所有问题，而不是使用所有技术解决一种问题。这样技术团队学习起来不会太吃力，且团队人力资源可以共享，测试难度和运维难度都会降低。

10.1. 模块化思维

技术思维另一个误区就是，拆整为零，模块化。例如,用户中心，商品中心，订单中心，物流中心 ……

```
  用户中心 ------ 商品中心,
     |   \         /    |
     |     \    /       |
     |       x          |
     |     /   \        |
     |   /       \      |
 订单中心 ------ 物流中心
```

看这个架构多么清晰

1. 用户中心： 负责用户注册，登录，找回密码
2. 商品中心：商品分类，商品搜索，商品列表，商品展示
3. 订单中心：订单报价，订单合并……
4. 物流中心：对接物流平台……

技术人员的成就感飘飘然，然票票。运维根据需求将上面四个中心使用四台高配置服务器部署起来。

市场部需求

1. 用户登录
2. 浏览商品
3. 下订单
4. 走物流
5. 用户积分+100

平时没有什么问题，订单量一大所有问题都暴漏出来， 积分添加失败，库存数据出错，物流下单失败……

这种模式的问题有很多，例如

1. 运维复杂，部署复杂，配置管理复杂
2. 排查问题难度搞
3. 分离后，通过网络连接，网络存在延迟和超时等等其他不可控因素
4. 分布式事务处理，复杂且难以保证
5. 分布式锁，并发的杀手

任何一个系统都不能简单的进行拆分，抓中拆分同样是我们教育的问题，导致思维方式产生问题。

15年前我就意识到这种问题所在，15年后去一下电商公司面试，发现他们仍然在采用这种模式。

11. 被遗忘的数据库

在持续集成和持续部署中数据库常常被忽略。

实施 DevOps 对于 DBA 都不那些诉求呢?

这里我列举一些DBA的诉求

- 数据库备份与恢复，备份文件的安全
- 数据库结构版本控制
- 数据库快照
- 注入扫描
- 撰改报警
- SQL 审计
- 数据库监控
- 脏数据处理

上面每一项都需要单独拿出来分析，例如监控。

数据库监控有可以细分为

- IP 地址，包括端口，服务
- 同步状态
- 连接数
- 缓存，命中率
- SQL语句调用统计

等等

总之 DBA 需要知道，谁，什么时候，登陆了数据库服务器，做了什么操作。随时可以备份数据，恢复数据。

另外还有数据文件一致性的需求

什么是数据文件一致性？举一个例子，用户头像是一张图片，存储在用户数据表中如下

```
 ID | USERNAME | ICON
-------------------------------
  1 | neo      | /images/neo/Avatar.jpg
```

可能存在数据存在，图片找不到；或者有图片，没有数据的情况。这里只是一个例子，实际场景更复杂，例如银行票据，合同等等。

12. 建立中心仓库

DevOps 需要一个核心仓库，用来管理构开发包，容器，以及建物等等。

仓库可以分为三种类型，分别是

1. 和基础设施库
2. 容器仓库
3. 软件依赖仓库

基础设施库包括: Yum,Apt,Snap

容器仓库包括 Docker, Helm

软件依赖仓库包括

1. Maven
2. Gradle
3. npm
4. PyPI
5. Ruby Gems
6. PHP composer
7. CPAN

为什么需要建立这些仓库呢?

首先构建物是公司的私有资产，不可能放在开放的仓库内。其次，使用外部仓库严重影响构建速度，例如下来速度慢和一些不可控的因素，挂起，闪断等等。

通常我们将私有自建的仓库和DevOps系统放在一起，以加速构建速度。

13. 缓存

缓存可以帮助构建程序显著提高执行速度，DevOps 涉及到的缓存包含

1. 源代码缓存
2. 软件开发包缓存
3. 构建物缓存

另外，软件开发包缓存和构建物缓存的版本通常是递增的，所有无需考虑缓存过期的问题，但是需要考虑下载过程中出现的损害。

常见的损坏包括

1. 源代码版本控制文件损坏，导致代码无法更新
2. 软件开发包依赖文件损坏，导致无法编译
3. 构建物损坏，导致无法部署，启动

14. 安全

DevOps 需要考虑几点安全问题

1. 隔离安全
2. 环境变量安全
3. 日志安全

对于单一用户，这些问题没有那么严重，但是对于多用户系统或基于 SaaS 的 DevOps 的平台来说这就是大问题。 否则会出现 A 用户可以访问 B 用户资源的问题。甚至做出一些恶意操作，下载源码，植入木马等等

第 12 章 Kubernetes & Docker 实施中你会遇到的问题

在项目中实施容器技术，你可以遇到下列问题。

1. 镜像会遇到的问题

目前docker 镜像，没有统一标准，体现在以下几个方面。

1.1. 镜像使用的OS发行版不统一

在使用过程中会遇到过各种版本的 OS。包括 alpine, debian, ubuntu, centos, oraclelinux, redhat 等等……

经过裁剪的 OS 面目全非，不完整

即使是镜像采用 CentOS 母版，很多镜像制作者会给操作系统减肥。经过优化后，已经不是官方版本，在使用过程中你会遇到各种麻烦。例如调试的时候需要 curl,wget,telnet,nslookup 等工具在镜像中没有。甚至 ps, top, free, find, netstat, ifconfig 命令都没有。

很多容器都不带 iptables 所以，即使带有iptables 在容器中修改规则也很麻烦。

1.2. 安装位置不统一

传统OS 以 CentOS为例，有严格的安装规范，例如:

通常安装位置是

- /etc/example 配置文件
- /bin/sbin 二进制文件
- /var/lib/example 数据文件
- /var/log/example 日志文件
- /var/run/example PID 文件
- /etc/sysconfig/example 启动参数文件
- /etc/system.d/example 启动脚本

或者被安装在

- /usr/local/etc 配置文件
- /usr/local/bin 可执行文件
- /usr/local/share 文档

最后一种是独立安装在:

/usr/local/example

容器镜像那可是五花八门，没有统一标准，如果不看 Dockerfile 根本不知道作者将文件安装到了哪里。

常常存储目录被放置在根目录。例如 /data

1.3. Linux 系统也存在BUG

在我的20年执业生涯中是遇到过 Linux 系统有BUG的，还向 Redhat 提交过 BUG。如果你采用的镜像有BUG，你想过怎么去debug 吗?

2. 容器会遇到的问题

2.1. 程序启动的区别

在Linux是一般是采用守护进程方式启动。启动后进入后台，启动采用 systemd 。

容器中启动通常是直接运行，这样的运行方式，相当于你在linux的 Shell 终端直接运行一样，是在前台运行，随时 CTRL + C 或者关闭终端窗口，程序就会退出。容器采用这种方式启动，就是为了让 docker 管理容器，docker 能够感知到容器的当前状态，如果程序退出，docker 将会重新启动这个容器。

守护进程方式需要记录 pid 即父进程ID，用于后面管理该进程，例如可以实现 HUP 信号处理。也就是 reload 操作，不用退出当前程序实现配置文件刷新。处理 HUP 信号，无需关闭 Socker 端口，也不会关闭线程或进程，用户体验更好。

容器是直接运行（前台运行），所以没有 PID 也不能实现 reload 操作。 配置文件更新需要重新启动容器，容器启动瞬间TCP Socker 端口关闭，此时用户会 timeout。甚至该服务可能会引起集群系统的雪崩效应。

很多镜像制作者更趋向使用环境变量传递启动参数。

当然你也可以在容器中使用 systemd ，这样做容器不能直接感知到容器的运行状态，systemctl stop example 后，容器仍然正常。需要做存活和健康检查。通过健康状态判断容器的工作情况。如果处于非健康状态，将该节点从负载均衡节点池中将它踢出去。

Linux 启动一个应用远远比docker 启动一个容器速度要快。因为物理机或者虚拟机的Linux操作系统已经启动，虚拟机也分配了资源，运行可执行文件基本上是瞬间启动。而 docker 启动容器，要分配资源（分配内存和CPU资源，新建文件系统），相当于创建一个虚拟机的过程，最后载入约200MB左右的镜像，并将镜像运行起来，所以启动所需时间较长，有时不可控，尤其是Java应用更为突出。

2.2. 存储面临的问题

传统 Linux 直接操作本地硬盘，IO性能最大化。

私有云还好办公有云处处受限。

自建的 Docker 或 Kubrnetes 可以使用宿主主机资源，公有云只能使用网络文件系统和分布式系统。

这也是我的架构中 KVM，Docker，Kubernetes，物理机混合使用的原因，根据业务场景的需要来选择哪种方案。

物理机上部署 docker 可以分配宿主主机的所有资源，适合做有状态的服务的存储持久化的需求。

私有云 Kubernetes 适合做 CPU密集型运算服务，虽然通过local 卷和 hostPath 可以绑定，但是管理起来不如 Docker 更方便。

NFS 基本是做实验用的，不能用在生产环境。我20年的职业生涯遇到过很多奇葩，例如 NFS 卡顿，NFS 用一段时间后访问不了，或者可以访问，文件内容是旧的等等。

无论是NFS是更先进的分布式文件系统，如果不是 10G以太网，基本都不能用在生产环境。10年前我用4电口1G网卡做端口聚合勉强可以用于生产环境，不过10年前的互联网生态跟当今不同，那时还是以图文为主，确切的说是文字为主，配图还很少。

所以涉及到存储使用分布式文件系统的前提是必须是 10G以上以太网或者8G以上的FC 存储。这样才不会有IO瓶颈。任何分布式文件系统都不可能比本地文件系统稳定，除了速度还有延迟等等。

10GB 电口，光口以太网已经出来十几年了，相对比较便宜，可以使用 4光口 10G网卡，然后做端口聚合，变成 40G 网口。

现在 40G光口交换机都在10-20万之间。一个40G的交换口可以分出四个10GB口。

如果使用40GB以上的以太网，那么总成本可能会超过物理机+虚拟机的解决方案。

2.3. 内部域名DNS

由于在集群环境中容器名称是随机，IP地址是不固定的，甚至端口也是动态的。为了定位到容器的节点，通常集群中带有DNS功能，为每个节点分配一个域名，在其他容器中使用域名即可访问到需要的容器。

看似没有问题，我的职业生涯中就遇到过DNS的问题，bind,dnsmseq我都用过，都出现过事故。解析卡顿，ping www.domain.com 后迟迟解析不出IP。最长一次用了几分钟才解析到IP地址。

所以后面就非常谨慎，配置文件中我们仍然使用域名，因为修改配置文件可能需要 reload 应用，或者重新部署等等。域名写入配置，方便IP地址变更。例如 db.host=db.netkiller.cn 同时我们会在 /etc/hosts 中增加 xxx.xxx.xxx.xxx db.netkiller.cn 。这样主要使用 /etc/hosts 做解析，一旦漏掉 /etc/hosts 配置 DNS 还能工作。

故障分析，DNS 使用 UDP 协议 53 端口，UDP 在网络中传输不会返回状态，有无数种可能导致 DNS 解析失败。例如内部的交换机繁忙，背

板带宽不够（用户存储转发数据包，你可以理解就是交换机的内存），路由的问题等等……

2.4. 容器与网络

相比传统网络，容器中的网络环境是十分复杂的。传统网络中一个数据包仅仅经过路由器，交换机，达到服务器，最多在服务前在增加一些防火墙，负载均衡等设备。

容器网络部分实现方式SDN（软件定义网络）相比物理机（路由器、交换机、无服务）实现相对复杂。容器里面使用了IP转发，端口转发，软路由，lvs，7层负载均衡等等技术…… 调试起来非常复杂。docker 的 iptables 规则很头痛。

例如一个TCP/IP 请求，需要经过多层虚拟网络设备（docker0,bridge0,tun0……）层层转发，再经过4层和7层的各种应用拆包，封包，最终到达容器内部。

有兴趣你可以测试一下对比硬件设备，容器的网络延迟和吞吐量。

容器的任何操作都是占用宿主主机的资源的，所以存在更多不可控的因素。例如CPU负载瞬态变化可以影响虚拟网络，如UDP丢包。而物理机+交换机+路由器的方案不会有任何影响。

在我学习Cisco的时候一直想不通，三层交换机都能启用路由了，为什么不在增加NAT功能？路由器都提供了ACL为什么不增加防火墙功能？为什么要分成三个设备？

为什么在高端的箱式插卡设备里，交换，路由，防火墙都能实现。仅仅是因为价格？

电信级别的设备 24口交换机只会配满足24口交换的CPU和背板，卖出24口的价格。如果增加NAT功能，不仅会影响交换能力，网络设备本

身价格透明且敏感。

箱式网络设备定位园区，城域网，骨干网，不是用于接入服务器。价格空间很大，厂家采用堆料方式开发，功能要满足各种用户需求和各种业务的需要。用途比IDC接入更复杂。

2.5. 容器的管理

传统服务可以通过键盘和显示器本地管理，OpenSSH 远程管理，通过配置还能使用串口。

容器的管理让你抓狂 docker exec 和 kubectl exec 进入后与传统Linux差异非常大，这是镜像制作者造成了。

控制台环境

- 有些镜像没有初始化 shell 只有一个 $ 符号
- 没有彩色显示
- 可能不支持 UTF-8，中文乱码
- 可能不是标准 ANSI/XTerm 终端
- 键盘定义五花八门，可能不是美式104键盘
- 国家和时区并不是东八区，上海
- HOME 目录也是不是 /root

OS 不完整

- 想查看端口情况，发现 netstat 和 ss 命令没有。
- 想查看IP地址，发现 ifconfig, ip 命令没有。
- 想测试IP地址是否畅通，发现 ping, traceroute 没有。
- 想测试URL，发现 curl , wget 没有。

有些镜像 dnf,yum,apk,apt 可以使用，有些镜像把包管理也给阉割了，你想安装上述工具都安装不了。

卧槽！！！一万匹草泥马

然后就自己用 Dockerfile 编译，整出200MB的镜像，卧槽这么大。

你还会发现你编译的程序没有yum/dnf官方制作的好，它们做了编译器的优化。你如果不相信你可以编译一个 nginx 然后看看bin/nginx这个文件的大小，你再使用 dnf install nginx 安装，比较以下两个二进制文件。你会发现你编译出来的二进制文件足有8M大小，而dnf安装的可能只有4M左右。

2.6. 容器与安全

2.6.1. 网络安全

很多容器的镜像中是不包含 iptables 的，所以无法做颗粒度很细的容器内部网络安全设置。即使你制作的镜像带有iptables ，多数容器的侧咯，IP地址和端口是随机变化的。绑定IP地址又带了容器的复杂性。

一旦攻入一个容器，进入容器后，容器与容器间基本是畅通无阻。

2.6.2. 挂马风险

在容器中藏一个后门比物理机更容易，如上文所说很多容器中没有调试相关命令，限制了你排查后门的难度。所以Dockerfile 制作镜像，最好使用官方镜像衍生出你的镜像。

2.6.3. 隔离安全

容器间是隔离安全的，kubernetes 还有明明空间和RBAC等等。但是.....

有时我们为了产生持久化会使用本地卷，或将容器目录挂载到节点宿主主机中。这也带来一个安全问题，一是可能大家使用了相同的文件或目录名，相互覆盖对方的文件。二是不小心将敏感信息写入了到宿主主机，存在信息泄露风险。

除此之外 local volume, hostPath 以及没有用户认证的网络文件系统，都存在这种风险。例如NFS，因为Pod是随机IP地址，NFS服务器无法通过IP地址设置访问规则。

2.7. 容器与监控

谈到监控，跳不开 prometheus（普罗米修斯），它并不能覆盖到所有监控。

我曾经写过一篇文章《监控的艺术》网上可以搜到。

2.8. 容器与CI/CD

在DevOps场景中，使用 docker 或 kubernetes 做 CI/CD 是很扯淡的。

当 git 产生提交后，gitlab/jenkins 启动容器，下载代码，编译，打包，测试，产生构建物，编译 Dockerfile ，上传 docker 镜像到 registry，最后部署到容器执行。

卧槽！！！速度能急死你。

于是乎，我们做了 Cache。 不用每次都 pull 镜像，缓存 Maven 的 .m2 库，不再清理代码（mvn clean）提速不少，测试环境凑合用吧。 注意不mvn clean 有时会编译出错

至于生产环境，我就不说了，有多少人真用CD部署生产环境。

2.9.

3. 人员的问题

现实中真正精通容器应用的人很少，容器实在太复杂。Google 将 Kubernetes 设计成大而全系统，想用 Kubernetes 解决所有问题。它涵盖了几大块。

操作系统，虚拟化，软件定义网络，存储，容器管理，用户体系，权限体系……

我们的大学教育是本科教育专科化，本科教育本应该重视通识教育，我们的教育却按照专科标准教育。本科是面向学术的起点，专科是面向工作，解决实际问题。

你问一个中国大学生他会什么，他会说：我会Java，我会Linux……

反应到工作上，就是程序猿不懂运维知识，运维攻城狮不会写程序。员工更趋向深耕一个领域，很类似现在的医生教育和医院体系，专科化，割裂化，导致很多跨科的疾病难以诊断。

于是我提出了「多维度架构」。

4. 最后总结

使用物理机，虚拟机，学习成本，试错成本，部署成本远远低于容器技术，对于事故排查可能容器需要更多的时间。Google 官方也曾经说过，未来 kubernetes 重点可能会转向虚拟机。

掌握容器技术需要很多跨界知识，对人员的要求非常高，深耕自己领域程序猿或运维工程狮都无法全面掌握容器技术，冒然跟风蹭热度，项目只会变成试炼场，成功员工打怪升级，点亮技能树。

我个人认为容器更适合CPU密集型的业务，前期制作符合你需求的镜像，可能需要花费很长时间，我的架构中 KVM，Docker，Kubernetes，物理机混合使用，根据业务场景的需要来选择最佳方案。

第 13 章 多维度架构之微服务

1. 微服务安全吗?

微服务安全吗? 其实存在很多隐患，常规的做法是将微服务置于私有局域网中，通过网关报漏服务。如果破坏者一旦进入了你的私有局域网中，微服务是及其危险的。

1.1. 配置中心的隐患

配置中心的安全隐患

配置中心有以下几种安全隐患

1. 配置中心报漏在公网IP之下
2. 配置中心没有做用户验证
3. 配置文件中存在敏感信息
4. 明文传输内容

配置有泄漏敏感信息的隐患，你的配置中心是不是也这样?

```
iMac:workspace neo$ curl http://localhost:8888/netkiller-dev-master.json
{"sms":{"gateway":{"url":"https://sms.netkiller.cn/v1","username":"netkiller","password":"123456"
```

给配置中心增加SSL和HTTP认证，可以让配置中心更安全。

```
iMac:resources neo$ curl -i -k https://config:s3cr3t@localhost:8888/netkiller-dev.json
HTTP/2 200
set-cookie: JSESSIONID=9E77660C8DC7669121C8D122A48D8737; Path=/; Secure; HttpOnly
x-content-type-options: nosniff
x-xss-protection: 1; mode=block
cache-control: no-cache, no-store, max-age=0, must-revalidate
pragma: no-cache
expires: 0
strict-transport-security: max-age=31536000 ; includeSubDomains
x-frame-options: DENY
content-type: application/json
content-length: 100
date: Mon, 07 Sep 2020 08:24:39 GMT

{"sms":{"gateway":{"url":"https://sms.netkiller.cn/v1","username":"netkiller","password":"123456"
```

我们将 HTTP2 SSL 应用在配置中心后，就不担心配置文件被嗅探器抓到。

1.2. 注册中心的隐患

注册中心一不小心就被公网IP报漏出去，甚至有被恶意注册的风险。

注册中心有以下几种安全隐患

1. 注册中心没有做用户验证，任何人都能访问
2. 注册中心报漏在公网IP之下，被恶意注册的风险。
3. 从openfeign 访问 euerka server 明文传输内容

你的注册中心是不是这样的?

```
iMac:workspace neo$ curl http://localhost:8761/eureka/apps
<applications>
  <versions__delta>1</versions__delta>
  <apps__hashcode>UP_1_</apps__hashcode>
  <application>
    <name>WEBFLUX</name>
    <instance>
      <instanceId>192.168.3.85:webflux</instanceId>
      <hostName>192.168.3.85</hostName>
      <app>WEBFLUX</app>
      <ipAddr>192.168.3.85</ipAddr>
      <status>UP</status>
      <overriddenstatus>UNKNOWN</overriddenstatus>
      <port enabled="true">8080</port>
      <securePort enabled="false">443</securePort>
      <countryId>1</countryId>
      <dataCenterInfo class="com.netflix.appinfo.InstanceInfo$DefaultDataCenterInfo">
        <name>MyOwn</name>
      </dataCenterInfo>
      <leaseInfo>
        <renewalIntervalInSecs>30</renewalIntervalInSecs>
        <durationInSecs>90</durationInSecs>
        <registrationTimestamp>1599106511367</registrationTimestamp>
        <lastRenewalTimestamp>1599106931380</lastRenewalTimestamp>
        <evictionTimestamp>0</evictionTimestamp>
        <serviceUpTimestamp>1599106511367</serviceUpTimestamp>
      </leaseInfo>
      <metadata>
        <management.port>8080</management.port>
      </metadata>
      <homePageUrl>http://192.168.3.85:8080/</homePageUrl>
      <statusPageUrl>http://192.168.3.85:8080/actuator/info</statusPageUrl>
      <healthCheckUrl>http://192.168.3.85:8080/actuator/health</healthCheckUrl>
      <vipAddress>webflux</vipAddress>
      <secureVipAddress>webflux</secureVipAddress>
      <isCoordinatingDiscoveryServer>false</isCoordinatingDiscoveryServer>
      <lastUpdatedTimestamp>1599106511368</lastUpdatedTimestamp>
      <lastDirtyTimestamp>1599106511299</lastDirtyTimestamp>
      <actionType>ADDED</actionType>
    </instance>
  </application>
</applications>
```

经过安全加固后

Eureka Web 界面进入需要输入用户名和密码，HTTP2 SSL 加密传输页面内容。

https://localhost:8761

```
iMac:resources neo$ curl -k https://eureka:s3cr3t@localhost:8761/eureka/apps
<applications>
  <versions__delta>1</versions__delta>
  <apps__hashcode>UP_2_</apps__hashcode>
```

```
<application>
  <name>MICROSERVICE-RESTFUL</name>
  <instance>
    <instanceId>192.168.3.85:microservice-restful:8081</instanceId>
    <hostName>192.168.3.85</hostName>
    <app>MICROSERVICE-RESTFUL</app>
    <ipAddr>192.168.3.85</ipAddr>
    <status>UP</status>
    <overriddenstatus>UNKNOWN</overriddenstatus>
    <port enabled="true">8081</port>
    <securePort enabled="false">443</securePort>
    <countryId>1</countryId>
    <dataCenterInfo class="com.netflix.appinfo.InstanceInfo$DefaultDataCenterInfo">
      <name>MyOwn</name>
    </dataCenterInfo>
    <leaseInfo>
      <renewalIntervalInSecs>30</renewalIntervalInSecs>
      <durationInSecs>90</durationInSecs>
      <registrationTimestamp>1599532959290</registrationTimestamp>
      <lastRenewalTimestamp>1599533499404</lastRenewalTimestamp>
      <evictionTimestamp>0</evictionTimestamp>
      <serviceUpTimestamp>1599532959290</serviceUpTimestamp>
    </leaseInfo>
    <metadata>
      <management.port>8081</management.port>
    </metadata>
    <homePageUrl>http://192.168.3.85:8081/</homePageUrl>
    <statusPageUrl>http://192.168.3.85:8081/actuator/info</statusPageUrl>
    <healthCheckUrl>http://192.168.3.85:8081/actuator/health</healthCheckUrl>
    <vipAddress>microservice-restful</vipAddress>
    <secureVipAddress>microservice-restful</secureVipAddress>
    <isCoordinatingDiscoveryServer>false</isCoordinatingDiscoveryServer>
    <lastUpdatedTimestamp>1599532959291</lastUpdatedTimestamp>
    <lastDirtyTimestamp>1599532959204</lastDirtyTimestamp>
    <actionType>ADDED</actionType>
  </instance>
</application>
<application>
  <name>OPENFEIGN</name>
  <instance>
    <instanceId>192.168.3.85:openfeign:8088</instanceId>
    <hostName>192.168.3.85</hostName>
    <app>OPENFEIGN</app>
    <ipAddr>192.168.3.85</ipAddr>
    <status>UP</status>
    <overriddenstatus>UNKNOWN</overriddenstatus>
    <port enabled="true">8088</port>
    <securePort enabled="false">443</securePort>
    <countryId>1</countryId>
    <dataCenterInfo class="com.netflix.appinfo.InstanceInfo$DefaultDataCenterInfo">
      <name>MyOwn</name>
    </dataCenterInfo>
    <leaseInfo>
      <renewalIntervalInSecs>30</renewalIntervalInSecs>
      <durationInSecs>90</durationInSecs>
      <registrationTimestamp>1599533216972</registrationTimestamp>
      <lastRenewalTimestamp>1599533517001</lastRenewalTimestamp>
      <evictionTimestamp>0</evictionTimestamp>
      <serviceUpTimestamp>1599533216972</serviceUpTimestamp>
    </leaseInfo>
    <metadata>
      <management.port>8088</management.port>
    </metadata>
    <homePageUrl>http://192.168.3.85:8088/</homePageUrl>
    <statusPageUrl>http://192.168.3.85:8088/actuator/info</statusPageUrl>
    <healthCheckUrl>http://192.168.3.85:8088/actuator/health</healthCheckUrl>
    <vipAddress>openfeign</vipAddress>
    <secureVipAddress>openfeign</secureVipAddress>
    <isCoordinatingDiscoveryServer>false</isCoordinatingDiscoveryServer>
    <lastUpdatedTimestamp>1599533216972</lastUpdatedTimestamp>
    <lastDirtyTimestamp>1599533216920</lastDirtyTimestamp>
    <actionType>ADDED</actionType>
```

```
    </instance>
  </application>
</applications>
```

1.3. Eureka 客户端

Eureka Client 的安全配置与Eureka Server/Config Server 类似

Eureka 客户端有以下几种安全隐患

1. 服务报漏在公网IP之下，任何人都不经过 Eureka Server 和 Openfeign 绕开后直接访问服务
2. 明文传输内容

我们给 Eureka Client 增加 HTTP/2 SSL 然后再注册到 Eureka Server，我通常会关闭 Eureka Client 端口，只保留 SSL 端口。

```
iMac:Architect neo$ curl -k https://eureka:s3cr3t@localhost:8761/eureka/apps
<applications>
  <versions__delta>1</versions__delta>
  <apps__hashcode>UP_2_</apps__hashcode>
  <application>
    <name>MICROSERVICE-RESTFUL</name>
    <instance>
      <instanceId>192.168.3.85:microservice-restful:8081</instanceId>
      <hostName>192.168.3.85</hostName>
      <app>MICROSERVICE-RESTFUL</app>
      <ipAddr>192.168.3.85</ipAddr>
      <status>UP</status>
      <overriddenstatus>UNKNOWN</overriddenstatus>
      <port enabled="false">8081</port>
      <securePort enabled="true">8081</securePort>
      <countryId>1</countryId>
      <dataCenterInfo class="com.netflix.appinfo.InstanceInfo$DefaultDataCenterInfo">
        <name>MyOwn</name>
      </dataCenterInfo>
      <leaseInfo>
        <renewalIntervalInSecs>30</renewalIntervalInSecs>
        <durationInSecs>90</durationInSecs>
        <registrationTimestamp>1599547853553</registrationTimestamp>
        <lastRenewalTimestamp>1599548033559</lastRenewalTimestamp>
        <evictionTimestamp>0</evictionTimestamp>
        <serviceUpTimestamp>1599547853554</serviceUpTimestamp>
      </leaseInfo>
      <metadata>
        <management.port>8081</management.port>
      </metadata>
      <homePageUrl>http://192.168.3.85:8081/</homePageUrl>
      <statusPageUrl>http://192.168.3.85:8081/actuator/info</statusPageUrl>
      <healthCheckUrl>http://192.168.3.85:8081/actuator/health</healthCheckUrl>
      <secureHealthCheckUrl>https://192.168.3.85:8081/actuator/health</secureHealthCheckUrl>
      <vipAddress>microservice-restful</vipAddress>
      <secureVipAddress>microservice-restful</secureVipAddress>
      <isCoordinatingDiscoveryServer>false</isCoordinatingDiscoveryServer>
      <lastUpdatedTimestamp>1599547853554</lastUpdatedTimestamp>
      <lastDirtyTimestamp>1599547853483</lastDirtyTimestamp>
      <actionType>ADDED</actionType>
    </instance>
  </application>
  <application>
    <name>OPENFEIGN</name>
    <instance>
      <instanceId>192.168.3.85:openfeign:8088</instanceId>
      <hostName>192.168.3.85</hostName>
```

```
      <app>OPENFEIGN</app>
      <ipAddr>192.168.3.85</ipAddr>
      <status>UP</status>
      <overriddenstatus>UNKNOWN</overriddenstatus>
      <port enabled="true">8088</port>
      <securePort enabled="true">8088</securePort>
      <countryId>1</countryId>
      <dataCenterInfo class="com.netflix.appinfo.InstanceInfo$DefaultDataCenterInfo">
        <name>MyOwn</name>
      </dataCenterInfo>
      <leaseInfo>
        <renewalIntervalInSecs>30</renewalIntervalInSecs>
        <durationInSecs>90</durationInSecs>
        <registrationTimestamp>1599547953476</registrationTimestamp>
        <lastRenewalTimestamp>1599547953476</lastRenewalTimestamp>
        <evictionTimestamp>0</evictionTimestamp>
        <serviceUpTimestamp>1599547953476</serviceUpTimestamp>
      </leaseInfo>
      <metadata>
        <management.port>8088</management.port>
      </metadata>
      <homePageUrl>http://192.168.3.85:8088/</homePageUrl>
      <statusPageUrl>http://192.168.3.85:8088/actuator/info</statusPageUrl>
      <healthCheckUrl>http://192.168.3.85:8088/actuator/health</healthCheckUrl>
      <secureHealthCheckUrl>https://192.168.3.85:8088/actuator/health</secureHealthCheckUrl>
      <vipAddress>openfeign</vipAddress>
      <secureVipAddress>openfeign</secureVipAddress>
      <isCoordinatingDiscoveryServer>false</isCoordinatingDiscoveryServer>
      <lastUpdatedTimestamp>1599547953476</lastUpdatedTimestamp>
      <lastDirtyTimestamp>1599547953435</lastDirtyTimestamp>
      <actionType>ADDED</actionType>
    </instance>
  </application>
</applications>
```


从上面配置中可以看到 port 已经禁用，也就意味着无法再通过 http:// 访问，securePort 是启用状态，只接受 https:// 访问。

```
      <port enabled="false">8081</port>
      <securePort enabled="true">8081</securePort>
```

最好还要设置防火墙，只允许 Eureka Server 才能访问 Eureka Client。防止通过其他服务做为跳板，进入局域网，直接访问 Eureka Client。

1.4. 最终总结

为了防止不小心 公网IP保留微服务，我们需要将实例与局域网IP地址绑定，这样服务只能从局域网IP访问，即使服务器映射了公网IP地址也不用担心。

禁用 HTTP 访问，全部改为 HTTPS 访问

```
H5 / App
    ^
    |
 HTTP2 ssl
    |
    V
Openfeign <--- HTTP2 ssl ---> Eureka Server
```

```
     ^
     |
     |
 HTTP2 ssl
     |
     |
     V
Eureka Client  <--- HTTP2 ssl ---> Config Server
```

如果可以，尽量为节点增加用户认证。

2. 熔断器解决了什么问题?

3. 微服务的性能

3.1. 微服务的开销

4. 多维度架构之微服务拆分

最近在群里有人问关闭分布式事务的话题，详细听了他们需求后。我呵呵一笑，大约在15年前我就遇到过这种问题。

起因是这样的，这是一个电商系统，架构师给出的架构是这样的：

```
用户中心：负责用户注册，登录，用户信息，钱包管理……
商品中心：负责商品的管理，包括展示，价格和库存管理……
广告中心：负责商品推广，促销……
物流中心：负责订单的物流……
等等中心：负责等等……
```

每个中心都有一个独立域名例如：

```
user.domain.com
product.domain.com
ad.domain.com
search.domain.com
m.domain.com
……
```

这种架构设计会存在一个问题，用户每下一个订单，都需要连接多个中心，做一连串调用，最终完成下订单这个功能。因为用户可能操作过程中终止购物流程，或者不可抗因素导致流程无法继续。为此需要设计了一种分布式事务系统，用来解决事务回滚的问题。

所谓的分布式事务，是指跨服务器实现数据库生成与回滚操作，例如：用户购物，浏览商品，添加购物车，选择物流方式…… 这些数据产生在不同服务器上，如果用户取消订单，数据将依次反向回滚。

无独有偶，另一个跨境电商公司的同事也遇到了这种问题，苦苦找不到解决方案，想起了我，询问我的意见。

有时候你会发现，人们会陷入思维边界的陷阱，全力以赴在错误的方向上，无法自拔。

首先，划分中心的架构思维，之所以会出现这种划分方法，我认为跟我们的教育方式有关，导致了多数人都会沿用这种思维定式。

其次，分布式服务的确能解决他们遇到的问题，能想到分布式事务，证明他们智商没有问题。但分布式事务不是最优解，是最差解决方案。

最后，出现了南辕北辙，在错误方向的道路上越走越远。

4.1. 分布式事务之路

大约在15年前我们也遇到了这个问题，幸好我及时出手纠正了架构设计的误区，从而没有走上分布式事务之路。

那时还没有微服务的概念，也没有容器技术，我们主要使用物理服务器，在服务器上运行多个实例。从BAT高薪挖来的架构师的思路跟上面一样，将应用划分成各种中心，并且要求每个中心都部署在独立物理机上。划分中心这种方式也与当时的开发模式有关，采用敏捷开发，分成多个小组，每个小组负责一个中心，小组间定义好信通接口，然后所有小组马力全开，活就一起开干了。现在想想简单又粗暴，就如人体器官一样，五脏六腑的联系不是通过一条神经实现的，他们的联系十分复杂。所以我们不能单独思考每个中心，然后就认为把它们合起来就是一个整体。

成本控制

随之时间推移，服务器用量越来越多，且服务器上运行的实例长时间都达不到 20% 资源利用率。为此我们开始尝试刀片服务器，和多实例+负载均衡方式运行。

运维的挑战

每个中心都需要与其他中心通信，配置文件非常复杂，这给运维带来不小负担。为此我们开发自动化发布系统。

故障排查难点

因为参与的节点多，系统每天都会出现各种问题，同时每次故障排查及其耗时。例如每个节点都会产生日志，排查故障时，要一个节点一个节点检查。看查看所有节点上的日志。为此我们开发了一个日志同步程序，能够增量和差异同步日志到日志中心。

如果再继续下去，我们一定会去研发分布式事务。

此时有一个更好的机会等着我，于是我提出了离职申请，反正是准备离开了，也不怕得罪人，我想我应该在离职之前把这些问题跟公司说一下。

4.2. 微服务拆分法则

我向公司反映了目前面临的所有问题，并且提出了两个概念：

1. 基于工作流拆分服务：服务的拆分法则，基于工作流拆分服务，确保该工作流运行在一个实例中。
2. 服务器即是服务池：所有物理机应该是一个服务池，根据我们的需求，可以将它部署成任何服务。

4.2.1. 基于工作流拆分服务

上面提出的两点，直到今天也仍然适用，例如在微服务的拆分中。在我的职业生涯中，这两个概念始终在指导我的团队。下面我详细说明两个概念怎样应用到实际的工作中。

我们还以电商系统举例，用户下单购物的工作流，如果是按照中心划分，流程可能是这样的：

用户 –> 商品中心(浏览) –> 搜索中心（过滤）–> 用户中心（添加购物车）–> 物流中心 （物流方式） –> 结算中心（支付结

数据流在，商品中心，搜索中心，用户中心…… 服务器中不断传递，网络延迟，网络超时，网络故障等等任何错误都可能影响用户体验。

如果是运行在一个实例中呢？确切的说，我们需要让工作流运行在一个服务器上，一个CPU、内存和硬盘上。这样就没有分布式事务的需求了，数据库的事务处理解决了所有的问题，就这么简单！！！

基于这种法则，我们将几套工作流归类，放在一个实例中，放在今天就是微服务。同样微服务的拆分也尽量满足一套工作流在一个微服务客户端上，避免请求过程出现，一个微服务调用另一个微服务的情况。

4.2.2. 服务池的概念

服务器即是服务池的意思是，做到服务于服务器无关，与IP地址无关，通过运维手段，可以将服务器部署成任何服务，这样可以最大化利用硬件资源，不至于一些服务器资源闲置，而另一些则满负荷工作。这样更容易调配服务器资源。

这种概念在今天的容器中得到了更好的实现。

4.3. 最后总结

还是那句话架构设计是做减法，不是堆技术。你需要从整体考虑，整体不等于个体的总和，这就是多维度思维。

分布式事务目前有成熟的解决方案，但是能不用在高并发，长工作流的需求上，这种方案增加了系统的复杂度。导致开发复杂，测试难度大，运维难，故障多。

5. 接口安全

5.1. Restful 安全问提

Restful 的通信安全有很多中解决方案,例如

1. HTTP Basic Auth 认证
2. Cooke / Session 认证
3. Token 认证
4. Oauth / OpenID

等等，每一种方案都很成熟，这里不依依解释，如果不了解，请去搜索引擎查找相关资料。这里我谈谈在实施微服务项目中的心得，首先项目采用 Spring cloud 方案，Spring cloud 有自己的RestController 控制器，我们需要遵循他的规范开发，这就限制了很多传统的认证加密方法不能应用到 Spring cloud中。

例如传统restful 使用 POST 方式提交，POST 数据格式如下：

```
name=Neo&age=23&md5=xxxxxxx
```

然后做 token 校验。

而 Spring cloud 使用 raw 格式的数据做POST提交，例如

```
@RequestMapping(value = "/member/create", method =
RequestMethod.POST)
public void create(@RequestBody Member member)
```

我们不想在Spring框架上做额外的改动，又想解决信息的安全问题。

5.2. 第一个阶段采用 HTTP Basic Auth

这个方案简单，实施起来最为方便，因为项目比较紧急，所以就采用了这个方案，这个方案既可以在运维方处理，也可以在开发方处理，对于 Spring boot 只需引入Spring Security 简单配置，立即生效。

实现方式请参考： Spring boot with Spring security

5.3. 第二阶段 HTTP Basic Auth + SSL

上面的方案适合在防火墙内部的服务器间通信，如果跨机房或者在广域网上就不在安全了，通过嗅探器抓包，包括 http basic auth 的用户和密码，以及接口数据没有安全可言。为Web 服务器增加 SSL 证书，可以解决信息安全问提。

证书可以使用CA机构颁发的证书，也可以自己生成证书。

证书可以配置在Web服务器上如Nginx， 实现方式请参考：
http://www.netkiller.cn/www/nginx/conf.html#http2 《Netkiller Web 手札》

也可以配置在 Spring boot 中， 实现方式请参考： Spring boot with HTTPS SSL

这个方案可以满足绝大部分用户的需求。

5.4. 第三阶段 HTTP2 + HTTP Basic Auth + Oauth2

由于需要为手机端提供 restful 服务，之前的方式已经不能满足我们的需求，之前的方式更适合提供私有服务，不适合提供公共服务。所谓私有服务是指它的使用范围限制在企业内部，或者事业部间共享服务，总的来说可以通过防火墙控制服务区域。

对于公共服务 OpenID/Oauth 更适合，我们不关心用户地理位，终端设备的情况。实现方式请参考： Spring boot with Oauth2

5.5. 第三阶段，终极版诞生，HTTP2 + HTTP Basic Auth + Oauth2 + Jwt

1. SSL 双向认证
2. HTTP Basic Auth 认证
3. Oauth2 认证

这是我们最终的方案，双向认证是服务器与客户端两端都需要证书才能通信。

```
App(IOS/Android) -->  SSL 双向认证 --> SLB/Proxy --> Feign Client
```

file:///Users/neo/tmp/spring/security/oauth2.html#oauth2.jwt

第 14 章 多维度架构之远程异地灾备

金融交易系统异地灾备方案

本文从灾备角度出发，并不局限于灾备，而是从多维度让你看到完成灾备需要做哪些工作，解决哪些问题。

灾备不仅仅是运维的工作，一个完善的灾备系统需要多个部门的努力，网络拓扑需要做特别的调整，服务器需要特别的部署，软件方面需要配合灾备做特别的开发，才能满足灾备的需求，最终实现真正的7*24*365灾备，并且尽量做到无人值守故障转移。

1. 背景

随着各种金融交易的推出，金融市场必将非常活跃，交易量的快速增加对交易系统产生了更大的压力，如何使交易系统平稳运行，成为金融公司的头等大事。 随着客户风险意识的增强，保证关键核心业务系统持续成功运作的需要，将发生风险的损失降到最低，保证企业的核心竞争力，金融公司对交易系统的可靠性提出了更高的要求。 目前金融公司普遍安装了热备份系统，达到了一定程度上的备份，但无法应对整个机房或大楼的灾难事故，这就需要在异地建立一套金融交易系统，一旦主系统发生问题，且无法启动热备的情况下，马上切换到异地灾备系统进行交易。

1.1. 建立容灾系统需要考虑哪些因素

与简单地将一部分数据从一台机器拷贝到另一台机器相比，企业内业务数据的复制要复杂的多。灾备系统必须满足以下所有业务需求：

1. 数据的高可用性：灾备系统应具有可靠性，同时灾备系统的计算机系统故障不应影响业务的正常操作，达到 (Recovery Time

Objective) “恢复时间目标”的要求。

2. 减少数据丢失：达到 RPO (Recovery Point Objective)“恢复点目标”的要求。作为灾备的系统，最主要的功能是将交易过程中的数据能在最短的延时情况下，同步到灾备系统，最低限度丢失数据。
3. 高性能：灾备系统不应影响数据源系统的正常运行，应当高效地利用网络，并允许各点采用最优化的方法存取本地数据。
4. 提高投资回报率：利用备份系统来支撑其他业务，以提高系统的利用率

1.2. 目前容灾系统的实现方式

传统的容灾方式

1. 基于存储复制的容灾，通过复制磁盘IO块复制的方式，从生产中心向容灾中心进行数据容灾，根据复制设备的不同，又可以分为：基于主机，基于磁盘阵列，基于智能SAN，虚拟存储设备
2. 基于数据库的容灾，通过复制数据库日志或者数据文件方式，从生产中心向容灾中心进行数据容灾
3. 基于应用的容灾，应用指向2个同时运作的数据中心，在任意一个中心活动情况下继续工作

以上的容灾方案都显得过时，这些老旧的容灾方案依靠IOE(IBM，Oracle, EMC)提供的解决方案，IOE的方案通常是通用很强，你只能按照他们要求实施，IOE并不清楚我们的业务逻辑，所以方案不一定是最好的，另外一旦使用了IOE产品，就会被他们绑架，金钱投入无休止。

随着开源软件技术发展，IOE的产品优势越来越不明显，技术固步自封，影响到了用户甚至成为企业发展的瓶颈，IOE投资回报率没有性价比，不符合当下互联网时代。

1.3. 系统灾难恢复等级划分

系统灾难恢复有国际标准，我国也有自己的系统灾难恢复标准，一般分为六个等级，但我觉得都较为过时，不符合互联网时代 。

我国灾难恢复等级划分国信办文件 2005 “重要信息系统灾难恢复指南”

1. “第1级”：数据介质转移（备份介质异地存放和定期更新）
2. “第2级”：备用场地支持（异地介质存放和备份中心支持）
3. “第3级”：电子传送和部分设备支持
4. “第4级”：电子传送和完整设备支持
5. “第5级”：实时数据传送及完整设备支持
6. “第6级”：零数据丢失

我认为等级划分应该为

灾难恢复等级划分

1. 第一级：冷备份

2. 第二级：双机热设备份（HA高可用, 通常为 Active Standby）

3. 第三级：双活互备份（Active-Active 双机互备）

4. 第四级：负载均衡（随时新增节点，移除节点，横向扩展）

5. 第五级：异地灾备

6. 第六级：全球负载均衡（分布式灾备）

从第二级开始就具备“零数据丢失”，远远高于国家标准与国际标准。一级可能需要人工介入，但从第二级开始都是无人值守，到第三级可以达到7*24*365不停机运转，实现智能故障转移。

1.4. 做灾备你面临最大的挑战是什么?

What are your biggest disaster recovery challenges?

1. Scalability(可扩展性)
2. Availability(可用性)
3. Performance(性能)
4. Flexibility(灵活性)

2. 灾备整体解决方案

双机设备已经推出历史舞台，基于主备的数据中心方案也显得过时，我只推荐双活的备份方案。

2.1. 双活互备方案

双活互备不分主次机房，两个机房同时对外提供服务，当一方出现故障时才会将用户自动切换到另一个机房。

双活互备方案的优点

1. 切换成功率高：灾备系统的核心部件基本处于运行状态，不存在灾备切换时起不来的情况。
2. 切换时间快：由于切换的时候，省去了灾备系统启动各服务性程序，与交易所数据同步等操作，可以进一步缩短切换的时间。
3. 数据实时性高：数据的实时性可以与主系统达到同步。
4. 数据丢失率低：可以保证主系统中只要已经报入交易所的委托，灾备系统都可以获取到。

双活互备方案的缺点

1. Active-Active 可以会出现数据不同步，两机交易服务器数据出现差异
2. 实时同步对网络环境要求比较高，这个不用过于担心，目前ＩＤＣ的网络环境越来越好。

提示

另外需要注意的是当双活启动时，平均分配用户到两套交易系统上，如果一个交易系统出现故障，可能会增加另一个交易系统访问压力，

所以要注意监控两个交易系统服务器的使用情况。

2.2. 三机房互备方案

在双活互备方案基础上，我们还可以进一步扩展，做到三地互备，甚至更多的备份，但考虑到成本一般不会超过三个机房。

两机房也好，三机房也罢，接入线路一定要选择不同供应商。

3. 数据中心网络

满足灾备前提是网络畅通无阻，同时网络拓扑设计能够支持灾备。

数据中心要求

1. 供电要求：双路供电，电力来自不同的发电厂，UPS 后备电源，柴油发电机组

2. 空调要求：通常从地面送风的机房比较好

3. 室内气体监控：数据中心应该具备气体监控，室内粉尘，湿度监控

4. 消防：二氧化碳灭火器

5. 机柜：机柜有很多规格，虽然都能放入机架服务器，但有些比较小，没有提供电源线与网路槽。220V电源与网线混在一起可能造成一定的数据丢包。

6. 网络设备：我常常考察一个机房会看他们的核心出口设备（是否有顶级的Cisio设备）

3.1. 单机房高可用双活互备解决方案

单机房最大的优势是网络连接比较方便，很多公司购买的机柜相邻，可能从机柜上方走线。

图 14.1. 单机房高可用双活互备解决方案

单机房高可用双活互备解决方案

双机热备这种我认为是过时的技术，常常主系统出现故障时，你会发现备用系统无法工作。所以我设计的系统都是AA(Active-Active)所有节点都对外提供服务，能够更早的发现问题。

3.2. 双机房互备异地灾备方案

图 14.2. 双机房异地灾备方案
双机房异地灾备方案

异地灾备通常将两个机房打通，但是由于线路带宽有限(通常是1G双绞线或光纤)我能做太复杂连接。通常我们将这条线主要用户数据复制，状态同步，其他服务器独立工作。

3.3. 三机房互备异地灾备方案

图 14.3. 三机房互备异地灾备方案
三机房互备异地灾备方案

三机房安全级别更高，采用三角路由方案，任何时刻都有两条链路是畅通的，通过路由表优化决定一下跳，而且可以绕过故障节点。

三机房有三个入口，通过智能DNS将用户解析到距离自己最近的节点上，用户也可以在交易软件端手动选择。

三机房的数据库采用环形复制

4. 服务器部署

一旦数据中心网络部分完成，下一步就是部署各种服务器，同时服务器的部署必须满足灾备的要求，否则无法实现灾备切换与故障转移。

服务器与交换机连接，按照服务器重要程度或者是否具备故障转移，有两种方案，如具备分布式部署的服务器可能只连接一条网线，有些服务器无法分布式部署，我们是两个网卡分别连接两条网线到两台交换机，同时网卡绑定为一个适配器。

4.1. 网站

网站提供新闻，资讯，开户，转帐，活动，报表，在线客服等等功能

图 14.4. 动态页面方案
动态页面方案

上面是动态页面方案，分为SSL服务器，WEB服务器，API接口服务器，Admin服务器。全部具备水平扩展与动态伸缩。下面分别详细说明他们的功能，为什么这样部署。

动态页面服务器

1. SSL：负责处理用户请求的加密与解密，我们将它与WEB分离，这样能够更好的伸缩扩展，同时提供故障转移功能，我们可以部署很多台这样的服务器。在不同城市，然后通过智能DNS将用户解析到距离用户最近的节点上。

2. WEB：动态页面服务器

3. API： Service 服务器，可以通过SOAP, JSON, MQ等等方式访问，WEB任何操作都要请求API服务器完成，WEB服务器不允许

直接操作数据库，全交由API处理。API具有运用层防火墙的功能与ACL访问控制列表。

4. Admin：网站后台

5. HA：负载均衡软件或设备

与门户网站不同，我们的网站访问量并不大，性能瓶颈基本不存在，更多是考虑高可用。

静态页面以及CDN部门这里就不谈了，非常简单，有兴趣可能到 http://netkiller.github.io/ 找相关资料。

4.2. 数据源

在交易过程中出现报盘中断是大忌，我们必须尽一切努力，让交易系统能源源不断的接收价格数据。为此我需要重点考虑数据源的灾备。

图 14.5. 数据源灾备解决方案

数据源灾备解决方案

我们设计了一个DataFeed程序，能够多方接收数据，因为每个数据源提供的产品不同。我们在这里可以合并，筛选，过滤等等。这个应用可以实现水平扩展，我们可以放在2个以上的机房当中。

接下来在交易服务器中，设置多个数据源分别指向Data Feed 1,2~n。与Trader相同机房的DataFeed的优先级总是最高。

4.3. 数据库

数据的部署要考虑几个问题，我一直不建议主备方案，而是采用双活方案。所以我们要考虑两个机房或者三个机房他们数据复制问题，性能瓶颈问题，数据安全问题等等

图 14.6. 数据库灾备解决方案

数据库灾备解决方案

上图我的方案中采用了Master-Master双活方案，同时每个Master都会有一个Slave。Master 主要用于主要的业务，Slave主要用户数据分析，报表查询等等非实时，且长时间允许等等查询服务

数据库访问需要经过SLB服务器，为数据库提供负载均衡，故障转移，水平扩展等等功能。

所有的数据操作只能通过API完成，不能直接连接数据库并通过SQL存取，上面已经提到过，这里不再多讲。

提示

如果是三个以上机房，将采用环形复制，大致原理Master A向Master B复制，Master B向 Master C复制，Master C在向Master A 复制，形成一个环路。

如果数据库查询如果出现瓶颈，增加Slave即可。

5. 软件开发

软件的设计都要围绕着灾备进行，不能分布式运行，就不合格。

5.1. 交易软件分布式交易

交易软件通常使用TCP私有协议，与我们常见的HTTP协议不同，HTTP是无状态，即用户请求-处理数据-渲染HTML页面-销毁并释放内存，我们很容易开发基于负载均衡的横向水平扩展的应用程序。 而交易软件采用的是持久TCP连接，实时传输用户请求，并将结果会送给用户，中间过程不允许中款，且用户状态数据存储在该交易服务器的内存中，如果我们采用常规手段做负载均衡，就会出现用户被分配到另一台服务器上，而无法继续交易。

我们首先要做的就是将用户状态持久化，有点类似我们在HTTP应用中将Session持久化方法，将这些数据存储到公共的共享服务器上。同时该服务器也需要做高可用等等，保证无单点故障。

第二步，我们解决分布式交易，分布式交易是指可以在交易服务器上任意一个节点上均可交易。方法有很多，复杂程度也不同。

通常解决分布式运行的方法，无非是同步|互斥排他锁|共享存储等等，其实就是两多线程的很多技术扩展到了互联网上。使用节点同步解决共享内存访问问题，使用远程锁解决解决多节点访问问题等等。

5.1.1. 分布式交易解决方案一

解决挂单问题，通常交易是即时完成的，较难处理的就是挂单。

下面的方案是最简单的，开发难度最低。

数据库表中增加一个字段例如

```
-----------------------------------------
Id  | node | status | xxxxx
-----------------------------------------
1 | trader1 | N | xxxxx
2 | trader2 | Y | xxxxx
-----------------------------------------
```

trader1 表示该记录产生在trader1 必须在Trader 1上完成成交

trader2 表示该记录产生在trader2 必须在Trader 2上完成成交

测试用例

1. Trader1 上登陆，下单，然后退出
2. 在Trader2 上登陆如果1 | trader1 | xxxxx 没有成交，将node 改为trader2 同时将挂单数据载入到Trader2服务器的内存中。
3. Trader1 的中的挂单将失效
4. 如果登陆Trader2前1 | trader1 | xxxxx 已经成交，就放弃载入内存。

提示

更新记录状态必须增加node条件

```
Update tab set status=Y where id=1 and node =<my node> …
```

5.1.2. 分布式交易解决方案一

双向通知的含义是Trader 1状态发生改变会通知Trader 2， 反之Trader 2状态发生改变会通知 Trader 1。

图 14.7. 双向通知解决方案
双向通知解决方案

上图可以看出通知与数据库持久关系图，在交易周期内实时接收通知，一旦收到通知，便根据通知中的指令做出反应，改变当前交易服务器的状态。

这种设计可以采用日志复制，然后重做日志的方式，非常累西数据库同步技术，缺点是可能有延迟。设计需要考虑进去。如果超过三个节点就要采用环形复制

我曾经想过通过IP组播，广播技术实现通知发布，这样就可以不设置目的地地址，同时增加了部署的灵活性，而且对于广播节点数量没有限制。比较适合三个以上的机房部署。

5.1.3. 分布式交易解决方案一

消息对列，即将所有交易服务器接入到消息队列中，所有通知均推送进入消息队列，由消息队列服务器统一管理通知消息，同时消息具备持久化。

图 14.8. 消息对列解决方案
消息对列解决方案

这种方法将消息交换MQ去处理，免去了自行开发消息中间件，目前MQ技术也比较成熟。

5.2. 交易终端

5.2.1. 用户分流

用户分流是指，不同用户登录到不同的服务器上进行交易。因为我们的交易系统是可以水平扩展的，可以有很多台交易服务器并且都是Active状态，任意一台服务器都能登陆交易，我们可以使用哈希算法基于用户ID，将用户分流到指定服务器。

用户分流还可以使用智能DNS技术，将用户分流到距离自己最近的交易服务器上。

还可以根据路由TTL值与Ping值进行优先级分配。

5.2.2. 会话保持

交易终端与交易服务器一旦连接，出现超时中断，人为退出，下一次连接还应该上一次连接的那台交易服务器。

如果是交易服务器无法连接将会重新分配一台，并记录配置

如果交易服务器采用的是双向通知，或者消息队列技术，可不必记录登陆服务器，可能按照最小连连接数算法分配服务器。

5.3. API 应用程序接口

上文反复提到了API，关于API请参考我的另一边文章 《CVS开发框架》，这里不再引用。

图 14.9. CVS开发框架

CVS开发框架

框架讲述的设计思路，你可以使用你最熟悉语言技术重新实现。

5.4. 大数据的问题

我们这个行业不是门户网站，所以数据量不会非常大，但我们要未雨绸缪，我们在设计阶段就将这些问题考虑进去，使将来不会困扰我们。

数据量最大的当属交易数据，有些金融产品双向交易，交易时间长，所以交易数据的累积相当可观。

虽然你可以定时删除，但我不建议。我认为交易数据需要永久保留，为什么？ 当交易数据积累到一定数量级，我们就可以从多角度数据挖掘，为决策提供数据支持。

5.4.1. 第一个阶分区表

第一个阶段数据分区存储，实施规划是五年之内。五年之内你可以使用分区这种技术存储数据。分许能够保持索引的连续，方便复杂的数据库查询。

5.4.2. 第二个阶分库分表

当数据庞大到无论怎么优化都无法提高查询性能是，这时你就要考虑数据库拆分了。

数据库拆分需要遵循几个原则，不仅仅是按照年份或月份分库分表。

数据库拆分

1. 基于用户拆分，要能保证查询某个用户的数据不需要跨库，也不需要联合多表查询，这样会降低查询效率。
2. 基于业务拆分，某些业务所用到的表，集中放在一台服务器上，保证数据查询不需要跨库，或者联合多表查询。

图 14.10. 传统的分表方案

传统的分表方案

传统方法，将数据日期切分，这回带来索引的不连续问题，且查询时必须加带日期，以便定位到指定的表中。如果需要做数据挖掘，需要统计四年的数据，必须查询四次，效率大打折扣。

图 14.11. 推荐的分表方案
推荐的分表方案

新的拆分方案，通过哈西算法均匀的将用户分配到指定数据库中或表中。这样所有针对该用户的操作都能在同一个数据库中完成。

图 14.12. 基于功能分表方案
基于功能分表方案

根据不同的功能拆分数据库或表，也是一种非常不错选择。

5.5. 数据校对

由于网络传输丢包和延迟等客观情况，当交易数据量比较大且并发请求过多时，可能导致数据无法同步成功或丢失，造成两个交易系统数据不一致。 两个交易数据库的数据不对等，为了解决该问题，可采开发一个数据校对工具，对数据进行比对和修复。确保数据一致性和完整性。

6. 自动化运维

自动化运维保障服务器快速部署，改善灾备的响应速度。尤其是需要重建某些服务器的时候。

自动化运维应具备

1. 自动化安装
2. 自动化配置
3. 自动化运行
4. 自动化备份
5. 自动化诊断
6. 自动化监控

自动化运维可以降低人为因素产生的故障，如误删除。

7. 灾备培训和演练

演练的主要目的是，验证两个交易系统能够互备，相互做对方的备份，我们将当前登录的系统称为主系统，另一次叫做灾备系统（仅仅是观看的角度）。

另一个目的是，双活系统能够自动规避很多小灾难，灾难没有扩大前系统已经自动修复了，长此以往系统管理员就会麻木，懒惰，侥幸。当更大的故障发生或双活不能正常工作时，由于管理员长期不再状态，就会产生难以预期的后果。

7.1. 培训内容

1. 灾备原理
2. 灾备中心系统结构（系统部署图、网络结构图）
3. 灾备系统运营维护
4. 公司操作流程
5. 切换操作流程
6. 数据检查与验证

7.1.1. 培训对象

公司灾备应急小组所有成员

其他感兴趣员工

7.1.2. 操作流程

公司操作流程

1. 报告灾难发生并申请灾备切换
2. 确认灾难发生并同意灾备切换（主交易中心无法进行正常交易）
3. 获取授权
4. 做切换前准备
5. 切换到灾备系统
6. 切换后检查
7. 切换后报告
8. 发布消息（通知营业部、客户）
9. 进行正常交易

7.2. 演练环境设置

业务准备

1. 确定演练时间

2. 通知参与演练的客户

3. 通知参与演练的业务部门

技术准备

1. 做好演练前的数据备份

2. 检查灾备各应用服务状态

参与部门

1. 开发部门

2. 测试部门

3. 业务部门

模拟灾难: 在演练开始时间，关闭主交易中心的所有的应用，模拟灾难发生。此时，公司应完全按照“公司操作流程”和“灾备系统切换流程”来切换灾备系统。

演练过程: 见“切换操作”和“公司操作流程”

7.3. 演练级别与方式

演练级别定义

1. 接入线路故障
2. 网络设备故障
3. 服务器故障
4. 数据库故障

演练的方式有两种

演练方式

1. 有计划的演练，通知各部门，灾备应急小组人员各就位，然后开始演练
2. 突击演练，在没有通知其他部门的情况下，临时启动演练，抢修主交易系统。

第一种方式适合灾备演练初期，因为各部门都有准备，是在良好的环境下进行，大家都坐在电脑前，看着严控数据，有条不紊按部就班。

但故障是随时都能出现的，是无法预知的。所以我们还需要实行“突击演练”。“突击演练”最近接真是故障发生的场景，可能大家会手忙脚乱，打破之前的各种流程。你会接到各部门领导的电话，可能你当时

还在睡觉或外出路上，你没有地方上网，你的脑子一片空白，这才是真实场景。

7.4. 开始演练

7.4.1. 切换前操作

切换前操作

1. 确认数据库已经备份无误
2. 检查监控系统是否正工作，短信网关是否正常工作

主要的工作室数据备份，其他工作可能作为演练的一部们，工作流程等各种问题在演练中暴漏出来，才能对后面的工作改进有所帮助。

7.4.2. 切换操作

双活互备切换操作，要进行两次，第一次将A机房视为主机房，从A向灾备机房B切换。完成后检查无误，再反向操作一次。

切换要模拟各种故障，通常采用的手段就是拔网线，关闭服务器，关闭应用，使其每条灾备链路都能跑通。

数据中心故障

1. 模拟机房停电，将另一侧灾备机房关机

2. 接入链路故障

3. 防火墙设备故障

4. 交换机故障设备故障

服务器故障

1. 网站故障
2. 数据源故障
3. 交易服务器故障
4. 数据库故障

切换过程中，每一次动作，都应该收到来自监控系统的报警并精确描述的故障。这也是完善监控系统绝佳机会。

7.4.3. 切换后检查

由于数据实时采集同步存在延时以及线路中断可能，导致同步时存在数据丢失，故灾备系统中的数据与主交易系统的数据可能存在不一致，故客户端连上灾备系统后，需要重新登录，获取灾备系统中的数据进行确认，在确认正确后，即可重新进行交易。

检查项目

1. 网站访问压力是否正常
2. 交易系统的压力是否正常
3. 数据同步是否正常
4. 数据源是否正常

测试部门介入项目

1. 开户，入金，出金
2. 行情数据
3. 各种交易情况验证
4. 交易记录
5. 各种报表
6. 多方面的数据核对

7.5. 演练结果检查

应用系统服务状态检查

1. 主库备库数据是否一致
2. 接口能否正常工作
3. 网上交易能否正常进行
4. 交易压力是否正常

公司需要进行多方面的数据核对，包括

1. 委托单是否一致,委托回报是否完整
2. 成交回报是否完整,交易记录是否完整
3. 行情数据是否正确
4. 客户资金是否正常

7.6. 可能存在的风险

当主交易系统切换到灾备系统后，必然会有一些影响业务的情况会发生，公司需要事先有所准备，并制定相关应急预案。

7.6.1. 主交易系统短期无法恢复

由于发生重大事件，可能造成灾害发生时切换到灾备中心的交易系统无法切换回主交易系统。导致交易需要长时间在灾备中心运行，对灾备中心形成压力。对于此情况，公司应该制定详细的预案，保证灾备中心能承担起正常交易压力，使用户交易正常进行。 当确认灾备中心无法在短时间切换回主交易中心时，公司应立即启动预案，并和服务

部门联系，请求技术支持。通过增加服务器、增加访问带宽、交易数据备份等方式，使灾备中心转换为主交易系统，长期承担主交易系统交易工作。

7.6.2. 切换灾备系统后业务的影响

当主交易系统需要切换到灾备系统时，为避免一部分人仍然连接到主交易系统，一部分人连接到灾备系统，切换前必须关闭主交易系统的应用服务器或者段外网络。

在切换以后，有部分业务将受影响，以下为部分情况：

1. 各类终端需要重新登录，如果有业务流水没有同步到灾备系统，则会丢失部分业务数据。
2. 客户的尚未触发的条件单不再有效也不会再触发，这个必须和客户声明，避免引起纠纷；
3. 第三方交易系统在灾备切换后，也需要进行切换，否则会无法使用。
4.

7.6.3. 数据不同步产生的影响

由于数据同步时，存在延时，故有可能灾备系统和主系统之间线路中断后，有部分已经发生的业务没有同步到灾备系统中，部分业务也会受些影响， 故客户端切换到灾备系统中时需要重新登录系统，以便重新查询数据库中的数据进行核对。

相关影响如下：

1. 双活互备相互切换，导致两套数据中的数据不一致，系统管理员可以重新同步数据库；

2. 主交易系统中已经生成但没有发往交易所的委托单没有同步，此时灾备系统中将没有这笔记录，且资金是不冻结的；
3. 主交易系统中已经发往交易所的委托单没有同步，此时灾备系统中将没有这笔记录，且资金是不冻结的，需要补发此笔委托的委托回报；

7.7. 灾备系统备份

当灾备中心承担交易时，灾备系统中的所有交易数据将进行备份，供事后的查询和备案。公司可以采用热备系统，备份灾备中心的交易数据。并做好灾备中心数据及时转移到安全位置。确保灾备中心数据不会丢失。

7.8. 系统运营维护

应用系统服务状态检查

1. 每日实时监控
2. 每日无人值守备份文件传输到备机上
3. 保证交易系统升级版本相同
4. 每月做一次灾备切换预演，确保备份系统能正常工作
5. 严重灾难记录

8. FAQ

8.1. 实现双活最大的障碍是什么?

实现双活最大的障碍是人，不是技术。你是否有足够的权限起到很大的作用。

很多企业动不动讲流程，跨部门沟通十分困难，甚至设置了很多部门防火墙（无形的墙），部门领导平级更是问题，这样会无休止争论下去，无法决策。

另外很多企业管理层不懂技术，无法做决策，有些领导更是怕担责任。

开发人员都会抵触你方案，因为在现有的稳定系统上增加改动且他并不理解的工作，后续的测试，验收都受到影响。

运维也很不情愿，只要是生产环境任何调整都存在一定的风险。

如果你的企业向上面一样，企业已经在走下坡路了，不做事就不出事，拒绝创新.....

总之，装B的人不在少数。

Many people think they are full of niubility and like to play zhuangbility, which only reflect their shability and erbility.

8.2. 双活怎么解决数据冲突问题

此文章已发布，收到很多读者的提问，双活就是双写那麽怎么解决同时写入时的数据冲突问题。

如有下面两个服务器，每个服务器有各自的ID，它们产生的数据类似下面

```
A服务器：1, 3, 5, 7
B服务器：2, 4, 6, 8
```

如果是三地数据中心，将产生类似如下的数据

```
A服务器：1, 4, 7, 10
B服务器：2, 5, 8, 11
C服务器：3, 6, 9, 12
```

第 15 章 多维度架构之应用防火墙

1. 什么是应用防火墙

应用防火墙用于保护应用及服务不受恶意访问和攻击。

应用防火墙有别于网络防火墙，防火墙防火墙偏重对IP地址和端口端访问控制。

应用防火墙有有别于7层防火墙，7层防火墙虽然能实现拆包，根据协议，做出访问控制。

应用防火墙的核心功能除了局别7层防火墙的特点，颗粒度可以做到更细。

例如开发过程中我们有很多需求，直接在功能模块中实现。所谓应用防火墙就是将这些功能做成一个独立模块，实现共享和复用。

2. 功能需求

2.1. 计数器

计数器的需求很常见，功能简单，就是记录访问数量，计数器也是水军主要战场。

计数器需求:

1. 阅读量
2. 点赞
3. 喜欢
4. 回复数
5. 转发
6. 完播

对于网防火墙可以通过IP访问策略进行封杀，但是我国由于IP地址有限，主要的上网方式是NAT（网络地址转换），例如一个公司的办公室内所有电脑都是通过一个IP地址出去的。封杀IP地址容易误伤。

使用应用防火墙就容易很多，可以使用用户+COOKIE+IP地址的方案。

2.2. 访问控制列表 ACL

访问控制即“通过”，“拒绝”

1. 黑名单

2. 白名单

2.3. 用户认证

用户认证模块化，通过插件可以支持多种用户认证

1. AAA
2. LDAP
3. MySQL

2.4. 协议

应用防火墙无需拆包，因为我们是直接调用他的API。

1. IP地址，端口号
2. URL(GET)
3. POST
4. Cookie
5. HTTP Header
6. 协议(HTTP,JASON,AJAX,SOAP,XML-RPM...)

3. 简单实现

应用防火墙我提供了一个思路，不便提供代码。

下面的代码是10年前写的，没有100%实现，因为该代码不会影响竞业，供大家参考。

```
<?php
/*
* ====================================
* Website: http://netkiller.github.com
* Author: neo <netkiller@msn.com>
* Email: netkiller@msn.com
* ====================================
*/

class Logging {
        protected $file;
        public function __construct($logfile = "/tmp/debug.log"){
                $this->file = fopen($logfile,"a+");
        }
        public function __destruct() {
        //fclose($this->file);
    }
        public function close() {
        fclose($this->file);
    }
        private function write($msg){
                        fwrite($this->file,date('Y-m-d H:i:s').' '.$msg."\r\n");
        }
        public function info($msg){
                $this->write(__FUNCTION__.' '.$msg);
        }
        public function warning($msg){
                $this->write(__FUNCTION__.' '.$msg);
        }
        public function error($msg){
                $this->write(__FUNCTION__.' '.$msg);
        }
        public function debug($msg){
                $this->write(__FUNCTION__.' '.$msg);
        }

}

class Permission{
        protected $_PERMISSION = array();

        public function __construct($login){
                $test =
                array(
                        'neo' => array(
                                'News'=> array(
                                        'add' => 'Y',
                                        'remove' => 'N',
                                        'update' => 'Y'
                                        ),
                                'RSS'=> array(
                                        'add' => 'Y',
                                        'remove' => 'N',
                                        'update' => 'Y'
                                        )
```

```
                                ),
                        'jam' => array(
                                'News'=> array(
                                        'add' => 'Y',
                                        'remove' => 'N',
                                        'update' => 'Y'
                                        ),
                                'RSS'=> array(
                                        'add' => 'Y',
                                        'remove' => 'N',
                                        'update' => 'Y'
                                        )
                                )
                );
                //print_r($test);
                $this->load($test[$login]);
        }
        public function load($arr){
                $this->_PERMISSION = $arr;
        }

        public function is_allowed($class, $fun){
                $class  = trim($class);
                $fun    = trim($fun);
                //echo $class, $fun;
                //print_r($this->_PERMISSION);
                if(array_key_exists($class,$this->_PERMISSION)){
                        if(array_key_exists($fun,$this->_PERMISSION[$class])){
                                if($this->_PERMISSION[$class][$fun] == 'Y') return true;
                                //return in_array("Y",$this->_PERMISSION[$class][$fun]);
                        }
                }
                return false;
        }
        public function is_denied($class, $fun){
                return (!$this->is_allowed($class, $fun));
        }
        public function scan(){
                return true;
        }
}

class News extends Permission{

        private $logging;
        public function __construct(){
                parent::__construct('neo');
                $this->logging = new Logging('/tmp/news.log');
        }
        public function __destruct() {
                $this->logging->debug('news->get permission denied!!!');
                $this->logging->close();
    }
        public function add(){
                if(!$this->is_allowed(__CLASS__,__FUNCTION__)) return;
                print("Allowed!!! \r\n");
                $this->logging->info('news->add ok');
        }
        public function get(){
                if( $this->is_denied(__CLASS__,__FUNCTION__)) {
                        print("Denied!!! \r\n");
                        $this->logging->warning('news->get permission denied!!!');
                }

        }
}
```

```
$news = new News();
$news->add();
$news->get();
```

3.1. 权限控制与实现

权限来自下面数组数据，这里仅仅提供一个例子，管理权限你可以单独实现一个class，实现供权限管理功能，最终后转化为下面的数据结构即可。例如你可以将权限写入数据库，最终拼装如下数字让Permission顺利load即可。

```
array(
                        'neo' => array(
                                'News'=> array(
                                        'add' => 'Y',
                                        'remove' => 'N',
                                        'update' => 'Y'
                                        ),
                                'RSS'=> array(
                                        'add' => 'Y',
                                        'remove' => 'N',
                                        'update' => 'Y'
                                        )
                                ),
                        'jam' => array(
                                'News'=> array(
                                        'add' => 'Y',
                                        'remove' => 'N',
                                        'update' => 'Y'
                                        ),
                                'RSS'=> array(
                                        'add' => 'Y',
                                        'remove' => 'N',
                                        'update' => 'Y'
                                        )
                                )
                );
```

public function is_allowed($class, $fun) 用户判断权限是否合法。

3.2. 演示

这里提供了一个 News 类，用于演示怎样控制每个function的权限。

同时还提供了一个简单的 Logging 类用于记录程序运行日志。

有了上面的例子就可以将News应用于SOAP一类Web Service上，用来控制每个方法的权限

3.3. 增加7 Layer防火墙

上面仅仅对于方法控制权限，接下来我们为程序增加7层防火墙功能

```php
<?php
/*
* ====================================
* Website: http://netkiller.github.com
* Author: neo <netkiller@msn.com>
* Email: netkiller@msn.com
* ====================================
*/
class Firewall{

        protected $status;
        protected $policy;
        protected $chain;
        protected $rule;
        protected $match;
        private $debug;
        //$get,$post,$cookie,$server;

        public function __construct() {
                $this->name     = "Firewall";
        }

        public function __destruct() {
                //print "Destroying " . $this->name . "\n";
        }

        public function enable(){
                $this->status = true;
        }
        public function disable(){
                $this->status = false;
        }

        public function get(){
                if($this->status){
                        $this->chain    = $_GET;
                        return($this);
                }else{
                        return($this->status);
                }
        }

        public function post(){
                if($this->status){
                        $this->chain    = $_GET;
                        return($this);
                }else{
                        return($this->status);
                }
                $this->chain    = $_POST;
        }

        public function cookie() {
                if($this->status){
                        $this->chain = $_COOKIE;
                        return($this);
                }else{
                        return($this->status);
                }

        }

        public function server(){
                if($this->status){
                        $this->chain = $_SERVER;
```

```
                        return($this);
                }else{
                        return($this->status);
                }
        }

        public function match($key, $value){
                if($this->debug) print_r($this->chain);
                $this->match = false;
                if(!array_key_exists($this->chain, $key)){
                        if($this->chain[$key] == $value){
                                $this->match = true;
                        }
                }
                return($this);
        }
        public function policy($p){
                $this->policy = $p;
        }
        public function counter($tm, $cnt){
                return($this);
        }
        public function allow($fun = null){
                if($this->status && $this->match){
                        if($fun){
                                $fun();
                        }
                }
                $this->destroy();
                return($this->status);
        }
        public function deny($fun = null){
                if($this->status && $this->match){
                        if($fun){
                                $fun();
                        }
                }
                $this->destroy();
                return($this->status);
        }
        public function debug($tmp){
                $this->debug = $tmp;
        }
        public function ip($ipaddr){
                return $this->server()->match('REMOTE_ADDR', $ipaddr);
        }
        public function destroy(){
                $this->chain = array();
                $this->match = false;
        }
};

#include_once('firewall.php')
$fw = new Firewall();

$fw->debug(true);
$fw->debug(false);
$fw->enable();
//$fw->disable();
function test(){
        echo 'OK';
};
function allow(){
        echo 'allow';
};
function deny(){
        echo 'deny';
```

```
};
//$fw->policy('blacklist');

$fw->ip('192.168.3.17')->allow('allow');
$fw->ip('192.168.3.17')->deny('deny');

$fw->counter('1m',5)->match('id','1000')->deny('test');

/*
$fw->ip('172.16.0.0/24')->allow();
$fw->ip('172.16.0.0','255.255.255.0')->allow();

$fw->header(array('User-Agent' => 'MSIE5'))->deny()
*/
$fw->get()->match('id','1000')->deny('test');
$fw->get()->match('name','chen')->allow('test');
//$fw->get()->match(array('id' => '1000'))->deny();
/*
$fw->post()->data(array('action'=>'/login.php'))->allow()
$fw->cookie()->data(array('userid'=>'test'))->deny()
*/
$fw->server()->match('HTTP_REFERER', 'http://www.mydomain.com/index.html')->allow('test');
$fw->server()->match('REQUEST_METHOD', 'GET')->deny('test');

$fw->disable();
//$fw->destroy();
```

这里仅仅给你一个思路，我并没有写完程序。例如控制IP请求次数可以如下实现，请自行改善程序

```
<?php
/*
* ====================================
* Website: http://netkiller.github.com
* Author: neo <netkiller@msn.com>
* Email: netkiller@msn.com
* ====================================
*/
require 'SharedConfigurations.php';

$single_server = array(
    'host'     => '127.0.0.1',
    'port'     => 6379,
    'database' => 0
);

$multiple_servers = array(
    array(
       'host'     => '127.0.0.1',
       'port'     => 6379,
       'database' => 15,
       'alias'    => 'first',
    ),
    array(
       'host'     => '127.0.0.1',
       'port'     => 6380,
       'database' => 15,
       'alias'    => 'second',
    ),
);
```

```
$client = new Predis\Client($single_server, array('prefix' => 'fw:'));

$key=$_SERVER['REMOTE_ADDR'];

if(!$client->exists($key)){
        $client->setex($key, 20, 1);
}else{
        $client->incrby($key,1);
}

$counter = $client->get($key);

if($counter > 10){
        echo 'Deny';
}

print_r($client->get($key));

//var_dump($client->keys('*'));
```

第 16 章 数据库与应用程序间通信

本章讲解数据库与应用程序间通信，这里会涉及到

1. 管道通信

你是否想过当数据库中的数据发生变化的时候出发某种操作？但因数据无法与其他进程通信（传递信号）让你放弃，而改用每隔一段时间查询一次数据变化的方法？下面的插件可以解决你的问题。

1.1. 背景

你是否有这样的需求：

你需要监控访问网站的IP，当同一个IP地址访问次数过多需要做出处理，例如拉黑，直接丢进iptables 防火墙规则连中。你的做法只能每个一段时间查询一次数据库，并且判断是否满足拉黑需求？

你是否需要监控某些数据发生变化，并通知其他程序作出处理。例如新闻内容修改后，需要立即做新页面静态化处理，生成新的静态页面

你使用数据库做队列，例如发送邮件，短信等等。你要通知发送程序对那些手机或者短线发送数据

1.2. 解决思路

需要让数据库与其他进程通信，传递信号

例如，发送短信这个需求，你只要告诉发短信的机器人发送的手机号码即可，机器人永远守候那哪里，只要命令一下立即工作。

监控数据库变化的需求原理类似，我们需要有一个守护进程等待命令，一旦接到下达命令便立即生成需要的静态页面

这里所提的方案是采用fifo(First In First Out)方案，通过管道相互传递信号，使两个进程协同工作，这样的效率远比定时任务高许多。fifo是用于操作系统内部进程间通信，如果跨越操作系统需要使用Socket，还有一个新名词MQ(Message queue).

这里只做fifo演示, 将本程序改为Socket方案，或者直接集成成熟的MQ也是分分钟可以实现。

1.3. Mysql plugin

我开发了几个 UDF, 共4个 function

UDF

fifo_create(pipename)

创建管道.成功返回true,失败返回flase.

fifo_remove(pipename)

删除管道.成功返回true,失败返回flase.

fifo_read(pipename)

读操作.

fifo_write(pipename,message)

写操作 pipename管道名,message消息正文.

有了上面的function后你就可以在begin,commit,rollback 直接穿插使用，实现在事物处理期间做你爱做的事。也可以用在触发器与EVENT定时任务中。

1.4. plugin 的开发与使用

编译UDF你需要安装下面的软件包

```
sudo apt-get install pkg-config
sudo apt-get install libmysqlclient-dev

sudo apt-get install gcc gcc-c++ make automake autoconf
```

https://github.com/netkiller/mysql-fifo-plugin

编译udf，最后将so文件复制到 /usr/lib/mysql/plugin/

```
git clone https://github.com/netkiller/mysql-image-plugin.git
cd mysql-image-plugin

gcc -O3  -g  -I/usr/include/mysql -I/usr/include  -fPIC -lm -lz -shared -o fifo.so fifo.c
sudo mv fifo.so /usr/lib/mysql/plugin/
```

装载

```
create function fifo_create returns string soname 'fifo.so';
create function fifo_remove returns string soname 'fifo.so';
create function fifo_read returns string soname 'fifo.so';
create function fifo_write returns string soname 'fifo.so';
```

卸载

```
drop function fifo_create;
drop function fifo_remove;
drop function fifo_read;
drop function fifo_write;
```

1.5. 插件如何使用

插件有很多种用法，这里仅仅一个例

```
CREATE TABLE `demo` (
        `id` INT(11) NULL DEFAULT NULL,
        `name` CHAR(10) NULL DEFAULT NULL,
        `mobile` VARCHAR(50) NULL DEFAULT NULL
)
```

```
COLLATE='utf8_general_ci'
ENGINE=InnoDB;

INSERT INTO `demo` (`id`, `name`, `mobile`) VALUES
        (1, 'neo', '13113668891'),
        (2, 'jam', '13113668892'),
        (3, leo', '13113668893');
```

我们假设有一个demo这样的表,我使用shell写了一个守护进程用于处理数据库送过来的数据

```
#!/bin/bash
#######################################
# Homepage: http://netkiller.github.io
# Author: neo <netkiller@msn.com>
#######################################
NAME=demo
PIPE=/tmp/myfifo
#######################################
LOGFILE=/tmp/$NAME.log
PIDFILE=/tmp/${NAME}.pid
#######################################

function start(){
        if [ -f "$PIDFILE" ]; then
                exit 2
        fi

        if [ ! -f "$LOGFILE" ]; then
                > ${LOGFILE}
        fi

        for (( ; ; ))
        do
            while read line
            do
                                NOW=$(date '+%Y-%m-%d %H:%M:%S')

                echo "[${NOW}] [OK] ${line}" >> ${LOGFILE}

            done < $PIPE
        done &
        echo $! > $PIDFILE
}
function stop(){
        [ -f $PIDFILE ] && kill `cat $PIDFILE` && rm -rf $PIDFILE
}

case "$1" in
  start)
        start
        ;;
  stop)
        stop
        ;;
  status)
        ps ax | grep ${0} | grep -v grep | grep -v status
        ;;
  restart)
        stop
        start
        ;;
  *)
        echo $"Usage: $0 {start|stop|status|restart}"
        exit 2
esac

exit $?
```

启动守护进程

```
$ ./sms.sh start
$ ./sms.sh status
  596 pts/5    S      0:00 /bin/bash ./sms.sh start
```

监控日志，因为守护进程没有输出，完成人户后写入日志。

```
$ tail -f /tmp/demo.log
```

开始推送任务

```
mysql> select fifo_write('/tmp/myfifo',concat(mobile,'\r\n')) from demo;
+-----------------------------------------------+
| fifo_write('/tmp/myfifo',concat(mobile,'\r\n')) |
+-----------------------------------------------+
| true                                          |
| true                                          |
| true                                          |
+-----------------------------------------------+
3 rows in set (0.00 sec)
```

现在看看日志的变化

```
$ tail -f /tmp/demo.log
[2013-12-16 14:55:48] [OK] 13113668891
[2013-12-16 14:55:48] [OK] 13113668892
[2013-12-16 14:55:48] [OK] 13113668893
```

我们再将上面的例子使用触发器进一步优化

```
CREATE TABLE `demo_sent` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
        `mobile` VARCHAR(50) NOT NULL,
        `status` ENUM('true','false') NOT NULL DEFAULT 'false',
        `ctime` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
        PRIMARY KEY (`id`)
)
COLLATE='utf8_general_ci'
ENGINE=InnoDB

CREATE DEFINER=`dba`@`%` TRIGGER `demo_after_insert` AFTER INSERT ON `demo` FOR EACH ROW BEGIN
        insert into demo_sent(mobile,status) select new.mobile,fifo_write('/tmp/myfifo',concat(ne
END
```

测试

```
mysql> insert into demo(name,mobile) values('jerry','13322993040');
```

```
Query OK, 1 row affected (0.05 sec)
```

日志变化

```
$ tail -f /tmp/demo.log
[2013-12-16 14:55:48] [OK] 13113668891
[2013-12-16 14:55:48] [OK] 13113668892
[2013-12-16 14:55:48] [OK] 13113668893
[2013-12-16 14:55:48] [OK] 13322993040
```

1.6. 部署相关问题

我们可以采用主从数据库，将任务放在专用的从库上执行

我们可以创建很多个管道，用于做不同的工作，例如插入一个任务，更新一个任务，发短信一个任务，处理模板与静态化一个任务等等。

2. 消息队列

这里选择使用ZeroMQ的原因主要考虑的是性能问题，其他MQ方案可能会阻塞数据库。

2.1. 背景

之前我发表过一篇文章 http://netkiller.github.io/journal/mysql.plugin.fifo.html

该文章中提出了通过fifo 管道，实现数据库与其他进程的通信。属于 IPC 机制(同一个OS/服务器内)，后我有采用ZeroMQ重新实现了一个 RPC 机制的方案，同时兼容IPC（跨越OS/服务器）

各种缩写的全称 IPC(IPC :Inter-Process Communication 进程间通信)，ITC(ITC : Inter Thread Communication 线程间通信)与RPC(RPC: Remote Procedure Calls远程过程调用)。

支持协议

```
inproc://my_publisher
tcp://server001:5555
ipc:///tmp/feeds/0
```

2.2. 应用场景

如果你想处理数据，由于各种原因你不能在程序中实现，你可以使用这个插件。当数据库中的数据发生变化的时候出发某种操作,你可以使用这个插件。

有时候你的项目可能是外包的，项目结束后外包方不会在管你，你有无法改动现有代码，或者根本不敢改。你可以使用这个插件

采用MQ技术对数据库无任何压力，与采用程序处理并无不同，省却了写代码

处理方法，可以采用同步或者异步方式

例 16.1. 发送短信

发送短信、邮件，只需要查询出相应手机号码，发送到MQ的服务端，服务端接收到手机号码后，放入队列中，多线程程序从队列中领取任务，发送短信。

```
select zmq_client('tcp://localhost:5555',mobile) from demo where subscribed='Y' ...;
```

传递多个参数，可以使用符号分隔

```
select zmq_client('tcp://localhost:5555',concat(name,',',mobile,', news')) from demo;
select zmq_client('tcp://localhost:5555',concat(name,'|',mobile,'|news')) from demo;
```

json格式

```
select zmq_client('tcp://localhost:5555',concat('{name:',name,', tel:',mobile,', template:news}')
```

建议采用异步方式，MQ端接收到任务立即反馈“成功”信息，因为我们不太关心是否能发送成功，本身就是盲目性的发送，手机号码是否可用我们无从得知，短信或者邮件的发送到达率不是100%，所以当进入队列后，让程序自行处理，将成功或者失败信息记录到日志中即可。

例 16.2. 处理图片

首先查询出需要处理图片，然后将路径与分辨率传递给MQ另一端的处理程序

```
select zmq_client('tcp://localhost:5555',concat(image,',800x600}')) from demo;
```

建议采用异步方式，MQ端接收到任务立即反馈“成功”信息

例 16.3. 身份证号码校验

```
select zmq_client('tcp://localhost:5555',id_number) from demo;
```

可以采用同步方案，因为MQ款处理几乎不会延迟，直接将处理结构反馈

例 16.4. 静态化案例

情景模拟，你的项目是你个电商项目，采用外包模式开发，项目已经开发完成。外包放不再负责维护，你现在要做静态化。增加该功能，你要检查多处与商品表相关的造作。

于其改代码，不如程序从外部处理，这样更保险。我们只要写一个程序将动态 URL 下载保存成静态即可，当数据发生变化的时候重新下载覆盖即可

```
CREATE DEFINER=`dba`@`%` TRIGGER `demo_after_insert` AFTER INSERT ON `demo` FOR EACH ROW BEGIN
        select zmq_client('tcp://localhost:5555', NEW.id);
END
CREATE DEFINER=`dba`@`%` TRIGGER `demo_after_update` AFTER UPDATE ON `demo` FOR EACH ROW BEGIN
        select zmq_client('tcp://localhost:5555', NEW.id);
END
CREATE DEFINER=`dba`@`%` TRIGGER `demo_after_delete` AFTER DELETE ON `demo` FOR EACH ROW BEGIN
        select zmq_client('tcp://localhost:5555', NEW.id);
END
```

MQ 另一端的服务会下载http://www.example.com/goods.php?cid=111&id=100, 然后生成html页面，http://www.example.com/111/100.html

插入会新建页面，更新会覆盖页面，删除会删除页面

这样无论商品的价格，属性改变，静态化程序都会做出相应的处理。

例 16.5. 数据同步案例

我们有多个数据库，A 库里面的数据发生变化后，要同步书库到B库，或者处理结果，或者数据转换后写入其他数据库中

方法也是采用触发器或者EVENT处理

2.3. Mysql plugin

我开发了几个 UDF, 共4个 function

UDF

zmq_client(sockt,message)

sockt .成功返回true,失败返回flase.

有了上面的function后你就可以在begin,commit,rollback 直接穿插使用，实现在事物处理期间做你爱做的事。也可以用在触发器与EVENT定时任务中。

2.4. plugin 的开发与使用

编译UDF你需要安装下面的软件包

```
sudo apt-get install pkg-config
sudo apt-get install libmysqlclient-dev

sudo apt-get install gcc gcc-c++ make cmake
```

https://github.com/netkiller/mysql-zmq-plugin

编译udf，最后将so文件复制到 /usr/lib/mysql/plugin/

```
git clone https://github.com/netkiller/mysql-zmq-plugin.git
cd mysql-zmq-plugin

cmake .
make && make install
```

装载

```
create function zmq_client returns string soname 'libzeromq.so';
create function zmq_publish returns string soname 'libzeromq.so';
```

卸载

```
drop function zmq_client;
drop function zmq_publish;
```

确认安装成功

```
mysql> SELECT * FROM `mysql`.`func` where name like 'zmq%';
+-------------+-----+--------------+----------+
| name        | ret | dl           | type     |
+-------------+-----+--------------+----------+
| zmq_client  |   0 | libzeromq.so | function |
| zmq_publish |   0 | libzeromq.so | function |
+-------------+-----+--------------+----------+
2 rows in set (0.00 sec)
```

2.5. 插件如何使用

插件有很多种用法，这里仅仅一个例

编译zeromq server 测试程序

```
cd test
cmake .
make
```

启动服务进程

```
./server
```

发送Hello world!

```
mysql> select zmq_client('tcp://localhost:5555','Hello world!');
+--------------------------------------------------+
| zmq_client('tcp://localhost:5555','Hello world!') |
+--------------------------------------------------+
| Hello world! OK                                  |
+--------------------------------------------------+
1 row in set (0.01 sec)
```

查看服务器端是否接收到信息。

```
$ ./server
Received: Hello world!
```

我们再将上面的例子使用触发器进一步优化

```
mysql> select zmq_client('tcp://localhost:5555',mobile) from demo;
+-------------------------------------------+
| zmq_client('tcp://localhost:5555',mobile) |
+-------------------------------------------+
| 13113668891 OK                            |
| 13113668892 OK                            |
| 13113668893 OK                            |
| 13322993040 OK                            |
| 13588997745 OK                            |
+-------------------------------------------+
5 rows in set (0.03 sec)
```

服务器端已经接收到数据库发过来的信息

```
$ ./server
Received: Hello world!
```

```
Received: 13113668891
Received: 13113668892
Received: 13113668893
Received: 13322993040
Received: 13588997745
```

我们可以拼装json或者序列化数据，发送给远端

```
mysql> select zmq_client('tcp://localhost:5555',concat('{name:',name,', tel:',mobile,'}')) from d
+-----------------------------------------------------------------------------+
| zmq_client('tcp://localhost:5555',concat('{name:',name,', tel:',mobile,'}')) |
+-----------------------------------------------------------------------------+
| {name:neo, tel:13113668891} OK                                              |
| {name:jam, tel:13113668892} OK                                              |
| {name:leo, tel:13113668893} OK                                              |
| {name:jerry, tel:13322993040} OK                                            |
| {name:tom, tel:13588997745} OK                                              |
+-----------------------------------------------------------------------------+
5 rows in set (0.03 sec)
```

返回数据取决于你服务端怎么编写处理程序，你可以返回true/false等等。

触发器以及事务处理，这里就不演示了

3. 数据库与外界文件

你是是不是在开发中常常遇到，删除了数据库记录后，发现该记录对应的图片没有删除，或者删除了图片，数据库中仍有数据存在，你的网站脏数据（图片）成几何数增长，阅读下文这里为你提供了一个完美决方案。

3.1. 背景

我以电商网站为例，一般的网站产品数据存放在数据库中，商品图片是上传到文件服务器，然后通过http服务器浏览商品图片。这是最基本的也是最常见做法。

稍复杂的方案是，如果图片数量庞大，会使用分布式文件系统方案。但是这些方案都不能保证数据的完整性，极易产生脏数据（垃圾数据）。脏数据是指当你删除了数据库表中的记录后，图片仍然存在，或者手工删除了图片，而数据库中的记录仍然存在。

将图片放入数据库中存放在BLOB的方法可以解决脏数据问题，典型的案例是公安的身份证系统。但这种方案的前提是，图片不能太大，数量不多，访问量不大。 这显然不适合电商网站。

2009年我在走秀网工作，商品图片与缩图文件900GB到2012离职已经有10TB，每天有成百上千的商品上架下架，很多商品下架后永远不会再上架，这些批量下架的商品数据不会删除，仅仅标记为删除，总是期望以后能继续使用，实际上再也不会有人过问，另一方面随着品类经理频繁更换，员工离职，这些商品会石沉大海，再也无人问均。这些商品所对应的图片也就脏数据主要来源。新的品类经理上任后，会重新拍照，上传新图片。

总之，删除数据库中的数据不能将图片删除就会产生脏数据。很多采用删除数据的时候去检查图片如果存在先删除图片，再删除数据的方法。这种方案也非完美解决方案，存在这图片先被删除，程序出错SQL没有运行，或者反之。

3.2. 解决思路

如果删除图片能够成为事物处理中的一个环节，所有问题都能迎刃而解，可彻底解决脏数据的烦恼。

3.3. 解决方案

mysql plugin 开发 udf。我写几个function

UDF

image_check(filename)

    检查图片是否存在.

image_remove(filename)

    删除图片.

image_rename(oldfile,newfile)

    更改图片文件名.

image_md5sum(filename)

md5sum 主要用户图片是否被更改过.

image_move(filename,filename)

移动图片的位置

有了上面的function后你就可以在begin,commit,rollback 直接穿插使用，实现在事物处理期间做你爱做的事。

3.4. plugin 的开发与使用

编译UDF你需要安装下面的软件包

```
sudo apt-get install pkg-config
sudo apt-get install libmysqlclient-dev

sudo apt-get install gcc gcc-c++ make automake autoconf
```

https://github.com/netkiller/mysql-image-plugin

编译udf，最后将so文件复制到 /usr/lib/mysql/plugin/

```
git clone https://github.com/netkiller/mysql-image-plugin.git
cd mysql-image-plugin/src

gcc -I/usr/include/mysql -I./ -fPIC -shared -o image.so image.c
sudo mv image.so /usr/lib/mysql/plugin/
```

装载

```
create function image_check returns boolean soname 'images.so';
create function image_remove returns boolean soname 'images.so';
create function image_rename returns boolean soname 'images.so';
create function image_md5sum returns string soname 'images.so';
create function image_move returns string soname 'images.so';
```

卸载

```
drop function image_check;
drop function image_remove;
drop function image_rename;
drop function image_md5sum;
drop function image_move;
```

3.5. 在事务中使用该插件

插入图片流程，上传图片后，通过插件检查图片是否正确上传，然后插入记录

```
begin;
IF image_check('/path/to/images.jpg') THEN
        insert into images(product_id,thumbnail,original) values(1000,'thumbnail/path/to/images.j
        commit;
ELSE
        image_remove('/path/to/images.jpg');
```

```
END IF
rollback;
```

删除商品采用image_move 方案，当出现异常rollback后还可以还原被删除的图片

```
begin;
IF image_check('/path/to/images.jpg') THEN
        select thumbnail,original into @thumbnail,@original from images where id='1000' for delet
        delete from images where id='1000';
        select image_move(@thumbnail,'recycle/path/to/');
        select image_move(@original,'recycle/path/to/');
        commit;
END IF

rollback;
select image_move('recycle/path/to/images.jpg','path/to/images.jpg');
```

我们可以使用EVENT定时删除回收站内的图片

```
image_remove('recycle/path/to/images.jpg');
```

3.6. 通过触发器调用图片处理函数

通过触发器更能保证数据完整性

```
1. insert 触发器的任务：  插入记录的时候通过image_check检查图片是否正常上传，如果非没有上传，数据插入失败。
2. delete 触发器的任务：  检查删除记录的时候，首先去删除图片，删除成功再删除该记录。
```

触发器进一步优化

```
1. insert 触发器的任务：  插入记录的时候通过image_check检查图片是否正常上传，如果非没有上传，数据插入失败。如果上
2. delete 触发器的任务：  检查删除记录的时候，首先去改图片文件名，然后删除该记录，最后删除图片，删除成功。如果中间玛
```

4. Socket 方式

TCP 方式还不如使用现在有的消息队列，所以数据库通过 Socket与应用程序通信，我推荐 UDP 方式。

UDP 有个好处，丢出去就不管了，性能非常好。并且可以实现组播，广播。下面是 UDP的例子

4.1. UDP

https://github.com/netkiller/mysql-udp-plugin

下载 mysql-udp_sendto-plugin 然后编译安装代码

```
# cmake .
# make && make install
```

安装

```
create function udp_sendto returns string soname 'libudp_sendto.so';
```

卸载

```
drop function udp_sendto;
```

使用演示，首先使用nc命令监听一个UDP端口，用来接收数据库发送过来的数据。数据结构请自行定义。这里仅仅是演示，可以采用json, 逗号分隔等等方式。

```
# nc -luv 4000
```

在数据库中使用下面SQL发送数据给应用程序

```
select udp_sendto('192.168.2.1','4000','hello');
```

第 17 章 多维度架构之消息队列

1. 你是怎样使用消息队列的?

十有八九的人是这样理解消息队列的，一段生产消息，另一端消费消息。多用于批量执行或者发送通知。

2. 你是否真正理解了消息队列?

2.1. 消息队列并不是实时的

消息队列并不是实时的，它不能替代传统 TCP/UDP Socket 通信。

```
消息生产者 –> 消息队列服务器 –> 消息消费者
```

消息队列中有 Switch (交换机) 的概念，事实上消息队列的工作方式的确跟网络交换机类似，网络交换机是TCP/UDP包的接收存储和转发，消息队列是消息的存储与转发。

从生产者到达消费者是有延迟的，尤其是多个生产者持续生产和多个消费者持续消费，消息服务器会出现瓶颈，消息会出现堆积情况，这时消息的生产和消费都会出现延迟，且时间不可控。

类似消息通知这种需求，是非实时的需求，不考虑发送延迟和达到时间，可以使用消息队列解决方案，否则还是使用 TCP Socket。

2.2. 不能替代异步执行

当使用消息队列替代异步执行的时候也会出现前面所说的执行时间不可控。所以不是所有场景都适合使用消息队列的，更多时候我们开启一个线程去异步执行效果可能更好。

3. 使用的合理吗?

我曾经接手一个项目，这个项目中用户注册发送手机验证码和邮箱验证码使用了消息队列记录，真实让人哭笑不得，为了使用消息队列而使用消息队列。

4. 是否有必要使用消息队列?

两个团队分别负责商品信息和价格还有库存功能模块的开发，商品信息是静态化方案，商品价格和库存是AJAX动态载入，公司重金从杭州淘宝挖了一名架构师，并给了首席架构师的头衔。

最终架构师给出的方案，更新商品信息时，后通过 ActiveMQ 通知刷新缓存更新价格和库存，这样的方案是否可行？我认为是可有可无，这个方案中消息队列并不是刚需。

出现了 这个需求是因为，商品促销需要经常调整价格和库存量。修改商品信息这个后台运营的操作，这种操作分成两类，一类是临时单品调价，另一类是批量调价。

我想问是否有必要更新商品价格？换个思路可以吗？是否还有其他更好的方案?

这里还有一个故事，于是促销需要，商品需要统一调价，程序猿给了一条SQL：

```
update goods set price=price+10 where category_id = xxx
```

高薪聘用的 DBA 竟然执行了，然后就悲剧了。

让我们换个思路去解决这个问题，我们专门来建一个促销打折表 discount 用来存储需要做促销的商品：

```
+----------------------------------------+
| discount                               |
```

```
+-----------------------------------------+
| Id | goods_id | price   | ctime | mtime |
+-----------------------------------------+
| 1  | 100      | + 12.5 | xxx   | xxx   |
| 2  | 101      | - 0.5  | xxx   | xxx   |
+-----------------------------------------+
```

当前价格 = 商品价格 + 折扣价格

相比 goods 表 discount 的数据量是非常小的，这样一来就无需异步执行，商品调价在Service 直接更新discount 表即可，刷新缓存也不过是一行代码，删除 cache key 在一个实例中完成原子性更好。

5. 最后总结

就如上面所说的更新商品信息需求，增加了消息队列之后，开发复杂了，测试复杂了，运维复杂了，监控复杂了，出现故障排查时间变长了。

架构设计中我们应该收缩技术栈，做减法，而不是做加法，强上消息队列?

第 18 章 多维度架构之分库分表

分库和分表是架构必经之路，我想问问你是怎么分库和分表?

很多系统在设计之初就没有考虑过后期的分库与分表，甚至开发团队没有架构和DBA人员，开发团队也比较年轻，对于数据库的架构定义非常随意，满足当前需求即可。

实际上数据库结构等同于建筑里面的地基，地基没有打好，后面的优化都是徒劳的，最终不得不重构数据库结构。

那么你是怎样分库分表的?

任何系统数据流都是漏斗形状的，数据库是漏斗末端，架构设计是尽量在前端计算，合并，拆分，分流，缓存，最终将有价值的数据写入数据库。数据库的访问是结果集越小越好。

基于这种认识，通常分库和分表，我们想到的就是首先垂直分表，这种方式简单易操作。

当前（本年度数据库）（热数据）
2019年数据
2018年数据
以此类推

或者按照月份分表

当前（热数据）
10月数据
9月数据
以此类推

这样分表可以缩小结果集，能快速解决查询瓶颈问题。但是新的挑战来了，由于分表后，索引是独立不连续的，历史数据的查询或遍历数据变的复杂了，要么使用联合查询，要么一张表一张表的遍历。

同理水平分表也是粗暴的将一些尺寸较大的列独立成新表，以降低单个表的容量尺寸。

如果是单纯的数据查找，还是能忍受，我们可能根据时间来选择查询的表，如果是复杂的SQL操作，就只能逐一查询，在程序中二次计算，合并等等操作。

这种分库或分表的思路，理论上属于数据归档。将热数据放在当前数据库中，将很少查询的冷数据放在另一个库中。但是对于 user 这种表就无能为力，你不知道那个用户什么时候会做登录操作。

1. 分表需要从业务角度考虑

分表需要从业务角度考虑，数据库服务于业务逻辑。

由于我即负责产品也负责架构，长期的工作中，总结出一套分库分表的策略。

我分表策略是:

1. 从UI角度出发
2. 从业务流角度出发

举例一，用户表分表?

用户表怎样水平拆分呢？ 用户登录的逻辑是这样的，第一个UI 输入用户名和密码，提交后验证密码，用户是否过期，记录登录时间，IP地址。第二个UI，载入用户资料，包括用户姓名，年龄，性别等等…..

数据库我是这样设计的:

```
+-----------------------------------------------+
| Member                                        |
+-----------------------------------------------+
| id | username | password | ip | ctime | mtime |
+-----------------------------------------------+
```

就这么简单，会员表服务于第一UI

```
+-----------------------------------------------+
| Profile                                       |
+-----------------------------------------------+
| id | member_id | name | sex | age | mtime |…… |
+-----------------------------------------------+
```

服务于第二个UI，通过 member_id 关联数据

怎么样？用户登录过程并不会去访问 profile 表，只有登录成功才会访问。

用户表怎样垂直拆分呢？ 将 username 做crc32/md5/sha1 运算，使用哪种随你，取出第一个字符用于分表。

例如 neo, netkiller 两个用户被分到 member_n 表中，jerry,jam 被分到member_j 表中，配套的还有 profile_n, profile_j 等等。我们甚至还可以使用外键约束 member_n 和 profile_n 两个表。

当用户登录时，对用户ID做一次 hash 运算，就知道去哪个表中找到该用户的数据。

举例二，商品表如何分表?

商品信息表数据量非常大，我们可以基于品类分库或分表，我们UI设计中，一般只有首页才会将不同分类的产品聚合到一起。进入品类分类页面后，只会访问该品类的数据表。

这就是从业务流的角度进行分表，用户操作是逐渐被引导至我们想呈现的页面。

举例三，海量用户如何分库?

海量用户分库的思路是，用户被分配到指定数据库，该用户所有的数据都会产生在该数据库中，也可以理解为基于数据库隔离用户。

基于该思路分库，这样表名保持不变。

例如 neo, netkiller 被分配到 schema_n.member, schema_n.profile

例如 jerry, jam 被分配到 schema_j.member, schema_j.profile

一旦用户登录，便被锁定到指定的数据库，接下来所有操作，用户产生的数据，都被存储在该数据库中。索引连续，外键约束，触发器，存储过程，均不影响使用。

使用分区表

使用分区别将分区数据写入挂载的SSD盘上。例如 /opt/data/ 下面挂载了四块SSD，目录名是 partition1~4。

```
CREATE TABLE your_table (id INT, cdate DATE)
engine='InnoDB'
PARTITION BY LIST(YEAR(cdate))
(
  PARTITION p2020 VALUES IN (2020)
    DATA DIRECTORY = '/opt/data/partition1',

  PARTITION p1999 VALUES IN (1999)
    DATA DIRECTORY = '/opt/data/partition2',

  PARTITION p1998 VALUES IN (1998)
    DATA DIRECTORY = '/opt/data/partition3',

  PARTITION p1997 VALUES IN (1997, 1996, 1995)
    DATA DIRECTORY = '/opt/data/partition4'
);
```

2. 最后总结

分库和分表不是简单的切割，而是需要从业务的角度出发，从产品经理视角，你需要展现什么数据，什么样的数据库结构能更好的为UI服务。或者我们应该设计什么样的UI才能更好的展现数据。

多维度架构的核心就是跨界，用跨界知识解决架构中存在的问题。

第 19 章 多维度架构之压力测试

压力测试中存在的问题

1. (What) 什么是压力测试

软件压力测试是一种基本的质量保证行为，它是每个重要软件测试工作的一部分。软件压力测试的基本思路很简单： 不是在常规条件下运行手动或自动测试，而是在计算机数量较少或系统资源匮乏的条件下运行测试。 通常要进行软件压力测试的资源包括内部内存、CPU 可用性、磁盘空间和网络带宽。

压力测试涵盖，性能测试，负载测试，并发测试等等，这些测试点常常交织耦合在一起。

1.1. 压力测试存在哪些问题

我归纳一下有几点：

1. 操作系统默认安装，在未做任何优化的情况下实施压力测试
2. 未考虑磁盘IO对软件的影响
3. 未考虑网络带宽对软件的影响
4. 网络软件测试，没有考虑到TCP特点
5. 各种超时参数优化
6. 测试客户端未优化
7. 并发理解有误

8. WEB服务器，数据库，等等服务器未优化

如果上面几项没有做优化，压力测试数据基本没有任何参考价值，任何一项没有优化，都会导致你的压力测试数据出现偏差。 下面我来逐条说明：

1. 操作系统问题： 操作系统是大众化软件，出厂优化都是面向大众，不可能为某个领域做单独优化。所以我们第一步需要优化操作系统。 Linux 系统优化内核参数，Windows 系统优化注册表等等。

2. 磁盘IO： 这是最容易出现瓶颈的地方，常常是CPU还没有达到极限，磁盘已经不堪重负。

3. 网络IO： 与磁盘IO相同

4. TCP连接： 几乎所有 B/S, C/S 软件都是采用多线程，或者多进程技术。这种技术有个特点，开发者将程序设计为线程可自动伸缩模式，开启进程后会启动少量线程，当连接不断提高后，线程数逐渐增加，随着线程运行结束后，线程逐渐减少。 这样的设计会更有效地利用硬件资源，在程序空闲时将硬件资源让给其他进程。少有软件设计为开启服务独占资源。 这样测试软件做压力测试，不能一次并发很多请求，而是要采用逐渐增加的方式，否则第一次测试会有一部分并发不能及时响应，导致测试数据偏差。另外也你可以多做几次压力请求（让多线程工作起来），从第三次开始记录测试数据，忽略前面两次的测试数据。

提示：另一个问题是TCP连接复用，这也是一个重要配置项。如果这项没有配置，我想测试出的数据也会有偏差

1. 超时参数： 超时参数在压力测试中是非常重要的参数，例如从WEB到数据库连接超时是60秒，如果有一个SQL查询超过300

秒，那么后面的请求会持续排队等待，当连接数达到数据库的最大连接时，接下来的所有请求都是失败的。 通常我们的WEB服务器超时不会超过30秒，有时我设置为10秒，一旦出现超时，宁可让该连接Timeout，不要让他影响整体服务。

2. 客户端： 很多网络软件需要从客户端发出压力测试请求，所以客户端的优化也是必须的，否则客户端压力出不去，服务端压力进不来。

3. 并发： 很多人认为并发，就是同一时间内的最大连接数，这是错误的。如果你写过多线程程序，就会发现多线程运行时有规律的。是顺序排队运行的，根本不是同时运行的。 所以并发是指，相对时间内能完成的连接总和，例如，每秒并发，每分钟并发等等，通常我们已秒为单位。 我们目前使用的操作系统叫分时操作系统，这种系统的特点就是可能实现多用户，多任务。操作系统将进程排队（优先级）轮询运行，只不过这个操作太快了，使你认为多个进程在同时运行。

4. 服务器优化： 主要B/S软件压力测试，WEB，缓存，数据库等等服务器，都需要逐一优化到最佳状态

2. (Why) 为什么做压力测试

如果在软件设计阶段都将这些问题元素都考虑进去，同时开发阶段严格执行。那么开发出些软件几乎不用做这个劳人伤神的压力测试。

所以在软件设计阶段就要考虑，灵活性，扩展性，可靠性与性能，还要考虑高可用与负载均衡。

同时软件优化伴随开发，持续集成，持续测试，持续部署。

3. (Where) 在哪里做压力测试

有些软件需要封闭的环境测试，不能在共享资源的环境中做测试。所以你有必要做Vlan隔离，甚至独立的路由器与交换机在封闭网络中测试。

4. (When) 什么时间做压力测试

任何时间都可能做压力测试，为什么我将“时间”重点提出呢？目前受地球自转影响，经常闰秒，你不的不考虑这个问题。

5. (Who) 压力测试过程参与人员

1. 运维部门
2. 开发部门
3. 测试部门

6. (How) 如何做压力测试

下面我们举一些例子，讲述压力测试方法，限于篇幅不可能面面俱到，我仅仅是给你提供思路。

测试前你需要一些监控工具，实时监控服务器的资源变化。

例如 Web 服务器压力测试，测试场景是 nginx：

```
worker_processes  8;                 处理器数
worker_rlimit_nofile 65530;          允许最多打开文件数
worker_connections  4096;            最大连接数数为
keepalive_timeout  65;               开启复用连接
gzip  on;                            压缩传输数据
```

怎么测试呢？你要获得最大化性能吗？还是相对性能？我们通常需要的是满足需求就好的相对性能，而不是最大化性能。为什么呢？因为要获得最大化性能是要做出很多配置牺牲的，例如关闭日志，禁止访问时间等等。

按照上面的配置你的测试用例应该是，每次并发4000 请求 8000~10000 次， 你不能并发8000 请求 4000 这样测试。很是很多人常常犯的错误，所以测试者需要连接系统的配置参数，不能盲目使用数字实验。

上面我说过线程的开启时随着请求，逐渐增加的，所以首次发起测试数据是不准确的，通过pstree命令可以看到线程数量。等第三次以后线程逐渐增加到4096个，并且之前开启的TCP可以复用，这时测试的结果比较有说服力。

延伸阅读《Netkiller Web 手札》《Netkiller Testing 手札》《Netkiller Linux 手札》

部分 II. Database Modeling Design

RDBMS / ORDBMS / OODBMS / HDMS 数据库设计

下面数据库设计实例中，大部分使用MySQL,PostgreSQL为例,少部分以Oracle为例。

第 20 章 关系型数据库设计

MySQL 数据库设计案例

1. 数据字典

我不建议使用传统的《数据字典》，我的做法是E-R图加数据库注释

注释伴随表，视图，触发器，过程等等，便于维护

2. 用户帐号表

用户帐号或通行证系统设计，下面以我的数库为例讲解。

我一般使用两个表 passport，profile 完成网站会员系统。

首先说说passport表，你也要以使用user或member等等命名，这个表设计尽可能地简单，不要使用过多字段。仅保存登录所必须用到的字段，如user,password,nickname,email... 登录帐号和密码做复合索引。

然后是profile表，这个表与passport是1:1关系，保存用户详细信息

这样设计可以保证海量用户登录时的速度。

```
+----------+
| user     |
|----------|
|id        | <---+
|user      |     |
|passwd    |     |
|nickname  |     |
|status    |     |
+----------+     |
                1:1
+----------+     |
| profile  |     |
|----------|     |
|user_id   | o---+
|name      |
|sex       |
|address   |
|telphone  |
|status    |
+----------+
```

2.1. 用户注册键盘跟踪表设计

该表的功能是，防止用户注册过程中流逝，记录已经填写的数据。

```
CREATE TABLE `signup_keyloggers` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '唯一ID',
        `cookie` VARCHAR(32) NOT NULL COMMENT 'cookie id',
        `type` ENUM('baidu','google') NOT NULL COMMENT '推广账号类型',
        `field` ENUM('Name','Mobile','Email') NOT NULL COMMENT '字段名',
        `value` VARCHAR(50) NOT NULL COMMENT '值',
        `status` ENUM('New','Sent','Ignored','Called','Processed') NOT NULL DEFAULT 'New' COMMENT
        `operator` VARCHAR(10) NOT NULL COMMENT '操作人',
        `ctime` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
        `mtime` TIMESTAMP NOT NULL DEFAULT '0000-00-00 00:00:00' COMMENT '状态修改时间',
        PRIMARY KEY (`id`),
        UNIQUE INDEX `unique_index` (`type`, `cookie`, `field`, `value`)
)
COMMENT='用户注册键盘记录器'
COLLATE='utf8_general_ci'
ENGINE=InnoDB;
```

当用户注册成功会根据cookie id 删除该表中的数据。

当数据被记录后，客服就可以对客户回访，并修改状态status，忽略 Ignored，邮件发送Sent， 电话回访Called等等

3. 分类表设计

3.1. 树形分类表

```
 +-----------+
 | category  |
 |-----------|
 |id         | <---+
 |title      |     |
 |description|    1:n
 |status     |     |
 |parent_id  | o---+
 +-----------+
```

```
CREATE TABLE `category` (
        `id` SMALLINT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
        `name` VARCHAR(10) NOT NULL,
        `description` VARCHAR(255) NULL,
        `status` ENUM('enable','desable') NOT NULL DEFAULT 'enable',
        `parent_id` SMALLINT(10) UNSIGNED NOT NULL DEFAULT '0',
        PRIMARY KEY (`id`),
        CONSTRAINT `FK1` FOREIGN KEY (`parent_id`) REFERENCES `category` (`id`)
)
COMMENT='goods category'
ENGINE=InnoDB
ROW_FORMAT=DEFAULT
```

3.2. 多对多分类

多对多分类,主要用于满足，一个产品/文章属于多个分类的需求。

```
      +------------+
      | category   |
      |------------|
 +--> |id          | <---+
 |    |title       |     |     +----------------------+
1:n   |description |    1:n    | categroy_has_product |
 |    |status      |     |     +----------------------+
 +--o |parent_id   |     |     | id                   |
      +------------+     +---o | category_id          |
                         +---o | product_id           |
      +------------+     |     +----------------------+
      | product    |    1:n
      +------------+     |
      |id          | <---+
      |price       |
      |quantity    |
      |...         |
      |status      |
      +------------+
```

3.3. 快速检索子分类设计

上面我刚刚讲过怎样实现“不限子树的分类树”，我们可以实现不限层次的无线分类表。

```
 +-----------+
 | category  |
 |-----------|
 |id         | <---+
 |title      |     |
 |description|    1:n
 |status     |     |
 |parent_id  | o---+
 +-----------+
```

问题出来了，当我需要读取一个分类（任意分类）下的所有子分类，怎样实现，很多人会说用“递归”。当然“递归”可是现实我们的需求，在几百个分类的项目中，使用递归也不是不可以的，但是当数量非常庞大时怎么办?

当然有更好的解决方案，请看下面

```
 +-----------+
 | category  |
 |-----------|
 |id         | <---+
 |title      |     |
 |description|    1:n
 |status     |     |
 |parent_id  | o---+
 |path       |
 +-----------+
```

```
+-----------------------------------------------------------------------+
| category                                                              |
+----+-----------+-----------------------+--------+-----------+-----------+
| id | name      | description           | status | parent_id | path      |
+----+-----------+-----------------------+--------+-----------+-----------+
|  1 | 中国     | 中华人民共和家                                  | Y      |      NULL | 1/
|  4 | 广东省 | 广东省                                                  | Y      |         1 |
|  5 | 深圳市 | NULL                  | Y      |         4 | 1/4/5     |
|  6 | 宝安区 | NULL                  | Y      |         5 | 1/4/5/6   |
|  7 | 龙华镇 | NULL                  | Y      |         6 | 1/4/5/6/7 |
+----+-----------+-----------------------+--------+-----------+-----------+
```

```
CREATE TABLE `category` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '分类ID',
        `name` VARCHAR(50) NOT NULL COMMENT '分类名称',
        `description` VARCHAR(200) NULL DEFAULT NULL COMMENT '分类描述',
        `status` ENUM('Y','N') NOT NULL DEFAULT 'Y' COMMENT '分类状态有继承性',
        `parent_id` INT(10) NULL DEFAULT '1' COMMENT '分类父ID',
        `path` VARCHAR(255) NOT NULL COMMENT '分类递归路径索引',
        INDEX `PK` (`id`),
        INDEX `relation` (`id`, `parent_id`),
        INDEX `FK_category_category` (`parent_id`),
        INDEX `path` (`path`)
```

```
)
COMMENT='分类表'
ENGINE=InnoDB
ROW_FORMAT=DEFAULT
AUTO_INCREMENT=0

insert into category(`name`,`description`,`status`,`parent_id`,`path`) values('中国','中华人民共和家
```

```
ALTER TABLE `category`
        ADD CONSTRAINT `FK_category_category` FOREIGN KEY (`parent_id`) REFERENCES `category` (`i
```

抽取广东子树

```
select * from category where path like '1/4%';
```

```
mysql> select * from category where path like '1/4%';
+----+-----------+-------------+--------+-----------+-----------+
| id | name      | description | status | parent_id | path      |
+----+-----------+-------------+--------+-----------+-----------+
|  4 | 广东省 | 广东省   | Y      |         1 | 1/4       |
|  5 | 深圳市 | NULL        | Y      |         4 | 1/4/5     |
|  6 | 宝安区 | NULL        | Y      |         5 | 1/4/5/6   |
|  7 | 龙华镇 | NULL        | Y      |         6 | 1/4/5/6/7 |
+----+-----------+-------------+--------+-----------+-----------+
4 rows in set (0.00 sec)
```

3.4. 计算节点数量

```
DROP TABLE IF EXISTS `test`;
CREATE TABLE IF NOT EXISTS `test` (
  `id` int(11) DEFAULT NULL,
  `pid` int(11) DEFAULT NULL,
  `name` char(50) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `test` (`id`, `pid`, `name`) VALUES
        (1, 0, 'A'),
        (2, 1, 'B'),
        (3, 1, 'C'),
        (4, 0, 'D'),
        (5, 0, 'E'),
        (6, 5, 'F');

select (select t2.name from test t2 where t2.id=t1.pid) as name, count(pid) as sum from test t1 w
```

统计所有节点包括数量为零的

```
select t1.name, (select count(t2.name) from test t2 where t2.pid=t1.id) as sum from test t1
```

3.5. Example

例 20.1. identity_card 身份证归属地表

```
CREATE TABLE `identity_card` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '唯一主键',
        `pid` INT(10) UNSIGNED NOT NULL DEFAULT '0' COMMENT '父ID',
        `path` VARCHAR(50) NOT NULL COMMENT '路径',
        `number` VARCHAR(18) NOT NULL COMMENT '身份证号码段',
        `zone` VARCHAR(50) NOT NULL COMMENT '行政区域',
        `status` ENUM('Y','N') NOT NULL DEFAULT 'N' COMMENT '状态',
        `modified` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMME
        PRIMARY KEY (`id`),
        INDEX `FK_identity_card_identity_card` (`pid`),
        INDEX `path` (`path`),
        INDEX `number` (`number`),
        CONSTRAINT `FK_identity_card_identity_card` FOREIGN KEY (`pid`) REFERENCES `identity_card
)
COMMENT='identity card number'
COLLATE='utf8_general_ci'
ENGINE=InnoDB;
```

"id"	"pid"	"path"	"number"	"zone"	"status"	"modified"
"1012"	"1"	"1.1012"	"330000"	"浙江省"	"Y"	"2012-05-16 17:18:14"
"1041"	"1012"	"1.1012.1041"	"330300"	"温州市"	"Y"	"2012-05-16 17:44:18"
"1052"	"1041"	"1.1012.1041.1052"	"330381"	"瑞安市"	"Y"	"2012-05-16 17:44:25"
"1367"	"1"	"1.1367"	"360000"	"江西省"	"Y"	"2012-05-16 16:57:23"
"1451"	"1367"	"1.1367.1451"	"360900"	"宜春市"	"Y"	"2012-05-16 17:44:58"
"1990"	"1"	"1.1990"	"430000"	"湖南省"	"Y"	"2012-05-16 16:50:50"
"1991"	"1990"	"1.1990.1991"	"430100"	"长沙市"	"Y"	"2012-05-16 16:50:54"
"2124"	"1990"	"1.1990.2124"	"431300"	"娄底市"	"Y"	"2012-05-16 16:54:45"

4. 文章表设计

看具体情况，拆分表，可按“日”，“月”，“年”等等

```
      +-----------+
      | category  |
      |-----------|
  +-->|id         | <---+
  |   |title      |     |
  |   |description|    1:n
  |   |status     |     |
  |   |parent_id  | o---+
  |   +-----------+
  |
 1:n
  |
  |   +-----------------+            +------------------+
  |   | article_2008_01 |            | feedback_2008_01 |
  |   |-----------------|            |------------------|
  |   |id               |<--1:n--+   |id                |
  |   |title            |        |   |title             |
  |   |content          |        |   |content           |
  |   |datetime         |        |   |datetime          |
  |   |status           |        |   |status            |
  +--o|category_id      |        +--o|news_id           |
  +--o|user_id          |        +-->|user_id           |
  |   +-----------------+        |   +------------------+
  |                              |
 1:n  +----------+     +---1:n---+
  |   | user     |     |
  |   |----------|     |
  +-->|id        | <---+
      |user      |
      |passwd    |
      |nickname  |
      |status    |
      +----------+
```

4.1. 分区表设计

分区表可以通过表空间，等等技术实现，优点是解决了Union查询问题，保证了数据的一致性。

```
     +-----------+
     | category  |
     |-----------|
 +-->|id         | <---+
 |   |title      |     |
 |   |description|    1:n
 |   |status     |     |
 |   |parent_id  | o---+
 |   +-----------+
 |
1:n
 |
 |   +-----------------+            +-----------------+
 |   | article         |            | feedback        |
 |   |-----------------|            |-----------------|
 |   |id               |<--1:n--+   |id               |
 |   |title            |        |   |title            |
 |   |content          |        |   |content          |
 |   |datetime         |        |   |datetime         |
 |   |status           |        |   |status           |
 +--o|category_id      |        +--o|news_id          |
 +--o|user_id          |        +-->|user_id          |
 |   +-----------------+        |   +-----------------+
 |   | 2007,2008,2009  |        |   | 2007,2008,2009  |
 |   +-----------------+        |   +-----------------+
 |                              |
1:n  +----------+     +---1:n---+
 |   | user     |     |
 |   |----------|     |
 +-->|id        | <---+
     |user      |
     |passwd    |
     |nickname  |
     |status    |
     +----------+
```

4.2. Title性能优化

显示title前20个汉字并在后尾添加省略号。

5. 评论表

```
+----------+
| user     |
|----------|
|id        | <---+
|user      |     |
|passwd    |     |
|nickname  |     |
|status    |     |
+----------+     |
                1:n
+-----------+    |      +-----------+
| feedback  |    |      | news      |
|-----------|    |      |-----------|
|id         |    |  +-->|id         |
|title      |    |  |   |title      |
|content    |    |  |   |content    |
|datetime   |    | 1:n  |datetime   |
|status     |    |  |   |status     |
|user_id    |o---+  |   |user_id    |
|news_id    |o------+   +-----------+
+-----------+
```

6. 记录点击率，阅读次数，及评分表

```
+--------------+              +--------------+
| article      |              | article_rank |
|--------------|              |--------------|
|id            | <---1:1---o  |article_id    |
|title         |              |click         |
|content       |              |read          |
|datetime      |              |score         |
|status        |              |...           |
|category_id   |              |...           |
|user_id       |              |...           |
+--------------+              +--------------+
```

7. 产品属性表

7.1. 简单实现

```
+------------+           +--------------------------+           +-----------------------+
| product    |           | product_attribute        |           |product_attribute_key  |
+------------+           +--------------------------+           +-----------------------+
|id          | <--1:1--o |product_id                |     +---> |id                     |
|price       |           |product_attribute_key_id  | o---+     |name                   |
|quantity    |           |product_attribute_value_id| o---+     +-----------------------+
|...         |           +--------------------------+     |     +-----------------------+
|category_id |                                         1:n     |product_attribute_value|
+------------+                                            |     +-----------------------+
                                                          +---> |id                     |
                                                                |name                   |
                                                                +-----------------------+
```

7.2. 实现属性组管理

product attribute group

```
+------------+           +--------------------------+           +--------------------------+
| category   |           | product_attribute_group  |           | product_attribute        |
+------------+           +--------------------------+           +--------------------------+
|id          |     +---> |id                        | <--1:n--o |product_attribute_group_id|
|title       |     |     |name                      |           |product_attribute_key_id  | o---
|description |    1:1    |status                    |           |product_attribute_value_id| o---
|status      |     |     +--------------------------+           +--------------------------+
|parent_id   |     |                                                                          1
|default_pag | o---+
+------------+
```

7.3. 可编辑属表

product attribute group

```
    +------------+           +------------------+           +--------------------------+
    | category   |           | attribute_group  |           | group_has_attribute      |
    +------------+           +------------------+           +--------------------------+
 +->|id          |     +-->  |id                | <--1:n--o |attribute_group_id        |     +-+-
 |  |title       |     |     |name              |           |attribute_key_id          | o---+ |
 |  |description |    1:1    |status            |           |                          |       |
 |  |status      |     |     +------------------+           +--------------------------+       |
 |  |parent_id   |     |                                                                       |
 |  |default_pag | o---+                                                                       |
 |  +------------+                                                                             |
1:n                                                                                            |
 |  +-------------+             +--------------------------+                                  |
 |  | product     |             | product_attribute        |                                  |
 |  +-------------+             +--------------------------+                                  |
 |  |id           |         +-> |product_id                |                                  |
 |  |price        |         |   |attribute_key_id          | o---------------1:n---------------+
 |  |quantity     |         |   |attribute_value           |
 |  |...          |         |   +--------------------------+
```

```
 +-o|category_id  |         |
    |attr_group_id| <--1:n--o
    +-------------+
```

product_attribute_key			
1	color	list	red,green,blue
2	sex	bool	Female,Male
3	qty	input	''

8. 商品库存表

```
+------------+              +----------------+               +------------------+
| product    |              | product_store  |               | user_order       |
+------------+              +----------------+               +------------------+
|id          | <--+         |id              | <---+         |id                |
|price       |    +--1:1--o |product_id      |     |         |user_id           |
|quantity    |              |sn              |     +--1:n--o |product_store_id  |
|...         |              |status          |               |                  |
|category_id |              +----------------+               +------------------+
+------------+
```

product 是产品表总表，product_store每个产品一条记录，同时将sn编号对应到物理产品，这时记录库存需要

```
select count(id) from product_store where product_id='xxxxx' and status = 'sell'
```

商品销售

```
begin;
select id from product_store where status = 'sale' and product_id='xxxxx' for update;
insert into user_order(user_id,product_store_id) values('xxxxxx','xxxxx');
update product_store set status = 'sold' where status = 'sale' and product_id='xxxxx';
commit;
```

售出的商品与用户的订单项一一对应的。

注意上面，这里使用了排它锁与事务处理，防止一个商品卖给两个人。

根据上面的思路我们可以将商品属性与product_store表进行一对一匹配，这样每个商品都有它自己的商品属性，甚至价格也可以移到product_store表中，例如不同颜色售价不同。

9. 国际化语言表

```
     +-----------+              +---------------+
     | category  |    .---+     | category_lang |
     |-----------|   /    |     +---------------+
 +-->|id         | <---+  +--o  |category_id    |
 |   |title      |     |        |language_id    | o---+
 |   |description|    1:n       |name           |     |      +-------------+
 |   |status     |     |        +---------------+     .      | language    |
 |   |parent_id  | o---+                               \     +-------------+
 |   +-----------+                                      >--> |id           |
1:n                                                    /     |lang         |
 |   +------------+                                   '      |status       |
 |   | product    |                                   |      +-------------+
 |   +------------+            +--------------+       |
 |   |id          | <---+      | product_lang |       |
 |   |price       |     |      +--------------+       |
 |   |quantity    |     +---o  |product_id    |       |
 |   |...         |            |language_id   | o-----+
 +-o |category_id |            |name          |
     +------------+            +--------------+
```

10. Workflow

```
   +------------+     +---------------+     +-----------+
   | user       |     | role_has_user |     | role      |
   +------------+     +---------------+     +-----------+
   |id          |o-+  |id             |  +->|id         |<-+
   |node_id     |  +->|user_id        |  |  |name       |  |
   |up_id       |     |role_id        |o-+  |description|  |
   +------------+         +---------------+     +-----------+  |
                                                                 |
   +----------------+     +------------+                        |
   | workflow       |     | job        |                        |
   +----------------+     +------------+                        |
+->|id              |  +->|id          |                        |
|  |job_id          |o-+  |name        |                        |
+-o|up_id           |     |role_id     |o-----------------------+
   |                |     |description |
   +----------------+     +------------+
```

11. 内容版本控制

主表

```
CREATE TABLE `article` (
        `article_id` MEDIUMINT(8) UNSIGNED NOT NULL AUTO_INCREMENT,
        `cat_id` SMALLINT(5) NOT NULL DEFAULT '0',
        `title` VARCHAR(150) NOT NULL DEFAULT '',
        `content` LONGTEXT NOT NULL,
        `author` VARCHAR(30) NOT NULL DEFAULT '',
        `keywords` VARCHAR(255) NOT NULL DEFAULT '',
        PRIMARY KEY (`article_id`),
        INDEX `cat_id` (`cat_id`)
)
ENGINE=MyISAM
ROW_FORMAT=DEFAULT
AUTO_INCREMENT=1
```

本版控制表，用于记录每次变动

```
CREATE TABLE `article_history` (
        `id` MEDIUMINT(8) UNSIGNED NOT NULL AUTO_INCREMENT,
        `article_id` MEDIUMINT(8) UNSIGNED NOT NULL,
        `cat_id` SMALLINT(5) NOT NULL DEFAULT '0',
        `title` VARCHAR(150) NOT NULL DEFAULT '',
        `content` LONGTEXT NOT NULL,
        `author` VARCHAR(30) NOT NULL DEFAULT '',
        `keywords` VARCHAR(255) NOT NULL DEFAULT '',
        PRIMARY KEY (`id`),
        INDEX `article_id` (`article_id`)
)
ENGINE=MyISAM
ROW_FORMAT=DEFAULT
AUTO_INCREMENT=1
```

版本控制触发器

```
DROP TRIGGER article_history;

DELIMITER //
CREATE TRIGGER article_history BEFORE update ON article FOR EACH ROW
BEGIN
        INSERT INTO article_history SELECT * FROM article WHERE article_id = OLD.article_id;
END; //
DELIMITER;
```

12. logging 日志表的设计

```
CREATE TABLE `logging` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
        `tag` ENUM('unknow','www','user','admin') NOT NULL DEFAULT 'unknow' COMMENT '日志标签',
        `time` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '产生时间',
        `facility` ENUM('bank','unionpay','sms','email') NOT NULL COMMENT '类别',
        `priority` ENUM('info','warning','error','critical','exception','debug') NOT NULL COMMENT
        `message` VARCHAR(512) NOT NULL COMMENT '内容',
        PRIMARY KEY (`id`)
)
COMMENT='日志表'
COLLATE='utf8_general_ci'
ENGINE=InnoDB
AUTO_INCREMENT=2;
```

分区日志表

```
delimiter $$

CREATE TABLE `logging` (
  `tag` enum('unknow','login','info','admin','cron','manual') NOT NULL DEFAULT 'unknow' COMMENT '
  `asctime` datetime NOT NULL COMMENT '产生时间',
  `facility` enum('account','bank','unionpay','sms','email','unknow') NOT NULL DEFAULT 'unknow' C
  `priority` enum('info','warning','error','critical','exception','debug') NOT NULL DEFAULT 'debu
  `message` varchar(512) NOT NULL COMMENT '内容',
  `operator` varchar(50) NOT NULL DEFAULT 'computer' COMMENT '操作者'
) ENGINE=InnoDB DEFAULT CHARSET=utf8
/*!50100 PARTITION BY RANGE (YEAR(asctime))
SUBPARTITION BY HASH (MONTH(asctime))
(PARTITION p0 VALUES LESS THAN (1990) ENGINE = InnoDB,
 PARTITION p1 VALUES LESS THAN (2000) ENGINE = InnoDB,
 PARTITION p2 VALUES LESS THAN MAXVALUE ENGINE = InnoDB) */$$
```

分表+分区，每年分表一次，每个分区中保存一个月的数据

```
delimiter $$

CREATE TABLE `logging_2013` (
  `tag` enum('unknow','login','info','admin','cron','manual') NOT NULL DEFAULT 'unknow' COMMENT '
  `asctime` datetime NOT NULL COMMENT '产生时间',
  `facility` enum('account','bank','unionpay','sms','email','unknow') NOT NULL DEFAULT 'unknow' C
  `priority` enum('info','warning','error','critical','exception','debug') NOT NULL DEFAULT 'debu
  `message` varchar(512) NOT NULL COMMENT '内容',
  `operator` varchar(50) NOT NULL DEFAULT 'computer' COMMENT '操作者'
) ENGINE=InnoDB DEFAULT CHARSET=utf8
/*!50100 PARTITION BY LIST (MONTH(asctime))
SUBPARTITION BY KEY (facility)
(PARTITION part0 VALUES IN (1) ENGINE = InnoDB,
 PARTITION part1 VALUES IN (2) ENGINE = InnoDB,
 PARTITION part2 VALUES IN (3) ENGINE = InnoDB,
 PARTITION part3 VALUES IN (4) ENGINE = InnoDB,
 PARTITION part4 VALUES IN (5) ENGINE = InnoDB,
 PARTITION part5 VALUES IN (6) ENGINE = InnoDB,
 PARTITION part6 VALUES IN (7) ENGINE = InnoDB,
 PARTITION part7 VALUES IN (8) ENGINE = InnoDB,
```

```
 PARTITION part8 VALUES IN (9) ENGINE = InnoDB,
 PARTITION part9 VALUES IN (10) ENGINE = InnoDB,
 PARTITION part10 VALUES IN (11) ENGINE = InnoDB,
 PARTITION part11 VALUES IN (12) ENGINE = InnoDB) */$$
```

命名分区

```
delimiter $$

CREATE TABLE `logging_2012` (
  `tag` enum('unknow','login','info','admin','cron','manual') NOT NULL DEFAULT 'unknow' COMMENT '
  `asctime` datetime NOT NULL COMMENT '产生时间',
  `facility` enum('account','bank','unionpay','sms','email','unknow') NOT NULL DEFAULT 'unknow' C
  `priority` enum('info','warning','error','critical','exception','debug') NOT NULL DEFAULT 'debu
  `message` varchar(512) NOT NULL COMMENT '内容',
  `operator` varchar(50) NOT NULL DEFAULT 'computer' COMMENT '操作者'
) ENGINE=InnoDB DEFAULT CHARSET=utf8
/*!50100 PARTITION BY LIST (MONTH(asctime))
SUBPARTITION BY KEY (facility)
(PARTITION January VALUES IN (1) ENGINE = InnoDB,
 PARTITION February VALUES IN (2) ENGINE = InnoDB,
 PARTITION March VALUES IN (3) ENGINE = InnoDB,
 PARTITION April VALUES IN (4) ENGINE = InnoDB,
 PARTITION May VALUES IN (5) ENGINE = InnoDB,
 PARTITION June VALUES IN (6) ENGINE = InnoDB,
 PARTITION July VALUES IN (7) ENGINE = InnoDB,
 PARTITION August VALUES IN (8) ENGINE = InnoDB,
 PARTITION September VALUES IN (9) ENGINE = InnoDB,
 PARTITION October VALUES IN (10) ENGINE = InnoDB,
 PARTITION November VALUES IN (11) ENGINE = InnoDB,
 PARTITION December VALUES IN (12) ENGINE = InnoDB) */$$
```

13. uuid 替代传统序列 id

```
DROP TABLE IF EXISTS `uuid_test`;
CREATE TABLE IF NOT EXISTS `uuid_test` (
  `uuid` varchar(36) NOT NULL,
  `name` varchar(20) NOT NULL,
  PRIMARY KEY (`uuid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='uuid 测试';
```

插入触发器

```
DROP TRIGGER IF EXISTS `uuid_test_insert`;
SET @OLDTMP_SQL_MODE=@@SQL_MODE, SQL_MODE='';
DELIMITER //
CREATE TRIGGER `uuid_test_insert` BEFORE INSERT ON `uuid_test` FOR EACH ROW BEGIN
        IF new.uuid is null or new.uuid = '' or length(new.uuid) != 36 THEN
                set new.uuid=uuid();
        END IF;
END//
DELIMITER ;
SET SQL_MODE=@OLDTMP_SQL_MODE;
```

ID放撰改触发器

```
DROP TRIGGER IF EXISTS `uuid_test_update`;
SET @OLDTMP_SQL_MODE=@@SQL_MODE, SQL_MODE='';
DELIMITER //
CREATE TRIGGER `uuid_test_update` BEFORE UPDATE ON `uuid_test` FOR EACH ROW BEGIN
        set new.uuid = old.uuid;
END//
DELIMITER ;
SET SQL_MODE=@OLDTMP_SQL_MODE;
```

14. 动态配置表

很多时候我们需要使用数据库存储配置项，由于各种原因我们无法使用配置文件来完成，例如在一个有很多节点集群环境中使用文件配置文件时非常不方便。

```
DROP TABLE IF EXISTS `config`;
CREATE TABLE IF NOT EXISTS `config` (
  `key` varchar(50) NOT NULL,
  `value` varchar(50) NOT NULL,
  `operator` varchar(50) NOT NULL DEFAULT 'dba',
  `mtime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`key`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='网站动态配置文件';
```

```
INSERT INTO `config` (`key`, `value`, `operator`, `mtime`) VALUES
        ('cache.apc', 'ON', 'dba', '2013-07-18 16:17:13'),
        ('cache.file.path', '/tmp', 'dba', '2013-07-18 16:17:57'),
        ('cache.redis', 'YES', 'dba', '2013-07-18 16:17:22'),
        ('payment.alipay.status', 'Enabled', 'dba', '2013-07-18 16:15:15'),
        ('payment.alipay.url', 'http://xx.comx.com', 'dba', '2013-07-18 16:16:38'),
        ('payment.yeepay.status', 'Enabled', 'dba', '2013-07-18 16:15:17'),
        ('payment.99bill.status', 'Enabled', 'dba', '2013-07-18 16:15:10'),
        ('payment.zqpay.status', 'Disabled', 'dba', '2013-07-18 16:15:20');
```

配置项key的写法很讲究

```
单个配置
database.host=localhost
database.user=user
database.pass=pass

多个配置
database.1.host=localhost
database.1.user=user
database.1.pass=pass
database.1.status=1

database.2.host=localhost
database.2.user=user
database.2.pass=pass
database.2.status=1

优化配置项，例如：payment.alipay.status 可以这样优化
payment.status.alipay
payment.status.yeepay
```

这样做的目的是为了更好的使用like进行查询

```
select `key`,`value` from config where `key` like 'payment.status.%';
select `key`,`value` from config where `key` like 'database.?.status';
```

14.1. 配置表历史记录

我有一个表，里面只有固定行数的行记录，这些数据就是配置参数，我们将配置文件保存在数据库中，因为需要做负载均衡而不能使用文件配置文件。

有这样一个需求，这个记录每次修改都要保存历史记录，用于审计等等。我是这样设计该表的

```
CREATE TABLE `config_fee` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
        `level` INT(11) NULL DEFAULT NULL COMMENT '层级',
        `type` ENUM('Deposit','Withdrawing') NOT NULL DEFAULT 'Withdrawing' COMMENT '类型，存款，取
        `min_fee` FLOAT(10,2) NOT NULL COMMENT '最低手续费',
        `max_fee` FLOAT(10,2) NOT NULL COMMENT '最高手续费',
        `ratio` FLOAT(10,2) NOT NULL COMMENT '手续费比例',
        `operator` VARCHAR(10) NOT NULL COMMENT '操作者',
        `status` ENUM('Current','Trash') NOT NULL DEFAULT 'Current',
        `ctime` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
        `mtime` TIMESTAMP NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP,
        PRIMARY KEY (`id`)
)
COMMENT='手续费管理'
COLLATE='utf8_general_ci'
ENGINE=InnoDB;
```

数据记录的形态

```
mysql> select type,operator,status,ctime,mtime from config_mtf_fee;
+---------+----------+---------+---------------------+---------------------+
| type    | operator | status  | ctime               | mtime               |
+---------+----------+---------+---------------------+---------------------+
| Deposit | 141      | Trash   | 2014-08-28 11:10:17 | 2014-08-28 11:10:57 |
| Deposit | 141      | Trash   | 2014-08-28 11:10:17 | 2014-08-28 11:10:57 |
| Deposit | 141      | Trash   | 2014-08-28 11:10:17 | 2014-08-28 11:10:57 |
| Deposit | 141      | Trash   | 2014-08-28 11:10:17 | 2014-08-28 11:10:57 |
| Deposit | 324      | Current | 2014-08-28 11:10:54 | 2014-08-28 11:10:59 |
+---------+----------+---------+---------------------+---------------------+
2 rows in set (0.00 sec)
```

如上图所示，状态 Current 是当前记录，而Trash是废弃的历史记录。

每次修改数据，首先将Current改为Trash，然后插入一条新数据状态为Current，我们只会使用最后一条状态为current的数据。

15. 验证码

用户注册，登陆等等需要验证码，下面的方案是，请求验证码生成一个随机验证码，存在code中，identity可以存储来源IP/手机号码/Cookie等等用户校验， 5分钟内如果没有被使用就会删除，如果5分钟内被使用也会删除。type 可以存放www,user,bbs,admin.....等等，将所有验证码放在captcha表中统一管理。

这个方案主要是考虑没有memcache/redis/apc/xcache等等缓存环境下的解决方案，你也可以将下表改造一下，增加ttl字段用于存放生存时间，而不是采用5分钟一刀切的方案。

```
CREATE TABLE `captcha` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
        `type` ENUM('user','admin') NOT NULL DEFAULT 'admin' COMMENT '验证码类型',
        `identity` VARCHAR(32) NOT NULL COMMENT '唯一身份识别md5摘要',
        `code` VARCHAR(6) NOT NULL COMMENT '验证码',
        `ctime` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT
        PRIMARY KEY (`id`),
        UNIQUE INDEX `code` (`code`),
        UNIQUE INDEX `identity` (`identity`)
)
COMMENT='验证码'
COLLATE='utf8_general_ci'
ENGINE=InnoDB;



CREATE EVENT `captcha` ON SCHEDULE
                EVERY 5 MINUTE STARTS '2013-07-08 16:27:03'
        ON COMPLETION PRESERVE
        ENABLE
        COMMENT ''
        DO BEGIN
        delete from captcha where type='myid' and ctime < DATE_ADD(now(), INTERVAL -5 MINUTE);
END
```

16. 手机归属地数据库表

members_location 表与 members 表是一对一关系，该表只负责存储归属地信息

```
DROP TABLE IF EXISTS `members_location`;
CREATE TABLE IF NOT EXISTS `members_location` (
  `id` int(10) unsigned NOT NULL,
  `province` varchar(50) NOT NULL,
  `city` varchar(50) NOT NULL,
  `ctime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`),
  KEY `province` (`province`),
  KEY `city` (`city`),
  CONSTRAINT `FK_members_location_members` FOREIGN KEY (`id`) REFERENCES `members` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

当某些用户符合条件需要查询归属地是，只要将其插入到 members_mobile 表即可。该表使用黑洞引擎并不会存储手机号码，所以明文手机号码安全得到了保障。

```
DROP TABLE IF EXISTS `members_mobile`;
CREATE TABLE IF NOT EXISTS `members_mobile` (
  `id` int(10) NOT NULL,
  `number` varchar(11) NOT NULL
) ENGINE=BLACKHOLE DEFAULT CHARSET=utf8;
```

当有数据进入到 members_mobile 时出发器 members_mobile_insert 会工作，去 mobile_location 表中查询归属地后保存在 members_location 表中

```
DROP TRIGGER IF EXISTS `members_mobile_insert`;
SET @OLDTMP_SQL_MODE=@@SQL_MODE, SQL_MODE='';
DELIMITER //
CREATE TRIGGER `members_mobile_insert` BEFORE INSERT ON `members_mobile` FOR EACH ROW BEGIN
        insert into members_location(id,province,city) select NEW.id,mobile_location.province,mob
END//
DELIMITER ;
SET SQL_MODE=@OLDTMP_SQL_MODE;
```

mobile_location 是存储手机号段与归属地信息的数据库

```
DROP TABLE IF EXISTS `mobile_location`;
CREATE TABLE IF NOT EXISTS `mobile_location` (
  `id` varchar(50) NOT NULL,
  `province` varchar(50) DEFAULT NULL,
  `city` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

17. 数据检查

17.1. 身份证校验

该函数能够检查身份证号码是否正确

```
CREATE DEFINER=`neo`@`%` FUNCTION `check_id_number`(`idnumber` CHAR(18))
        RETURNS enum('true','false')
        LANGUAGE SQL
        NOT DETERMINISTIC
        NO SQL
        SQL SECURITY DEFINER
        COMMENT ''
BEGIN
DECLARE status ENUM('true','false') default 'false';
DECLARE verify CHAR(1);
DECLARE sigma INT;
DECLARE remainder INT;

IF length(idnumber) = 18 THEN
        set sigma = cast(substring(idnumber,1,1) as UNSIGNED) * 7
                +cast(substring(idnumber,2,1) as UNSIGNED) * 9
                +cast(substring(idnumber,3,1) as UNSIGNED) * 10
                +cast(substring(idnumber,4,1) as UNSIGNED) * 5
                +cast(substring(idnumber,5,1) as UNSIGNED) * 8
                +cast(substring(idnumber,6,1) as UNSIGNED) * 4
                +cast(substring(idnumber,7,1) as UNSIGNED) * 2
                +cast(substring(idnumber,8,1) as UNSIGNED) * 1
                +cast(substring(idnumber,9,1) as UNSIGNED) * 6
                +cast(substring(idnumber,10,1) as UNSIGNED) * 3
                +cast(substring(idnumber,11,1) as UNSIGNED) * 7
                +cast(substring(idnumber,12,1) as UNSIGNED) * 9
                +cast(substring(idnumber,13,1) as UNSIGNED) * 10
                +cast(substring(idnumber,14,1) as UNSIGNED) * 5
                +cast(substring(idnumber,15,1) as UNSIGNED) * 8
                +cast(substring(idnumber,16,1) as UNSIGNED) * 4
                +cast(substring(idnumber,17,1) as UNSIGNED) * 2;
        set remainder = MOD(sigma,11);
        set verify = (case remainder
                when 0 then '1' when 1 then '0' when 2 then 'X' when 3 then '9'
                when 4 then '8' when 5 then '7' when 6 then '6' when 7 then '5'
                when 8 then '4' when 9 then '3' when 10 then '2' else '/' end
        );

END IF;

IF right(idnumber,1) = verify THEN
        set status = 'true';
END IF;

RETURN status;
```

```
END
```

首先我们使用正确身份证号码进行测试，返回true

```
mysql> select check_id_number('330702198003090915');
+----------------------------------------+
| check_id_number('330702198003090915') |
+----------------------------------------+
| true                                   |
+----------------------------------------+
1 row in set (0.01 sec)
```

长度不符合18位直接返回false.

```
mysql> select check_id_number('33070219800309');
+-----------------------------------+
| check_id_number('33070219800309') |
+-----------------------------------+
| false                             |
+-----------------------------------+
1 row in set (0.00 sec)

mysql> select check_id_number('3307021980030909145788 9');
+---------------------------------------------+
| check_id_number('3307021980030909145788 9') |
+---------------------------------------------+
| false                                       |
+---------------------------------------------+
1 row in set, 1 warning (0.00 sec)
```

随便改译为数，校验失败返回 false

```
mysql> select check_id_number('330702198003090914');
+----------------------------------------+
| check_id_number('330702198003090914') |
+----------------------------------------+
| false                                  |
+----------------------------------------+
1 row in set (0.00 sec)
```

18. 创建与修改时间

ctime 为创建时间, mtime 是修改时间默认CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP

```
CREATE TABLE `trades_platform` (
        `id` INT(10) UNSIGNED NULL DEFAULT NULL,
        `type` ENUM('A','B','C') NOT NULL,
        `login` VARCHAR(10) NOT NULL,
        `password` VARCHAR(10) NOT NULL,
        `ctime` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
        `mtime` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
)
COMMENT='平台'
ENGINE=InnoDB;
```

我们还可以让mtime默认为NULL,这样更节省存储空间。

```
        `ctime` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
        `mtime` TIMESTAMP NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP,
```

19. 在线用户表

该表的功能是显示在线用户，控制多点登陆，防止异常退出

```
CREATE TABLE IF NOT EXISTS `employees_online` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `username` varchar(20) NOT NULL COMMENT 'User ID',
  `ipaddr` varchar(15) NOT NULL COMMENT 'IP Address',
  `expire` datetime NOT NULL COMMENT 'Expire time',
  `status` enum('Login','Logout','Offline') NOT NULL DEFAULT 'Login' COMMENT 'Current Status',
  `message` varchar(255) NOT NULL COMMENT 'Leave a message',
  `ctime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT 'Created Time',
  `mtime` timestamp NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP COMMENT 'Modified Time',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='show listing of last logged in users';

DELIMITER //
CREATE DEFINER=`dba`@`192.168.%` EVENT `employees_online` ON SCHEDULE EVERY 15 MINUTE STARTS '201
        update employees_online set `status` = 'Offline' where expire < now() and ctime > DATE_AD
END//
DELIMITER ;

SET @OLDTMP_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_ENGINE_SUBSTITUTION';
DELIMITER //
CREATE TRIGGER `employees_online_before_delete` BEFORE DELETE ON `employees_online` FOR EACH ROW
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Permission denied', MYSQL_ERRNO = 1001;
END//
DELIMITER ;
SET SQL_MODE=@OLDTMP_SQL_MODE;

SET @OLDTMP_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_ENGINE_SUBSTITUTION';
DELIMITER //
CREATE TRIGGER `employees_online_before_update` BEFORE UPDATE ON `employees_online` FOR EACH ROW
        SET NEW.`id` = OLD.id;
        SET NEW.`username` = OLD.username;
        SET NEW.`ipaddr` = OLD.ipaddr;
        SET NEW.`message` = OLD.message;
        SET NEW.`ctime` = OLD.ctime;
END//
DELIMITER ;
SET SQL_MODE=@OLDTMP_SQL_MODE;
```

登陆日志将永久保存，防止数据被删除由触发器employees_online_before_delete负责

防止有人撰改登陆信息，由触发器employees_online_before_update负责，主要是防止撰改登陆名与IP地址，这样讲不能从其他电脑登陆，必须用户Logout才能在其他电脑登陆。

20. HTML TO Text

实现 PHP strip_tags 函数的功能。

```
CREATE DEFINER=`dba`@`%` FUNCTION `strip_tags`(`$str` TEXT)
RETURNS text CHARSET utf8
LANGUAGE SQL
NOT DETERMINISTIC
CONTAINS SQL
SQL SECURITY DEFINER
COMMENT ''
BEGIN
    DECLARE $start, $end INT DEFAULT 1;
    LOOP
        SET $start = LOCATE("<", $str, $start);
        IF (!$start) THEN RETURN $str; END IF;
        SET $end = LOCATE(">", $str, $start);
        IF (!$end) THEN SET $end = $start; END IF;
        SET $str = INSERT($str, $start, $end - $start + 1, "");
    END LOOP;
END
```

```
mysql> select strip_tags('<span><i>hello</i> <b>world</b>!!! <br /><a href="//www.netkiller/">net
+----------------------------------------------------------------------+
| strip_tags('<span>hel<b>lo <a href="world">wo<>rld</a> <<x>again<.') |
+----------------------------------------------------------------------+
| hello world again.                                                   |
+----------------------------------------------------------------------+
1 row in set

mysql> select strip_tags('<span style="color:red"><i>hello</i> <b id="world" >world</b>!!! <br />
+--------------------------+
| TEXT                     |
+--------------------------+
| hello world!!! netkiller |
+--------------------------+
1 row in set (0.00 sec)
```

21. SNS 数据库设计

这里讲解SNS交友社区的数据库设计与实现

我们要实现下面几个功能

1. 朋友之间的关系，多对多关系
2. 朋友之间的维度，如3度4度....
3. 朋友的查找

```
CREATE DATABASE `sns` /*!40100 COLLATE 'utf8_general_ci' */
```

21.1. people 表

people 是存储人，你可以用为user,member都可以

```
CREATE TABLE `people` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
        `name` VARCHAR(50) NOT NULL,
        PRIMARY KEY (`id`)
)
COMMENT='Social Network Site - Six Degrees of Separation - http://www.netkiller.cn'
COLLATE='utf8_general_ci'
ENGINE=InnoDB;
```

存储具体的这人

21.2. firend 表

这个表的功能主要是维持朋友之间的关系网，这里使用了多对多方式并且使用外键防止产生脏数据。

```
CREATE TABLE `friend` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
        `people_id` INT(10) UNSIGNED NOT NULL,
        `friend_id` INT(10) UNSIGNED NOT NULL,
        `ctime` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
        PRIMARY KEY (`id`),
        UNIQUE INDEX `unique` (`people_id`, `friend_id`),
        INDEX `FK_firend_people` (`people_id`),
        INDEX `FK_firend_people_2` (`friend_id`),
        CONSTRAINT `FK_firend_people` FOREIGN KEY (`people_id`) REFERENCES `people` (`id`),
        CONSTRAINT `FK_firend_people_2` FOREIGN KEY (`friend_id`) REFERENCES `people` (`id`)
)
COMMENT='Social Network Site - Six Degrees of Separation - http://www.netkiller.cn'
COLLATE='utf8_general_ci'
ENGINE=InnoDB;
```

21.3. 演示

首先初始化用户数据

```
INSERT INTO `people` (`id`, `name`) VALUES
        (1, 'Neo'),
        (2, 'Luke'),
        (3, 'Jack'),
        (4, 'Joey'),
        (5, 'Jam'),
        (6, 'John');
```

建立朋友之间的关系

```
INSERT INTO `friend` (`id`, `people_id`, `friend_id`) VALUES
        (1, 1, 2),
        (2, 1, 3),
        (3, 1, 4),
        (4, 1, 5),
        (5, 1, 6),
        (6, 2, 1),
        (7, 2, 3);
```

现在就可以查找你的朋友了

```
select people.* from friend, people where friend.people_id = 1 and friend.friend_id = people.id;
```

查找朋友的朋友就比较麻烦了，必须使用递归方法，一层一层查下去，反复执行SQL效率是很低的，所以我们准备了第三张表。

21.4. network 表

关系网表，主要功能是弥补firend表，用于快速检索（在不使用递归的情况下）

```
CREATE TABLE `network` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
        `people_id` INT(10) UNSIGNED NOT NULL,
        `following_id` INT(10) UNSIGNED NOT NULL,
        `friend_id` INT(10) UNSIGNED NULL DEFAULT NULL,
        `degrees` VARCHAR(250) NOT NULL,
        `ctime` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
        PRIMARY KEY (`id`),
        UNIQUE INDEX `unique` (`people_id`, `friend_id`, `following_id`),
        INDEX `FK_firend_people` (`people_id`),
        INDEX `FK_firend_people_2` (`friend_id`),
        INDEX `FK_friend_people_following_id` (`following_id`),
        CONSTRAINT `FK_firend_people` FOREIGN KEY (`people_id`) REFERENCES `people` (`id`),
        CONSTRAINT `FK_friend_people_following_id` FOREIGN KEY (`following_id`) REFERENCES `peopl
        CONSTRAINT `FK_friend_people_friend_id` FOREIGN KEY (`friend_id`) REFERENCES `people` (`i
)
COMMENT='Social Network Site - Six Degrees of Separation - http://www.netkiller.cn'
COLLATE='utf8_general_ci'
ENGINE=InnoDB;
```

following 一个朋友, Neo following Jam

```
INSERT INTO `people` (`id`, `name`) VALUES
        (1, 'Neo'),
        (2, 'Luke'),
        (3, 'Jack'),
        (4, 'Joey'),
        (5, 'Jam'),
        (6, 'John');

INSERT INTO `network` (`people_id`, `following_id`, `friend_id`, `degrees`) VALUES ( 1, 5, NULL,
```

之前Neo已经 following Jam，接下来查找Jam的朋友，现在Neo following John, John 是 Jam 的朋友，
friend_id = NULL 表示 Jam 尚未有朋友

```
select * from network where people_id=1 and friend_id = 5;

INSERT INTO `sns`.`network` (`people_id`, `following_id`, `friend_id`, `degrees`) VALUES ('1', '6
```

Neo following Joey, Joey 是 Luke 的朋友， 所以 Luke可能是 Neo的朋友

```
INSERT INTO `sns`.`network` (`people_id`, `following_id`, `friend_id`, `degrees`) VALUES ('1', '4
```

查询不同维度下的所有好友，查询出的用户ID需要处理。

```
select * from network where people_id=1 and degrees like "1.%";
select * from network where people_id=1 and degrees like "1.2%";
select * from network where people_id=1 and degrees like "1.2.%";
```

至此社区管理网就建立起来了

上面的例子演示了 people_id=1 即 Neo 的关系网

22. 数据库与缓存

22.1. 什么是数据库缓存?

这里讲的缓存是数据库本身的缓存，并不是外部缓存例如Redis/Memcache等等。

数据库的数据分为冷数据和热数据库，通俗的讲冷数据是存储在磁盘上不经常查询的数据；而热数据是频繁查询的数据，这部分数据会被缓存到内存中。

22.2. 为什么缓存数据呢?

因为频繁查询相同结果集的数据时，每次到磁盘上查找数据是非常耗时的，所以数据库将频繁查询且返回相同结果集的数据放到内存中，可以减少磁盘访问操作。

22.3. 什么时候使用数据库缓存

频繁访问且返回相同结果集的情况下使用缓存。

偶尔查询一次且间隔时间较长的情况下不要使用缓存。

尺寸较大的结果集不建议使用缓存，因为数据太大太大，缓存不足以存储，会导致频繁载入与销毁，命中率低。

通常数据库默认情况是开启缓存的，也就是说正常的select查询，如果符合缓存规则就会经过缓存。

当一条SQL查询时如果结果集在内存中称作“命中”

22.4. 涉及缓存的地方有哪些

数据库本身，查看数据库缓存状态

数据库应用程序接口（ODBC、JDBC.....）

22.5. 谁来控制数据库缓存

通常DBA只能控制数据库缓存是否开启，分配多少内存给缓存使用，过期销毁时间，以及策略等等.

上面我已经说过，通常数据库默认都开启缓存，所以更多的时候我们的操作是禁用缓存。这就需要开发人员来通过特定的SQL操作来控制数据库缓存。

22.6. 怎么控制数据库缓存

以 MySQL 为例

```
mysql> show variables like '%query_cache%';
+------------------------------+---------+
| Variable_name                | Value   |
+------------------------------+---------+
| have_query_cache             | YES     |
| query_cache_limit            | 1048576 |
| query_cache_min_res_unit     | 4096    |
```

```
| query_cache_size             | 1048576 |
| query_cache_type             | OFF     |
| query_cache_wlock_invalidate | OFF     |
+------------------------------+---------+
6 rows in set (0.04 sec)
```

编辑 my.cnf 文件，加入配置项 query_cache_type=1 然后重启mysql服务

```
mysql> show variables like '%query_cache%';
+------------------------------+---------+
| Variable_name                | Value   |
+------------------------------+---------+
| have_query_cache             | YES     |
| query_cache_limit            | 1048576 |
| query_cache_min_res_unit     | 4096    |
| query_cache_size             | 1048576 |
| query_cache_type             | ON      |
| query_cache_wlock_invalidate | OFF     |
+------------------------------+---------+
6 rows in set (0.00 sec)
```

query_cache_type | ON 表示缓存已经开启。

22.6.1. SQL_CACHE 缓存

默认情况 select 查询操作只要符合数据库缓存规则那么结果集就会被缓存，如果你的数据库没有开启缓存，请参考下面

```
set session query_cache_type=on;

flush tables;
show status like 'qcache_q%';
select sql_cache * from member where id=1;
show status like 'qcache_q%';
select sql_cache * from member where id=1;
show status like 'qcache_q%';
```

例 20.2. 演示 SQL_CACHE

```
mysql> flush tables;
Query OK, 0 rows affected (0.00 sec)

mysql> show status like 'qcache_q%';
+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| Qcache_queries_in_cache | 0     |
+-------------------------+-------+
1 row in set (0.00 sec)

mysql> select sql_cache * from member where id=1;
+----+-----+---------------------+------------+--------+-------+------+---------+------+--------+
| id | age | ctime               | ip_address | mobile | mtime | name | picture | sex  | status |
+----+-----+---------------------+------------+--------+-------+------+---------+------+--------+
|  1 |   1 | 2017-08-24 17:05:43 | 1          | NULL   | NULL  | 1    | 1       | 1    | Enable |
```

```
+----+-----+---------------------+------------+--------+-------+------+---------+------+--------+
1 row in set (0.00 sec)

mysql> show status like 'qcache_q%';
+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| Qcache_queries_in_cache | 1     |
+-------------------------+-------+
1 row in set (0.01 sec)

mysql> select sql_cache * from member where id=1;
+----+-----+---------------------+------------+--------+-------+------+---------+------+--------+
| id | age | ctime               | ip_address | mobile | mtime | name | picture | sex  | status |
+----+-----+---------------------+------------+--------+-------+------+---------+------+--------+
|  1 |   1 | 2017-08-24 17:05:43 | 1          | NULL   | NULL  | 1    | 1       | 1    | Enable |
+----+-----+---------------------+------------+--------+-------+------+---------+------+--------+
1 row in set (0.00 sec)

mysql> show status like 'qcache_q%';
+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| Qcache_queries_in_cache | 1     |
+-------------------------+-------+
1 row in set (0.01 sec)
```

我们可以看到 Qcache_queries_in_cache 值由0转为1表示缓存已经生效。

22.6.2. 禁止缓存 SQL_NO_CACHE

这里我们主要讲怎样禁止缓存，使查询出的结果集不进入缓存。

```
SELECT SQL_NO_CACHE * FROM table where id=xxxx
```

下面的用法比较安全，切换到其他数据库也能正常工作

```
SELECT /*!40001 SQL_NO_CACHE */ * FROM table
```

```
set session query_cache_type=on;

flush tables;
show status like 'qcache_q%';
select sql_no_cache * from member where id=1;
show status like 'qcache_q%';
select sql_no_cache * from member where id=1;
show status like 'qcache_q%';
```

例 20.3. 演示 SQL_NO_CACHE

```
mysql> flush tables;
Query OK, 0 rows affected (0.00 sec)

mysql> show status like 'qcache_q%';
+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| Qcache_queries_in_cache | 0     |
```

```
+-------------------------+-------+
1 row in set (0.00 sec)

mysql> select sql_no_cache * from member where id=1;
+----+-----+---------------------+------------+--------+-------+------+---------+------+--------+
| id | age | ctime               | ip_address | mobile | mtime | name | picture | sex  | status |
+----+-----+---------------------+------------+--------+-------+------+---------+------+--------+
|  1 |   1 | 2017-08-24 17:05:43 | 1          | NULL   | NULL  | 1    | 1       | 1    | Enable |
+----+-----+---------------------+------------+--------+-------+------+---------+------+--------+
1 row in set (0.00 sec)

mysql> show status like 'qcache_q%';
+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| Qcache_queries_in_cache | 0     |
+-------------------------+-------+
1 row in set (0.00 sec)

mysql> select sql_no_cache * from member where id=1;
+----+-----+---------------------+------------+--------+-------+------+---------+------+--------+
| id | age | ctime               | ip_address | mobile | mtime | name | picture | sex  | status |
+----+-----+---------------------+------------+--------+-------+------+---------+------+--------+
|  1 |   1 | 2017-08-24 17:05:43 | 1          | NULL   | NULL  | 1    | 1       | 1    | Enable |
+----+-----+---------------------+------------+--------+-------+------+---------+------+--------+
1 row in set (0.00 sec)

mysql> show status like 'qcache_q%';
+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| Qcache_queries_in_cache | 0     |
+-------------------------+-------+
1 row in set (0.00 sec)
```

使用 sql_no_cache 查询 Qcache_queries_in_cache 值始终是 0

22.6.3. 关闭缓存 set session query_cache_type=off

我们使用 set session query_cache_type=off 可以关闭本次查询缓存。

```
set session query_cache_type=off;

flush tables;
show status like 'qcache_q%';
select sql_cache * from member where id=1;
show status like 'qcache_q%';
select sql_cache * from member where id=1;
show status like 'qcache_q%';
```

例 20.4. 演示 query_cache_type=off 关闭查询缓存

```
mysql> set session query_cache_type=off;
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> flush tables;
Query OK, 0 rows affected (0.00 sec)

mysql> show status like 'qcache_q%';
+-------------------------+-------+
| Variable_name           | Value |
```

```
+-------------------------+-------+
| Qcache_queries_in_cache | 0     |
+-------------------------+-------+
1 row in set (0.00 sec)

mysql> select sql_cache * from member where id=1;
+----+-----+---------------------+------------+--------+-------+------+---------+------+--------+
| id | age | ctime               | ip_address | mobile | mtime | name | picture | sex  | status |
+----+-----+---------------------+------------+--------+-------+------+---------+------+--------+
|  1 |   1 | 2017-08-24 17:05:43 | 1          | NULL   | NULL  | 1    | 1       | 1    | Enable |
+----+-----+---------------------+------------+--------+-------+------+---------+------+--------+
1 row in set (0.00 sec)

mysql> show status like 'qcache_q%';
+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| Qcache_queries_in_cache | 0     |
+-------------------------+-------+
1 row in set (0.00 sec)

mysql> select sql_cache * from member where id=1;
+----+-----+---------------------+------------+--------+-------+------+---------+------+--------+
| id | age | ctime               | ip_address | mobile | mtime | name | picture | sex  | status |
+----+-----+---------------------+------------+--------+-------+------+---------+------+--------+
|  1 |   1 | 2017-08-24 17:05:43 | 1          | NULL   | NULL  | 1    | 1       | 1    | Enable |
+----+-----+---------------------+------------+--------+-------+------+---------+------+--------+
1 row in set (0.00 sec)

mysql> show status like 'qcache_q%';
+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| Qcache_queries_in_cache | 0     |
+-------------------------+-------+
1 row in set (0.00 sec)
```

23. PostgreSQL 所特有数据库设计

PostgreSQL 数据库 ORDBMS / OODBMS 等特有属性

对象相关数据库管理系统(ORDBMS Object – Oriented Relative DBMS)

23.1. 国家地区表的设计

```
 +-----------+
 | city      |
 |-----------|
 |id         | <---+
 |name       |     |
 |description|    1:n
 |status     |     |
 |parent_id  | o---+
 +-----------+
```

例 20.5. 递归查询实例 city 表

定义结构

```
CREATE TABLE city
(
  id serial NOT NULL,
  name character varying,
  parent_id integer,
  status boolean,
  CONSTRAINT city_pkey PRIMARY KEY (id),
  CONSTRAINT city_parent_id_fkey FOREIGN KEY (parent_id)
      REFERENCES city (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
ALTER TABLE city
  OWNER TO sys;
```

插入数据

```
INSERT INTO city (id, name, parent_id, status) VALUES (1, '广东', NULL, NULL);
INSERT INTO city (id, name, parent_id, status) VALUES (2, '湖南', NULL, NULL);
INSERT INTO city (id, name, parent_id, status) VALUES (3, '深圳', 1, NULL);
INSERT INTO city (id, name, parent_id, status) VALUES (4, '东莞', 1, NULL);
INSERT INTO city (id, name, parent_id, status) VALUES (5, '福田', 3, NULL);
INSERT INTO city (id, name, parent_id, status) VALUES (6, '南山', 3, NULL);
INSERT INTO city (id, name, parent_id, status) VALUES (7, '宝安', 3, NULL);
INSERT INTO city (id, name, parent_id, status) VALUES (8, '西乡', 7, NULL);
INSERT INTO city (id, name, parent_id, status) VALUES (9, '福永', 7, NULL);
INSERT INTO city (id, name, parent_id, status) VALUES (10, '龙华', 7, NULL);
INSERT INTO city (id, name, parent_id, status) VALUES (11, '长沙', 2, NULL);
```

```
INSERT INTO city (id, name, parent_id, status) VALUES (12, '湘潭', 2, NULL);
INSERT INTO city (id, name, parent_id, status) VALUES (13, '常德', 2, NULL);
INSERT INTO city (id, name, parent_id, status) VALUES (14, '桃源', 13, NULL);
INSERT INTO city (id, name, parent_id, status) VALUES (15, '汉寿', 13, NULL);
INSERT INTO city (id, name, parent_id, status) VALUES (16, '黑龙江', NULL, NULL);
INSERT INTO city (id, name, parent_id, status) VALUES (17, '伊春', 16, NULL);
INSERT INTO city (id, name, parent_id, status) VALUES (18, '哈尔滨', 16, NULL);
INSERT INTO city (id, name, parent_id, status) VALUES (19, '齐齐哈尔', 16, NULL);
INSERT INTO city (id, name, parent_id, status) VALUES (20, '牡丹江', 16, NULL);
INSERT INTO city (id, name, parent_id, status) VALUES (21, '佳木斯', 16, NULL);
INSERT INTO city (id, name, parent_id, status) VALUES (22, '民治', 10, NULL);
INSERT INTO city (id, name, parent_id, status) VALUES (23, '上塘', 10, NULL);
```

查询

```
WITH RECURSIVE path(id, name, path, idpath, parent_id, status) AS (
  SELECT id, name, '/' || name , '/' || id , parent_id, status FROM city WHERE parent_id is null
  UNION
  SELECT
    city.id,
    city.name,
    parentpath.path ||
      CASE parentpath.path
        WHEN '/' THEN ''
        ELSE '/'
      END || city.name,
    parentpath.idpath ||
     CASE parentpath.idpath
        WHEN '/' THEN ''
        ELSE '/'
      END || city.id,
    city.parent_id, city.status
  FROM city, path as parentpath
  WHERE city.parent_id = parentpath.id
)

SELECT * FROM path;
```

结果输出

```
 id |   name    |           path             |    idpath    | parent_id | status
----+----------+----------------------------+--------------+-----------+--------
  1 | 广东      | /广东                       | /1           |           |
  2 | 湖南      | /湖南                       | /2           |           |
 16 | 黑龙江    | /黑龙江                     | /16          |           |
  3 | 深圳      | /广东/深圳                  | /1/3         |         1 |
  4 | 东莞      | /广东/东莞                  | /1/4         |         1 |
 11 | 长沙      | /湖南/长沙                  | /2/11        |         2 |
 12 | 湘潭      | /湖南/湘潭                  | /2/12        |         2 |
 13 | 常德      | /湖南/常德                  | /2/13        |         2 |
 17 | 伊春      | /黑龙江/伊春                | /16/17       |        16 |
 18 | 哈尔滨    | /黑龙江/哈尔滨              | /16/18       |        16 |
 19 | 齐齐哈尔  | /黑龙江/齐齐哈尔            | /16/19       |        16 |
 20 | 牡丹江    | /黑龙江/牡丹江              | /16/20       |        16 |
 21 | 佳木斯    | /黑龙江/佳木斯              | /16/21       |        16 |
  5 | 福田      | /广东/深圳/福田             | /1/3/5       |         3 |
```

```
  6 | 南山      | /广东/深圳/南山            | /1/3/6       |          3 |
  7 | 宝安      | /广东/深圳/宝安            | /1/3/7       |          3 |
 14 | 桃源      | /湖南/常德/桃源            | /2/13/14     |         13 |
 15 | 汉寿      | /湖南/常德/汉寿            | /2/13/15     |         13 |
  8 | 西乡      | /广东/深圳/宝安/西乡       | /1/3/7/8     |          7 |
  9 | 福永      | /广东/深圳/宝安/福永       | /1/3/7/9     |          7 |
 10 | 龙华      | /广东/深圳/宝安/龙华       | /1/3/7/10    |          7 |
 22 | 民治      | /广东/深圳/宝安/龙华/民治 | /1/3/7/10/22 |         10 |
 23 | 上塘      | /广东/深圳/宝安/龙华/上塘 | /1/3/7/10/23 |         10 |
(23 rows)
```

23.2. 话题讨论表的设计

例 20.6. 话题讨论表的设计

http://justcramer.com/2010/05/30/scaling-threaded-comments-on-django-at-disqus/

```
create table comments (
    id SERIAL PRIMARY KEY,
    message VARCHAR,
    author VARCHAR,
    parent_id INTEGER REFERENCES comments(id)
);
insert into comments (message, author, parent_id)
    values ('This thread is really cool!', 'David', NULL), ('Ya David, we love it!', 'Jason', 1),
    ('Very interesting post!', 'thedz', NULL), ('You sir, are wrong', 'Chris', 5), ('Agreed', 'G'
```

```
WITH RECURSIVE cte (id, message, author, path, parent_id, depth)  AS (
    SELECT  id,
        message,
        author,
        array[id] AS path,
        parent_id,
        1 AS depth
    FROM    comments
    WHERE   parent_id IS NULL

    UNION ALL

    SELECT  comments.id,
        comments.message,
        comments.author,
        cte.path || comments.id,
        comments.parent_id,
        cte.depth + 1 AS depth
    FROM    comments
    JOIN cte ON comments.parent_id = cte.id
    )
    SELECT id, message, author, path, depth FROM cte ORDER BY path;
```

输出结果

```
 id |           message           | author |  path    | depth
----+-----------------------------+--------+----------+-------
  1 | This thread is really cool! | David  | {1}      |     1
  2 | Ya David, we love it!       | Jason  | {1,2}    |     2
```

```
  4 | gift Jason                  | Anton   | {1,2,4} |     3
  3 | I agree David!              | Daniel  | {1,3}   |     2
  5 | Very interesting post!      | thedz   | {5}     |     1
  6 | You sir, are wrong          | Chris   | {5,6}   |     2
  7 | Agreed                      | G       | {5,7}   |     2
  8 | Fo sho, Yall                | Mac     | {5,8}   |     2
(8 rows)
```

23.3. 账户表/余额表/消费储蓄表

此表适用于购物车等金钱来往账面等等。

```
-- Table: account

-- DROP TABLE account;

CREATE TABLE account
(
  id integer NOT NULL DEFAULT nextval('trade_id_seq'::regclass),
  no character varying(10) NOT NULL, -- 账号
  balance money NOT NULL DEFAULT 0.00, -- 余额
  datetime timestamp without time zone NOT NULL DEFAULT (now())::timestamp(0) without time zone,
  CONSTRAINT account_pkey PRIMARY KEY (id)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE account
  OWNER TO dba;
COMMENT ON COLUMN account.no IS '账号';
COMMENT ON COLUMN account.balance IS '余额';


-- Index: account_no_idx

-- DROP INDEX account_no_idx;

CREATE INDEX account_no_idx
  ON account
  USING btree
  (no COLLATE pg_catalog."default");
```

账户结余计算

```
select acc.*, (select sum(balance)+acc.balance from account as ac where ac.id < acc.id) as profit

test=# select acc.*, (select sum(balance)+acc.balance from account as ac where ac.id < acc.id) as
 id |  no  | balance  |      datetime       | profit
----+------+----------+---------------------+---------
  1 | 1000 |    $0.00 | 2013-10-09 10:51:10 |
  2 | 1000 |   $12.60 | 2013-10-09 10:51:22 |  $12.60
  4 | 1000 |   $16.80 | 2013-10-09 10:51:42 |  $29.40
  5 | 1000 |  $100.00 | 2013-10-09 10:51:49 | $129.40
  6 | 1000 |  $200.00 | 2013-10-09 10:56:35 | $329.40
  7 | 1000 |   $50.45 | 2013-10-09 10:57:23 | $379.85
  8 | 1000 |   $75.50 | 2013-10-09 10:57:31 | $455.35
  9 | 1000 |  -$55.30 | 2013-10-09 10:59:28 | $400.05
 10 | 1000 | -$200.00 | 2013-10-09 10:59:44 | $200.05
(9 rows)
```

24. 数据库并行访问控制

这里主要讲述有关开发中遇到的数据库并行问题

24.1. 防止并行显示

背景

我们有一个order订单表，工作流如下

```
创建订单 -> 订单分配 -> 订单审核 -> 批准 -> 发货 ... 等等
```

有多个岗位，每个岗位上有多个工作人员。需要实现相同岗位上的工作人员看到的订单不能重复，防止多人同时操作一个订单。

```
id | user | sn    | status
-----------------------------------
1  | neo  | x001  | new
2  | jam  | x002  | new
3  | sam  | x003  | new
4  | tom  | x004  | new
5  | ann  | x005  | new
6  | leo  | x006  | new
7  | ant  | x007  | new
8  | cat  | x008  | new
```

正常情况只要是多人一起打开订单页面就会显示上面的订单，并且每个人显示的内容都相同。

```
CREATE TABLE `orders` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
        `name` VARCHAR(50) NOT NULL,
        `sn` INT(10) UNSIGNED ZEROFILL NOT NULL,
        `status` ENUM('New','Pending','Processing','Success','Failure') NOT NULL DEFAULT 'New',
        PRIMARY KEY (`id`),
        UNIQUE INDEX `sn` (`sn`)
)
COMMENT='订货单'
COLLATE='utf8_general_ci'
ENGINE=InnoDB
```

```
INSERT INTO `orders` (`id`, `name`, `sn`, `status`) VALUES
        (1, 'neo', 0000000001, 'New'),
        (2, 'jam', 0000000002, 'New'),
        (3, 'sam', 0000000003, 'New'),
        (4, 'tom', 0000000004, 'New'),
        (5, 'ann', 0000000005, 'New'),
        (6, 'leo', 0000000006, 'New'),
        (7, 'ant', 0000000007, 'New'),
        (8, 'cat', 0000000008, 'New');
```

表 20.1. 工作流模拟

操作	订单审核员 A

显示未处理订单,这里模拟两个人同时点开页面的情景	`begin;`
	`select id from orders where status='New' limit 5 for update;`
	`update orders set status='Pending' where status='New' and id in (1,2,3,4,5);`
	`select * from orders where status='Pending' and id in (1,2,3,4,5) order by id asc limit 5;`
	`commit;`
	
	首先查询出数据库中的前五条记录，然后更新为Pending状态，防止他人抢占订单。

处理订单，模拟两个人点击审批通过按钮是的情景	`begin;`
	`select * from orders where status='Pending' and id='1' for update;`
	`update orders set status='Processing' where status='Pending' and id=1;`
	`commit;`
	
	更新状态Pending到Processing

处理成功与失败的情况	`begin;`
	`select * from orders where status='Processing' and id='1' for update;`
	`update orders set status='Success' where status='Processing' and id=1;`
	`commit;`

处理Pending状态的订单，可能产生冲突，不用担心有行锁，防止重复处理。	`begin;`
	`select * from orders where status='Processing' and id='5' for update;`
	`update orders set status='Failure' where status='Processing' and id=5;`
	`commit;`

有一种情况，用户查看了列表并未及时处理订单，就会有很多Pending状态的订单，这是需要有人处理这些订单，但查询Pending时，可能同一时刻有人在审批订单，我们通过排他锁避免重复处理。

上面以MySQL为例，每次都需要使用for update 查出要处理的订单，如果是PostgreSQL 可以使用update + returning 来返回修改的数据，更为方便。

25. Sharding

Sharding是近几年提出的概念，可以做分表，分库切割，通过hash值定位。但都存在一个问题，数据连续性，索引无法跨表。

Oracle 在8.x中就支持分区功能，MySQL在5.1.x中也是闲类似功能，PostgreSQL 因存储结构设计的较好，基本不需要做分区。

25.1. horizontal

```
ALTER TABLE `goods`  DROP INDEX `goods_sn_2`;
ALTER TABLE goods PARTITION BY RANGE (goods_id) (
    PARTITION p0 VALUES LESS THAN (10000),
    PARTITION p1 VALUES LESS THAN (20000),
    PARTITION p2 VALUES LESS THAN (30000),
    PARTITION p3 VALUES LESS THAN (40000),
    PARTITION p4 VALUES LESS THAN MAXVALUE
);

ALTER TABLE goods PARTITION BY HASH(goods_id) PARTITIONS 10;

ALTER TABLE goods  PARTITION BY KEY (is_on_sale) PARTITIONS 2;

ALTER TABLE goods PARTITION BY HASH(YEAR(FROM_UNIXTIME(add_time))) PARTITIONS 4;
```

25.2. vertical

25.3. 新闻数据库分表案例

这里我通过一个新闻网站为例，解决分表的问题

避免开发中经常拼接表，我采用一个一劳永逸的方法，建立一个 news 表使用黑洞引擎，然后通过出发器将数据分流到匹配的表中。同时采用uuid替代数字序列，可以保证未来数年不会出现ID用尽。

```
CREATE TABLE IF NOT EXISTS `news` (
  `uuid` varchar(36) NOT NULL COMMENT '唯一ID',
  `title` varchar(50) NOT NULL COMMENT '新闻标题',
  `body` text NOT NULL COMMENT '新闻正文',
  `ctime` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00' COMMENT '创建时间',
  `mtime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT '修改时
  `atime` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00' COMMENT '访问时间',
  PRIMARY KEY (`uuid`)
) ENGINE=BLACKHOLE DEFAULT CHARSET=utf8;
```

该表仅仅用于举例，结构比较简单。接下来创建年份分表，你也可以每个月一个表，根据你的许下灵活调整。表结构与上面的news表相同，注意 ENGINE=InnoDB。

```
CREATE TABLE IF NOT EXISTS `news_2012` (
  `uuid` varchar(36) NOT NULL COMMENT '唯一ID',
```

```
  `title` varchar(50) NOT NULL COMMENT '新闻标题',
  `body` text NOT NULL COMMENT '新闻正文',
  `ctime` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00' COMMENT '创建时间',
  `mtime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT '修改时
  `atime` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00' COMMENT '访问时间',
  PRIMARY KEY (`uuid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='news 表';

CREATE TABLE IF NOT EXISTS `news_2013` (
  `uuid` varchar(36) NOT NULL COMMENT '唯一ID',
  `title` varchar(50) NOT NULL COMMENT '新闻标题',
  `body` text NOT NULL COMMENT '新闻正文',
  `ctime` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00' COMMENT '创建时间',
  `mtime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT '修改时
  `atime` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00' COMMENT '访问时间',
  PRIMARY KEY (`uuid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='news 表';
```

uuid 索引表，主要的功能是通过uuid查询出该记录在那张表中。更好的方案是将数据放入solr中处理，包括标题与内容搜索等等。

```
CREATE TABLE `news_index` (
        `uuid` VARCHAR(36) NOT NULL,
        `tbl_name` VARCHAR(10) NOT NULL,
        PRIMARY KEY (`uuid`)
)
COMMENT='news uuid 索引表'
COLLATE='utf8_general_ci'
ENGINE=InnoDB;
```

news_insert 过程，用于向目标表中插入数据，可以单独call 但不建议。因为insert 远比 call 更通用，要考虑移植性与通用性

```
DELIMITER //
CREATE DEFINER=`neo`@`%` PROCEDURE `news_insert`(IN `uuid` vARCHAR(36), IN `title` VARCHAR(50), I
BEGIN
        if year(ctime) = '2012' then
                insert into news_2012(uuid,title,body,ctime) values(uuid,title, body, ctime);
        end if;
        if year(ctime) = '2013' then
                insert into news_2013(uuid,title,body,ctime) values(uuid,title, body, ctime);
        end if;
        insert into news_index values(uuid, year(ctime));
END//
DELIMITER ;
```

插入触发器，负责获取 uuid 然后调用存储过程

```
SET @OLDTMP_SQL_MODE=@@SQL_MODE, SQL_MODE='';
DELIMITER //
CREATE TRIGGER `news_before_insert` BEFORE INSERT ON `news` FOR EACH ROW BEGIN
        IF new.uuid is null or new.uuid = '' or length(new.uuid) != 36 THEN
```

```
                set new.uuid=uuid();
        END IF;
        call news_insert(new.uuid,new.title,new.body,new.ctime);
END//
DELIMITER ;
SET SQL_MODE=@OLDTMP_SQL_MODE;
```

这个触发器用户保护表中的 uuid 值不被修改。

```
SET @OLDTMP_SQL_MODE=@@SQL_MODE, SQL_MODE='';
DELIMITER //
CREATE TRIGGER `news_before_update` BEFORE UPDATE ON `news_2013` FOR EACH ROW BEGIN
        set new.uuid = old.uuid;
END//
DELIMITER ;
SET SQL_MODE=@OLDTMP_SQL_MODE;
```

26. 数据库安全

26.1. 数据库结构版本控制

http://netkiller.github.io/journal/mysql.struct.html

26.1.1. 什么是数据库结构版本控制

首先说说什么是数据库结构，什么事版本控制。

数据库结构是指数据库表结构，数据库定义语言导出的DDL语句。主要由CREATE TABLE, DROP TABLE等等构成。

再来说说什么事版本控制，如果你从事开发工作应该会很容易理解，版本控制就是记录每一次变化，可以随时查看历史记录，并可回撤到指定版本。

26.1.2. 为什么要做数据库结构本版控制

软件开发过程中需要常常对数据库结构作调整，这是无法避免的，甚至很多想起启动后，需求还不明确，开发人员只能按照所理解需求创建表。需求往往会发生变化，一旦变化，代码需要修改，表结构也避免不了。 我们常常刚改好数据库结构，需求部门有发来通知，不用修改了，维持原有设计。甚至是过了几周再次回撤。

所以我们要将数据库结构的变化进行版本控制，通常的做法是DBA人工管理，但我觉完全可以自动化的工作，没有必要浪费人力资源，且自动化不会犯错更稳定，仅仅需要人工定期查看工作状态即可。

26.1.3. 何时做数据库结构本版控制

任何时候都可以部署下面的脚本，对现有系统无任何影响。

26.1.4. 在哪里做数据库结构本版控制

可以在版本控制服务器上，建议GIT仓库push到远程。

26.1.5. 谁来负责数据库结构本版控制

DBA与配置管理员都可以做，通常DBA不接触版本库这块，建议创建一个backup用户给配置管理员。

26.1.6. 怎样做数据库结构本版控制

26.1.6.1. 安装脚本

首先下载脚本 https://github.com/oscm/devops/blob/master/shell/backup.mysql.struct.sh

```
wget https://raw.githubusercontent.com/oscm/devops/master/shell/backup.mysql.struct.sh
mv backup.mysql.struct.sh /usr/local/bin
chmod +x /usr/local/bin/backup.mysql.struct
```

创建备份用户

```
CREATE USER 'backup'@'localhost' IDENTIFIED BY 'SaJePoM6BAPOmOFOd7Xo3e1A52vEPE';
GRANT SELECT, LOCK TABLES  ON *.* TO 'backup'@'localhost';
FLUSH PRIVILEGES;
SHOW GRANTS FOR 'backup'@'localhost';
```

配置脚本

```
BACKUP_HOST="localhost"                 数据库主机
BACKUP_USER="backup"                    备份用户
BACKUP_PASS="chen"                              备份密码
BACKUP_DBNAME="test aabbcc"             版本控制那些数据库，多个数据库使用空格分隔
BACKUP_DIR=~/backup                             数据库结构放在那里
```

初始化仓库

```
# /usr/local/bin/backup.mysql.struct init
Initialized empty Git repository in /www/database/struct/.git/
```

26.1.6.2. 启动脚本，停止脚本

```
# /usr/local/bin/backup.mysql.struct
Usage: /usr/local/bin/backup.mysql.struct {init|start|stop|status|restart}
```

开始脚本

```
# /usr/local/bin/backup.mysql.struct start
```

查看状态

```
# /usr/local/bin/backup.mysql.struct status
 9644 pts/1    S      0:00 /bin/bash /usr/local/bin/backup.mysql.struct start
```

停止脚本

```
# /usr/local/bin/backup.mysql.struct status
```

26.1.6.3. 查看历史版本

通过 git log 命令查看历史版本

```
# cd /www/database/struct/

# git status
# On branch master
nothing to commit (working directory clean)
```

```
# git log
commit d38fc624c21cad0e2f55f0228bff0c1be981827c
Author: root <root@slave.example.com>
Date:   Wed Dec 17 12:33:55 2014 +0800

    2014-12-17.04:33:55
```

这里仅仅将数据库结构版本控制，关于版本控制软件更多细节，延伸阅读 《Netkiller Version 手札》

26.2. 保护表

保护表中的数据不被删除，当记录被用户删除时会提示"Permission denied" 权限拒绝

```
CREATE DEFINER=`root`@`192.168.%` TRIGGER `member_before_delete` BEFORE DELETE ON `member` FOR EA
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Permission denied', MYSQL_ERRNO = 1001;
END
```

26.3. 保护表字段

通过触发器，使之无法修改某些字段的数据，同时不影响修改其他字段。

```
DROP TRIGGER IF EXISTS `members`;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='';
DELIMITER //
CREATE TRIGGER `members` BEFORE UPDATE ON `members` FOR EACH ROW BEGIN
set new.name = old.name;
set new.cellphone = old.cellphone;
set new.email = old.email;
set new.password = old.password;
END//
DELIMITER ;
SET SQL_MODE=@OLD_SQL_MODE;
```

再举一个例子

```
CREATE TABLE `account` (
`id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
`user` VARCHAR(50) NOT NULL DEFAULT '0',
`cash` FLOAT NOT NULL DEFAULT '0',
PRIMARY KEY (`id`)
)
COLLATE='utf8_general_ci'
ENGINE=InnoDB;
```

每一次数据变化新增一条数据

```
INSERT INTO `test`.`account` (`user`, `cash`) VALUES ('neo', -10);
INSERT INTO `test`.`account` (`user`, `cash`) VALUES ('neo', -5);
INSERT INTO `test`.`account` (`user`, `cash`) VALUES ('neo', 30);
INSERT INTO `test`.`account` (`user`, `cash`) VALUES ('neo', -20);
```

保护用户的余额不被修改

```
DROP TRIGGER IF EXISTS `account`;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='';
DELIMITER //
CREATE TRIGGER `account` BEFORE UPDATE ON `account` FOR EACH ROW BEGIN
set new.cash = old.cash;
END//
DELIMITER ;
SET SQL_MODE=@OLD_SQL_MODE;
```

26.4. 时间一致性

经常会因为每个服务器的时间不同，导致插入数据有问题，虽然可以采用ntp服务同步时间，但由于各种因素仍然会出问题，怎么解决？我建议以数据库时间为准。

MySQL 5.6 之前的版本

默认值为当前时间

```
CREATE TABLE `tdate` (
`id` INT(11) NOT NULL AUTO_INCREMENT,
`ctime` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
`mtime` TIMESTAMP NOT NULL DEFAULT '0000-00-00 00:00:00' COMMENT '修改时间
PRIMARY KEY (`id`)
)
COLLATE='utf8_general_ci'
ENGINE=InnoDB;
```

MySQL不允许一个表拿有两个默认时间。我一无法兼顾修改时间，我们舍弃创建时间，当有数据变化ON UPDATE CURRENT_TIMESTAMP自动修改时间

```
CREATE TABLE `tdate` (
`id` INT(11) NOT NULL AUTO_INCREMENT,
`ctime` TIMESTAMP NOT NULL DEFAULT '0000-00-00 00:00:00' COMMENT '创建时间
`mtime` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TI
PRIMARY KEY (`id`)
)
COLLATE='utf8_general_ci'
ENGINE=InnoDB;
```

插入创建时间 insert into tdate(ctime) values(CURRENT_TIMESTAMP); 不要采用 insert into tdate(ctime) values('2013-12-02 08:20:06');这种方法，尽量让数据库处理时间。

MySQL 5.6 之后版本，可以实现创建时间为系统默认，修改时间创建的时候默认为空，当修改数据的时候更新时间。

```
CREATE TABLE `tdate` (
        `id` INT(11) NOT NULL AUTO_INCREMENT,
        `ctime` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
        `mtime` TIMESTAMP NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP COMMENT '修改时间',
        PRIMARY KEY (`id`)
)
COLLATE='utf8_general_ci'
ENGINE=InnoDB;
```

26.5. 为数据安全而分库

我们通常使用一个数据库开发，该数据库包含了前后台所有的功能，我建议将前后台等等功能进行分库然后对应各种平台分配用户权限，例如

我们创建三个数据库cms,frontend,backend 同时对应创建三个用户 cms,frontend,backend 三个用户只能分别访问自己的数据库，注意在系统的设计之初你要考虑好这样的划分随之系统需要做相应的调整。

```
CREATE DATABASE `cms` /*!40100 COLLATE 'utf8_general_ci' */;
CREATE DATABASE `frontend` /*!40100 COLLATE 'utf8_general_ci' */;
CREATE DATABASE `backend` /*!40100 COLLATE 'utf8_general_ci' */;
```

backend 负责后台，权限最高

```
mysql> SHOW GRANTS FOR 'backend'@'localhost';
+-----------------------------------------------------------------------
| Grants for backend@localhost |
+-----------------------------------------------------------------------
| GRANT USAGE ON *.* TO 'backend'@'localhost' |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `cms`.* TO 'backend'@'localhost
| GRANT SELECT, INSERT, UPDATE, DELETE ON `frontend`.* TO 'backend'@'loca
| GRANT SELECT, INSERT, UPDATE, DELETE, CREATE ON `backend`.* TO 'backend
+-----------------------------------------------------------------------
4 rows in set (0.04 sec)
```

frontend 是前台权限，主要是用户用户中心，用户注册，登录，用户信息资料编辑，查看新闻等等

```
mysql> SHOW GRANTS FOR 'frontend'@'localhost';
+-----------------------------------------------------------------------
| Grants for frontend@localhost |
+-----------------------------------------------------------------------
| GRANT USAGE ON *.* TO 'frontend'@'localhost' |
| GRANT SELECT, INSERT, UPDATE ON `frontend`.* TO 'frontend'@'localhost'
| GRANT SELECT ON `cms`.`news` TO 'frontend'@'localhost' |
+-----------------------------------------------------------------------
3 rows in set (0.00 sec)
```

cms 用户是网站内容管理，主要负责内容更新，但登陆CMS后台需要`backend`.`Employees`表用户认证，所以他需要读取权限，但不允许修改其中的数据。

```
mysql> SHOW GRANTS FOR 'cms'@'localhost';
+-----------------------------------------------------------------------+
| Grants for cms@localhost |
+-----------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'cms'@'localhost' |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `cms`.* TO 'cms'@'localhost' |
| GRANT SELECT ON `backend`.`Employees` TO 'cms'@'localhost' |
+-----------------------------------------------------------------------+
3 rows in set (0.00 sec)
```

26.6. 内容版本控制,撰改留痕

主表

```
CREATE TABLE `article` (
```

```
`article_id` MEDIUMINT(8) UNSIGNED NOT NULL AUTO_INCREMENT,
`cat_id` SMALLINT(5) NOT NULL DEFAULT '0',
`title` VARCHAR(150) NOT NULL DEFAULT '',
`content` LONGTEXT NOT NULL,
`author` VARCHAR(30) NOT NULL DEFAULT '',
`keywords` VARCHAR(255) NOT NULL DEFAULT '',
PRIMARY KEY (`article_id`),
INDEX `cat_id` (`cat_id`)
)
ENGINE=MyISAM
ROW_FORMAT=DEFAULT
AUTO_INCREMENT=1
```

用于记录每次修改变动，通过该表，可以追朔数据库记录被什么时候修改过，修改了那些内容。

```
CREATE TABLE `article_history` (
`id` MEDIUMINT(8) UNSIGNED NOT NULL AUTO_INCREMENT,
`article_id` MEDIUMINT(8) UNSIGNED NOT NULL,
`cat_id` SMALLINT(5) NOT NULL DEFAULT '0',
`title` VARCHAR(150) NOT NULL DEFAULT '',
`content` LONGTEXT NOT NULL,
`author` VARCHAR(30) NOT NULL DEFAULT '',
`keywords` VARCHAR(255) NOT NULL DEFAULT '',
PRIMARY KEY (`id`),
INDEX `article_id` (`article_id`)
)
ENGINE=MyISAM
ROW_FORMAT=DEFAULT
AUTO_INCREMENT=1
```

版本控制触发器

```
DROP TRIGGER article_history;

DELIMITER //
CREATE TRIGGER article_history BEFORE update ON article FOR EACH ROW
BEGIN
INSERT INTO article_history SELECT * FROM article WHERE article_id = OLD.
END; //
DELIMITER;
```

进一步优化，我们可以为 history 历史表增加时间字段，用于记录被撰改那一时刻的时间。

```
CREATE TABLE `article_history` (
        `id` MEDIUMINT(8) UNSIGNED NOT NULL AUTO_INCREMENT,
        `article_id` MEDIUMINT(8) UNSIGNED NOT NULL,
        `cat_id` SMALLINT(5) NOT NULL DEFAULT '0',
        `title` VARCHAR(150) NOT NULL DEFAULT '',
        `content` LONGTEXT NOT NULL,
        `author` VARCHAR(30) NOT NULL DEFAULT '',
        `keywords` VARCHAR(255) NOT NULL DEFAULT '',
        `ctime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT 'Created Time',
        `mtime` timestamp NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP COMMENT 'Modified Time',
        PRIMARY KEY (`id`),
        INDEX `article_id` (`article_id`)
)
ENGINE=MyISAM
ROW_FORMAT=DEFAULT
AUTO_INCREMENT=1
```

我们还可以为该表（article_history）增加出发器，任何修改将被拒绝.

26.7. 数据库审计表

与上一章节所提到的历史表不同，历史表需要经常翻查所以我们需要用到索引。审计表通常是数据归档，不允许修改，且基本上很少访问。

```
CREATE TABLE `order` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT COMMENT '订单ID',
  `name` varchar(45) NOT NULL COMMENT '订单名称',
  `price` float NOT NULL COMMENT '价格',
  `ctime` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='订单表'
```

基于 order 表创建 order_audit 审计表

```
create table order_audit engine=archive as select * from `order`;
```

order_audit 表结构如下

```
CREATE TABLE `order_audit` (
  `id` int(10) unsigned NOT NULL DEFAULT '0' COMMENT '订单ID',
  `name` varchar(45) NOT NULL COMMENT '订单名称',
  `price` float NOT NULL COMMENT '价格',
  `ctime` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间'
) ENGINE=ARCHIVE DEFAULT CHARSET=utf8
```

创建插入和更新触发器，用于插入和修改的时候同事写入一份到归档表中。

```
DROP TRIGGER IF EXISTS `test`.`order_AFTER_INSERT`;

DELIMITER $$
USE `test`$$
CREATE DEFINER=`dba`@`%` TRIGGER `test`.`order_AFTER_INSERT` AFTER INSERT ON `order` FOR EACH ROW
BEGIN
        INSERT INTO order_audit SELECT * FROM `order` WHERE id = NEW.id;
END$$
DELIMITER ;
DROP TRIGGER IF EXISTS `test`.`order_AFTER_UPDATE`;

DELIMITER $$
USE `test`$$
CREATE DEFINER=`dba`@`%` TRIGGER `test`.`order_AFTER_UPDATE` AFTER UPDATE ON `order` FOR EACH ROW
BEGIN
        INSERT INTO order_audit SELECT * FROM `order` WHERE id = NEW.id;
END$$
DELIMITER ;
```

26.8. 用户/角色认证

本小节我们实现一个功能，当用户插入，修改或者删除数据时，判断该操作是否具备应有的权限。如果权限不符合就拒绝操作同时提示用户。

```
CREATE TABLE `staff` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '员工ID',
        `name` VARCHAR(50) NOT NULL COMMENT '员工名字',
        PRIMARY KEY (`id`)
)
COMMENT='员工表'
COLLATE='utf8_general_ci'
ENGINE=InnoDB;

INSERT INTO `staff` (`id`, `name`) VALUES
        (1, 'Neo'),
        (2, 'Luke'),
        (2, 'Jack');
```

staff 是员工表与下面的staff_has_role配合使用，形成员工与权限一对多关系。

```
CREATE TABLE `staff_has_role` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
        `staff_id` INT(10) UNSIGNED NOT NULL COMMENT '员工ID',
        `role` ENUM('Create','Update','Delete') NOT NULL COMMENT '角色',
        PRIMARY KEY (`id`),
        INDEX `FK_staff_has_role_staff` (`staff_id`),
        CONSTRAINT `FK_staff_has_role_staff` FOREIGN KEY (`staff_id`) REFERENCES `staff` (`id`) C
)
COLLATE='utf8_general_ci'
ENGINE=InnoDB;

INSERT INTO `staff_has_role` (`id`, `staff_id`, `role`) VALUES
        (1, 1, 'Create'),
        (2, 1, 'Delete'),
        (3, 1, 'Update'),
        (4, 2, 'Delete'),
        (5, 3, 'Create');
        (6, 3, 'Update');
```

权限表可以进一步优化，角色拥有组功能，实现颗粒度更细的权限控制，有情趣看前面的相关章节。

```
CREATE TABLE `product` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '唯一ID',
        `name` VARCHAR(10) NOT NULL COMMENT '名称',
        `sn` VARCHAR(10) NOT NULL COMMENT '序列号',
        `price` FLOAT NOT NULL COMMENT '价格',
        `amount` SMALLINT(6) NOT NULL COMMENT '数量',
        `staff_id` INT(10) UNSIGNED NOT NULL COMMENT '员工ID',
        `ctime` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
        `mtime` TIMESTAMP NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP COMMENT '修改时间',
        PRIMARY KEY (`id`),
        UNIQUE INDEX `sn` (`sn`),
```

```
        INDEX `FK_product_staff` (`staff_id`),
        CONSTRAINT `FK_product_staff` FOREIGN KEY (`staff_id`) REFERENCES `staff` (`id`)
)
COMMENT='产品表'
COLLATE='utf8_general_ci'
ENGINE=InnoDB;
```

以产品表为例，这里要实现的是对产品表记录的权限控制。例如Neo有用插入，修改和删除权限，Luke的Create与Update权限被吊销，只能删除他之前创建的数据。而Jack只有能创建于更新数据。

下面的三个触发器完成具体的权限控制。同样你可以进一步优化下面的代码的权限颗粒度，使之能控制到具体列，甚至具体的记录。

```
CREATE DEFINER=`root`@`%` TRIGGER `product_before_delete` BEFORE DELETE ON `product` FOR EACH ROW
        if not exists(select id from staff where id=OLD.staff_id and role="delete") then
                SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Permission denied', MYSQL_ERRNO = 100
        end if;
END

CREATE DEFINER=`root`@`%` TRIGGER `product_before_insert` BEFORE INSERT ON `product` FOR EACH ROW
         if not exists(select id from staff where id=NEW.staff_id and role="create") then
               SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = "The staff's role is not correct or it
         end if;
END

CREATE DEFINER=`root`@`%` TRIGGER `product_before_update` BEFORE UPDATE ON `product` FOR EACH ROW
        if not exists(select id from staff where id=NEW.staff_id and role="update") then
                SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = "The staff's role cannot update data."
        end if;
END
```

Neo 测试如下

```
INSERT INTO `test`.`product` (`name`, `sn`, `price`, `amount`, `staff_id`, `ctime`) VALUES ('Iphc
SELECT LAST_INSERT_ID();

UPDATE `test`.`product` SET `name`='HTC', `sn`='5544467', `price`='2000' WHERE  `id`=2;

DELETE FROM `test`.`product` WHERE  `id`=1;
```

Luke 测试如下:

```
INSERT INTO `test`.`product` (`name`, `sn`, `price`, `amount`, `staff_id`) VALUES ('Nokia', '6572
/* SQL错误（1001）：The staff's role is not correct or it does not exist. */

UPDATE `test`.`product` SET `name`='HTC', `sn`='5544467', `price`='2000', staff_id=2 WHERE  `id`=
/* SQL错误（1001）：The staff's role cannot update data. */
```

26.9. Token 认证

我们在staff表的基础上增加 token 字段

```
CREATE TABLE `staff` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '员工ID',
        `name` VARCHAR(50) NOT NULL COMMENT '员工名字',
        `token` VARCHAR(32) NOT NULL COMMENT 'Token 校验',
        PRIMARY KEY (`id`)
)
COMMENT='员工表'
COLLATE='utf8_general_ci'
ENGINE=InnoDB;
```

插入数据的时候增加一些干扰字符串，这里使用concat(NEW.id,'+',NEW.name,'-')

```
CREATE DEFINER=`root`@`%` TRIGGER `staff_before_insert` BEFORE INSERT ON `staff` FOR EACH ROW BEG

if md5(concat(NEW.id,'+',NEW.name,'-')) != NEW.token then
                SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Permission denied', MYSQL_ERRNO = 100
        end if;

END
```

注意表权限可以授权给用户，触发器权限不让普通用户查看。否则用户看到concat(NEW.id,'+',NEW.name,'-') 就没有意义了。

下面开始测试:

```
INSERT INTO `test`.`staff` (`name`, `token`) VALUES ('John', '678797066');
/* SQL错误（1001）：Permission denied */
```

下面再测试，首先生成一个正确的tokon, 然后使用该token插入数据:

```
-- 通过下面语句生成一个 Token
select md5(concat('5','+','Jam','-')) as token;

-- 使用上面的 Token 插入数据
INSERT INTO `test`.`staff` (`id`, `name`, `token`) VALUES (5, 'Jam', '1b033ce21cbadacabc9f0c38fb5

SELECT * FROM `test`.`staff` WHERE `id` = 5;
```

开发注意事项, Token 生成算法要保密，不要使用下面SQL提交数据

```
INSERT INTO `test`.`staff` (`id`, `name`, `token`) VALUES (5, 'Jam', md5(
```

应该分两步，一是计算Token，二是插入数据。可以将Token计算交给程序而不是SQL，并且封装在。jar(Java)中或者。so(PHP 扩展中).

26.10. 数据加密

数据库中有很多敏感字段，不允许随意查看，例如开发人员，运维人员，甚至DBA数据库管理员。另外加密主要是防止被黑客脱库（盗走）

敏感数据加密有很多办法，可以用数据库内部加密函数，也可以在外部处理后写入数据库。加密算法有很多种，但通常两类比较常用，一种是通过key加密解密，另一种是通过证书加密解密。

通常程序员负责写程序，程序交给运维配置，运维将key设置好，运维不能有数据库权限，DBA只能登陆数据库，没有key权限。

26.10.1. AES_ENCRYPT / AES_DECRYPT

这里介绍AES加密与解密简单用法

```
mysql> select AES_ENCRYPT('helloworld','key');
+---------------------------------+
| AES_ENCRYPT('helloworld','key') |
+---------------------------------+
|                                 |
+---------------------------------+
1 row in set (0.00 sec)

mysql> select AES_DECRYPT(AES_ENCRYPT('helloworld','key'),'key');
+---------------------------------------------------+
| AES_DECRYPT(AES_ENCRYPT('helloworld','key'),'key') |
+---------------------------------------------------+
| helloworld                                        |
+---------------------------------------------------+
1 row in set (0.00 sec)

mysql>
```

26.10.2. 加密字段

加密数据入库

```
CREATE TABLE `encryption` (
        `mobile` VARBINARY(16) NOT NULL,
        `key` VARCHAR(32) NOT NULL
)
ENGINE=InnoDB;

INSERT INTO encryption(`mobile`,`key`)VALUES( AES_ENCRYPT('13691851789',md5('13691851789')), md5(
select AES_DECRYPT(mobile,`key`), length(mobile) from encryption;
```

这里方便演示将key 写入了数据库，实际应用key应该存储在应用程序配置文件中。通常能把获得key的人不应该用数据库权限。

26.11. 开发加密插件开发

数据库内部提供的摘要函数MD5/SHA/CRC与现有的AES/DES加密函数以及不能满足我们的需求，所以我们有必要开发外挂插件实现数据加密。

这里有一个例子，是我早年开发的 https://github.com/netkiller/mysql-safenet-plugin 这个UDF是链接 Safenet 设备，实现数据库加密记录。

saftnet.h

```
my_bool safenet_encrypt_init(UDF_INIT *initid, UDF_ARGS *args, char *message);
char *safenet_encrypt(UDF_INIT *initid, UDF_ARGS *args, char *result, unsigned long *length, char
void safenet_encrypt_deinit(UDF_INIT *initid);

my_bool safenet_decrypt_init(UDF_INIT *initid, UDF_ARGS *args, char *message);
char *safenet_decrypt(UDF_INIT *initid, UDF_ARGS *args, char *result, unsigned long *length, char
void safenet_decrypt_deinit(UDF_INIT *initid);

my_bool safenet_config_init(UDF_INIT *initid, UDF_ARGS *args, char *message);
char *safenet_config(UDF_INIT *initid, UDF_ARGS *args, char *result, unsigned long *length, char
void safenet_config_deinit(UDF_INIT *initid);
```

safenet.c

```
/*
Homepage: http://netkiller.github.io/
Author: netkiller<netkiller@msn.com>
*/

#include <mysql.h>
#include <string.h>

#include <stdio.h>
#include <stdlib.h>
#include <curl/curl.h>
#include "safenet.h"

#define SAFENET_URL "http://localhost/safe/interface"
#define SAFENET_KEY "Web01-key"

char *safe_url;
char *safe_key;


void get_safenet_env(){
    if (getenv("SAFENET_URL")){
        safe_url = getenv("SAFENET_URL");
    }else{
        safe_url = SAFENET_URL;
    }
    if (getenv("SAFENET_KEY")){
        safe_key = getenv("SAFENET_KEY");
    }else{
        safe_key = SAFENET_KEY;
    }
}

/* CURL FUNCTION BEGIN*/
struct string {
  char *ptr;
  size_t len;
};

void init_string(struct string *s) {
  s->len = 0;
  s->ptr = malloc(s->len+1);
  if (s->ptr == NULL) {
    fprintf(stderr, "malloc() failed\n");
```

```
    exit(EXIT_FAILURE);
  }
  s->ptr[0] = '\0';
}

size_t writefunc(void *ptr, size_t size, size_t nmemb, struct string *s)
{
  size_t new_len = s->len + size*nmemb;
  s->ptr = realloc(s->ptr, new_len+1);
  if (s->ptr == NULL) {
    fprintf(stderr, "realloc() failed\n");
    exit(EXIT_FAILURE);
  }
  memcpy(s->ptr+s->len, ptr, size*nmemb);
  s->ptr[new_len] = '\0';
  s->len = new_len;

  return size*nmemb;
}

char * safenet(char *url, char *mode, char *key, char *in )
{
    CURL *curl;
    CURLcode res;
    char *fields;
    char *data;

//  curl_global_init(CURL_GLOBAL_ALL);

    /* get a curl handle */
    curl = curl_easy_init();
    if(curl) {
        struct string s;
        init_string(&s);

        asprintf(&fields, "mode=%s&keyname=%s&input=%s", mode, key, in);

        curl_easy_setopt(curl, CURLOPT_URL, url);
        curl_easy_setopt(curl, CURLOPT_USERAGENT, "safenet/1.0 by netkiller <netkiller@msn.com>")
        curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, writefunc);
        curl_easy_setopt(curl, CURLOPT_WRITEDATA, &s);
        curl_easy_setopt(curl, CURLOPT_POSTFIELDS, fields);

        /* Perform the request, res will get the return code */
        res = curl_easy_perform(curl);
        /* Check for errors */
        if(res != CURLE_OK)
          fprintf(stderr, "curl_easy_perform() failed: %s\n",
                  curl_easy_strerror(res));

        asprintf(&data, "%s", s.ptr);
        //printf("Encrypt: %s\n", data);

        free(s.ptr);
        /* always cleanup */
        curl_easy_cleanup(curl);
    }
    else{
        strcpy(data,"");
    }

    return data;
  //curl_global_cleanup();
}
/* CURL FUNCTION END*/

/* ------------------------ safenet encrypt ----------------------------- */

my_bool safenet_encrypt_init(UDF_INIT *initid, UDF_ARGS *args, char *message)
{

  if (args->arg_count != 1)
```

```
  {
    strncpy(message,
            "two arguments must be supplied: safenet_encrypt('<data>').",
            MYSQL_ERRMSG_SIZE);
    return 1;
  }
  get_safenet_env();
  args->arg_type[0]= STRING_RESULT;

  return 0;
}

char *safenet_encrypt(UDF_INIT *initid, UDF_ARGS *args,
                __attribute__ ((unused)) char *result,
               unsigned long *length,
                __attribute__ ((unused)) char *is_null,
                __attribute__ ((unused)) char *error)
{

    char *data;
    data = safenet(safe_url, "encrypt", safe_key, args->args[0]);
    *length = strlen(data);
    return ((char *)data);

}

void safenet_encrypt_deinit(UDF_INIT *initid)
{
  return;
}

/* ------------------------ safenet decrypt ----------------------------- */

my_bool safenet_decrypt_init(UDF_INIT *initid, UDF_ARGS *args, char *message)
{

  if (args->arg_count != 1)
  {
    strncpy(message,
            "two arguments must be supplied: safenet_decrypt('<data>').",
            MYSQL_ERRMSG_SIZE);
    return 1;
  }

  get_safenet_env();
  args->arg_type[0]= STRING_RESULT;

  return 0;
}

char *safenet_decrypt(UDF_INIT *initid, UDF_ARGS *args,
                __attribute__ ((unused)) char *result,
               unsigned long *length,
                __attribute__ ((unused)) char *is_null,
                __attribute__ ((unused)) char *error)
{

    char *data;
    if(strlen(args->args[0]) != 512){
        data = args->args[0];
    }else{
        data = safenet(safe_url, "decrypt", safe_key, args->args[0]);
    }
    *length = strlen(data);
    return ((char *)data);

}

void safenet_decrypt_deinit(UDF_INIT *initid)
{
  return;
}
```

```
/* ------------------------ safenet config ----------------------------- */

my_bool safenet_config_init(UDF_INIT *initid, UDF_ARGS *args, char *message)
{

    get_safenet_env();
    return 0;
}

char *safenet_config(UDF_INIT *initid, UDF_ARGS *args,
                __attribute__ ((unused)) char *result,
               unsigned long *length,
                __attribute__ ((unused)) char *is_null,
                __attribute__ ((unused)) char *error)
{

  char *config;
  asprintf(&config, "SAFENET_URL=%s, SAFENET_KEY=%s", safe_url, safe_key);
  *length = strlen(config);
  return ((char *)config);
}

void safenet_config_deinit(UDF_INIT *initid)
{
   return;
}
```

CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8)
PROJECT(safenet)
ADD_LIBRARY(safenet SHARED safenet.c)
INCLUDE_DIRECTORIES(/usr/include/mysql)
TARGET_LINK_LIBRARIES(safenet curl)
INSTALL(PROGRAMS libsafenet.so DESTINATION /usr/lib64/mysql/plugin/)
```

Installation Plugin

```
yum install -y libcurl-devel

cd src
cmake .
make
make install

cat > /etc/sysconfig/mysqld <<EOF
export SAFENET_URL=http://host.localdomain/safe/interface
export SAFENET_KEY=Web01-key
EOF
```

Create Function

```
create function safenet_encrypt returns string soname 'libsafenet.so';
create function safenet_decrypt returns string soname 'libsafenet.so';
create function safenet_config returns string soname 'libsafenet.so';
```

Example

```
mysql> select safenet_encrypt('Helloworld!!!');
+-----------------------------------------------------------------------------------------------
| safenet_encrypt('Helloworld!!!')
+-----------------------------------------------------------------------------------------------
| 994BAB7BC417F0559A09ECE94EDCB695AC1D5705F7ABA9F3562158F5AFAC4720FA9B3E53F30DF65C1726E0F02A93A9C
+-----------------------------------------------------------------------------------------------
1 row in set (0.58 sec)

mysql> select safenet_decrypt(safenet_encrypt('Helloworld!!!'));
+--------------------------------------------------+
| safenet_decrypt(safenet_encrypt('Helloworld!!!')) |
+--------------------------------------------------+
| Helloworld!!!                                    |
+--------------------------------------------------+
1 row in set (0.31 sec)

mysql> select safenet_config();
```

Drop Function

```
drop function safenet_encrypt;
drop function safenet_decrypt;
drop function safenet_config;
```

26.12. 数据区块链

背景：例如我们需要一个排行榜，存储活动的报名顺序或者考试成绩。我们防止有人作弊或者撰改，包括DBA在内。

任务：1.数据检查，2.发现撰改，2.风险提示

方案：使用链表指针方案,将数据看成一个链条，中间任何改动，就如同链条被剪断，改动之处之后的数据全部视为无效。

结果：达到数据后发现是否撰改，提示风险目的

```
CREATE TABLE `top100_list` (
        `id` INT,
        `name` VARBINARY(16) NOT NULL,
        ......
        ......
        `extend` VARCHAR(32) NULL
)
ENGINE=InnoDB;
```

演示数据

```
id | extend | ...
1 | 0 | ...
```

```
2 | 1 | ...
3 | 2 | ...
4 | 3 | ...
5 | 4 | ...
```

extend 始终集成上一条记录，保证数据是连续的。但这样还不够，这样只能防止数据被删除，如果其他字段被修改呢

```
id | extend | ...
1 | NULL | ...
2 | crc32(...) | ...
3 | crc32(...) | ...
4 | crc32(...) | ...
5 | crc32(...) | ...
```

我们使用crc算法运算上一条一整行的数据，你还可以使用 salt 技术干扰，这个 salt 只有软件部署者知道，DBA和开发人员不得而知。

对于一般数据crc32 可能做到性能和安全性平衡，如果安全要求更高可以使用 sha256 等等，甚至采用 RSA 非对称秘钥。

26.13. 状态保护

表中有一个 Status 字段，是一个状态机，你可以理解为工作流，工作流是有任务流向的，不能随意修改其状态。

```
CREATE TABLE `card` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
        `uuid` VARCHAR(36) NOT NULL COMMENT 'UUID' COLLATE 'utf8mb4_unicode_ci',
        `number` VARCHAR(36) NOT NULL COMMENT '充值卡号码' COLLATE 'utf8mb4_unicode_ci',
        `price` MEDIUMINT(8) UNSIGNED NOT NULL COMMENT '面值',
        `status` ENUM('New','Activated','Recharged','Discard') NOT NULL DEFAULT 'New' COMMENT '充
        `ctime` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
        `mtime` TIMESTAMP NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP COMMENT '修改时间',
        PRIMARY KEY (`id`),
        UNIQUE INDEX `number` (`number`),
        UNIQUE INDEX `uuid` (`uuid`)
)
COMMENT='充值卡表'
COLLATE='utf8mb4_unicode_ci'
ENGINE=InnoDB
AUTO_INCREMENT=1;
```

状态流向

```
        +----------+    +-----------+    +-----------+
        | New      | -> | Activated | -> | Recharged |
        +----------+    +-----------+    +-----------+
                 |                 |
                 V                 |
        +----------+          |
```

```
    | Discard  | <--------+
        +----------+
```

为此我们创建触发器保护状态正确走向。

```
CREATE DEFINER=`root`@`%` TRIGGER `card_before_update` BEFORE UPDATE ON `card` FOR EACH ROW BEGIN
        set new.uuid            = old.uuid;
        set new.number          = old.number;
        set new.price           = old.price;
        set new.ctime           = old.ctime;

        IF old.status = "New" THEN
                IF new.status NOT IN ("Activated","Discard") THEN
                        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Status denied', MYSQL_ERRNO =
                END IF;
        END IF;
        IF old.status = "Activated" THEN
                IF new.status NOT IN ("Recharged") THEN
                        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Status denied', MYSQL_ERRNO =
                END IF;
        END IF;
        IF old.status = "Recharged" THEN
                set new.status  = old.status;
        END IF;
END
```

保护记录不被删除

```
CREATE DEFINER=`root`@`%` TRIGGER `card_before_delete` BEFORE DELETE ON `card` FOR EACH ROW BEGIN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Permission denied', MYSQL_ERRNO = 1001;
END
```

这个方案很容易移植到其他场景中，例如购物，发货，收货等等

26.14. 数据归档

MySQL 提供 ARCHIVE 引擎，ARCHIVE归档的数据不能够修改，这个引擎只提供插入操作

```
CREATE TABLE `logging` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
        `tag` ENUM('unknow','www','user','admin') NOT NULL DEFAULT 'unknow' COMMENT '日志标签',
        `time` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '产生时间',
        `facility` ENUM('card','payment','sms','blockchain') NOT NULL COMMENT '类别',
        `priority` ENUM('info','warning','error','critical','exception','debug') NOT NULL COMMENT
        `message` VARCHAR(1024) NOT NULL COMMENT '内容',
        PRIMARY KEY (`id`)
)
COMMENT='日志表'
COLLATE='utf8_general_ci'
ENGINE=ARCHIVE
AUTO_INCREMENT=1;
```

27. MySQL 大数据操作注意事项

http://netkiller.github.io/journal/mysql.parallel.html

27.1. 关于 delete

delete from mytable 必死无疑，你需要分批删除，尽量缩小每个批次删除的记录数，delete 是可以并行执行的，你可以同时运行多个删除操作

```
mysql> show processlist;
+--------+-----------------+---------------------+-----------+---------+-------+-----------------
| Id     | User            | Host                | db        | Command | Time  | State
+--------+-----------------+---------------------+-----------+---------+-------+-----------------
|      1 | event_scheduler | localhost           | NULL      | Daemon  |    52 | Waiting for next
| 115986 | dba             | localhost           | example   | Query   |     0 | NULL
| 117446 | dba             | localhost           | example   | Query   |    20 | updating
| 117525 | dba             | localhost           | example   | Query   |     2 | updating
| 117526 | dba             | localhost           | example   | Query   |    49 | updating
| 117527 | dba             | localhost           | example   | Query   |     6 | updating
| 117528 | dba             | localhost           | example   | Query   |    64 | updating
| 117546 | dba             | localhost           | example   | Query   |    33 | updating
+--------+-----------------+---------------------+-----------+---------+-------+-----------------
23 rows in set (0.00 sec)
```

27.2. 关于 update

在电商领域常常遇到一个问题“调价”，经常需要调整一批商品的价格,程序猿一条语句搞定有没有?

```
update goods set price=price+10 where category_id = xxx
```

在开发，测试环境是可以通过测试的，一旦部署到生产环境，必死无疑

27.3. 关于创建索引

大表创建索引需要很久的时间，通常要经历 manage keys 与 copy to tmp table 的过程

```
mysql> show processlist;
+--------+-----------------+---------------------+----------+---------+-------+------------------
| Id     | User            | Host                | db       | Command | Time  | State
+--------+-----------------+---------------------+----------+---------+-------+------------------
|      1 | event_scheduler | localhost           | NULL     | Daemon  |    47 | Waiting for next
| 115986 | dba             | localhost           | example  | Query   |     0 | NULL
| 118814 | dba             | 192.168.6.20:50459  | example  | Query   |     8 | copy to tmp table
+--------+-----------------+---------------------+----------+---------+-------+------------------
17 rows in set (0.00 sec)
```

删除索引，也需要经理 copy to tmp table 过程，漫长的等待

```
mysql> show processlist;
+--------+-----------------+---------------------+--------------+---------+-------+--------------
| Id     | User            | Host                | db           | Command | Time  | State
+--------+-----------------+---------------------+--------------+---------+-------+--------------
|      1 | event_scheduler | localhost           | NULL         | Daemon  |    11 | Waiting for n
| 115986 | dba             | localhost           | example      | Query   |     0 | NULL
| 118814 | dba             | 192.168.6.20:50459  | example      | Query   |     4 | copy to tmp t
+--------+-----------------+---------------------+--------------+---------+-------+--------------
17 rows in set (0.00 sec)
```

所以数据设计要深思熟虑，做到提前未雨绸缪，不要亡羊补牢

27.4. 关于 OPTIMIZE

OPTIMIZE 的操作是将当前表复制到临时表操作后再删除当前表，最后将临时表改名

```
mysql> show processlist;
+--------+-----------------+---------------------+---------------------------+---------+-------+-
| Id     | User            | Host                | db                        | Command | Time  |
+--------+-----------------+---------------------+---------------------------+---------+-------+-
|      1 | event_scheduler | localhost           | NULL                      | Daemon  |    14 |
| 115835 | dba             | 192.168.6.20:49664  | example                   | Query   |     9 |
| 115986 | dba             | localhost           | example                   | Query   |     0 |
+--------+-----------------+---------------------+---------------------------+---------+-------+-
17 rows in set (0.00 sec)
```

27.5. 关于切换引擎

转换ENGINE从MyISAM到InnoDB会经历creating table然后copy to tmp table在修改表名几个阶段，过程非常缓慢

```
mysql> show processlist;
+------+-----------------+---------------------+---------+---------+-------+---------------------
| Id   | User            | Host                | db      | Command | Time  | State
+------+-----------------+---------------------+---------+---------+-------+---------------------
|    1 | event_scheduler | localhost           | NULL    | Daemon  |    10 | Waiting for next act
| 3167 | dba             | 192.168.6.20:56723  | example | Query   |     2 | creating table
| 3172 | dba             | localhost           | example | Query   |     0 | NULL
+------+-----------------+---------------------+---------+---------+-------+---------------------
18 rows in set (0.00 sec)
```

copy to tmp table 过程

```
mysql> show processlist;
+------+-----------------+---------------------+---------+---------+-------+---------------------
| Id   | User            | Host                | db      | Command | Time  | State
+------+-----------------+---------------------+---------+---------+-------+---------------------
|    1 | event_scheduler | localhost           | NULL    | Daemon  |    21 | Waiting for next act
| 3167 | dba             | 192.168.6.20:56723  | example | Query   |    13 | copy to tmp table
| 3172 | dba             | localhost           | example | Query   |     0 | NULL
+------+-----------------+---------------------+---------+---------+-------+---------------------
18 rows in set (0.00 sec)
```

此时我们查看mysql data目录会看到临时表文件

```
# ll /var/lib/mysql/hx9999_real_history/

-rw-rw---- 1 mysql mysql      9522 May 16 17:17 #sql-c2f_c5f.frm
-rw-rw---- 1 mysql mysql        48 May 16 17:17 #sql-c2f_c5f.par
-rw-rw---- 1 mysql mysql 637534208 May 16 17:29 #sql-c2f_c5f#P#p0.ibd
-rw-rw---- 1 mysql mysql    180224 May 16 17:17 #sql-c2f_c5f#P#p1.ibd
-rw-rw---- 1 mysql mysql    180224 May 16 17:17 #sql-c2f_c5f#P#p2.ibd
-rw-rw---- 1 mysql mysql    180224 May 16 17:17 #sql-c2f_c5f#P#p3.ibd
-rw-rw---- 1 mysql mysql    180224 May 16 17:17 #sql-c2f_c5f#P#p4.ibd
-rw-rw---- 1 mysql mysql    180224 May 16 17:17 #sql-c2f_c5f#P#p5.ibd
-rw-rw---- 1 mysql mysql    180224 May 16 17:17 #sql-c2f_c5f#P#p6.ibd
-rw-rw---- 1 mysql mysql    180224 May 16 17:17 #sql-c2f_c5f#P#p7.ibd
```

27.6. 确保SELECT不被受阻

使用各种手段保证select操作不被受阻，只要select一直可以查询网站前端就能提供80%的功能，一旦select受阻一切都是浮云。

保证 select 操作优先于其他操作

```
UPDATE [LOW_PRIORITY] [IGNORE] tbl_name
SET col_name1=expr1 [, col_name2=expr2 ...]
[WHERE where_definition]
[ORDER BY ...]
[LIMIT row_count]
```

update的时候增加 LOW_PRIORITY 参数，可以降低更新语句的优先级。

my.cnf

```
[mysqld]
low_priority_updates=1
```

或者启动是添加--low-priority-updates参数

全局开启

```
SET @@global.low_priority_updates = 1;
```

适用于本次会话连接

```
SET @@session.low_priority_updates = 1;
```

第 21 章 参考例子

1. CMS 数据库设计

```
-- --------------------------------------------------------
-- 主机:                           192.168.6.1
-- 服务器版本:                       5.6.26-log - MySQL Community Server (GPL)
-- 服务器操作系统:                     Linux
-- HeidiSQL 版本:                  9.3.0.4998
-- --------------------------------------------------------

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET NAMES utf8mb4 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;

-- 导出 inf 的数据库结构
CREATE DATABASE IF NOT EXISTS `inf` /*!40100 DEFAULT CHARACTER SET utf8 */;
USE `inf`;


-- 导出  表 inf.album 结构
CREATE TABLE IF NOT EXISTS `album` (
  `id` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(50) NOT NULL,
  `folder` varchar(8) NOT NULL,
  `description` varchar(255) NOT NULL,
  `ctime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `mtime` timestamp NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`),
  UNIQUE KEY `folder` (`folder`),
  UNIQUE KEY `name` (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- 数据导出被取消选择。


-- 导出  表 inf.article 结构
CREATE TABLE IF NOT EXISTS `article` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT COMMENT '唯一值',
  `division_id` mediumint(8) unsigned NOT NULL COMMENT '所属事业部',
  `category_id` mediumint(8) unsigned DEFAULT NULL COMMENT '分类',
  `division_category_id` mediumint(8) unsigned NOT NULL COMMENT '事业部分类',
  `title` varchar(255) NOT NULL COMMENT '页面标题',
  `content` text NOT NULL COMMENT '内容',
  `author` varchar(50) DEFAULT NULL COMMENT '作者',
  `keyword` varchar(255) DEFAULT NULL COMMENT '关键字SEO',
  `description` varchar(255) DEFAULT NULL COMMENT '描述SEO',
  `image` varchar(100) DEFAULT NULL COMMENT '图片路径',
  `language` enum('cn','tw','en') NOT NULL DEFAULT 'cn' COMMENT '语言',
  `source` varchar(50) DEFAULT NULL COMMENT '来源',
  `share` enum('Y','N') NOT NULL DEFAULT 'N' COMMENT '分享',
  `attribute` mediumtext COMMENT '扩展属性',
  `visibility` enum('Visible','Hidden') NOT NULL DEFAULT 'Hidden' COMMENT '可见性',
  `status` enum('Enabled','Disabled','Deleted') NOT NULL DEFAULT 'Disabled' COMMENT '状态',
  `ctime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
  `mtime` timestamp NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP COMMENT '编辑时间',
  PRIMARY KEY (`id`),
  KEY `FK_article_category` (`category_id`),
  KEY `ctime` (`ctime`),
```

```
  KEY `division_category_id` (`division_category_id`),
  KEY `division_id` (`division_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='内容'
/*!50100 PARTITION BY KEY (id)
PARTITIONS 16 */;

-- 数据导出被取消选择。


-- 导出  表 inf.category 结构
CREATE TABLE IF NOT EXISTS `category` (
  `id` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
  `division_id` mediumint(8) unsigned NOT NULL COMMENT '分类所属事业部',
  `name` varchar(20) NOT NULL COMMENT '分类名称',
  `description` varchar(255) DEFAULT NULL COMMENT '分类表述',
  `language` enum('en','cn','tw') NOT NULL DEFAULT 'cn',
  `visibility` enum('Visible','Hidden') NOT NULL DEFAULT 'Hidden' COMMENT '可见性',
  `status` enum('Enabled','Disabled') NOT NULL DEFAULT 'Disabled' COMMENT '分类状态',
  `parent_id` mediumint(8) unsigned DEFAULT NULL COMMENT '父节点',
  `path` varchar(255) NOT NULL DEFAULT '/' COMMENT '路径',
  `ctime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
  `mtime` timestamp NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP COMMENT '修改时间',
  PRIMARY KEY (`id`),
  KEY `path` (`path`),
  KEY `FK_category_division` (`division_id`),
  KEY `FK_category_category` (`parent_id`),
  CONSTRAINT `FK_category_category` FOREIGN KEY (`parent_id`) REFERENCES `category` (`id`),
  CONSTRAINT `FK_category_division` FOREIGN KEY (`division_id`) REFERENCES `division` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='分类';

-- 数据导出被取消选择。


-- 导出  表 inf.category_has_template 结构
CREATE TABLE IF NOT EXISTS `category_has_template` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `category_id` mediumint(8) unsigned NOT NULL,
  `template_id` mediumint(8) unsigned NOT NULL,
  `ctime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`),
  UNIQUE KEY `category_id_template_id` (`category_id`,`template_id`),
  KEY `FK_category_has_template_template` (`template_id`),
  CONSTRAINT `FK_category_has_template_category` FOREIGN KEY (`category_id`) REFERENCES `category
  CONSTRAINT `FK_category_has_template_template` FOREIGN KEY (`template_id`) REFERENCES `template
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='分类模板';

-- 数据导出被取消选择。


-- 导出  表 inf.division 结构
CREATE TABLE IF NOT EXISTS `division` (
  `id` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(50) NOT NULL,
  `description` varchar(50) DEFAULT NULL,
  `username` varchar(50) NOT NULL,
  `password` varchar(32) NOT NULL,
  `url` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='事业部表';

-- 数据导出被取消选择。


-- 导出  过程 inf.netkiller 结构
DELIMITER //
//
DELIMITER ;
```

```
-- 导出  表 inf.netkiller_news 结构
CREATE TABLE IF NOT EXISTS `netkiller_news` (
  `no` int(11) NOT NULL AUTO_INCREMENT,
  `title` varchar(200) DEFAULT NULL,
  `publish` date DEFAULT NULL,
  `description` longtext,
  `language` char(2) DEFAULT NULL,
  `kind` char(2) DEFAULT NULL,
  `display` char(1) DEFAULT NULL,
  `updatetime` datetime DEFAULT NULL,
  `mis` varchar(20) DEFAULT NULL,
  `image_b1` varchar(50) DEFAULT NULL,
  `image_s1` varchar(50) DEFAULT NULL,
  `image_b2` varchar(50) DEFAULT NULL,
  `image_s2` varchar(50) DEFAULT NULL,
  `area` varchar(2) DEFAULT NULL,
  `image_b3` varchar(50) DEFAULT NULL,
  `image_s3` varchar(50) DEFAULT NULL,
  `image_b4` varchar(50) DEFAULT NULL,
  `image_s4` varchar(50) DEFAULT NULL,
  `expertsId` int(11) DEFAULT NULL,
  `endDate` date DEFAULT NULL,
  `category` char(1) DEFAULT NULL COMMENT '0代表全部1代表外汇2代表贵金属',
  `pair_id` varchar(40) DEFAULT NULL COMMENT '配对号',
  `curr_data` varchar(200) DEFAULT NULL COMMENT '以字符串的形式保存产品名称、目标、止损、买或者卖，和建议买
  `is_index_dis` char(1) DEFAULT NULL COMMENT '0代表显示 1代表不显示',
  `account` varchar(20) DEFAULT NULL,
  `notice_category` varchar(100) DEFAULT NULL,
  `title2` varchar(1000) DEFAULT NULL,
  `SEO_TITLE` varchar(400) DEFAULT NULL,
  `SEO_KEYWORDS` varchar(400) DEFAULT NULL,
  `SEO_DESCRIPTION` varchar(800) DEFAULT NULL,
  `publish2` date DEFAULT NULL,
  `author` varchar(20) DEFAULT NULL,
  `sort` int(11) DEFAULT NULL,
  `urlstatus` char(1) DEFAULT NULL,
  `url` varchar(200) DEFAULT NULL,
  `knowledge_type` varchar(20) DEFAULT NULL,
  `video` varchar(300) DEFAULT NULL COMMENT '视频',
  `audio` varchar(300) DEFAULT NULL COMMENT '音频',
  `video_image` varchar(300) DEFAULT NULL COMMENT '视频图片',
  `equipment` varchar(20) DEFAULT NULL,
  `praise` int(11) DEFAULT NULL COMMENT '赞同(点赞)',
  `not_praise` int(11) DEFAULT NULL COMMENT '不赞同(点赞)',
  `currency_type` varchar(20) DEFAULT NULL COMMENT '货币类型',
  `publish_mobile` datetime DEFAULT NULL,
  `audio_time` varchar(10) DEFAULT NULL,
  `source` char(1) DEFAULT NULL,
  PRIMARY KEY (`no`)
) ENGINE=FEDERATED DEFAULT CHARSET=utf8 CONNECTION='mysql://netkiller:netkiller@192.168.4.1:3306/

-- 数据导出被取消选择。


-- 导出  表 inf.netkiller_real_news 结构
CREATE TABLE IF NOT EXISTS `netkiller_real_news` (
  `no` int(11) NOT NULL AUTO_INCREMENT,
  `newsid` varchar(50) DEFAULT NULL COMMENT '新闻ID',
  `newstime` datetime DEFAULT NULL,
  `jointime` datetime DEFAULT NULL,
  `language` varchar(2) DEFAULT NULL,
  `name` varchar(200) DEFAULT NULL,
  `content` longtext,
  `type` int(11) DEFAULT NULL COMMENT '用来区分读取各个不同的xml文件',
  `SEO_TITLE` varchar(200) DEFAULT NULL,
  `SEO_KEYWORDS` varchar(200) DEFAULT NULL,
```

```
  `SEO_DESCRIPTION` varchar(500) DEFAULT NULL,
  PRIMARY KEY (`no`)
) ENGINE=FEDERATED DEFAULT CHARSET=utf8 CONNECTION='mysql://netkiller:netkiller@192.168.4.1:3306/

-- 数据导出被取消选择。


-- 导出  表 inf.netkiller_video 结构
CREATE TABLE IF NOT EXISTS `netkiller_video` (
  `no` int(11) NOT NULL AUTO_INCREMENT,
  `video` varchar(300) DEFAULT NULL,
  `smallimage` varchar(100) DEFAULT NULL,
  `largeimage` varchar(100) DEFAULT NULL,
  `display` char(1) DEFAULT NULL,
  `language` char(2) DEFAULT NULL,
  `updatetime` datetime DEFAULT NULL,
  `mis` varchar(20) DEFAULT NULL,
  `sort` int(11) DEFAULT NULL,
  `title` varchar(200) DEFAULT NULL,
  `description` longtext,
  `kind` char(2) DEFAULT NULL,
  `publish` date DEFAULT NULL,
  `source` char(1) DEFAULT NULL,
  `equipment` varchar(20) DEFAULT NULL,
  `expertsId` int(11) DEFAULT NULL,
  `author` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`no`)
) ENGINE=FEDERATED DEFAULT CHARSET=utf8 CONNECTION='mysql://netkiller:netkiller@192.168.4.1:3306/

-- 数据导出被取消选择。


-- 导出  表 inf.images 结构
CREATE TABLE IF NOT EXISTS `images` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `article_id` bigint(20) unsigned NOT NULL,
  `url` varchar(255) NOT NULL,
  `ctime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- 数据导出被取消选择。


-- 导出  表 inf.statistical 结构
CREATE TABLE IF NOT EXISTS `statistical` (
  `id` bigint(20) unsigned DEFAULT NULL,
  `click` bigint(20) unsigned DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='统计表';

-- 数据导出被取消选择。


-- 导出  表 inf.synchronous 结构
CREATE TABLE IF NOT EXISTS `synchronous` (
  `division_id` mediumint(8) unsigned NOT NULL COMMENT '事业部',
  `category_id` mediumint(8) unsigned NOT NULL COMMENT '分类',
  `type` varchar(8) NOT NULL COMMENT '事业部所属类型',
  `table` enum('news','real_news','video','info','t_hotpoint','goldnews','t_review') NOT NULL COM
  `lang` enum('en','cn','tw') NOT NULL DEFAULT 'cn',
  `position` bigint(20) unsigned NOT NULL DEFAULT '1' COMMENT '位置',
  `ctime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `mtime` timestamp NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`category_id`),
  UNIQUE KEY `category_id_type` (`category_id`,`type`),
  KEY `FK_synchronous_division` (`division_id`),
  CONSTRAINT `FK_synchronous_category` FOREIGN KEY (`category_id`) REFERENCES `category` (`id`),
  CONSTRAINT `FK_synchronous_division` FOREIGN KEY (`division_id`) REFERENCES `division` (`id`)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='数据同步设置';

-- 数据导出被取消选择。


-- 导出  表 inf.template 结构
CREATE TABLE IF NOT EXISTS `template` (
  `id` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
  `division_id` mediumint(8) unsigned NOT NULL COMMENT '模板所属分类',
  `name` varchar(50) NOT NULL COMMENT '模板名字',
  `decription` varchar(255) DEFAULT NULL COMMENT '简短描述',
  `content` text NOT NULL COMMENT '模板内容',
  `type` enum('Category','List','Detail','Video') NOT NULL DEFAULT 'Category' COMMENT '模板类型',
  `status` enum('Enabled','Disabled') NOT NULL DEFAULT 'Disabled' COMMENT '模板状态',
  `engine` enum('PHP','Smarty','Volt') NOT NULL DEFAULT 'PHP' COMMENT '模板引擎',
  `ctime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间 ',
  `mtime` timestamp NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP COMMENT '修改时间',
  PRIMARY KEY (`id`),
  KEY `FK_template_division` (`division_id`),
  CONSTRAINT `FK_template_division` FOREIGN KEY (`division_id`) REFERENCES `division` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='模板';

-- 数据导出被取消选择。


-- 导出  表 inf.template_history 结构
CREATE TABLE IF NOT EXISTS `template_history` (
  `id` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
  `template_id` mediumint(8) unsigned NOT NULL,
  `division_id` mediumint(8) unsigned NOT NULL COMMENT '模板所属分类',
  `name` varchar(50) NOT NULL COMMENT '模板名字',
  `decription` varchar(255) DEFAULT NULL COMMENT '简短描述',
  `content` text NOT NULL COMMENT '模板内容',
  `type` enum('Category','List','Detail','Video') NOT NULL DEFAULT 'Category' COMMENT '模板类型',
  `status` enum('Enabled','Disabled') NOT NULL DEFAULT 'Disabled' COMMENT '模板状态',
  `engine` enum('PHP','Smarty','Volt') NOT NULL DEFAULT 'PHP' COMMENT '模板引擎',
  `ctime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间 ',
  `mtime` timestamp NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP COMMENT '修改时间',
  PRIMARY KEY (`id`),
  KEY `FK_template_division` (`division_id`),
  KEY `FK_template_history_template` (`template_id`),
  CONSTRAINT `FK_template_history_division` FOREIGN KEY (`division_id`) REFERENCES `division` (`i
  CONSTRAINT `FK_template_history_template` FOREIGN KEY (`template_id`) REFERENCES `template` (`i
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='模板';

-- 数据导出被取消选择。


-- 导出  表 inf.video 结构
CREATE TABLE IF NOT EXISTS `video` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `division_id` mediumint(8) unsigned NOT NULL COMMENT '所属事业部',
  `category_id` mediumint(8) unsigned NOT NULL COMMENT '隶属分类',
  `title` varchar(255) NOT NULL COMMENT '标题',
  `description` varchar(1024) DEFAULT NULL COMMENT '描述',
  `thumbnail` varchar(255) DEFAULT NULL COMMENT '缩图',
  `image` varchar(255) DEFAULT NULL COMMENT '图片',
  `video` varchar(255) NOT NULL COMMENT '视频',
  `author` varchar(32) DEFAULT NULL COMMENT '作者',
  `language` enum('cn','tw','en') NOT NULL DEFAULT 'cn' COMMENT '语言',
  `player` enum('youku','JW Player') NOT NULL,
  `visibility` enum('Visible','Hidden') NOT NULL DEFAULT 'Hidden' COMMENT '可见否',
  `ctime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
```

```
  `mtime` timestamp NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP COMMENT '修改时间',
  PRIMARY KEY (`id`),
  KEY `FK_videos_division` (`division_id`),
  KEY `FK_videos_category` (`category_id`),
  CONSTRAINT `FK_videos_category` FOREIGN KEY (`category_id`) REFERENCES `category` (`id`),
  CONSTRAINT `FK_videos_division` FOREIGN KEY (`division_id`) REFERENCES `division` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='视频';

-- 数据导出被取消选择。


-- 导出  触发器 inf.category_before_insert 结构
SET @OLDTMP_SQL_MODE=@@SQL_MODE, SQL_MODE='STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION';
DELIMITER //
CREATE TRIGGER `category_before_insert` BEFORE UPDATE ON `category` FOR EACH ROW BEGIN
        IF old.parent_id IS NULL THEN
                -- new.parent_id IS NOT NULL
                set new.parent_id = NULL;
        END IF;
        IF new.id = new.parent_id THEN
                set new.parent_id = old.parent_id;
        END IF;
END//
DELIMITER ;
SET SQL_MODE=@OLDTMP_SQL_MODE;


-- 导出  触发器 inf.template_before_update 结构
SET @OLDTMP_SQL_MODE=@@SQL_MODE, SQL_MODE='STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION';
DELIMITER //
CREATE TRIGGER `template_before_update` BEFORE UPDATE ON `template` FOR EACH ROW BEGIN
        INSERT INTO template_history(   `template_id`,  `division_id`,  `name`,  `decription`,  `
        VALUES (old.id, old.division_id, old.name, old.decription, old.content, old.type, old.sta
END//
DELIMITER ;
SET SQL_MODE=@OLDTMP_SQL_MODE;
/*!40101 SET SQL_MODE=IFNULL(@OLD_SQL_MODE, '') */;
/*!40014 SET FOREIGN_KEY_CHECKS=IF(@OLD_FOREIGN_KEY_CHECKS IS NULL, 1, @OLD_FOREIGN_KEY_CHECKS) *
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
```

2. 数据属性例子

常用属性参考

2.1. 布尔状态

布尔状态

```
Enabled, Disabled, Yes, No
```

```
Rejected, Approved
Allowed, Denied
```

```
Success,Failure
```

```
'Succeed','Failed'
```

2.2. 流状态

```
Canceled, Canceled Reversal
Complete
Expired, Extension
Chargeback, Refunded, Reversed
Shipped, Voided
```

状态流

```
New -> Pending -> Processing -> Processed
```

使用案例

```
'New' -> 'Unassigned' -> 'Assigned' -> 'Pending' -> 'Invalid'
```

混合使用案例

```
'New' -> 'Processing' -> 'Canceled' / 'Completed' / 'Failed' / Deleted
```

```
'New','Processing','Cancelling','Canceled','Completed','Failed','Hidden'
```

2.3. 商品属性

这些属性很常见，你可以将它插入到你的数据库中，方便日后使用

2.3.1. 鞋

鞋

20
20.5
21
21.5
22
22.5
23
24
25
25.5
26
26.5
27
27.5
28
28.5
29
30
31
32
33
33.5
34
34.5
35
35.5
36
36.5
36 2/3
37
37 1/3
37.5
38
38
38.5
38 2/3
39
39 1/3
39.5
40
40.5
40 2/3
41
41 1/3
41.5
42
42.5
42 2/3
43
43 1/3
43.5
44
44.5
44 2/3
45
45 1/3
45.5
46
46.5
46 2/3
47
47 1/3
47.5
48
48.5
49
49.5
50
51

52

2.3.2. 裤子

26
26.5
27
27.5
28
28.5
29
30
31
32
33
33.5
34
34.5
35
35.5
36

2.3.3. 服装

42
44
46
48
50

其他服装

F
XF
XXS
XS
S
M
L
XL
XXL
XXXL
XXXXL
XXXXXL

2.3.4. 内衣

70A
70B
70C
70D
72A
72S
74A
74S
75A
75B
75C
75D

75E
76A
76S
78A
78S
80A
80B
80C
80D
80E
80S
82A
82S
83S
84A
84B
84S
85A
85B
85C
85D
85E
86S
88A
88B
88S
90B
90C
90D
90S
92S
94S
96S
98S

文胸

A罩
B罩
C罩
D罩

2.3.5. 隐形眼镜

100
125
150
175
200
225
250
275
300
325
350
375
400
425
450
475
500
550
600
650

700
750
800
850
900
950
1000

2.3.6. 戒指

11
12
13
14
15
16
17
18
19

2.4. 手机号码分配

移动：134，135，136，137，138，139，150，151，152，157，158，159，187，188，147，182

联通：130，131，132，155，156，185，186，140

电信：180，159，133，153 ，189

中国电信发布中国3G号码段，中国联通185，186；中国移动188，187；中国电信189，180共6个号段。目前，3G业务专属的180-

中国移动拥有号码段为：139、138、137、136、135、134、159、158、157（3G）、151、150、188（3G）、187（3G）；13

中国联通拥有号码段为：130、131、132、156（3G）、186（3G）、185（3G）；6个号段

中国电信拥有号码段为：133、153、189（3G）、180（3G）；4个号码段

2.5. 身份证

```
LOAD DATA LOW_PRIORITY LOCAL INFILE 'C:\\Users\\neo\\Desktop\\id.csv' INTO TABLE `identity_card`

CREATE TEMPORARY TABLE tmp1 select * from identity_card where number like '%0000';
update  identity_card id,tmp1 t1 set id.province=t1.zone where left(id.number,2) = left(t1.number
CREATE TEMPORARY TABLE tmp2 select * from identity_card where number like '%00';
update  identity_card id,tmp2 t1 set id.city=t1.zone where left(id.number,4) = left(t1.number,4);
update  identity_card set county=zone;
update  identity_card set county='' where county=city;
update  identity_card set city='' where province=city;

CREATE TABLE `identity_location` (
        `prefix` VARCHAR(32) NOT NULL COMMENT '身份证号码段',
        `province` VARCHAR(50) NULL DEFAULT NULL COMMENT '省份',
        `city` VARCHAR(50) NULL DEFAULT NULL COMMENT '城市',
        `county` VARCHAR(50) NULL DEFAULT NULL COMMENT '县',
```

```
        `status` ENUM('Y','N') NOT NULL DEFAULT 'N' COMMENT '状态',
        `mtime` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT
        PRIMARY KEY (`prefix`)
)
COMMENT='identity card number'
COLLATE='utf8_general_ci'
ENGINE=InnoDB;

insert into identity_location(prefix,province,city,county) select md5(number) as prefix,province,
```

```
110000,北京市
110100,市辖区
110101,东城区
110102,西城区
110105,朝阳区
110106,丰台区
110107,石景山区
110108,海淀区
110109,门头沟区
110111,房山区
110112,通州区
110113,顺义区
110114,昌平区
110115,大兴区
110116,怀柔区
110117,平谷区
110200,县
110228,密云县
110229,延庆县
120000,天津市
120100,市辖区
120101,和平区
120102,河东区
120103,河西区
120104,南开区
120105,河北区
120106,红桥区
120110,东丽区
120111,西青区
120112,津南区
120113,北辰区
120114,武清区
120115,宝坻区
120116,滨海新区
120200,县
120221,宁河县
120223,静海县
120225,蓟县
130000,河北省
130100,石家庄市
130101,市辖区
130102,长安区
130103,桥东区
130104,桥西区
130105,新华区
130107,井陉矿区
130108,裕华区
```

130121,井陉县
130123,正定县
130124,栾城县
130125,行唐县
130126,灵寿县
130127,高邑县
130128,深泽县
130129,赞皇县
130130,无极县
130131,平山县
130132,元氏县
130133,赵县
130181,辛集市
130182,藁城市
130183,晋州市
130184,新乐市
130185,鹿泉市
130200,唐山市
130201,市辖区
130202,路南区
130203,路北区
130204,古冶区
130205,开平区
130207,丰南区
130208,丰润区
130209,曹妃甸区
130223,滦县
130224,滦南县
130225,乐亭县
130227,迁西县
130229,玉田县
130281,遵化市
130283,迁安市
130300,秦皇岛市
130301,市辖区
130302,海港区
130303,山海关区
130304,北戴河区
130321,青龙满族自治县
130322,昌黎县
130323,抚宁县
130324,卢龙县
130400,邯郸市
130401,市辖区
130402,邯山区
130403,丛台区
130404,复兴区
130406,峰峰矿区
130421,邯郸县
130423,临漳县
130424,成安县
130425,大名县
130426,涉县
130427,磁县
130428,肥乡县
130429,永年县
130430,邱县
130431,鸡泽县

130432,广平县
130433,馆陶县
130434,魏县
130435,曲周县
130481,武安市
130500,邢台市
130501,市辖区
130502,桥东区
130503,桥西区
130521,邢台县
130522,临城县
130523,内丘县
130524,柏乡县
130525,隆尧县
130526,任县
130527,南和县
130528,宁晋县
130529,巨鹿县
130530,新河县
130531,广宗县
130532,平乡县
130533,威县
130534,清河县
130535,临西县
130581,南宫市
130582,沙河市
130600,保定市
130601,市辖区
130602,新市区
130603,北市区
130604,南市区
130621,满城县
130622,清苑县
130623,涞水县
130624,阜平县
130625,徐水县
130626,定兴县
130627,唐县
130628,高阳县
130629,容城县
130630,涞源县
130631,望都县
130632,安新县
130633,易县
130634,曲阳县
130635,蠡县
130636,顺平县
130637,博野县
130638,雄县
130681,涿州市
130682,定州市
130683,安国市
130684,高碑店市
130700,张家口市
130701,市辖区
130702,桥东区
130703,桥西区
130705,宣化区

130706,下花园区
130721,宣化县
130722,张北县
130723,康保县
130724,沽源县
130725,尚义县
130726,蔚县
130727,阳原县
130728,怀安县
130729,万全县
130730,怀来县
130731,涿鹿县
130732,赤城县
130733,崇礼县
130800,承德市
130801,市辖区
130802,双桥区
130803,双滦区
130804,鹰手营子矿区
130821,承德县
130822,兴隆县
130823,平泉县
130824,滦平县
130825,隆化县
130826,丰宁满族自治县
130827,宽城满族自治县
130828,围场满族蒙古族自治县
130900,沧州市
130901,市辖区
130902,新华区
130903,运河区
130921,沧县
130922,青县
130923,东光县
130924,海兴县
130925,盐山县
130926,肃宁县
130927,南皮县
130928,吴桥县
130929,献县
130930,孟村回族自治县
130981,泊头市
130982,任丘市
130983,黄骅市
130984,河间市
131000,廊坊市
131001,市辖区
131002,安次区
131003,广阳区
131022,固安县
131023,永清县
131024,香河县
131025,大城县
131026,文安县
131028,大厂回族自治县
131081,霸州市
131082,三河市
131100,衡水市

131101,市辖区
131102,桃城区
131121,枣强县
131122,武邑县
131123,武强县
131124,饶阳县
131125,安平县
131126,故城县
131127,景县
131128,阜城县
131181,冀州市
131182,深州市
140000,山西省
140100,太原市
140101,市辖区
140105,小店区
140106,迎泽区
140107,杏花岭区
140108,尖草坪区
140109,万柏林区
140110,晋源区
140121,清徐县
140122,阳曲县
140123,娄烦县
140181,古交市
140200,大同市
140201,市辖区
140202,城区
140203,矿区
140211,南郊区
140212,新荣区
140221,阳高县
140222,天镇县
140223,广灵县
140224,灵丘县
140225,浑源县
140226,左云县
140227,大同县
140300,阳泉市
140301,市辖区
140302,城区
140303,矿区
140311,郊区
140321,平定县
140322,盂县
140400,长治市
140401,市辖区
140402,城区
140411,郊区
140421,长治县
140423,襄垣县
140424,屯留县
140425,平顺县
140426,黎城县
140427,壶关县
140428,长子县
140429,武乡县
140430,沁县

140431,沁源县
140481,潞城市
140500,晋城市
140501,晋城市市辖区
140502,城区
140521,沁水县
140522,阳城县
140524,陵川县
140525,泽州县
140581,高平市
140600,朔州市
140601,市辖区
140602,朔城区
140603,平鲁区
140621,山阴县
140622,应县
140623,右玉县
140624,怀仁县
140700,晋中市
140701,市辖区
140702,榆次区
140721,榆社县
140722,左权县
140723,和顺县
140724,昔阳县
140725,寿阳县
140726,太谷县
140727,祁县
140728,平遥县
140729,灵石县
140781,介休市
140800,运城市
140801,市辖区
140802,盐湖区
140821,临猗县
140822,万荣县
140823,闻喜县
140824,稷山县
140825,新绛县
140826,绛县
140827,垣曲县
140828,夏县
140829,平陆县
140830,芮城县
140881,永济市
140882,河津市
140900,忻州市
140901,市辖区
140902,忻府区
140921,定襄县
140922,五台县
140923,代县
140924,繁峙县
140925,宁武县
140926,静乐县
140927,神池县
140928,五寨县
140929,岢岚县

140930,河曲县
140931,保德县
140932,偏关县
140981,原平市
141000,临汾市
141001,市辖区
141002,尧都区
141021,曲沃县
141022,翼城县
141023,襄汾县
141024,洪洞县
141025,古县
141026,安泽县
141027,浮山县
141028,吉县
141029,乡宁县
141030,大宁县
141031,隰县
141032,永和县
141033,蒲县
141034,汾西县
141081,侯马市
141082,霍州市
141100,吕梁市
141101,市辖区
141102,离石区
141121,文水县
141122,交城县
141123,兴县
141124,临县
141125,柳林县
141126,石楼县
141127,岚县
141128,方山县
141129,中阳县
141130,交口县
141181,孝义市
141182,汾阳市
150000,内蒙古自治区
150100,呼和浩特市
150101,市辖区
150102,新城区
150103,回民区
150104,玉泉区
150105,赛罕区
150121,土默特左旗
150122,托克托县
150123,和林格尔县
150124,清水河县
150125,武川县
150200,包头市
150201,市辖区
150202,东河区
150203,昆都仑区
150204,青山区
150205,石拐区
150206,白云鄂博矿区
150207,九原区

150221,土默特右旗
150222,固阳县
150223,达尔罕茂明安联合旗
150300,乌海市
150301,市辖区
150302,海勃湾区
150303,海南区
150304,乌达区
150400,赤峰市
150401,市辖区
150402,红山区
150403,元宝山区
150404,松山区
150421,阿鲁科尔沁旗
150422,巴林左旗
150423,巴林右旗
150424,林西县
150425,克什克腾旗
150426,翁牛特旗
150428,喀喇沁旗
150429,宁城县
150430,敖汉旗
150500,通辽市
150501,市辖区
150502,科尔沁区
150521,科尔沁左翼中旗
150522,科尔沁左翼后旗
150523,开鲁县
150524,库伦旗
150525,奈曼旗
150526,扎鲁特旗
150581,霍林郭勒市
150600,鄂尔多斯市
150601,市辖区
150602,东胜区
150621,达拉特旗
150622,准格尔旗
150623,鄂托克前旗
150624,鄂托克旗
150625,杭锦旗
150626,乌审旗
150627,伊金霍洛旗
150700,呼伦贝尔市
150701,市辖区
150702,海拉尔区
150721,阿荣旗
150722,莫力达瓦达斡尔族自治旗
150723,鄂伦春自治旗
150724,鄂温克族自治旗
150725,陈巴尔虎旗
150726,新巴尔虎左旗
150727,新巴尔虎右旗
150781,满洲里市
150782,牙克石市
150783,扎兰屯市
150784,额尔古纳市
150785,根河市
150800,巴彦淖尔市

150801,市辖区
150802,临河区
150821,五原县
150822,磴口县
150823,乌拉特前旗
150824,乌拉特中旗
150825,乌拉特后旗
150826,杭锦后旗
150900,乌兰察布市
150901,市辖区
150902,集宁区
150921,卓资县
150922,化德县
150923,商都县
150924,兴和县
150925,凉城县
150926,察哈尔右翼前旗
150927,察哈尔右翼中旗
150928,察哈尔右翼后旗
150929,四子王旗
150981,丰镇市
152200,兴安盟
152201,乌兰浩特市
152202,阿尔山市
152221,科尔沁右翼前旗
152222,科尔沁右翼中旗
152223,扎赉特旗
152224,突泉县
152500,锡林郭勒盟
152501,二连浩特市
152502,锡林浩特市
152522,阿巴嘎旗
152523,苏尼特左旗
152524,苏尼特右旗
152525,东乌珠穆沁旗
152526,西乌珠穆沁旗
152527,太仆寺旗
152528,镶黄旗
152529,正镶白旗
152530,正蓝旗
152531,多伦县
152900,阿拉善盟
152921,阿拉善左旗
152922,阿拉善右旗
152923,额济纳旗
210000,辽宁省
210100,沈阳市
210101,市辖区
210102,和平区
210103,沈河区
210104,大东区
210105,皇姑区
210106,铁西区
210111,苏家屯区
210112,东陵区
210113,沈北新区
210114,于洪区
210122,辽中县

210123,康平县
210124,法库县
210181,新民市
210200,大连市
210201,市辖区
210202,中山区
210203,西岗区
210204,沙河口区
210211,甘井子区
210212,旅顺口区
210213,金州区
210224,长海县
210281,瓦房店市
210282,普兰店市
210283,庄河市
210300,鞍山市
210301,市辖区
210302,铁东区
210303,铁西区
210304,立山区
210311,千山区
210321,台安县
210323,岫岩满族自治县
210381,海城市
210400,抚顺市
210401,市辖区
210402,新抚区
210403,东洲区
210404,望花区
210411,顺城区
210421,抚顺县
210422,新宾满族自治县
210423,清原满族自治县
210500,本溪市
210501,市辖区
210502,平山区
210503,溪湖区
210504,明山区
210505,南芬区
210521,本溪满族自治县
210522,桓仁满族自治县
210600,丹东市
210601,市辖区
210602,元宝区
210603,振兴区
210604,振安区
210624,宽甸满族自治县
210681,东港市
210682,凤城市
210700,锦州市
210701,市辖区
210702,古塔区
210703,凌河区
210711,太和区
210726,黑山县
210727,义县
210781,凌海市
210782,北镇市

210800,营口市
210801,市辖区
210802,站前区
210803,西市区
210804,鲅鱼圈区
210811,老边区
210881,盖州市
210882,大石桥市
210900,阜新市
210901,市辖区
210902,海州区
210903,新邱区
210904,太平区
210905,清河门区
210911,细河区
210921,阜新蒙古族自治县
210922,彰武县
211000,辽阳市
211001,市辖区
211002,白塔区
211003,文圣区
211004,宏伟区
211005,弓长岭区
211011,太子河区
211021,辽阳县
211081,灯塔市
211100,盘锦市
211101,市辖区
211102,双台子区
211103,兴隆台区
211121,大洼县
211122,盘山县
211200,铁岭市
211201,市辖区
211202,银州区
211204,清河区
211221,铁岭县
211223,西丰县
211224,昌图县
211281,调兵山市
211282,开原市
211300,朝阳市
211301,市辖区
211302,双塔区
211303,龙城区
211321,朝阳县
211322,建平县
211324,喀喇沁左翼蒙古族自治县
211381,北票市
211382,凌源市
211400,葫芦岛市
211401,市辖区
211402,连山区
211403,龙港区
211404,南票区
211421,绥中县
211422,建昌县
211481,兴城市

220000,吉林省
220100,长春市
220101,市辖区
220102,南关区
220103,宽城区
220104,朝阳区
220105,二道区
220106,绿园区
220112,双阳区
220122,农安县
220181,九台市
220182,榆树市
220183,德惠市
220200,吉林市
220201,市辖区
220202,昌邑区
220203,龙潭区
220204,船营区
220211,丰满区
220221,永吉县
220281,蛟河市
220282,桦甸市
220283,舒兰市
220284,磐石市
220300,四平市
220301,市辖区
220302,铁西区
220303,铁东区
220322,梨树县
220323,伊通满族自治县
220381,公主岭市
220382,双辽市
220400,辽源市
220401,市辖区
220402,龙山区
220403,西安区
220421,东丰县
220422,东辽县
220500,通化市
220501,市辖区
220502,东昌区
220503,二道江区
220521,通化县
220523,辉南县
220524,柳河县
220581,梅河口市
220582,集安市
220600,白山市
220601,市辖区
220602,浑江区
220605,江源区
220621,抚松县
220622,靖宇县
220623,长白朝鲜族自治县
220681,临江市
220700,松原市
220701,市辖区
220702,宁江区

220721,前郭尔罗斯蒙古族自治县
220722,长岭县
220723,乾安县
220724,扶余县
220800,白城市
220801,市辖区
220802,洮北区
220821,镇赉县
220822,通榆县
220881,洮南市
220882,大安市
222400,延边朝鲜族自治州
222401,延吉市
222402,图们市
222403,敦化市
222404,珲春市
222405,龙井市
222406,和龙市
222424,汪清县
222426,安图县
230000,黑龙江省
230100,哈尔滨市
230101,市辖区
230102,道里区
230103,南岗区
230104,道外区
230108,平房区
230109,松北区
230110,香坊区
230111,呼兰区
230112,阿城区
230123,依兰县
230124,方正县
230125,宾县
230126,巴彦县
230127,木兰县
230128,通河县
230129,延寿县
230182,双城市
230183,尚志市
230184,五常市
230200,齐齐哈尔市
230201,市辖区
230202,龙沙区
230203,建华区
230204,铁锋区
230205,昂昂溪区
230206,富拉尔基区
230207,碾子山区
230208,梅里斯达斡尔族区
230221,龙江县
230223,依安县
230224,泰来县
230225,甘南县
230227,富裕县
230229,克山县
230230,克东县
230231,拜泉县

230281,讷河市
230300,鸡西市
230301,市辖区
230302,鸡冠区
230303,恒山区
230304,滴道区
230305,梨树区
230306,城子河区
230307,麻山区
230321,鸡东县
230381,虎林市
230382,密山市
230400,鹤岗市
230401,市辖区
230402,向阳区
230403,工农区
230404,南山区
230405,兴安区
230406,东山区
230407,兴山区
230421,萝北县
230422,绥滨县
230500,双鸭山市
230501,市辖区
230502,尖山区
230503,岭东区
230505,四方台区
230506,宝山区
230521,集贤县
230522,友谊县
230523,宝清县
230524,饶河县
230600,大庆市
230601,市辖区
230602,萨尔图区
230603,龙凤区
230604,让胡路区
230605,红岗区
230606,大同区
230621,肇州县
230622,肇源县
230623,林甸县
230624,杜尔伯特蒙古族自治县
230700,伊春市
230701,市辖区
230702,伊春区
230703,南岔区
230704,友好区
230705,西林区
230706,翠峦区
230707,新青区
230708,美溪区
230709,金山屯区
230710,五营区
230711,乌马河区
230712,汤旺河区
230713,带岭区
230714,乌伊岭区

230715,红星区
230716,上甘岭区
230722,嘉荫县
230781,铁力市
230800,佳木斯市
230801,市辖区
230803,向阳区
230804,前进区
230805,东风区
230811,郊区
230822,桦南县
230826,桦川县
230828,汤原县
230833,抚远县
230881,同江市
230882,富锦市
230900,七台河市
230901,市辖区
230902,新兴区
230903,桃山区
230904,茄子河区
230921,勃利县
231000,牡丹江市
231001,市辖区
231002,东安区
231003,阳明区
231004,爱民区
231005,西安区
231024,东宁县
231025,林口县
231081,绥芬河市
231083,海林市
231084,宁安市
231085,穆棱市
231100,黑河市
231101,市辖区
231102,爱辉区
231121,嫩江县
231123,逊克县
231124,孙吴县
231181,北安市
231182,五大连池市
231200,绥化市
231201,市辖区
231202,北林区
231221,望奎县
231222,兰西县
231223,青冈县
231224,庆安县
231225,明水县
231226,绥棱县
231281,安达市
231282,肇东市
231283,海伦市
232700,大兴安岭地区
232721,呼玛县
232722,塔河县
232723,漠河县

310000,上海市
310100,市辖区
310101,黄浦区
310104,徐汇区
310105,长宁区
310106,静安区
310107,普陀区
310108,闸北区
310109,虹口区
310110,杨浦区
310112,闵行区
310113,宝山区
310114,嘉定区
310115,浦东新区
310116,金山区
310117,松江区
310118,青浦区
310120,奉贤区
310200,县
310230,崇明县
320000,江苏省
320100,南京市
320101,市辖区
320102,玄武区
320103,白下区
320104,秦淮区
320105,建邺区
320106,鼓楼区
320107,下关区
320111,浦口区
320113,栖霞区
320114,雨花台区
320115,江宁区
320116,六合区
320124,溧水县
320125,高淳县
320200,无锡市
320201,市辖区
320202,崇安区
320203,南长区
320204,北塘区
320205,锡山区
320206,惠山区
320211,滨湖区
320281,江阴市
320282,宜兴市
320300,徐州市
320301,市辖区
320302,鼓楼区
320303,云龙区
320305,贾汪区
320311,泉山区
320312,铜山区
320321,丰县
320322,沛县
320324,睢宁县
320381,新沂市
320382,邳州市

320400,常州市
320401,市辖区
320402,天宁区
320404,钟楼区
320405,戚墅堰区
320411,新北区
320412,武进区
320481,溧阳市
320482,金坛市
320500,苏州市
320501,市辖区
320505,虎丘区
320506,吴中区
320507,相城区
320508,姑苏区
320509,吴江区
320581,常熟市
320582,张家港市
320583,昆山市
320585,太仓市
320600,南通市
320601,市辖区
320602,崇川区
320611,港闸区
320612,通州区
320621,海安县
320623,如东县
320681,启东市
320682,如皋市
320684,海门市
320700,连云港市
320701,市辖区
320703,连云区
320705,新浦区
320706,海州区
320721,赣榆县
320722,东海县
320723,灌云县
320724,灌南县
320800,淮安市
320801,市辖区
320802,清河区
320803,淮安区
320804,淮阴区
320811,清浦区
320826,涟水县
320829,洪泽县
320830,盱眙县
320831,金湖县
320900,盐城市
320901,市辖区
320902,亭湖区
320903,盐都区
320921,响水县
320922,滨海县
320923,阜宁县
320924,射阳县
320925,建湖县

320981,东台市
320982,大丰市
321000,扬州市
321001,市辖区
321002,广陵区
321003,邗江区
321012,江都区
321023,宝应县
321081,仪征市
321084,高邮市
321100,镇江市
321101,市辖区
321102,京口区
321111,润州区
321112,丹徒区
321181,丹阳市
321182,扬中市
321183,句容市
321200,泰州市
321201,市辖区
321202,海陵区
321203,高港区
321281,兴化市
321282,靖江市
321283,泰兴市
321284,姜堰市
321300,宿迁市
321301,市辖区
321302,宿城区
321311,宿豫区
321322,沭阳县
321323,泗阳县
321324,泗洪县
330000,浙江省
330100,杭州市
330101,市辖区
330102,上城区
330103,下城区
330104,江干区
330105,拱墅区
330106,西湖区
330108,滨江区
330109,萧山区
330110,余杭区
330122,桐庐县
330127,淳安县
330182,建德市
330183,富阳市
330185,临安市
330200,宁波市
330201,市辖区
330203,海曙区
330204,江东区
330205,江北区
330206,北仑区
330211,镇海区
330212,鄞州区
330225,象山县

330226,宁海县
330281,余姚市
330282,慈溪市
330283,奉化市
330300,温州市
330301,市辖区
330302,鹿城区
330303,龙湾区
330304,瓯海区
330322,洞头县
330324,永嘉县
330326,平阳县
330327,苍南县
330328,文成县
330329,泰顺县
330381,瑞安市
330382,乐清市
330400,嘉兴市
330401,市辖区
330402,南湖区
330411,秀洲区
330421,嘉善县
330424,海盐县
330481,海宁市
330482,平湖市
330483,桐乡市
330500,湖州市
330501,市辖区
330502,吴兴区
330503,南浔区
330521,德清县
330522,长兴县
330523,安吉县
330600,绍兴市
330601,市辖区
330602,越城区
330621,绍兴县
330624,新昌县
330681,诸暨市
330682,上虞市
330683,嵊州市
330700,金华市
330701,市辖区
330702,婺城区
330703,金东区
330723,武义县
330726,浦江县
330727,磐安县
330781,兰溪市
330782,义乌市
330783,东阳市
330784,永康市
330800,衢州市
330801,市辖区
330802,柯城区
330803,衢江区
330822,常山县
330824,开化县

330825,龙游县
330881,江山市
330900,舟山市
330901,市辖区
330902,定海区
330903,普陀区
330921,岱山县
330922,嵊泗县
331000,台州市
331001,市辖区
331002,椒江区
331003,黄岩区
331004,路桥区
331021,玉环县
331022,三门县
331023,天台县
331024,仙居县
331081,温岭市
331082,临海市
331100,丽水市
331101,市辖区
331102,莲都区
331121,青田县
331122,缙云县
331123,遂昌县
331124,松阳县
331125,云和县
331126,庆元县
331127,景宁畲族自治县
331181,龙泉市
340000,安徽省
340100,合肥市
340101,市辖区
340102,瑶海区
340103,庐阳区
340104,蜀山区
340111,包河区
340121,长丰县
340122,肥东县
340123,肥西县
340124,庐江县
340181,巢湖市
340200,芜湖市
340201,市辖区
340202,镜湖区
340203,弋江区
340207,鸠江区
340208,三山区
340221,芜湖县
340222,繁昌县
340223,南陵县
340225,无为县
340300,蚌埠市
340301,市辖区
340302,龙子湖区
340303,蚌山区
340304,禹会区
340311,淮上区

340321,怀远县
340322,五河县
340323,固镇县
340400,淮南市
340401,市辖区
340402,大通区
340403,田家庵区
340404,谢家集区
340405,八公山区
340406,潘集区
340421,凤台县
340500,马鞍山市
340501,市辖区
340503,花山区
340504,雨山区
340506,博望区
340521,当涂县
340522,含山县
340523,和县
340600,淮北市
340601,市辖区
340602,杜集区
340603,相山区
340604,烈山区
340621,濉溪县
340700,铜陵市
340701,市辖区
340702,铜官山区
340703,狮子山区
340711,郊区
340721,铜陵县
340800,安庆市
340801,市辖区
340802,迎江区
340803,大观区
340811,宜秀区
340822,怀宁县
340823,枞阳县
340824,潜山县
340825,太湖县
340826,宿松县
340827,望江县
340828,岳西县
340881,桐城市
341000,黄山市
341001,市辖区
341002,屯溪区
341003,黄山区
341004,徽州区
341021,歙县
341022,休宁县
341023,黟县
341024,祁门县
341100,滁州市
341101,市辖区
341102,琅琊区
341103,南谯区
341122,来安县

341124,全椒县
341125,定远县
341126,凤阳县
341181,天长市
341182,明光市
341200,阜阳市
341201,市辖区
341202,颍州区
341203,颍东区
341204,颍泉区
341221,临泉县
341222,太和县
341225,阜南县
341226,颍上县
341282,界首市
341300,宿州市
341301,市辖区
341302,埇桥区
341321,砀山县
341322,萧县
341323,灵璧县
341324,泗县
341500,六安市
341501,市辖区
341502,金安区
341503,裕安区
341521,寿县
341522,霍邱县
341523,舒城县
341524,金寨县
341525,霍山县
341600,亳州市
341601,市辖区
341602,谯城区
341621,涡阳县
341622,蒙城县
341623,利辛县
341700,池州市
341701,市辖区
341702,贵池区
341721,东至县
341722,石台县
341723,青阳县
341800,宣城市
341801,市辖区
341802,宣州区
341821,郎溪县
341822,广德县
341823,泾县
341824,绩溪县
341825,旌德县
341881,宁国市
350000,福建省
350100,福州市
350101,市辖区
350102,鼓楼区
350103,台江区
350104,仓山区

350105,马尾区
350111,晋安区
350121,闽侯县
350122,连江县
350123,罗源县
350124,闽清县
350125,永泰县
350128,平潭县
350181,福清市
350182,长乐市
350200,厦门市
350201,市辖区
350203,思明区
350205,海沧区
350206,湖里区
350211,集美区
350212,同安区
350213,翔安区
350300,莆田市
350301,市辖区
350302,城厢区
350303,涵江区
350304,荔城区
350305,秀屿区
350322,仙游县
350400,三明市
350401,市辖区
350402,梅列区
350403,三元区
350421,明溪县
350423,清流县
350424,宁化县
350425,大田县
350426,尤溪县
350427,沙县
350428,将乐县
350429,泰宁县
350430,建宁县
350481,永安市
350500,泉州市
350501,市辖区
350502,鲤城区
350503,丰泽区
350504,洛江区
350505,泉港区
350521,惠安县
350524,安溪县
350525,永春县
350526,德化县
350527,金门县
350581,石狮市
350582,晋江市
350583,南安市
350600,漳州市
350601,市辖区
350602,芗城区
350603,龙文区
350622,云霄县

350623,漳浦县
350624,诏安县
350625,长泰县
350626,东山县
350627,南靖县
350628,平和县
350629,华安县
350681,龙海市
350700,南平市
350701,市辖区
350702,延平区
350721,顺昌县
350722,浦城县
350723,光泽县
350724,松溪县
350725,政和县
350781,邵武市
350782,武夷山市
350783,建瓯市
350784,建阳市
350800,龙岩市
350801,市辖区
350802,新罗区
350821,长汀县
350822,永定县
350823,上杭县
350824,武平县
350825,连城县
350881,漳平市
350900,宁德市
350901,市辖区
350902,蕉城区
350921,霞浦县
350922,古田县
350923,屏南县
350924,寿宁县
350925,周宁县
350926,柘荣县
350981,福安市
350982,福鼎市
360000,江西省
360100,南昌市
360101,市辖区
360102,东湖区
360103,西湖区
360104,青云谱区
360105,湾里区
360111,青山湖区
360121,南昌县
360122,新建县
360123,安义县
360124,进贤县
360200,景德镇市
360201,市辖区
360202,昌江区
360203,珠山区
360222,浮梁县
360281,乐平市

360300,萍乡市
360301,市辖区
360302,安源区
360313,湘东区
360321,莲花县
360322,上栗县
360323,芦溪县
360400,九江市
360401,市辖区
360402,庐山区
360403,浔阳区
360421,九江县
360423,武宁县
360424,修水县
360425,永修县
360426,德安县
360427,星子县
360428,都昌县
360429,湖口县
360430,彭泽县
360481,瑞昌市
360482,共青城市
360500,新余市
360501,市辖区
360502,渝水区
360521,分宜县
360600,鹰潭市
360601,市辖区
360602,月湖区
360622,余江县
360681,贵溪市
360700,赣州市
360701,市辖区
360702,章贡区
360721,赣县
360722,信丰县
360723,大余县
360724,上犹县
360725,崇义县
360726,安远县
360727,龙南县
360728,定南县
360729,全南县
360730,宁都县
360731,于都县
360732,兴国县
360733,会昌县
360734,寻乌县
360735,石城县
360781,瑞金市
360782,南康市
360800,吉安市
360801,市辖区
360802,吉州区
360803,青原区
360821,吉安县
360822,吉水县
360823,峡江县

360824,新干县
360825,永丰县
360826,泰和县
360827,遂川县
360828,万安县
360829,安福县
360830,永新县
360881,井冈山市
360900,宜春市
360901,市辖区
360902,袁州区
360921,奉新县
360922,万载县
360923,上高县
360924,宜丰县
360925,靖安县
360926,铜鼓县
360981,丰城市
360982,樟树市
360983,高安市
361000,抚州市
361001,市辖区
361002,临川区
361021,南城县
361022,黎川县
361023,南丰县
361024,崇仁县
361025,乐安县
361026,宜黄县
361027,金溪县
361028,资溪县
361029,东乡县
361030,广昌县
361100,上饶市
361101,市辖区
361102,信州区
361121,上饶县
361122,广丰县
361123,玉山县
361124,铅山县
361125,横峰县
361126,弋阳县
361127,余干县
361128,鄱阳县
361129,万年县
361130,婺源县
361181,德兴市
370000,山东省
370100,济南市
370101,市辖区
370102,历下区
370103,市中区
370104,槐荫区
370105,天桥区
370112,历城区
370113,长清区
370124,平阴县
370125,济阳县

370126,商河县
370181,章丘市
370200,青岛市
370201,市辖区
370202,市南区
370203,市北区
370205,四方区
370211,黄岛区
370212,崂山区
370213,李沧区
370214,城阳区
370281,胶州市
370282,即墨市
370283,平度市
370284,胶南市
370285,莱西市
370300,淄博市
370301,市辖区
370302,淄川区
370303,张店区
370304,博山区
370305,临淄区
370306,周村区
370321,桓台县
370322,高青县
370323,沂源县
370400,枣庄市
370401,市辖区
370402,市中区
370403,薛城区
370404,峄城区
370405,台儿庄区
370406,山亭区
370481,滕州市
370500,东营市
370501,市辖区
370502,东营区
370503,河口区
370521,垦利县
370522,利津县
370523,广饶县
370600,烟台市
370601,市辖区
370602,芝罘区
370611,福山区
370612,牟平区
370613,莱山区
370634,长岛县
370681,龙口市
370682,莱阳市
370683,莱州市
370684,蓬莱市
370685,招远市
370686,栖霞市
370687,海阳市
370700,潍坊市
370701,市辖区
370702,潍城区

370703,寒亭区
370704,坊子区
370705,奎文区
370724,临朐县
370725,昌乐县
370781,青州市
370782,诸城市
370783,寿光市
370784,安丘市
370785,高密市
370786,昌邑市
370800,济宁市
370801,市辖区
370802,市中区
370811,任城区
370826,微山县
370827,鱼台县
370828,金乡县
370829,嘉祥县
370830,汶上县
370831,泗水县
370832,梁山县
370881,曲阜市
370882,兖州市
370883,邹城市
370900,泰安市
370901,市辖区
370902,泰山区
370911,岱岳区
370921,宁阳县
370923,东平县
370982,新泰市
370983,肥城市
371000,威海市
371001,市辖区
371002,环翠区
371081,文登市
371082,荣成市
371083,乳山市
371100,日照市
371101,市辖区
371102,东港区
371103,岚山区
371121,五莲县
371122,莒县
371200,莱芜市
371201,市辖区
371202,莱城区
371203,钢城区
371300,临沂市
371301,市辖区
371302,兰山区
371311,罗庄区
371312,河东区
371321,沂南县
371322,郯城县
371323,沂水县
371324,苍山县

371325,费县
371326,平邑县
371327,莒南县
371328,蒙阴县
371329,临沭县
371400,德州市
371401,市辖区
371402,德城区
371421,陵县
371422,宁津县
371423,庆云县
371424,临邑县
371425,齐河县
371426,平原县
371427,夏津县
371428,武城县
371481,乐陵市
371482,禹城市
371500,聊城市
371501,市辖区
371502,东昌府区
371521,阳谷县
371522,莘县
371523,茌平县
371524,东阿县
371525,冠县
371526,高唐县
371581,临清市
371600,滨州市
371601,市辖区
371602,滨城区
371621,惠民县
371622,阳信县
371623,无棣县
371624,沾化县
371625,博兴县
371626,邹平县
371700,菏泽市
371701,市辖区
371702,牡丹区
371721,曹县
371722,单县
371723,成武县
371724,巨野县
371725,郓城县
371726,鄄城县
371727,定陶县
371728,东明县
410000,河南省
410100,郑州市
410101,市辖区
410102,中原区
410103,二七区
410104,管城回族区
410105,金水区
410106,上街区
410108,惠济区
410122,中牟县

410181,巩义市
410182,荥阳市
410183,新密市
410184,新郑市
410185,登封市
410200,开封市
410201,市辖区
410202,龙亭区
410203,顺河回族区
410204,鼓楼区
410205,禹王台区
410211,金明区
410221,杞县
410222,通许县
410223,尉氏县
410224,开封县
410225,兰考县
410300,洛阳市
410301,市辖区
410302,老城区
410303,西工区
410304,瀍河回族区
410305,涧西区
410306,吉利区
410311,洛龙区
410322,孟津县
410323,新安县
410324,栾川县
410325,嵩县
410326,汝阳县
410327,宜阳县
410328,洛宁县
410329,伊川县
410381,偃师市
410400,平顶山市
410401,市辖区
410402,新华区
410403,卫东区
410404,石龙区
410411,湛河区
410421,宝丰县
410422,叶县
410423,鲁山县
410425,郏县
410481,舞钢市
410482,汝州市
410500,安阳市
410501,市辖区
410502,文峰区
410503,北关区
410505,殷都区
410506,龙安区
410522,安阳县
410523,汤阴县
410526,滑县
410527,内黄县
410581,林州市
410600,鹤壁市

410601,市辖区
410602,鹤山区
410603,山城区
410611,淇滨区
410621,浚县
410622,淇县
410700,新乡市
410701,市辖区
410702,红旗区
410703,卫滨区
410704,凤泉区
410711,牧野区
410721,新乡县
410724,获嘉县
410725,原阳县
410726,延津县
410727,封丘县
410728,长垣县
410781,卫辉市
410782,辉县市
410800,焦作市
410801,市辖区
410802,解放区
410803,中站区
410804,马村区
410811,山阳区
410821,修武县
410822,博爱县
410823,武陟县
410825,温县
410882,沁阳市
410883,孟州市
410900,濮阳市
410901,市辖区
410902,华龙区
410922,清丰县
410923,南乐县
410926,范县
410927,台前县
410928,濮阳县
411000,许昌市
411001,市辖区
411002,魏都区
411023,许昌县
411024,鄢陵县
411025,襄城县
411081,禹州市
411082,长葛市
411100,漯河市
411101,市辖区
411102,源汇区
411103,郾城区
411104,召陵区
411121,舞阳县
411122,临颍县
411200,三门峡市
411201,市辖区
411202,湖滨区

411221,渑池县
411222,陕县
411224,卢氏县
411281,义马市
411282,灵宝市
411300,南阳市
411301,市辖区
411302,宛城区
411303,卧龙区
411321,南召县
411322,方城县
411323,西峡县
411324,镇平县
411325,内乡县
411326,淅川县
411327,社旗县
411328,唐河县
411329,新野县
411330,桐柏县
411381,邓州市
411400,商丘市
411401,市辖区
411402,梁园区
411403,睢阳区
411421,民权县
411422,睢县
411423,宁陵县
411424,柘城县
411425,虞城县
411426,夏邑县
411481,永城市
411500,信阳市
411501,市辖区
411502,浉河区
411503,平桥区
411521,罗山县
411522,光山县
411523,新县
411524,商城县
411525,固始县
411526,潢川县
411527,淮滨县
411528,息县
411600,周口市
411601,市辖区
411602,川汇区
411621,扶沟县
411622,西华县
411623,商水县
411624,沈丘县
411625,郸城县
411626,淮阳县
411627,太康县
411628,鹿邑县
411681,项城市
411700,驻马店市
411701,市辖区
411702,驿城区

411721,西平县
411722,上蔡县
411723,平舆县
411724,正阳县
411725,确山县
411726,泌阳县
411727,汝南县
411728,遂平县
411729,新蔡县
419000,省直辖县级行政区划
419001,济源市
420000,湖北省
420100,武汉市
420101,市辖区
420102,江岸区
420103,江汉区
420104,硚口区
420105,汉阳区
420106,武昌区
420107,青山区
420111,洪山区
420112,东西湖区
420113,汉南区
420114,蔡甸区
420115,江夏区
420116,黄陂区
420117,新洲区
420200,黄石市
420201,市辖区
420202,黄石港区
420203,西塞山区
420204,下陆区
420205,铁山区
420222,阳新县
420281,大冶市
420300,十堰市
420301,市辖区
420302,茅箭区
420303,张湾区
420321,郧县
420322,郧西县
420323,竹山县
420324,竹溪县
420325,房县
420381,丹江口市
420500,宜昌市
420501,市辖区
420502,西陵区
420503,伍家岗区
420504,点军区
420505,猇亭区
420506,夷陵区
420525,远安县
420526,兴山县
420527,秭归县
420528,长阳土家族自治县
420529,五峰土家族自治县
420581,宜都市

420582,当阳市
420583,枝江市
420600,襄阳市
420601,市辖区
420602,襄城区
420606,樊城区
420607,襄州区
420624,南漳县
420625,谷城县
420626,保康县
420682,老河口市
420683,枣阳市
420684,宜城市
420700,鄂州市
420701,市辖区
420702,梁子湖区
420703,华容区
420704,鄂城区
420800,荆门市
420801,市辖区
420802,东宝区
420804,掇刀区
420821,京山县
420822,沙洋县
420881,钟祥市
420900,孝感市
420901,市辖区
420902,孝南区
420921,孝昌县
420922,大悟县
420923,云梦县
420981,应城市
420982,安陆市
420984,汉川市
421000,荆州市
421001,市辖区
421002,沙市区
421003,荆州区
421022,公安县
421023,监利县
421024,江陵县
421081,石首市
421083,洪湖市
421087,松滋市
421100,黄冈市
421101,市辖区
421102,黄州区
421121,团风县
421122,红安县
421123,罗田县
421124,英山县
421125,浠水县
421126,蕲春县
421127,黄梅县
421181,麻城市
421182,武穴市
421200,咸宁市
421201,市辖区

421202,咸安区
421221,嘉鱼县
421222,通城县
421223,崇阳县
421224,通山县
421281,赤壁市
421300,随州市
421301,市辖区
421303,曾都区
421321,随县
421381,广水市
422800,恩施土家族苗族自治州
422801,恩施市
422802,利川市
422822,建始县
422823,巴东县
422825,宣恩县
422826,咸丰县
422827,来凤县
422828,鹤峰县
429000,省直辖县级行政区划
429004,仙桃市
429005,潜江市
429006,天门市
429021,神农架林区
430000,湖南省
430100,长沙市
430101,市辖区
430102,芙蓉区
430103,天心区
430104,岳麓区
430105,开福区
430111,雨花区
430112,望城区
430121,长沙县
430124,宁乡县
430181,浏阳市
430200,株洲市
430201,市辖区
430202,荷塘区
430203,芦淞区
430204,石峰区
430211,天元区
430221,株洲县
430223,攸县
430224,茶陵县
430225,炎陵县
430281,醴陵市
430300,湘潭市
430301,市辖区
430302,雨湖区
430304,岳塘区
430321,湘潭县
430381,湘乡市
430382,韶山市
430400,衡阳市
430401,市辖区
430405,珠晖区

430406,雁峰区
430407,石鼓区
430408,蒸湘区
430412,南岳区
430421,衡阳县
430422,衡南县
430423,衡山县
430424,衡东县
430426,祁东县
430481,耒阳市
430482,常宁市
430500,邵阳市
430501,市辖区
430502,双清区
430503,大祥区
430511,北塔区
430521,邵东县
430522,新邵县
430523,邵阳县
430524,隆回县
430525,洞口县
430527,绥宁县
430528,新宁县
430529,城步苗族自治县
430581,武冈市
430600,岳阳市
430601,市辖区
430602,岳阳楼区
430603,云溪区
430611,君山区
430621,岳阳县
430623,华容县
430624,湘阴县
430626,平江县
430681,汨罗市
430682,临湘市
430700,常德市
430701,市辖区
430702,武陵区
430703,鼎城区
430721,安乡县
430722,汉寿县
430723,澧县
430724,临澧县
430725,桃源县
430726,石门县
430781,津市市
430800,张家界市
430801,市辖区
430802,永定区
430811,武陵源区
430821,慈利县
430822,桑植县
430900,益阳市
430901,市辖区
430902,资阳区
430903,赫山区
430921,南县

430922,桃江县
430923,安化县
430981,沅江市
431000,郴州市
431001,市辖区
431002,北湖区
431003,苏仙区
431021,桂阳县
431022,宜章县
431023,永兴县
431024,嘉禾县
431025,临武县
431026,汝城县
431027,桂东县
431028,安仁县
431081,资兴市
431100,永州市
431101,市辖区
431102,零陵区
431103,冷水滩区
431121,祁阳县
431122,东安县
431123,双牌县
431124,道县
431125,江永县
431126,宁远县
431127,蓝山县
431128,新田县
431129,江华瑶族自治县
431200,怀化市
431201,市辖区
431202,鹤城区
431221,中方县
431222,沅陵县
431223,辰溪县
431224,溆浦县
431225,会同县
431226,麻阳苗族自治县
431227,新晃侗族自治县
431228,芷江侗族自治县
431229,靖州苗族侗族自治县
431230,通道侗族自治县
431281,洪江市
431300,娄底市
431301,市辖区
431302,娄星区
431321,双峰县
431322,新化县
431381,冷水江市
431382,涟源市
433100,湘西土家族苗族自治州
433101,吉首市
433122,泸溪县
433123,凤凰县
433124,花垣县
433125,保靖县
433126,古丈县
433127,永顺县

433130,龙山县
440000,广东省
440100,广州市
440101,市辖区
440103,荔湾区
440104,越秀区
440105,海珠区
440106,天河区
440111,白云区
440112,黄埔区
440113,番禺区
440114,花都区
440115,南沙区
440116,萝岗区
440183,增城市
440184,从化市
440200,韶关市
440201,市辖区
440203,武江区
440204,浈江区
440205,曲江区
440222,始兴县
440224,仁化县
440229,翁源县
440232,乳源瑶族自治县
440233,新丰县
440281,乐昌市
440282,南雄市
440300,深圳市
440301,市辖区
440303,罗湖区
440304,福田区
440305,南山区
440306,宝安区
440307,龙岗区
440308,盐田区
440400,珠海市
440401,市辖区
440402,香洲区
440403,斗门区
440404,金湾区
440500,汕头市
440501,市辖区
440507,龙湖区
440511,金平区
440512,濠江区
440513,潮阳区
440514,潮南区
440515,澄海区
440523,南澳县
440600,佛山市
440601,市辖区
440604,禅城区
440605,南海区
440606,顺德区
440607,三水区
440608,高明区
440700,江门市

440701,市辖区
440703,蓬江区
440704,江海区
440705,新会区
440781,台山市
440783,开平市
440784,鹤山市
440785,恩平市
440800,湛江市
440801,市辖区
440802,赤坎区
440803,霞山区
440804,坡头区
440811,麻章区
440823,遂溪县
440825,徐闻县
440881,廉江市
440882,雷州市
440883,吴川市
440900,茂名市
440901,市辖区
440902,茂南区
440903,茂港区
440923,电白县
440981,高州市
440982,化州市
440983,信宜市
441200,肇庆市
441201,市辖区
441202,端州区
441203,鼎湖区
441223,广宁县
441224,怀集县
441225,封开县
441226,德庆县
441283,高要市
441284,四会市
441300,惠州市
441301,市辖区
441302,惠城区
441303,惠阳区
441322,博罗县
441323,惠东县
441324,龙门县
441400,梅州市
441401,市辖区
441402,梅江区
441421,梅县
441422,大埔县
441423,丰顺县
441424,五华县
441426,平远县
441427,蕉岭县
441481,兴宁市
441500,汕尾市
441501,市辖区
441502,城区
441521,海丰县

441523,陆河县
441581,陆丰市
441600,河源市
441601,市辖区
441602,源城区
441621,紫金县
441622,龙川县
441623,连平县
441624,和平县
441625,东源县
441700,阳江市
441701,市辖区
441702,江城区
441721,阳西县
441723,阳东县
441781,阳春市
441800,清远市
441801,市辖区
441802,清城区
441821,佛冈县
441823,阳山县
441825,连山壮族瑶族自治县
441826,连南瑶族自治县
441827,清新县
441881,英德市
441882,连州市
441900,东莞市
442000,中山市
445100,潮州市
445101,市辖区
445102,湘桥区
445121,潮安县
445122,饶平县
445200,揭阳市
445201,市辖区
445202,榕城区
445221,揭东县
445222,揭西县
445224,惠来县
445281,普宁市
445300,云浮市
445301,市辖区
445302,云城区
445321,新兴县
445322,郁南县
445323,云安县
445381,罗定市
450000,广西壮族自治区
450100,南宁市
450101,市辖区
450102,兴宁区
450103,青秀区
450105,江南区
450107,西乡塘区
450108,良庆区
450109,邕宁区
450122,武鸣县
450123,隆安县

450124,马山县
450125,上林县
450126,宾阳县
450127,横县
450200,柳州市
450201,市辖区
450202,城中区
450203,鱼峰区
450204,柳南区
450205,柳北区
450221,柳江县
450222,柳城县
450223,鹿寨县
450224,融安县
450225,融水苗族自治县
450226,三江侗族自治县
450300,桂林市
450301,市辖区
450302,秀峰区
450303,叠彩区
450304,象山区
450305,七星区
450311,雁山区
450321,阳朔县
450322,临桂县
450323,灵川县
450324,全州县
450325,兴安县
450326,永福县
450327,灌阳县
450328,龙胜各族自治县
450329,资源县
450330,平乐县
450331,荔浦县
450332,恭城瑶族自治县
450400,梧州市
450401,市辖区
450403,万秀区
450404,蝶山区
450405,长洲区
450421,苍梧县
450422,藤县
450423,蒙山县
450481,岑溪市
450500,北海市
450501,市辖区
450502,海城区
450503,银海区
450512,铁山港区
450521,合浦县
450600,防城港市
450601,市辖区
450602,港口区
450603,防城区
450621,上思县
450681,东兴市
450700,钦州市
450701,市辖区

450702,钦南区
450703,钦北区
450721,灵山县
450722,浦北县
450800,贵港市
450801,市辖区
450802,港北区
450803,港南区
450804,覃塘区
450821,平南县
450881,桂平市
450900,玉林市
450901,市辖区
450902,玉州区
450921,容县
450922,陆川县
450923,博白县
450924,兴业县
450981,北流市
451000,百色市
451001,市辖区
451002,右江区
451021,田阳县
451022,田东县
451023,平果县
451024,德保县
451025,靖西县
451026,那坡县
451027,凌云县
451028,乐业县
451029,田林县
451030,西林县
451031,隆林各族自治县
451100,贺州市
451101,市辖区
451102,八步区
451121,昭平县
451122,钟山县
451123,富川瑶族自治县
451200,河池市
451201,市辖区
451202,金城江区
451221,南丹县
451222,天峨县
451223,凤山县
451224,东兰县
451225,罗城仫佬族自治县
451226,环江毛南族自治县
451227,巴马瑶族自治县
451228,都安瑶族自治县
451229,大化瑶族自治县
451281,宜州市
451300,来宾市
451301,市辖区
451302,兴宾区
451321,忻城县
451322,象州县
451323,武宣县

451324,金秀瑶族自治县
451381,合山市
451400,崇左市
451401,市辖区
451402,江洲区
451421,扶绥县
451422,宁明县
451423,龙州县
451424,大新县
451425,天等县
451481,凭祥市
460000,海南省
460100,海口市
460101,市辖区
460105,秀英区
460106,龙华区
460107,琼山区
460108,美兰区
460200,三亚市
460201,市辖区
460300,三沙市
460321,西沙群岛
460322,南沙群岛
460323,中沙群岛的岛礁及其海域
469000,省直辖县级行政区划
469001,五指山市
469002,琼海市
469003,儋州市
469005,文昌市
469006,万宁市
469007,东方市
469021,定安县
469022,屯昌县
469023,澄迈县
469024,临高县
469025,白沙黎族自治县
469026,昌江黎族自治县
469027,乐东黎族自治县
469028,陵水黎族自治县
469029,保亭黎族苗族自治县
469030,琼中黎族苗族自治县
500000,重庆市
500100,市辖区
500101,万州区
500102,涪陵区
500103,渝中区
500104,大渡口区
500105,江北区
500106,沙坪坝区
500107,九龙坡区
500108,南岸区
500109,北碚区
500110,綦江区
500111,大足区
500112,渝北区
500113,巴南区
500114,黔江区
500115,长寿区

500116,江津区
500117,合川区
500118,永川区
500119,南川区
500200,县
500223,潼南县
500224,铜梁县
500226,荣昌县
500227,璧山县
500228,梁平县
500229,城口县
500230,丰都县
500231,垫江县
500232,武隆县
500233,忠县
500234,开县
500235,云阳县
500236,奉节县
500237,巫山县
500238,巫溪县
500240,石柱土家族自治县
500241,秀山土家族苗族自治县
500242,酉阳土家族苗族自治县
500243,彭水苗族土家族自治县
510000,四川省
510100,成都市
510101,市辖区
510104,锦江区
510105,青羊区
510106,金牛区
510107,武侯区
510108,成华区
510112,龙泉驿区
510113,青白江区
510114,新都区
510115,温江区
510121,金堂县
510122,双流县
510124,郫县
510129,大邑县
510131,蒲江县
510132,新津县
510181,都江堰市
510182,彭州市
510183,邛崃市
510184,崇州市
510300,自贡市
510301,市辖区
510302,自流井区
510303,贡井区
510304,大安区
510311,沿滩区
510321,荣县
510322,富顺县
510400,攀枝花市
510401,市辖区
510402,东区
510403,西区

510411,仁和区
510421,米易县
510422,盐边县
510500,泸州市
510501,市辖区
510502,江阳区
510503,纳溪区
510504,龙马潭区
510521,泸县
510522,合江县
510524,叙永县
510525,古蔺县
510600,德阳市
510601,市辖区
510603,旌阳区
510623,中江县
510626,罗江县
510681,广汉市
510682,什邡市
510683,绵竹市
510700,绵阳市
510701,市辖区
510703,涪城区
510704,游仙区
510722,三台县
510723,盐亭县
510724,安县
510725,梓潼县
510726,北川羌族自治县
510727,平武县
510781,江油市
510800,广元市
510801,市辖区
510802,利州区
510811,元坝区
510812,朝天区
510821,旺苍县
510822,青川县
510823,剑阁县
510824,苍溪县
510900,遂宁市
510901,市辖区
510903,船山区
510904,安居区
510921,蓬溪县
510922,射洪县
510923,大英县
511000,内江市
511001,市辖区
511002,市中区
511011,东兴区
511024,威远县
511025,资中县
511028,隆昌县
511100,乐山市
511101,市辖区
511102,市中区
511111,沙湾区

511112,五通桥区
511113,金口河区
511123,犍为县
511124,井研县
511126,夹江县
511129,沐川县
511132,峨边彝族自治县
511133,马边彝族自治县
511181,峨眉山市
511300,南充市
511301,市辖区
511302,顺庆区
511303,高坪区
511304,嘉陵区
511321,南部县
511322,营山县
511323,蓬安县
511324,仪陇县
511325,西充县
511381,阆中市
511400,眉山市
511401,市辖区
511402,东坡区
511421,仁寿县
511422,彭山县
511423,洪雅县
511424,丹棱县
511425,青神县
511500,宜宾市
511501,市辖区
511502,翠屏区
511503,南溪区
511521,宜宾县
511523,江安县
511524,长宁县
511525,高县
511526,珙县
511527,筠连县
511528,兴文县
511529,屏山县
511600,广安市
511601,市辖区
511602,广安区
511621,岳池县
511622,武胜县
511623,邻水县
511681,华蓥市
511700,达州市
511701,市辖区
511702,通川区
511721,达县
511722,宣汉县
511723,开江县
511724,大竹县
511725,渠县
511781,万源市
511800,雅安市
511801,市辖区

511802,雨城区
511803,名山区
511822,荥经县
511823,汉源县
511824,石棉县
511825,天全县
511826,芦山县
511827,宝兴县
511900,巴中市
511901,市辖区
511902,巴州区
511921,通江县
511922,南江县
511923,平昌县
512000,资阳市
512001,市辖区
512002,雁江区
512021,安岳县
512022,乐至县
512081,简阳市
513200,阿坝藏族羌族自治州
513221,汶川县
513222,理县
513223,茂县
513224,松潘县
513225,九寨沟县
513226,金川县
513227,小金县
513228,黑水县
513229,马尔康县
513230,壤塘县
513231,阿坝县
513232,若尔盖县
513233,红原县
513300,甘孜藏族自治州
513321,康定县
513322,泸定县
513323,丹巴县
513324,九龙县
513325,雅江县
513326,道孚县
513327,炉霍县
513328,甘孜县
513329,新龙县
513330,德格县
513331,白玉县
513332,石渠县
513333,色达县
513334,理塘县
513335,巴塘县
513336,乡城县
513337,稻城县
513338,得荣县
513400,凉山彝族自治州
513401,西昌市
513422,木里藏族自治县
513423,盐源县
513424,德昌县

513425,会理县
513426,会东县
513427,宁南县
513428,普格县
513429,布拖县
513430,金阳县
513431,昭觉县
513432,喜德县
513433,冕宁县
513434,越西县
513435,甘洛县
513436,美姑县
513437,雷波县
520000,贵州省
520100,贵阳市
520101,市辖区
520102,南明区
520103,云岩区
520111,花溪区
520112,乌当区
520113,白云区
520114,小河区
520121,开阳县
520122,息烽县
520123,修文县
520181,清镇市
520200,六盘水市
520201,钟山区
520203,六枝特区
520221,水城县
520222,盘县
520300,遵义市
520301,市辖区
520302,红花岗区
520303,汇川区
520321,遵义县
520322,桐梓县
520323,绥阳县
520324,正安县
520325,道真仡佬族苗族自治县
520326,务川仡佬族苗族自治县
520327,凤冈县
520328,湄潭县
520329,余庆县
520330,习水县
520381,赤水市
520382,仁怀市
520400,安顺市
520401,市辖区
520402,西秀区
520421,平坝县
520422,普定县
520423,镇宁布依族苗族自治县
520424,关岭布依族苗族自治县
520425,紫云苗族布依族自治县
520500,毕节市
520502,七星关区
520521,大方县

520522,黔西县
520523,金沙县
520524,织金县
520525,纳雍县
520526,威宁彝族回族苗族自治县
520527,赫章县
520600,铜仁市
520602,碧江区
520603,万山区
520621,江口县
520622,玉屏侗族自治县
520623,石阡县
520624,思南县
520625,印江土家族苗族自治县
520626,德江县
520627,沿河土家族自治县
520628,松桃苗族自治县
522300,黔西南布依族苗族自治州
522301,兴义市
522322,兴仁县
522323,普安县
522324,晴隆县
522325,贞丰县
522326,望谟县
522327,册亨县
522328,安龙县
522600,黔东南苗族侗族自治州
522601,凯里市
522622,黄平县
522623,施秉县
522624,三穗县
522625,镇远县
522626,岑巩县
522627,天柱县
522628,锦屏县
522629,剑河县
522630,台江县
522631,黎平县
522632,榕江县
522633,从江县
522634,雷山县
522635,麻江县
522636,丹寨县
522700,黔南布依族苗族自治州
522701,都匀市
522702,福泉市
522722,荔波县
522723,贵定县
522725,瓮安县
522726,独山县
522727,平塘县
522728,罗甸县
522729,长顺县
522730,龙里县
522731,惠水县
522732,三都水族自治县
530000,云南省
530100,昆明市

530101,市辖区
530102,五华区
530103,盘龙区
530111,官渡区
530112,西山区
530113,东川区
530114,呈贡区
530122,晋宁县
530124,富民县
530125,宜良县
530126,石林彝族自治县
530127,嵩明县
530128,禄劝彝族苗族自治县
530129,寻甸回族彝族自治县
530181,安宁市
530300,曲靖市
530301,市辖区
530302,麒麟区
530321,马龙县
530322,陆良县
530323,师宗县
530324,罗平县
530325,富源县
530326,会泽县
530328,沾益县
530381,宣威市
530400,玉溪市
530402,红塔区
530421,江川县
530422,澄江县
530423,通海县
530424,华宁县
530425,易门县
530426,峨山彝族自治县
530427,新平彝族傣族自治县
530428,元江哈尼族彝族傣族自治县
530500,保山市
530501,市辖区
530502,隆阳区
530521,施甸县
530522,腾冲县
530523,龙陵县
530524,昌宁县
530600,昭通市
530601,市辖区
530602,昭阳区
530621,鲁甸县
530622,巧家县
530623,盐津县
530624,大关县
530625,永善县
530626,绥江县
530627,镇雄县
530628,彝良县
530629,威信县
530630,水富县
530700,丽江市
530701,市辖区

530702,古城区
530721,玉龙纳西族自治县
530722,永胜县
530723,华坪县
530724,宁蒗彝族自治县
530800,普洱市
530801,市辖区
530802,思茅区
530821,宁洱哈尼族彝族自治县
530822,墨江哈尼族自治县
530823,景东彝族自治县
530824,景谷傣族彝族自治县
530825,镇沅彝族哈尼族拉祜族自治县
530826,江城哈尼族彝族自治县
530827,孟连傣族拉祜族佤族自治县
530828,澜沧拉祜族自治县
530829,西盟佤族自治县
530900,临沧市
530901,市辖区
530902,临翔区
530921,凤庆县
530922,云县
530923,永德县
530924,镇康县
530925,双江拉祜族佤族布朗族傣族自治县
530926,耿马傣族佤族自治县
530927,沧源佤族自治县
532300,楚雄彝族自治州
532301,楚雄市
532322,双柏县
532323,牟定县
532324,南华县
532325,姚安县
532326,大姚县
532327,永仁县
532328,元谋县
532329,武定县
532331,禄丰县
532500,红河哈尼族彝族自治州
532501,个旧市
532502,开远市
532503,蒙自市
532523,屏边苗族自治县
532524,建水县
532525,石屏县
532526,弥勒县
532527,泸西县
532528,元阳县
532529,红河县
532530,金平苗族瑶族傣族自治县
532531,绿春县
532532,河口瑶族自治县
532600,文山壮族苗族自治州
532601,文山市
532622,砚山县
532623,西畴县
532624,麻栗坡县
532625,马关县

532626,丘北县
532627,广南县
532628,富宁县
532800,西双版纳傣族自治州
532801,景洪市
532822,勐海县
532823,勐腊县
532900,大理白族自治州
532901,大理市
532922,漾濞彝族自治县
532923,祥云县
532924,宾川县
532925,弥渡县
532926,南涧彝族自治县
532927,巍山彝族回族自治县
532928,永平县
532929,云龙县
532930,洱源县
532931,剑川县
532932,鹤庆县
533100,德宏傣族景颇族自治州
533102,瑞丽市
533103,芒市
533122,梁河县
533123,盈江县
533124,陇川县
533300,怒江傈僳族自治州
533321,泸水县
533323,福贡县
533324,贡山独龙族怒族自治县
533325,兰坪白族普米族自治县
533400,迪庆藏族自治州
533421,香格里拉县
533422,德钦县
533423,维西傈僳族自治县
540000,西藏自治区
540100,拉萨市
540101,市辖区
540102,城关区
540121,林周县
540122,当雄县
540123,尼木县
540124,曲水县
540125,堆龙德庆县
540126,达孜县
540127,墨竹工卡县
542100,昌都地区
542121,昌都县
542122,江达县
542123,贡觉县
542124,类乌齐县
542125,丁青县
542126,察雅县
542127,八宿县
542128,左贡县
542129,芒康县
542132,洛隆县
542133,边坝县

542200,山南地区
542221,乃东县
542222,扎囊县
542223,贡嘎县
542224,桑日县
542225,琼结县
542226,曲松县
542227,措美县
542228,洛扎县
542229,加查县
542231,隆子县
542232,错那县
542233,浪卡子县
542300,日喀则地区
542301,日喀则市
542322,南木林县
542323,江孜县
542324,定日县
542325,萨迦县
542326,拉孜县
542327,昂仁县
542328,谢通门县
542329,白朗县
542330,仁布县
542331,康马县
542332,定结县
542333,仲巴县
542334,亚东县
542335,吉隆县
542336,聂拉木县
542337,萨嘎县
542338,岗巴县
542400,那曲地区
542421,那曲县
542422,嘉黎县
542423,比如县
542424,聂荣县
542425,安多县
542426,申扎县
542427,索县
542428,班戈县
542429,巴青县
542430,尼玛县
542500,阿里地区
542521,普兰县
542522,札达县
542523,噶尔县
542524,日土县
542525,革吉县
542526,改则县
542527,措勤县
542600,林芝地区
542621,林芝县
542622,工布江达县
542623,米林县
542624,墨脱县
542625,波密县
542626,察隅县

542627,朗县
610000,陕西省
610100,西安市
610101,市辖区
610102,新城区
610103,碑林区
610104,莲湖区
610111,灞桥区
610112,未央区
610113,雁塔区
610114,阎良区
610115,临潼区
610116,长安区
610122,蓝田县
610124,周至县
610125,户县
610126,高陵县
610200,铜川市
610201,市辖区
610202,王益区
610203,印台区
610204,耀州区
610222,宜君县
610300,宝鸡市
610301,市辖区
610302,渭滨区
610303,金台区
610304,陈仓区
610322,凤翔县
610323,岐山县
610324,扶风县
610326,眉县
610327,陇县
610328,千阳县
610329,麟游县
610330,凤县
610331,太白县
610400,咸阳市
610401,市辖区
610402,秦都区
610403,杨陵区
610404,渭城区
610422,三原县
610423,泾阳县
610424,乾县
610425,礼泉县
610426,永寿县
610427,彬县
610428,长武县
610429,旬邑县
610430,淳化县
610431,武功县
610481,兴平市
610500,渭南市
610501,市辖区
610502,临渭区
610521,华县
610522,潼关县

610523,大荔县
610524,合阳县
610525,澄城县
610526,蒲城县
610527,白水县
610528,富平县
610581,韩城市
610582,华阴市
610600,延安市
610601,市辖区
610602,宝塔区
610621,延长县
610622,延川县
610623,子长县
610624,安塞县
610625,志丹县
610626,吴起县
610627,甘泉县
610628,富县
610629,洛川县
610630,宜川县
610631,黄龙县
610632,黄陵县
610700,汉中市
610701,市辖区
610702,汉台区
610721,南郑县
610722,城固县
610723,洋县
610724,西乡县
610725,勉县
610726,宁强县
610727,略阳县
610728,镇巴县
610729,留坝县
610730,佛坪县
610800,榆林市
610801,市辖区
610802,榆阳区
610821,神木县
610822,府谷县
610823,横山县
610824,靖边县
610825,定边县
610826,绥德县
610827,米脂县
610828,佳县
610829,吴堡县
610830,清涧县
610831,子洲县
610900,安康市
610901,市辖区
610902,汉滨区
610921,汉阴县
610922,石泉县
610923,宁陕县
610924,紫阳县
610925,岚皋县

610926,平利县
610927,镇坪县
610928,旬阳县
610929,白河县
611000,商洛市
611001,市辖区
611002,商州区
611021,洛南县
611022,丹凤县
611023,商南县
611024,山阳县
611025,镇安县
611026,柞水县
620000,甘肃省
620100,兰州市
620101,市辖区
620102,城关区
620103,七里河区
620104,西固区
620105,安宁区
620111,红古区
620121,永登县
620122,皋兰县
620123,榆中县
620200,嘉峪关市
620201,市辖区
620300,金昌市
620301,市辖区
620302,金川区
620321,永昌县
620400,白银市
620401,市辖区
620402,白银区
620403,平川区
620421,靖远县
620422,会宁县
620423,景泰县
620500,天水市
620501,市辖区
620502,秦州区
620503,麦积区
620521,清水县
620522,秦安县
620523,甘谷县
620524,武山县
620525,张家川回族自治县
620600,武威市
620601,市辖区
620602,凉州区
620621,民勤县
620622,古浪县
620623,天祝藏族自治县
620700,张掖市
620701,市辖区
620702,甘州区
620721,肃南裕固族自治县
620722,民乐县
620723,临泽县

620724,高台县
620725,山丹县
620800,平凉市
620801,市辖区
620802,崆峒区
620821,泾川县
620822,灵台县
620823,崇信县
620824,华亭县
620825,庄浪县
620826,静宁县
620900,酒泉市
620901,市辖区
620902,肃州区
620921,金塔县
620922,瓜州县
620923,肃北蒙古族自治县
620924,阿克塞哈萨克族自治县
620981,玉门市
620982,敦煌市
621000,庆阳市
621001,市辖区
621002,西峰区
621021,庆城县
621022,环县
621023,华池县
621024,合水县
621025,正宁县
621026,宁县
621027,镇原县
621100,定西市
621101,市辖区
621102,安定区
621121,通渭县
621122,陇西县
621123,渭源县
621124,临洮县
621125,漳县
621126,岷县
621200,陇南市
621201,市辖区
621202,武都区
621221,成县
621222,文县
621223,宕昌县
621224,康县
621225,西和县
621226,礼县
621227,徽县
621228,两当县
622900,临夏回族自治州
622901,临夏市
622921,临夏县
622922,康乐县
622923,永靖县
622924,广河县
622925,和政县
622926,东乡族自治县

622927,积石山保安族东乡族撒拉族自治县
623000,甘南藏族自治州
623001,合作市
623021,临潭县
623022,卓尼县
623023,舟曲县
623024,迭部县
623025,玛曲县
623026,碌曲县
623027,夏河县
630000,青海省
630100,西宁市
630101,市辖区
630102,城东区
630103,城中区
630104,城西区
630105,城北区
630121,大通回族土族自治县
630122,湟中县
630123,湟源县
632100,海东地区
632121,平安县
632122,民和回族土族自治县
632123,乐都县
632126,互助土族自治县
632127,化隆回族自治县
632128,循化撒拉族自治县
632200,海北藏族自治州
632221,门源回族自治县
632222,祁连县
632223,海晏县
632224,刚察县
632300,黄南藏族自治州
632321,同仁县
632322,尖扎县
632323,泽库县
632324,河南蒙古族自治县
632500,海南藏族自治州
632521,共和县
632522,同德县
632523,贵德县
632524,兴海县
632525,贵南县
632600,果洛藏族自治州
632621,玛沁县
632622,班玛县
632623,甘德县
632624,达日县
632625,久治县
632626,玛多县
632700,玉树藏族自治州
632721,玉树县
632722,杂多县
632723,称多县
632724,治多县
632725,囊谦县
632726,曲麻莱县
632800,海西蒙古族藏族自治州

632801,格尔木市
632802,德令哈市
632821,乌兰县
632822,都兰县
632823,天峻县
640000,宁夏回族自治区
640100,银川市
640101,市辖区
640104,兴庆区
640105,西夏区
640106,金凤区
640121,永宁县
640122,贺兰县
640181,灵武市
640200,石嘴山市
640201,市辖区
640202,大武口区
640205,惠农区
640221,平罗县
640300,吴忠市
640301,市辖区
640302,利通区
640303,红寺堡区
640323,盐池县
640324,同心县
640381,青铜峡市
640400,固原市
640401,市辖区
640402,原州区
640422,西吉县
640423,隆德县
640424,泾源县
640425,彭阳县
640500,中卫市
640501,市辖区
640502,沙坡头区
640521,中宁县
640522,海原县
650000,新疆维吾尔自治区
650100,乌鲁木齐市
650101,市辖区
650102,天山区
650103,沙依巴克区
650104,新市区
650105,水磨沟区
650106,头屯河区
650107,达坂城区
650109,米东区
650121,乌鲁木齐县
650200,克拉玛依市
650201,市辖区
650202,独山子区
650203,克拉玛依区
650204,白碱滩区
650205,乌尔禾区
652100,吐鲁番地区
652101,吐鲁番市
652122,鄯善县

652123,托克逊县
652200,哈密地区
652201,哈密市
652222,巴里坤哈萨克自治县
652223,伊吾县
652300,昌吉回族自治州
652301,昌吉市
652302,阜康市
652323,呼图壁县
652324,玛纳斯县
652325,奇台县
652327,吉木萨尔县
652328,木垒哈萨克自治县
652700,博尔塔拉蒙古自治州
652701,博乐市
652722,精河县
652723,温泉县
652800,巴音郭楞蒙古自治州
652801,库尔勒市
652822,轮台县
652823,尉犁县
652824,若羌县
652825,且末县
652826,焉耆回族自治县
652827,和静县
652828,和硕县
652829,博湖县
652900,阿克苏地区
652901,阿克苏市
652922,温宿县
652923,库车县
652924,沙雅县
652925,新和县
652926,拜城县
652927,乌什县
652928,阿瓦提县
652929,柯坪县
653000,克孜勒苏柯尔克孜自治州
653001,阿图什市
653022,阿克陶县
653023,阿合奇县
653024,乌恰县
653100,喀什地区
653101,喀什市
653121,疏附县
653122,疏勒县
653123,英吉沙县
653124,泽普县
653125,莎车县
653126,叶城县
653127,麦盖提县
653128,岳普湖县
653129,伽师县
653130,巴楚县
653131,塔什库尔干塔吉克自治县
653200,和田地区
653201,和田市
653221,和田县

653222,墨玉县
653223,皮山县
653224,洛浦县
653225,策勒县
653226,于田县
653227,民丰县
654000,伊犁哈萨克自治州
654002,伊宁市
654003,奎屯市
654021,伊宁县
654022,察布查尔锡伯自治县
654023,霍城县
654024,巩留县
654025,新源县
654026,昭苏县
654027,特克斯县
654028,尼勒克县
654200,塔城地区
654201,塔城市
654202,乌苏市
654221,额敏县
654223,沙湾县
654224,托里县
654225,裕民县
654226,和布克赛尔蒙古自治县
654300,阿勒泰地区
654301,阿勒泰市
654321,布尔津县
654322,富蕴县
654323,福海县
654324,哈巴河县
654325,青河县
654326,吉木乃县
659000,自治区直辖县级行政区划
659001,石河子市
659002,阿拉尔市
659003,图木舒克市
659004,五家渠市
710000,台湾省
810000,香港特别行政区
820000,澳门特别行政区

身份证校验码

将前面的身份证号码17位数分别乘以不同的系数。从第一位到第十七位的系数分别为：7 9 10 5 8 4 2 1 6 3 7 9 10 5 8

将这17位数字和系数相乘的结果相加；

用加出来和除以11，看余数是多少？；

余数只可能有0 1 2 3 4 5 6 7 8 9 10这11个数字。其分别对应的最后一位身份证的号码为1 0 X 9 8 7 6 5 4 3 2；

2.6. 银行卡

现行的16为卡号是国际标准，前6位是BIN，由国际卡组织分配，其中首位是4开头是VISA的卡；5开头Master；622开头的是中国银

第7位-15位各行根据自己需要按照地区归属分配，第16位为校验位。

其他非16位卡号都是各银行自己制订分配的。

现行的16为卡号是国际标准，前6位是BIN，由国际卡组织分配，其中首位是4的是VISA的卡，5是Master,622126~622925是银联
第7位-15位各行根据自己需要按照地区归属分配，第16位为校验位。
其他非16位卡号都是各银行自己制订分配的

银行卡卡号的编码规则是什么？

现行的16为卡号是国际标准，

前6位是BIN，由国际卡组织分配，其中首位是

4开头是VISA的卡，

5开头Master

622开头的是中国银联标准卡。

第7位-15位各行根据自己需要按照地区归属分配，

第16位为校验位。

其他非16位卡号都是各银行自己制订分配的。

第 22 章 NoSQL OOD(Object-Oriented Design)

1. MongoDB

1.1. 配置表 config

```
{
    "_id" : ObjectId("5799a8535a855eca473977e1"),
    "key" : "payment",
    "value" : {
        "alpay" : true,
        "tenpay" : false,
        "unionpay" : false,
    }
},

{
    "_id" : ObjectId("5799a8535a855eca47397723"),
    "key" : "signup",
    "value" : {
        "online" : true,
        "manual" : false
    }
}
```

1.2. 日志表

tag ENUM('unknow','www','user','admin') '日志标签',
time '插入时间',
facility ENUM('config','unionpay','sms','email') '类别',
priority ENUM('info','warning','error','critical','exception','debug') '级别',
message '内容'

```
/* 1 */
{
```

```
    "_id" : ObjectId("5799b0175a855eca473977e2"),
    "tag" : "www",
    "time" : "2016-07-30 12:12:00",
    "facility" : "config",
    "priority" : "info",
    "message" : "xxxxxxxxxxxxxxxxx"
}

/* 2 */
{
    "_id" : ObjectId("5799b10f5a855eca473977e3"),
    "tag" : "www",
    "time" : "2016-07-30 12:12:00",
    "facility" : "config",
    "priority" : "info",
    "message" : {
        "name" : "neo",
        "age" : 30,
        "sex" : true
    }
}
```

2. Cassandra

```
<Keyspaces>
  <Keyspace Name="Example">

    <KeysCachedFraction>0.01</KeysCachedFraction>

    <ColumnFamily CompareWith="BytesType" Name="User"/>
    <ColumnFamily CompareWith="UTF8Type" Name="UserProfile"/>
    <ColumnFamily CompareWith="UTF8Type" Name="Category"/>
    <ColumnFamily CompareWith="UTF8Type" Name="Article"/>
    <ColumnFamily CompareWith="UTF8Type" Name="ArticleComment" />
    <ColumnFamily CompareWith="UTF8Type" Name="Product"/>
    <ColumnFamily CompareWith="UTF8Type" Name="ProductComment" />
    <ColumnFamily CompareWith="UTF8Type" Name="ProductAttribute" CompareSubcolumnsWith="UTF8Ty
pe" ColumnType="Super" />
    <ColumnFamily ColumnType="Super"
              CompareWith="UTF8Type"
              CompareSubcolumnsWith="UTF8Type"
              Name="Address"
              Comment="A column family with supercolumns, whose column and subcolumn names are UTF8 stri
ngs"/>
  </Keyspace>
</Keyspaces>
```

2.1. User And Profile

```
set Example.User['neo']['uuid']='b5ac78c3-fd5c-40ca-acc2-04d483052fc4'
set Example.User['neo']['name']='neo'
set Example.User['neo']['passwd']='mNBhMPAH'
set Example.User['neo']['email']='openunix@163.com'
set Example.User['neo']['status']='Y'

get Example.User['neo']

set Example.User['jam']['uuid']='8e07adbd-2dea-40d0-a822-5909f14f9ba2'
set Example.User['jam']['name']='jam'
set Example.User['jam']['passwd']='mNBhMPAH'
set Example.User['jam']['email']='t1@163.com'
set Example.User['jam']['status']='Y'

get Example.User['jam']
```

```
set Example.UserProfile['b5ac78c3-fd5c-40ca-acc2-04d483052fc4']['name']='neo chen'
set Example.UserProfile['b5ac78c3-fd5c-40ca-acc2-04d483052fc4']['age']='30'
set Example.UserProfile['b5ac78c3-fd5c-40ca-acc2-04d483052fc4']['gender']='male'
set Example.UserProfile['b5ac78c3-fd5c-40ca-acc2-04d483052fc4']['Tel']='13113668890'
set Example.UserProfile['b5ac78c3-fd5c-40ca-acc2-04d483052fc4']['Cellphone']='13113668890'

get Example.UserProfile['b5ac78c3-fd5c-40ca-acc2-04d483052fc4']
```

2.2. Category

```
set Example.Category['85c1acb3-dc81-4626-aea9-c153dc80e74f']['uuid'] = '85c1acb3-dc81-4626-aea9-c
set Example.Category['85c1acb3-dc81-4626-aea9-c153dc80e74f']['name'] = '中国'
set Example.Category['85c1acb3-dc81-4626-aea9-c153dc80e74f']['description'] = '中华人民共和国'

set Example.Category['002f7fd4-455a-4f16-9cc8-38a43f9d285c']['uuid'] = '002f7fd4-455a-4f16-9cc8-3
set Example.Category['002f7fd4-455a-4f16-9cc8-38a43f9d285c']['name'] = '广东'
set Example.Category['002f7fd4-455a-4f16-9cc8-38a43f9d285c']['description'] = '广东省'
set Example.Category['002f7fd4-455a-4f16-9cc8-38a43f9d285c']['parent_uuid'] = '85c1acb3-dc81-4626

get Example.Category['002f7fd4-455a-4f16-9cc8-38a43f9d285c']
```

2.3. Article

```
set Example.Article['862f0f17-a697-49b3-9bca-68b0cfc873ec']['uuid'] = '862f0f17-a697-49b3-9bca-68
set Example.Article['862f0f17-a697-49b3-9bca-68b0cfc873ec']['title'] = '文章标题'
set Example.Article['862f0f17-a697-49b3-9bca-68b0cfc873ec']['content'] = '文章内容'
set Example.Article['862f0f17-a697-49b3-9bca-68b0cfc873ec']['author'] = 'Neo'
set Example.Article['862f0f17-a697-49b3-9bca-68b0cfc873ec']['datetime'] = '2010-5-10 12:00:00'

get Example.Article['862f0f17-a697-49b3-9bca-68b0cfc873ec']
```

2.4. Product and ProductAttribute

Product data

```
set Example.Product['b12e97e1-63b4-4042-a3f2-da60005ec081']['uuid'] = 'b12e97e1-63b4-4042-a3f2-da
set Example.Product['b12e97e1-63b4-4042-a3f2-da60005ec081']['name'] = 'Dell Optiplex 780'
set Example.Product['b12e97e1-63b4-4042-a3f2-da60005ec081']['description'] = 'Dell Computer'
set Example.Product['b12e97e1-63b4-4042-a3f2-da60005ec081']['price'] = '5000'
set Example.Product['b12e97e1-63b4-4042-a3f2-da60005ec081']['image'] = '/www/images/dell780.jpg'
set Example.Product['b12e97e1-63b4-4042-a3f2-da60005ec081']['category_uuid'] = 'b12e97e1-63b4-404

get Example.Product['b12e97e1-63b4-4042-a3f2-da60005ec081']
```

product attribute

```
set Example.ProductAttribute['b12e97e1-63b4-4042-a3f2-da60005ec081']['color']['box'] = 'silver'
set Example.ProductAttribute['b12e97e1-63b4-4042-a3f2-da60005ec081']['color']['display'] = 'black
set Example.ProductAttribute['b12e97e1-63b4-4042-a3f2-da60005ec081']['monitor']['size'] = '1440*9
set Example.ProductAttribute['b12e97e1-63b4-4042-a3f2-da60005ec081']['monitor']['power'] = '12v'
set Example.ProductAttribute['b12e97e1-63b4-4042-a3f2-da60005ec081']['parameter']['process'] = 'I
set Example.ProductAttribute['b12e97e1-63b4-4042-a3f2-da60005ec081']['parameter']['memory'] = '2G
set Example.ProductAttribute['b12e97e1-63b4-4042-a3f2-da60005ec081']['parameter']['harddisk'] = '
set Example.ProductAttribute['b12e97e1-63b4-4042-a3f2-da60005ec081']['parameter']['disc'] = 'DVD
set Example.ProductAttribute['b12e97e1-63b4-4042-a3f2-da60005ec081']['software']['os'] = 'Windows
set Example.ProductAttribute['b12e97e1-63b4-4042-a3f2-da60005ec081']['software']['compress'] = '7
set Example.ProductAttribute['b12e97e1-63b4-4042-a3f2-da60005ec081']['software']['media'] = 'Kmpl
set Example.ProductAttribute['b12e97e1-63b4-4042-a3f2-da60005ec081']['software']['game'] = 'mine'

get Example.ProductAttribute['b12e97e1-63b4-4042-a3f2-da60005ec081']
```

2.5. Address

```
set Example.Address['b5ac78c3-fd5c-40ca-acc2-04d483052fc4']['home']['street']='Longhua'
set Example.Address['b5ac78c3-fd5c-40ca-acc2-04d483052fc4']['home']['city']='Shenzhen'
```

```
set Example.Address['b5ac78c3-fd5c-40ca-acc2-04d483052fc4']['home']['zip']='518000'

set Example.Address['b5ac78c3-fd5c-40ca-acc2-04d483052fc4']['work']['street']='CheGongMiao'
set Example.Address['b5ac78c3-fd5c-40ca-acc2-04d483052fc4']['work']['city']='Shenzhen'
set Example.Address['b5ac78c3-fd5c-40ca-acc2-04d483052fc4']['work']['zip']='51800'

get Example.Address['b5ac78c3-fd5c-40ca-acc2-04d483052fc4']
```

2.6. 练习

```
division
 id
 name
 country_id

department
 id
 name
 up_id
 path

division_has_department

employee
 id
 ename
 name
 sex
 age
 department_id

devices
 name
 sn

devices_attribute


employee_has_devices
 employee_id
 devices_id
```

第 23 章 Spring Data 最佳实践

1. MySQL

ORM的出现解决了程序猿学习数据库学历成本，也加快了开发的速度。程序猿无需再学习数据库定义语言DDL以及数据库客户端，也无需关注建表这些繁琐的工作，同时也降低了数据库结构变更管理中与DBA频繁沟通的成本。。

在过去的两年中我们采用 Spring Data JPA 定义数据库，访问数据库，积累了很多经验，最终我们发现使用 Spring Data 实体定义完全可以代替 DBA 的建模工作。

下面我们采用案例，一个一个讲解，各种数据库实体关系的定义。相关数据库建模知识请先阅读 《Netkiller Architect 手札》 以及 《Netkiller Spring 手札》

1.1. 分类表

这是一个通用分类表，常见的父子关系加上path路径

```
 +-----------+
 | category  |
 |-----------|
 |id         | <---+
 |name       |     |
 |description|    1:n
 |status     |     |
 |pid        | o---+
 |path       |
 |status     |
 |ctime      |
 |mtime      |
 +-----------+
```

```
package cn.netkiller.domain;

import java.util.Date;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;

import org.springframework.format.annotation.DateTimeFormat;

import com.fasterxml.jackson.annotation.JsonFormat;
import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity
public class Category {
        @Id
```

```
@GeneratedValue(strategy = GenerationType.IDENTITY)
@Column(name = "id", unique = true, nullable = false, insertable = true, updatable = fals
public int id;
public String name;
public String description;
public String path;

@Column(columnDefinition = "enum('Enabled','Disabled') DEFAULT 'Enabled' COMMENT '状态'")
public String status;

@DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss")
@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone = "GMT+8")
@Column(columnDefinition = "TIMESTAMP DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间'")
public Date ctime;

@DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss")
@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone = "GMT+8")
@Column(columnDefinition = "TIMESTAMP NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP COMME
public Date mtime;

@ManyToOne(cascade = { CascadeType.PERSIST, CascadeType.REMOVE })
@JoinColumn(name = "pid", referencedColumnName = "id")
private Category categorys;

@JsonIgnore
@OneToMany(cascade = CascadeType.ALL, mappedBy = "category", fetch = FetchType.EAGER)
private Set<Category> category;

public int getId() {
        return id;
}

public void setId(int id) {
        this.id = id;
}

public String getName() {
        return name;
}

public void setName(String name) {
        this.name = name;
}

public String getDescription() {
        return description;
}

public void setDescription(String description) {
        this.description = description;
}

public String getPath() {
        return path;
}

public void setPath(String path) {
        this.path = path;
}

public String getStatus() {
        return status;
}

public void setStatus(String status) {
        this.status = status;
}

public Date getCtime() {
        return ctime;
}
```

```
        public void setCtime(Date ctime) {
                this.ctime = ctime;
        }

        public Date getMtime() {
                return mtime;
        }

        public void setMtime(Date mtime) {
                this.mtime = mtime;
        }

        public Category getCategorys() {
                return categorys;
        }

        public void setCategorys(Category categorys) {
                this.categorys = categorys;
        }

        public Set<Category> getCategory() {
                return category;
        }

        public void setCategory(Set<Category> category) {
                this.category = category;
        }

        @Override
        public String toString() {
                return "Category [id=" + id + ", name=" + name + ", description=" + description +
                                + status + ", ctime=" + ctime + ", mtime=" + mtime + ", categorys
                                + category + "]";
        }

}
```

期望结果

```
CREATE TABLE `category` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `ctime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '',
  `description` varchar(255) DEFAULT NULL,
  `mtime` timestamp NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP COMMENT '',
  `name` varchar(255) DEFAULT NULL,
  `path` varchar(255) DEFAULT NULL,
  `status` enum('Enabled','Disabled') DEFAULT 'Enabled' COMMENT '',
  `pid` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

1.2. 为字段增加索引

我们希望为 name 和 path 字段增加普通索引

```
package cn.netkiller.domain;

import java.util.Date;
import java.util.Set;
```

```
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Index;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.Table;

import org.springframework.format.annotation.DateTimeFormat;

import com.fasterxml.jackson.annotation.JsonFormat;
import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity
@Table(indexes = { @Index(name = "name", columnList = "name DESC"), @Index(name = "path", columnL
public class Category {
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        @Column(name = "id", unique = true, nullable = false, insertable = true, updatable = fals
        public int id;
        public String name;
        public String description;
        public String path;

        @Column(columnDefinition = "enum('Enabled','Disabled') DEFAULT 'Enabled' COMMENT '状态'")
        public String status;

        @DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss")
        @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone = "GMT+8")
        @Column(columnDefinition = "TIMESTAMP DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间'")
        public Date ctime;

        @DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss")
        @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone = "GMT+8")
        @Column(columnDefinition = "TIMESTAMP NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP COMME
        public Date mtime;

        @ManyToOne(cascade = { CascadeType.PERSIST, CascadeType.REMOVE })
        @JoinColumn(name = "pid", referencedColumnName = "id")
        private Category categorys;

        @JsonIgnore
        @OneToMany(cascade = CascadeType.ALL, mappedBy = "category", fetch = FetchType.EAGER)
        private Set<Category> category;

        public int getId() {
                return id;
        }

        public void setId(int id) {
                this.id = id;
        }

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        public String getDescription() {
                return description;
        }
```

```
        public void setDescription(String description) {
                this.description = description;
        }

        public String getPath() {
                return path;
        }

        public void setPath(String path) {
                this.path = path;
        }

        public String getStatus() {
                return status;
        }

        public void setStatus(String status) {
                this.status = status;
        }

        public Date getCtime() {
                return ctime;
        }

        public void setCtime(Date ctime) {
                this.ctime = ctime;
        }

        public Date getMtime() {
                return mtime;
        }

        public void setMtime(Date mtime) {
                this.mtime = mtime;
        }

        public Category getCategorys() {
                return categorys;
        }

        public void setCategorys(Category categorys) {
                this.categorys = categorys;
        }

        public Set<Category> getCategory() {
                return category;
        }

        public void setCategory(Set<Category> category) {
                this.category = category;
        }

        @Override
        public String toString() {
                return "Category [id=" + id + ", name=" + name + ", description=" + description +
                                + status + ", ctime=" + ctime + ", mtime=" + mtime + ", categorys
                                + category + "]";
        }

}
```

期望结果

```
CREATE TABLE `category` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `ctime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '????',
```

```
  `description` varchar(255) DEFAULT NULL,
  `mtime` timestamp NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP COMMENT '????',
  `name` varchar(255) DEFAULT NULL,
  `path` varchar(255) DEFAULT NULL,
  `status` enum('Enabled','Disabled') DEFAULT 'Enabled' COMMENT '??',
  `pid` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `name` (`name`),
  KEY `path` (`path`),
  KEY `FKeiel7nqjxu4kmefso9tm9qcsu` (`pid`),
  CONSTRAINT `FKeiel7nqjxu4kmefso9tm9qcsu` FOREIGN KEY (`pid`) REFERENCES `category` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

1.3. 复合索引

创建由多个字段组成的复合索引，如： "member_id", "articleId"

```
package cn.netkiller.api.model;

import java.io.Serializable;
import java.util.Date;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.persistence.UniqueConstraint;

import com.fasterxml.jackson.annotation.JsonFormat;

@Entity
@Table(name = "comment", uniqueConstraints = { @UniqueConstraint(columnNames = { "member_id", "ar
public class Comment implements Serializable {
        /**
         *
         */
        private static final long serialVersionUID = -1484408775034277681L;
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        @Column(name = "id", unique = true, nullable = false, insertable = true, updatable = fals
        private int id;

        @ManyToOne(cascade = { CascadeType.ALL })
        @JoinColumn(name = "member_id")
        private Member member;

        private int articleId;

        private String message;

        @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")
        @Temporal(TemporalType.TIMESTAMP)
        @Column(updatable = false)
        @org.hibernate.annotations.CreationTimestamp
        protected Date createDate;

        public int getId() {
                return id;
        }
```

```
        public void setId(int id) {
                this.id = id;
        }

        public Member getMember() {
                return member;
        }

        public void setMember(Member member) {
                this.member = member;
        }

        public int getArticleId() {
                return articleId;
        }

        public void setArticleId(int articleId) {
                this.articleId = articleId;
        }

        public String getMessage() {
                return message;
        }

        public void setMessage(String message) {
                this.message = message;
        }

        public Date getCreateDate() {
                return createDate;
        }

        public void setCreateDate(Date createDate) {
                this.createDate = createDate;
        }
}
```

期望结果

```
CREATE TABLE `comment` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `article_id` int(11) NOT NULL,
  `create_date` datetime DEFAULT NULL,
  `message` varchar(255) DEFAULT NULL,
  `member_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `UK5qxfiu92nwlvgli7bl3evl11m` (`member_id`,`article_id`),
  CONSTRAINT `FKmrrrpi513ssu63i2783jyiv9m` FOREIGN KEY (`member_id`) REFERENCES `member` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

1.4. 一对多实例

如下图，我们将实现 categroy 和 article 的一对多关系

```
      +-----------+
      | category  |
      |-----------|
  +-->|id         | <---+
  |   |title      |     |
  |   |description|    1:n
```

```
  |   |status     |     |
  |   |parent_id  | o---+
  |   +-----------+
  |
 1:n
  |
  |   +-----------------+            +-----------------+
  |   | article         |            | feedback        |
  |   |-----------------|            |-----------------|
  |   |id               |<--1:n--+   |id               |
  |   |title            |        |   |title            |
  |   |content          |        |   |content          |
  |   |datetime         |        |   |datetime         |
  |   |status           |        |   |status           |
  +--o|category_id      |        +--o|article_id       |
  +--o|member_id        |        +-->|member_id        |
  |   +-----------------+        |   +-----------------+
  |   | 2007,2008,2009  |        |   | 2007,2008,2009  |
  |   +-----------------+        |   +-----------------+
  |                              |
 1:n  +----------+     +---1:n---+
  |   | member   |     |
  |   |----------|     |
  +-->|id        | <---+
      |user      |
      |passwd    |
      |nickname  |
      |status    |
      +----------+
```

首先定义分类实体类

```
package cn.netkiller.domain;

import java.util.Date;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Index;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.Table;

import org.springframework.format.annotation.DateTimeFormat;

import com.fasterxml.jackson.annotation.JsonFormat;
import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity
@Table(indexes = { @Index(name = "name", columnList = "name DESC"), @Index(name = "path", columnL
public class Category {
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        @Column(name = "id", unique = true, nullable = false, insertable = true, updatable = fals
        public int id;
        public String name;
        public String description;
        public String path;

        @Column(columnDefinition = "enum('Enabled','Disabled') DEFAULT 'Enabled' COMMENT '状态'")
```

```
public String status;

@DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss")
@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone = "GMT+8")
@Column(columnDefinition = "TIMESTAMP DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间'")
public Date ctime;

@DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss")
@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone = "GMT+8")
@Column(columnDefinition = "TIMESTAMP NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP COMME
public Date mtime;

@ManyToOne(cascade = { CascadeType.PERSIST, CascadeType.REMOVE })
@JoinColumn(name = "pid", referencedColumnName = "id")
private Category categorys;

@JsonIgnore
@OneToMany(cascade = CascadeType.ALL, mappedBy = "category", fetch = FetchType.EAGER)
private Set<Category> category;

@JsonIgnore
@OneToMany(cascade = CascadeType.ALL, mappedBy = "category", orphanRemoval = true)
private Set<Article> article;

public int getId() {
        return id;
}

public void setId(int id) {
        this.id = id;
}

public String getName() {
        return name;
}

public void setName(String name) {
        this.name = name;
}

public String getDescription() {
        return description;
}

public void setDescription(String description) {
        this.description = description;
}

public String getPath() {
        return path;
}

public void setPath(String path) {
        this.path = path;
}

public String getStatus() {
        return status;
}

public void setStatus(String status) {
        this.status = status;
}

public Date getCtime() {
        return ctime;
}

public void setCtime(Date ctime) {
        this.ctime = ctime;
}
```

```
        public Date getMtime() {
                return mtime;
        }

        public void setMtime(Date mtime) {
                this.mtime = mtime;
        }

        public Category getCategorys() {
                return categorys;
        }

        public void setCategorys(Category categorys) {
                this.categorys = categorys;
        }

        public Set<Category> getCategory() {
                return category;
        }

        public void setCategory(Set<Category> category) {
                this.category = category;
        }

        @Override
        public String toString() {
                return "Category [id=" + id + ", name=" + name + ", description=" + description +
                                + status + ", ctime=" + ctime + ", mtime=" + mtime + ", categorys
                                + category + "]";
        }

}
```

定义文章实体类

```
package cn.netkiller.domain;

import java.io.Serializable;
import java.util.Date;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.Table;

import org.springframework.format.annotation.DateTimeFormat;

import com.fasterxml.jackson.annotation.JsonFormat;
import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity
@Table(name = "article")
public class Article implements Serializable {
        private static final long serialVersionUID = 7603772682950271321L;

        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
@Column(name = "id", unique = true, nullable = false, insertable = true, updatable = fals
public int id;
public String title;
@Column(name = "short")
public String shortTitle;
public String description;
public String author;
public int star;
public String tag;
public boolean share;
public boolean status;
public String content;

@ManyToOne(cascade = { CascadeType.PERSIST, CascadeType.REMOVE })
@JoinColumn(name = "category_id", referencedColumnName = "id")
private Category category;

@ManyToOne(cascade = { CascadeType.PERSIST, CascadeType.REMOVE })
@JoinColumn(name = "site_id", referencedColumnName = "id")
private Site site;

@ManyToOne(cascade = { CascadeType.ALL })
@JoinColumn(name = "member_id", referencedColumnName = "id")
private Member member;

@DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss")
@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone = "GMT+8")
@Column(columnDefinition = "TIMESTAMP DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间'")
public Date ctime;

@DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss")
@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone = "GMT+8")
@Column(columnDefinition = "TIMESTAMP NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP COMME
public Date mtime;

@JsonIgnore
@OneToMany(cascade = CascadeType.ALL, mappedBy = "article", fetch = FetchType.EAGER)
private Set<Comment> comment;

@JsonIgnore
@OneToMany(cascade = CascadeType.ALL, mappedBy = "article", fetch = FetchType.EAGER)
private Set<Favorites> favorites;

@JsonIgnore
@OneToMany(cascade = CascadeType.ALL, mappedBy = "article", orphanRemoval = true)
private Set<Statistics> statistics;

public int getId() {
        return id;
}

public void setId(int id) {
        this.id = id;
}

public String getTitle() {
        return title;
}

public void setTitle(String title) {
        this.title = title;
}

public String getDescription() {
        return description;
}

public void setDescription(String description) {
        this.description = description;
}
```

```
public Date getCtime() {
        return ctime;
}

public void setCtime(Date ctime) {
        this.ctime = ctime;
}

public String getShortTitle() {
        return shortTitle;
}

public void setShortTitle(String shortTitle) {
        this.shortTitle = shortTitle;
}

public String getAuthor() {
        return author;
}

public void setAuthor(String author) {
        this.author = author;
}

public int getStar() {
        return star;
}

public void setStar(int star) {
        this.star = star;
}

public String getTag() {
        return tag;
}

public void setTag(String tag) {
        this.tag = tag;
}

public boolean isShare() {
        return share;
}

public void setShare(boolean share) {
        this.share = share;
}

public boolean isStatus() {
        return status;
}

public void setStatus(boolean status) {
        this.status = status;
}

public String getContent() {
        return content;
}

public void setContent(String content) {
        this.content = content;
}

public Category getCategory() {
        return category;
}

public void setCategory(Category category) {
        this.category = category;
}
```

```
        public Member getMember() {
                return member;
        }

        public void setMember(Member member) {
                this.member = member;
        }

        public Set<Comment> getComment() {
                return comment;
        }

        public void setComment(Set<Comment> comment) {
                this.comment = comment;
        }

        public Set<Favorites> getFavorites() {
                return favorites;
        }

        public void setFavorites(Set<Favorites> favorites) {
                this.favorites = favorites;
        }

        public Set<Statistics> getStatistics() {
                return statistics;
        }

        public void setStatistics(Set<Statistics> statistics) {
                this.statistics = statistics;
        }

        public Site getSite() {
                return site;
        }

        public void setSite(Site site) {
                this.site = site;
        }

        public Date getMtime() {
                return mtime;
        }

        public void setMtime(Date mtime) {
                this.mtime = mtime;
        }

        @Override
        public String toString() {
                return "Article [id=" + id + ", title=" + title + ", shortTitle=" + shortTitle +
                                + ", author=" + author + ", star=" + star + ", tag=" + tag + ", s
                                + ", content=" + content + ", category=" + category + ", site=" +
                                + ", ctime=" + ctime + ", mtime=" + mtime + ", comment=" + commen
                                + ", statistics=" + statistics + "]";
        }

}
```

希望结果

```
CREATE TABLE `category` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `ctime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '????',
  `description` varchar(255) DEFAULT NULL,
  `mtime` timestamp NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP COMMENT '????',
```

```
  `name` varchar(255) DEFAULT NULL,
  `path` varchar(255) DEFAULT NULL,
  `status` enum('Enabled','Disabled') DEFAULT 'Enabled' COMMENT '??',
  `pid` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `name` (`name`),
  KEY `path` (`path`),
  KEY `FKeiel7nqjxu4kmefso9tm9qcsu` (`pid`),
  CONSTRAINT `FKeiel7nqjxu4kmefso9tm9qcsu` FOREIGN KEY (`pid`) REFERENCES `category` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `article` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `author` varchar(255) DEFAULT NULL,
  `content` varchar(255) DEFAULT NULL,
  `ctime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '????',
  `description` varchar(255) DEFAULT NULL,
  `mtime` timestamp NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP COMMENT '????',
  `share` bit(1) NOT NULL,
  `short` varchar(255) DEFAULT NULL,
  `star` int(11) NOT NULL,
  `status` bit(1) NOT NULL,
  `tag` varchar(255) DEFAULT NULL,
  `title` varchar(255) DEFAULT NULL,
  `category_id` int(11) DEFAULT NULL,
  `member_id` int(11) DEFAULT NULL,
  `site_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `FKy5kkohbk00g0w88fi05k2hcw` (`category_id`),
  KEY `FK6l9vkfd5ixw8o8kph5rj1k7gu` (`member_id`),
  KEY `FKrxbc33rok9m4n6pnbbwb3piwf` (`site_id`),
  CONSTRAINT `FK6l9vkfd5ixw8o8kph5rj1k7gu` FOREIGN KEY (`member_id`) REFERENCES `member` (`id`),
  CONSTRAINT `FKrxbc33rok9m4n6pnbbwb3piwf` FOREIGN KEY (`site_id`) REFERENCES `site` (`id`),
  CONSTRAINT `FKy5kkohbk00g0w88fi05k2hcw` FOREIGN KEY (`category_id`) REFERENCES `category` (`id`
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

现在我们已经将 categroy 与 article 两张表一对多关系建立起来。

1.5. ManyToMany 多对多

用户与角色就是一个多对多的关系，多对多是需要中间表做关联的。所以需要一个 user_has_role 表。

```
+----------+          +---------------+            +--------+
| users    |          | user_has_role |            | role   |
+----------+          +---------------+            +--------+
| id       | <------o | user_id       |      /---> | id     |
| name     |          | role_id       | o---+      | name   |
| password |          |               |            |        |
+----------+          +---------------+            +--------+
```

创建 User 表

```
package cn.netkiller.api.domain.test;

import java.io.Serializable;
import java.util.Set;

import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
```

```
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;
import javax.persistence.JoinColumn;

@Entity
@Table(name = "users")
public class Users implements Serializable {
        /**
         *
         */
        private static final long serialVersionUID = -2480194112597046349L;
        @Id
        @GeneratedValue(strategy = GenerationType.AUTO)
        private int id;
        private String name;
        private String password;

        @ManyToMany(fetch = FetchType.EAGER)
        @JoinTable(name = "user_has_role", joinColumns = { @JoinColumn(name = "user_id", referenc
        private Set<Roles> roles;

        public int getId() {
                return id;
        }

        public void setId(int id) {
                this.id = id;
        }

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        public String getPassword() {
                return password;
        }

        public void setPassword(String password) {
                this.password = password;
        }

        public Set<Roles> getRoles() {
                return roles;
        }

        public void setRoles(Set<Roles> roles) {
                this.roles = roles;
        }

        @Override
        public String toString() {
                return "Users [id=" + id + ", name=" + name + ", password=" + password + ", roles
        }

}
```

创建 Role 表

```
package cn.netkiller.api.domain.test;
```

```java
import java.io.Serializable;
import java.util.Set;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToMany;
import javax.persistence.Table;

@Entity
@Table(name = "roles")
public class Roles implements Serializable {
	private static final long serialVersionUID = 6737037465677800326L;
	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	private int id;
	private String name;
	@ManyToMany(mappedBy = "roles")
	private Set<Users> users;

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public Set<Users> getUsers() {
		return users;
	}

	public void setUsers(Set<Users> users) {
		this.users = users;
	}

	@Override
	public String toString() {
		return "Roles [id=" + id + ", name=" + name + ", users=" + users + "]";
	}

}
```

最终产生数据库表如下

```mysql
CREATE TABLE `users` (
	`id` INT(11) NOT NULL AUTO_INCREMENT,
	`name` VARCHAR(255) NULL DEFAULT NULL,
	`password` VARCHAR(255) NULL DEFAULT NULL,
	PRIMARY KEY (`id`)
)
COLLATE='utf8_general_ci'
ENGINE=InnoDB;

CREATE TABLE `roles` (
	`id` INT(11) NOT NULL AUTO_INCREMENT,
	`name` VARCHAR(255) NULL DEFAULT NULL,
```

```
        PRIMARY KEY (`id`)
)
COLLATE='utf8_general_ci'
ENGINE=InnoDB;

CREATE TABLE `user_has_role` (
        `user_id` INT(11) NOT NULL,
        `role_id` INT(11) NOT NULL,
        PRIMARY KEY (`user_id`, `role_id`),
        INDEX `FKsvvq61v3koh04fycopbjx72hj` (`role_id`),
        CONSTRAINT `FK2dl1ftxlkldulcp934i3125qo` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`
        CONSTRAINT `FKsvvq61v3koh04fycopbjx72hj` FOREIGN KEY (`role_id`) REFERENCES `roles` (`id`
)
COLLATE='utf8_general_ci'
ENGINE=InnoDB;
```

1.6. 外键级联删除

orphanRemoval = true 可以实现数据级联删除

```
package cn.netkiller.api.domain;

import java.io.Serializable;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Table;

import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity
@Table(name = "member")
public class Member implements Serializable {
        /**
         *
         */
        private static final long serialVersionUID = 1L;

        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        @Column(name = "id", unique = true, nullable = false, insertable = true, updatable = fals
        private int id;

        private String name;
        private String sex;
        private int age;
        private String wechat;

        @Column(unique = true)
        private String mobile;
        private String picture;
        private String ipAddress;

        @JsonIgnore
        @OneToMany(cascade = CascadeType.ALL, orphanRemoval = true, mappedBy = "member")
```

```
private Set<Comment> comment;
@JsonIgnore
@OneToMany(cascade = CascadeType.ALL, orphanRemoval = true, mappedBy = "member")
private Set<StatisticsHistory> statisticsHistory;

public Member() {
}

public Member(int id) {
        this.id = id;
}

public int getId() {
        return id;
}

public void setId(int id) {
        this.id = id;
}

public String getName() {
        return name;
}

public void setName(String name) {
        this.name = name;
}

public String getSex() {
        return sex;
}

public void setSex(String sex) {
        this.sex = sex;
}

public int getAge() {
        return age;
}

public void setAge(int age) {
        this.age = age;
}

public String getWechat() {
        return wechat;
}

public void setWechat(String wechat) {
        this.wechat = wechat;
}

public String getMobile() {
        return mobile;
}

public void setMobile(String mobile) {
        this.mobile = mobile;
}

public String getPicture() {
        return picture;
}

public void setPicture(String picture) {
        this.picture = picture;
}

public String getIpAddress() {
        return ipAddress;
}
```

```
        public void setIpAddress(String ipAddress) {
                this.ipAddress = ipAddress;
        }

        @Override
        public String toString() {
                return "Member [id=" + id + ", name=" + name + ", sex=" + sex + ", age=" + age +
        }

}
```

2. MongoDB

2.1. 枚举定义

```
package api.pojo;

public class Enum {

        public enum Authority {
                USER, EXPERTS, ORGANIZATION, ADMIN
        }

        public enum Status {
                Enabled, Disabled, Yes, No, Y, N
        }

        public enum Examine {
                New, Pending, Rejected, Approved
        }

        public enum Task {
                New, Processing, Canceled, Completed
        }

        public enum Sex {
                True, False, T, F, Male, Female, Boy, Girl
        }

        public Enum() {
                // TODO Auto-generated constructor stub
        }
}
```

2.2. 日志表

```
package api.domain;

import java.util.Date;

import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

@Document
public class Logging {

        public enum Priority {
                info, warning, error, critical, exception, debug
        }

        public enum Facility {
                app, sms, ios, android, api, db
        }

        @Id
        private String id;
        private String tag;
        private Date time;
        private Facility facility;
        private Priority priority;
        private String message;
```

```
        public Logging() {
                // TODO Auto-generated constructor stub
        }

        public String getId() {
                return id;
        }

        public void setId(String id) {
                this.id = id;
        }

        public String getTag() {
                return tag;
        }

        public void setTag(String tag) {
                this.tag = tag;
        }

        public Date getTime() {
                return time;
        }

        public void setTime(Date time) {
                this.time = time;
        }

        public Facility getFacility() {
                return facility;
        }

        public void setFacility(Facility facility) {
                this.facility = facility;
        }

        public Priority getPriority() {
                return priority;
        }

        public void setPriority(Priority priority) {
                this.priority = priority;
        }

        public String getMessage() {
                return message;
        }

        public void setMessage(String message) {
                this.message = message;
        }

        @Override
        public String toString() {
                return "Logging [id=" + id + ", tag=" + tag + ", time=" + time + ", facility=" +
        }

}
```

2.3. 地址与定位

```
package cn.netkiller.api.domain;

import org.springframework.data.geo.Point;
import org.springframework.data.redis.core.index.GeoIndexed;
```

```
import org.springframework.data.redis.core.index.Indexed;

public class Address {

        private @Indexed String city;
        private String country;
        private @GeoIndexed Point location;

        public Address(String city, String country, Point location) {
                this.city = city;
                this.country = country;
                this.location = location;
        }

        public String getCity() {
                return city;
        }

        public void setCity(String city) {
                this.city = city;
        }

        public String getCountry() {
                return country;
        }

        public void setCountry(String country) {
                this.country = country;
        }

        public Point getLocation() {
                return location;
        }

        public void setLocation(Point location) {
                this.location = location;
        }

}
```

部分 III. 运维篇

第 24 章 IDC

1. 网络设备配置管理与版本控制

http://netkiller.github.io/journal/network.ios.html

1.1. 背景

我们经常会频繁的配置网络设备，但有时候做了某些操作出现了异常，我们不清楚问题出在哪里，还原配置也不起作用，甚至你根本记得你改动了什么。

另外我们希望能够监控网络设备的配置变化，一个公司可能有很多网络工程师，他们都有权限操作路由交换或防火墙设备，我们要知道网络设备配置什么时候发生了变化，并通知其他几位同事。同时我们要对修改操作记录归档，方便日后查阅。

1.2. 怎样实现网络设备配置管理

我们每隔一段时间便将网络设备的配置导出存档，然后通过版本控制工具进行版本化管理，远离非常简单。

有了版本控制我们可能很方便的回撤操作。下面我来详细讲解怎样安装于配置该软件

```
$ git clone https://github.com/netkiller/logging.git
$ cd logging
$ python3 setup.py sdist
$ python3 setup.py install

$ sudo apt-get install expect
$ sudo chmod +x /usr/local/libexec/*
```

配置网络设备地址

打开 /usr/local/bin/cisco 文件，修改BACKUP_DIR，改为你的备份目录

```
$ vim /usr/local/bin/cisco

CFGFILE=$BASEDIR/etc/cisco.conf
BACKUP_DIR=~/.backup

$ cat /usr/local/etc/cisco.conf
192.168.50.1 mgmt EBopQ1X2vMkrl M8YJxvDiddG6QK
192.168.50.2 mgmt EBopQ1X2vMkrl M8YJxvDiddG6QK
192.168.50.3 mgmt EBopQ1X2vMkrl M8YJxvDiddG6QK
192.168.50.4 mgmt EBopQ1X2vMkrl M8YJxvDiddG6QK

$ sudo chmod 600 /usr/local/etc/cisco.conf
```

编辑/usr/local/etc/cisco.conf文件，格式如下：

```
host | username | password | enable password
```

初始化版本仓库

```
# Initialized empty Git repository in local.
$ cisco init

# Initialized empty Git repository from remote.
$ cisco init http://xxx.xxx.xxx.xxx/project/xxxx.git
$ cisco init user@host:/project/xxxx.git
```

启动，停止等操作

```
$ cisco
Usage: /usr/local/bin/cisco {init|start|stop|status|restart}

# cisco start
```

```
# cisco stop
```

查看网络设备配置变化

查看当前与上一个版本的变化

```
$ cd your_backup_dir
$ git diff HEAD HEAD~ route.running-config
```

查看当前与前面第三个版本的变化

```
$ cd your_backup_dir
$ git diff HEAD HEAD~3 route.running-config
```

1.3. 总结

该程序主要是备份网络设备的配置文件，当然也能起到监控做用，但备份间隔时间需要根据你的情况设定，如果太频繁也不太好，间隔太长可能起不到很好的监控作用。

关于监控与报警，你可以通过git diff 命令对比与上次配置文件的变化，通常是没有变化的，一旦发生变化便通过电子邮件与短信发出警报。至于怎么实现，不再本文讨论范围，有兴趣可以参考笔者的其他技术手札。

关于版本控制软件更多细节，延伸阅读 《Netkiller Version 手札》

2. 机房迁移

总结一下5年前的工作，在不写下来自己都快忘光了，工作关系现在已经不涉及运维这块的工作。

2.1. 拓扑确立

首先制定服务器拓扑图，拓扑图应该有两套，一套是物理拓扑图，另一套是基于业务的虚拟拓扑图。

物理拓扑图包含机柜，机位，例如防火墙，核心交换机，机柜交换机，服务器，存储等等他们之间的物理关系。如果是云主机也许标注出来。

接下来分配IP地址以及服务端口号

最后制定虚拟拓扑图，是各种服务间的关系图，由IP地址和端口组成，标住出他们之间的关系。

2.2. 存储规划

什么东西放在什么地方，怎么规划空间等等。

2.2.1. RAID Disk Group 规划

根据不同用途使用不同的RAID，这主要跟IO密集都与数据安全性有关。

Virtual Disk 技术很有用，我使用这种技术两RAID划分为两个设备，一个用来安装操作系统，另一个用于数据存储，方便系统重做。

SSD 机械故障为零，整体故障率低于传统硬盘。我通常做RAID0用与负载均衡场景。

2.2.2. 文件系统规划

我通常使用btrfs，LVM/EXT4已经过时。

/ 分区EXT4 安装操作系统，swap 分区不一定是内存2倍，因为现在的服务器都是8~16GB，OS很少能使用到交换分区，但是像Oracle这样强制交换分区为内存两倍。

其余所有空间分区格式化为btrfs mount 到 /srv 目录，在通过子卷(subvolume)分配给各个应用。

提示

子卷(subvolume) 有个特点是不能rm -rf 删除子卷的，也起到一定的安全性。

2.2.3. 目录规划

以Tomcat为例

Tomcat 的虚拟机功能基本没用，因为需要升级需要频繁启动，会影响其他业务，所以采用每个项目一个实例的方式。

```
/srv/apache-tomcat/ 是Tomcat目录
/srv/apache-tomcat/www.netkiller.cn 每个实例一个目录
/srv/apache-tomcat/other.netkiller.cn
```

以PHP为例

```
/srv/php-7.0.0
ln -s /srv/php-7.0.0 /srv/php
```

通过 /srv/php 符号链接可以任意切换PHP版本

代码目录与服务器目录分开

```
/www/netkiller.cn/www.netkiller.cn
/www/netkiller.cn/other.netkiller.cn
```

2.3. 设备上架

按照物理图谱图，对应机位安装设备，链接网线，整理机柜。

注意强弱电分离，以免强电磁场干扰弱电。以Dell系列服务器为例，电源通常在右边，网口在中间左边，这样电源走机会右侧理线架，网线走左侧理线架。

我通常每个机柜放两台千兆交换机，一台放在机柜最顶端，通过10GB万兆以太网链接至核心交换机，走核心业务数据；另一台放在机柜最底端，负责其他次要业务，例如远程控制口，数据库备份等等。

上电，接通电源，开机。观察机柜的电压/电流变化。

2.4. 操作系统初始化

安装操作系统，系统裁剪，内核优化，时区设置，配置history格式（记录每条发出命令的时间点），TCP栈优化

安装自动化运维客户端，监控客户端

2.5. 服务器及运行环境

通过脚本或者自动化运维工具按照并配置。

安装各种服务器软件如 nginx, apache-httpd, apache-tomcat .....

软件运行环境，例如Java，PHP, Node.js, Ruby, Python .....

安装数据库，配置复制策略，备份计划

2.6. 部署应用程序

配置管理员通过虚拟拓扑提供的IP地址，端口号以及运维提供的账号密码配置应用程序。

然后部署应用程序到远程服务器

2.7. 监控系统

应用程序部署完毕后不要急着测试，可能很多IP地址以及端口不通，这时候测试只能是频繁报BUG。

我们先让将监控系统建立起来，监控所有服务器IP地址与端口，以及各种应用服务监控。

硬件监控: 温度监控，风扇监控，RAID卡监控，内存监控，PCI设备监控...

操作监控：负载，CPU，内存，用户登陆监控，磁盘空间监控，网络流量监控，TCP/IP状态监控，进程数量，线程监控，僵尸进程，进程退出...

服务器监控：连接数，线程数，进程数，内存开销，节点状态...

日志监控：如果监控到日志中出现某些关键次，发出警报。

服务监控：HTTP，SMTP，POP，AJAX/JSON，XML

2.8. 日志中心

所有的日志应该实时同步到日志中心，便于开发与测试人员实时观察服务器的状态

2.9. 测试

当我们看到监控系统报表中的各种服务器都畅通无阻时就可以进行验收测试，测试的时候需要关注监控系统的图标，与日志中心的日志变化。

安全测试：硬件防火墙规则，服务器防火墙规则，SSL证书，服务器版本号隐藏，操作系统权限检查

压力与性能测试

业务功能测试

3. 网线怎样连接才合理

一般小IDC，或者小客户，IDC只给一条共享或独享线路。单先连接非常简单，一条网线进入核心机柜，直接插到防火墙/路由器上即可。

由于我们是大客户对网络要求比较高，IDC给了我们两条线路，一条是Active，一条是Stand By。两条线路怎么连接防火墙呢?

3.1. 单个硬件防火墙方案

Firewall Cisco ASA 5550 + Switch 3Layer

```
=================================
Design by Neo chan
Homepage: http://netkiller.sf.net
=================================

Internet     ____________________________________________
  |  |      /                                            |
  v  v    v                                               |
+-----------------------------------------------------+   |
|[1] [3] [5] [7] [] [] [] [] [] [] [] []     Switch |   |
|[2] [4] [6] [8] [] [] [] [] [] [] [] []  [SFP][SFP]|   |
+-----------------------------------------------------+   |
  |          ___________________________________________/
  v         /
+-----------------------------------------------------+
| [Wan]  [LAN]   [DMZ]                     Firewall |
|                                         [Console] |
+-----------------------------------------------------+

Vlan 2
------------------
G0/0/1-4

Port
------------------
G0/0/1          Internet Active Line
G0/0/3          Internet Stand by Line
G0/0/4          Debug, Mirror or Other

G0/0/2          Firewal Outside
G0/0/5          Firewal Inside

G0/0/6-24       Server
```

3.1.1. 防火墙

IDC 给你两条WLAN网线，一条是Active激活状态，如果这条网线出现中断，将自动切换到第二条网线上。

如果你直接将网线插到防火墙上，就意味着第二条备用线路你无法使用。我想出一个方案，如下图

防火墙Inside 口连接到交换机 5口

其他口连接服务器即可

3.1.2. 交换机

设置 G0/0/1 至 G0/0/4 为一个vlan，将两条WLAN网线分别插入1，3两个口

2口连接防火墙Outside口，

4口用于平时调试使用，可以直接插电脑等设备

3.2. 双防火墙方案

```
==================================
Design by Neo chan
Homepage: http://netkiller.sf.net
==================================

                Internet
  ____________________________________
  ^                                   ^
  |                                   |
  v                                   v
+----------------------------+      +----------------------------+
| [Wan]  [LAN]    Firewall A |      | [Wan]  [LAN]    Firewall B |
|                            | <--> |                            |
+----------------------------+      +----------------------------+
  +--------^                                  |
  |    ____________________________________/
  v   /
+----------------------------------------------------+
|[1] [3] [5] [7] [] [] [] [] [] [] [] []     Switch |
|[2] [4] [6] [8] [] [] [] [] [] [] [] []  [SFP][SFP]|
+----------------------------------------------------+
```

两个防火墙分别插一条线，两个防火墙做HSRP心跳。

交换机可以是一台，也可以是两台，我当时使用 Cisco 4507 交换机，分别于 ASA 防火墙连接。

3.3. 网卡

3.3.1. 内外隔离

双网卡方案,一般服务器会提供至少2块网卡。 使用两个交换机，一个交换机连接接防火墙，另一个交换机独立不接入Internet

```
+------------+    +--------------+    /---> [eth0 Server A eth1] <---\    +--------------+    /--
| Firewall   |--->| Switch - WAN |---> ---> [eth0 Server B eth1] <--- <---| Switch - LAN |---> --
+------------+    +--------------+    \---> [eth0 Server C eth1] <---/    +--------------+    \--
```

Internet 用户从防火墙进入，只能访问WAN交换机上的服务器，WAN上一般是WEB服务器，WEB服务器通过LAN交换机访问数据库，Memcache等服务器

这样既有效利用了网络IO，有能有效隔离不需要暴露在公网上的服务器还可以降低成本， WAN 可以使用100M交换机，LAN 可以使用1G交换机，因为内部数据传输远远大于外部。

另外 WAN与LAN也可以使用VLAN实现

3.3.2. 负载均衡

eth0与eth1 做bonding, eth2与3做bonding, 然后内外隔离

```
+--------------+    /---> [eth0/1 Server A eth2/3] <---\    +--------------+
| Switch - WAN |---> ---> [eth0/1 Server B eth2/3] <--- <---| Switch - LAN |
+--------------+    \---> [eth0/1 Server C eth2/3] <---/    +--------------+
```

3.3.3. 交叉互联

在交换机端口有限的情况可以采用交叉互联。

交叉线连接与通过交换机连接二者差异:

A与B两个服务器举例:

1. 交叉线连接A与B两个服务器，A发数据包，B接收数据包，如果接受方在接收包过程中出现异常（毫秒级），可能会堵塞，数据包会重新发包。交换机存储转发，仍然会接收数据放到背板缓存中，建立连接后交换机会处理一切。
2. A 服务器出现故障宕机，A网卡灯不亮，那么B服务器的网卡将检测，认为没有插网线，B网卡灯也是不亮状态。而通过交换机B网卡仍然工作

3.3.4. 网络适配器

3.3.4.1. 常见网络适配器品牌

Broadcom NetXtreme II Gigabit Ethernet Driver bnx2 v2.0.8-rh (Oct 11, 2010)

Emulex OneConnect 10Gbps NIC

Intel 10 Gigabit AT2 Server Adapter (E10G41AT2)

3.3.4.2. 1G 千兆以太网产品

目前服务器1G网卡市场90%都被Broadcom NetXtreme占领，不仅仅限于服务器网卡，Cisco的设备中用的也是Broadcom NetXtreme芯片

3.3.4.3. 10G 万兆以太网产品

万兆以太网标准很多，有10000BAST-T(使用双绞线连接)，还有SFP+(850nm 光纤连接)

Dell 有通过6类线连接的万兆交换机8024，服务器端Dell给用户配的是Intel万兆网卡，使用方法与千兆一样。

笔者有两个刀笼（刀片服务器），刀笼配置万兆模块通过4条10G SFP+ 连接到8024，然后服务器使用6类双绞线，通过Intel网卡连接8024。

光纤万兆网卡与千兆网卡使用上并无不同。如果指示灯不亮，请调换RX/TX光纤跳线

```
# dmesg | grep Emulex
Copyright(c) 2004-2009 Emulex. All rights reserved.
be2net 0000:18:00.0: eth0 - Emulex OneConnect 10Gbps NIC
be2net 0000:18:00.1: eth1 - Emulex OneConnect 10Gbps NIC
```

笔者使用过Emulex/Intel在Linux上无需驱动，光纤交换机Cisco 4507的万兆模块是Broadcom NetXtreme芯片的.

提示

无论是外形还是接口，万兆以太网与FC(Fibre Channel) HBA 卡很难区分，且卡上没有任何印刷文字提示，购买千万小心不要买错，最好与厂商反复确认。另外光纤交换机与FC交换机也容易混淆，我建议你网卡用Cisco交换机，存储用博科交换机

4. 记录思科路由器/防火墙/交换机日志

本程序用于收集，防火墙，路由器，交换机等日志

4.1. 开启日志

配置 Cisco ASA 5550 Firewall 防火墙，路由器和交换机操作方法大同小异。

```
logging enable
logging timestamp
logging trap warnings
logging host inside 172.16.0.5
logging facility local0
```

172.16.0.5 改为你的syslogd服务器地址

4.2. syslogd 服务器脚本

*注意：python版本必须3.0以上

chmod 700 syslogd

./syslogd

```
#!/srv/python/bin/python3
# -*- encoding: utf-8 -*-
# Cisco ASA Firewall - Syslog Server by neo
# Author: neo<neo.chen@live.com>

import logging
import socketserver
import threading

LOG_FILE = '/var/log/asa5550.log'

logging.basicConfig(level=logging.INFO,
                    format='%(message)s',
                    datefmt='',
                    filename=LOG_FILE,
```

```
                    filemode='a')

class SyslogUDPHandler(socketserver.BaseRequestHandler):

    def handle(self):
        data = bytes.decode(self.request[0].strip())
        socket = self.request[1]
        print( "%s : " % self.client_address[0], str(data))
        logging.info(str(data))
#        socket.sendto(data.upper(), self.client_address)

if __name__ == "__main__":
    try:
        HOST, PORT = "0.0.0.0", 514
        server = socketserver.UDPServer((HOST, PORT), SyslogUDPHandler)
        server.serve_forever(poll_interval=0.5)
    except (IOError, SystemExit):
        raise
    except KeyboardInterrupt:
        print ("Crtl+C Pressed. Shutting down.")
```

5. 影响网络流量的因素

数不清的用户在访问你的服务器

带宽与服务器可以随时增加，但也有限，如果你不清楚影响流量的因素，增加服务器也无济于事。瓶颈无处不在，你必须将各个瓶颈分析出来，并各个击破，才能达到你的目的。

正常的IDC硬件布局

```
user -> \                                      /--1G--> | server |
user ->  |--1G--> [Firewal] --1G--> [Switch] ----1G--> | server |
user -> /                                      \--1G--> | server |
```

5.1. 带宽

主流网络设备带宽均为1G，目前来看10G仍不普及，仅在存储领域封闭使用，价格非常昂贵

firewall (1G) - switch (Forwarding bandwidth / 1G) - server (NIC 1G)

5.1.1. 防火墙带宽

怎么能提高带宽呢?

首先是防火墙，这个设备非常重要。 100M 基本淘汰，10G 防火墙尚未普及，1G带宽如果不够怎么办? 答案是买2个，3个...

为什么不买10G的。在下面会谈到会话数，你一看就明白了。10G防火墙会话数不是1G防火墙会话数的十倍。

5.1.2. 交换机带宽

目前主流交换机 Cisco WS-C2960G-48TC-L，48个RJ45口与2或4个SFP光纤口均为1G带宽

一般中小企业1台交换机足够，再上一个台阶超过40台服务器，就会有出现多台交换机互连问题，使用以太网口与SFP光纤口的带宽是一样的，唯一区别是传输距离。

每个交换机后面都对应几十台服务器，每个服务器1G网卡，如果这些服务器满负荷传输，交换机与交换机间数据传输就会带来瓶颈。

通过端口聚合可以解决交换机间数据传输瓶颈，另种方式是交换机堆叠。

```
+----------+                          +----------+
|          |gi0/0/1            gi0/0/1|          |
| Switch A +--------------------------+ Switch B |
|          +--------------------------+          |
|          |gi0/0/2            gi0/0/2|          |
+----------+                          +----------+
```

比如你有5个机柜，将交换机放置到3号机柜，处于中间位置，所有交换机放入该机柜，然后堆叠，从中心机柜向两侧分线

对于不大不小的企业，直接采购IDC箱式交换机

5.1.2.1. 聚合端口

```
Example 1 : host to host at double speed

          +----------+                          +----------+
          |          |eth0                  eth0|          |
          | Host A   +--------------------------+  Host B  |
          |          +--------------------------+          |
          |          |eth1                  eth1|          |
          +----------+                          +----------+

  On each host :
     # modprobe bonding miimon=100
     # ifconfig bond0 addr
     # ifenslave bond0 eth0 eth1

Example 2 : host to switch at double speed

          +----------+                          +----------+
          |          |eth0                 port1|          |
          | Host A   +--------------------------+  switch  |
          |          +--------------------------+          |
```

```
          |          |eth1                 port2|          |
          +----------+                          +----------+

  On host A :                              On the switch :
     # modprobe bonding miimon=100           # set up a trunk on port1
     # ifconfig bond0 addr                     and port2
     # ifenslave bond0 eth0 eth1
```

Example 3: High Availability in a Multiple Switch Topology

```
               |                                     |
               |port3                           port3|
         +-----+----+                          +-----+----+
         |          |port2       ISL      port2|          |
         | switch A +--------------------------+ switch B |
         |          |                          |          |
         +-----+----+                          +-----++---+
               |port1                           port1|
               |             +-------+               |
               +-------------+ host1 +---------------+
                         eth0 +-------+ eth1
```

Example 4: Maximum Throughput in a Multiple Switch Topology

Multiple switches may be utilized to optimize for throughput when they are configured in parallel as part of an isolated network between two or more systems, for example:

```
                      +-----------+
                      |  Host A   |
                      +-+---+---+-+
                        |   |   |
               +--------+   |   +---------+
               |            |             |
        +------+---+  +-----+----+  +-----+----+
        | Switch A |  | Switch B |  | Switch C |
        +------+---+  +-----+----+  +-----+----+
               |            |             |
               +--------+   |   +---------+
                        |   |   |
                      +-+---+---+-+
                      |  Host B   |
                      +-----------+
```

Example 5: Using multiple host and multiple switches to build a "no single point of failure" solution.

```
                |                                     |
                |port3                           port3|
          +-----+----+                          +-----+----+
          |          |port7       ISL      port7|          |
          | switch A +--------------------------+ switch B |
          |          +--------------------------+          |
          |          |port8                port8|          |
          +----++----+                          +-----++---+
          port2||port1                           port1||port2
               ||             +-------+               ||
               |+-------------+ host1 +---------------+|
               |         eth0 +-------+ eth1            |
```

```
|                                       |
|                +-------+              |
+---------------+ host2 +----------------+
           eth0 +-------+ eth1
```

5.1.3. 服务器带宽

目前主流服务都配备2到4个网口，像IBM / HP / DELL 等品牌服务器你无需关心网卡问题.

这里主要是针对自行安装或使用PC服务器的用户，因为很多PC服务器使用Realtak网卡。那么Realtak与Broadcom的NetXtreme有什么不同?

建议你安装一个windows系统在服务器上，然后看看网卡驱动属性。Realtak 仅仅提供基本网络功能，QOS质量访问服务由驱动程序提供（软QOS）而NetXtreme 提供非常丰富的功能，并且都是硬件实现。

话题回到带宽上，linux 支持 bonding 网卡，可以帮你解决服务器网络通信带宽问题，bonding 还可以解决网卡故障转移，传输流量负载均衡等等。

在我的《Netkiller Linux 手札》中你可以找到具体的设置方法。

5.2. 会话数

firewall (nat session) - switch (Forwarding bandwidth) - os (ulimit,sysctl) - application (httpd,vsftpd,tomcat ...)

会话数，国人俗称并发数。当你的带宽没有满，但tcp不能建立连接，这时你就要考虑会话数了。

5.2.1. 防火墙会话数

购买防火墙的时候主要有两个指标，一是会话数，二是带宽，三是配备模块。售前工程师都会交代清楚。

例如 Cisco ASA 5550 会话数65万，2个1G接口，可选IPS模块等等...

使用下面命令可以查看当前会话数

```
show conn count
```

如果你网站或接口有100万访问量，那么即使带宽没有满，也无法在建立TCP连接，这时你需要增加线路，增加防火墙。

5.2.2. 服务器会话数

Linux 影响会话数的的参数与配置文件

/etc/security/limits.conf , /etc/security/limits.d

nofile - max number of open files 在POSIX系统中硬件，管道，Socket 均被看作是一个设备，如硬盘是块设备，显示器是字符设备，操作这些设备均使用c的open函数，被算作打开一个文件。所有设备都是如此，加上web服务器还要读取很多HTML文件，系统对nofile 开销是非常巨大的。

nproc - max number of processes 目前多线程是主流，使用多线程技术这个参数可以不关心。像Oracle,PostgreSQL, Apache prefork,你就需要关心这个参数

/etc/sysctl.conf , /etc/sysctl.d/

net.ipv4.ip_local_port_range = 1024 65500 可用端口范围

tcp 协议当你尝试主动与服务器建立连接，如：telnet 172.16.0.1 80,本地会开启一个大于1024小于65500的端口

client: localhost:1025 --- 172.16.0.1:80 server

以上参数要综合你的CPU处理能力，内存空间，硬盘IO等等，才能配置出合理数值

配置过大（小马拉大车），超出你的服务器处理能力，导致服务器无响应，最终只能重启

配置过小（大马拉小车），你的服务器长时间处于空间状态，CPU，内存没有得到合理使用

在我的《Netkiller Linux 手札》中你可以找到具体的设置方法。

5.2.3. 应用服务器会话数

连接数受限与limits与sysctl

Nginx

```
worker_processes 8; 处理器数
worker_rlimit_nofile 65530; 允许最多打开文件数
worker_connections 4096; 最大连接数数为
keepalive_timeout 65; 开启复用连接
```

apache : httpd/conf/extra/httpd-mpm.conf

```
<IfModule mpm_worker_module>
    ServerLimit         16
    ThreadLimit         256
    StartServers        8
    MaxClients          4096
    MinSpareThreads     64
    MaxSpareThreads     256
    ThreadsPerChild     256
    MaxRequestsPerChild 10000
</IfModule>
```

mysql : /etc/my.cnf

```
[mysqld]
max_connections=250
```

不依依列举，有兴趣看我的系列文档。

5.3. IO

IO (Input/Output) 输入/输出，在国内被泛指硬盘IO，没办法这里也不例外，也被指为硬盘IO

5.3.1. 硬盘 HDD

影响IO的几个参数：

硬盘转速与硬盘速率

RAID卡速率

以Dell为例，去官网查看一下
http://www.dell.com/content/topics/topic.aspx/global/products/pvaul/topics/en/us/raid_controller?c=us&l=en&cs=555

PERC H700 Integrated / Adapter: 6Gb/s SAS

SAS 硬盘接口 3Gbps，理论读写速度300MB/S，实际情况没有这么理想。

RAID0 / RAID10是提高IO最有效的手段，但是你从上面数据计算。6块SAS硬盘做Raid 0 传输速率可以达到18Gb/s，但RAID卡H700只能达到6Gb/s，整体带宽并没有提高。

这样做的意义是在Raid带宽与硬盘速度不变的情况下，读写所花费的时间减少了，提高了列队处理速度，减少IO排队。

IO的问题就是IO排队等待问题，而不是传输带宽不够用

5.3.2. 固态硬盘 SSD

5.3.3. 分布IO

在经济紧张的情况下，可以使用多块独立硬盘分布IO，每块硬件分别做独立存储，比如数据库可以采用这种方案：可以一块硬盘存数据，一块硬盘做索引，另一块做日志等等，禁止交叉。

在经济允许的情况下，你可以配置多个RAID卡，外挂DAS。或者采用集群加分布式文件系统方案

5.3.4. FC SAN

8Gb Fibre Channel

我曾经测试过本地硬盘（146G 15RPM * 8 做RAID10）

5.3.5. iSCSI / FCoE

http://zh.wikipedia.org/wiki/ISCSI

iSCSI 可以提供1GB，10GB数据传输，传输介质可以选择双绞线或者光纤

FCoE 通过以太网传输FC协议，与iSCSI有很多相似之处

5.3.6. InfiniBand 或 RDMA

提供10Gbps ~ 120Gbps 的IO速度

http://en.wikipedia.org/wiki/InfiniBand

http://www.infinibandta.org/

第 25 章 Server

1. Linux 系统安全与优化配置

1.1. Openssh 安全配置

这节主要讲与SSH有关的安全配置

1.1.1. 禁止root用户登录

只允许普通用户登陆，然后通过su命令切换到root用过。后面还会将怎样限制su命令

```
PermitRootLogin no
```

1.1.2. 限制SSH验证重试次数

超过3次socket连接会断开，效果不明显，有一点点用。

```
MaxAuthTries 3
```

1.1.3. 禁止证书登陆

证书登陆非常安全，但是很有可能正常用户在你不知道情况下，给你安装了一个证书，他随时都可能进入你的系统

任何一个有权限的用户都能很方便的植入一个证书到 .ssh/authorized_keys 文件中

```
PubkeyAuthentication no
AuthorizedKeysFile /dev/null
```

1.1.4. 使用证书替代密码认证

是不是自相矛盾？ 这个跟上面讲的正好相反，这里只允许使用key文件登陆。

```
PasswordAuthentication no
```

这种方式比起密码要安全的多，唯一要注意的地方就是证书被拷贝 ，建议你给证书加上 passphrase。

证书的 passphrase 是可以通过openssl工具将其剥离的，SSH证书我没有试过，但是原理都差不多。

1.1.5. 图形窗口客户端记忆密码的问题

当你使用XShell, Xftp, WinSCP, SecureCRT, SecureFX .....等等软件登录时，该软件都提供记住密码的功能，使你下次再登陆的时候无须输入密码就可以进入系统。这样做的确非常方便，

但是你是否想过你的电脑一旦丢失或者被其他人进入，那有多么危险。我之前每天背着笔记本电脑上班，上面安装着XShell并且密码全部记忆在里面。这使我意识到一点电脑丢失，有多么可怕。

禁止SSH客户端记住密码，你不要要求别人那么做。你也无法控制，最终我找到了一种解决方案。

```
ChallengeResponseAuthentication yes
```

每次登陆都回提示你输入密码。密码保存也无效。

1.1.6. 关闭 GSSAPI

```
GSSAPIAuthentication no
#GSSAPIAuthentication yes
#GSSAPICleanupCredentials yes
#GSSAPICleanupCredentials yes
#GSSAPIStrictAcceptorCheck yes
#GSSAPIKeyExchange no
```

1.1.7. 禁止SSH端口映射

禁止使用SSH映射Socks5翻墙等等

```
AllowTcpForwarding no
```

1.1.8. IP地址限制

只允许通过192.168.2.1,192.168.2.2 访问本机

```
# vim /etc/hosts.allow
sshd:192.168.2.1,192.168.2.2
```

禁止所有人访问本机

```
# vim /etc/hosts.deny
sshd:ALL
```

上面使白名单策略，你也可以采用黑名单策略。

1.1.9. 禁止SSH密码穷举

骇客常常使用骇客字典穷举你的SSH密码，使用下面脚本可以封杀频繁链接的IP地址

```
#!/bin/bash
#######################################
# Homepage: http://netkiller.github.io
# Author: neo <netkiller@msn.com>
#######################################
PIPE=/var/tmp/pipe
pidfile=/var/tmp/$0.pid
BLACKLIST=/var/tmp/black.lst
WHITELIST=/var/tmp/white.lst

LOGFILE=/var/log/secure
DAY=5
########################################

if [ -z "$( egrep "CentOS|7." /etc/centos-release)" ]; then
        echo 'Only for CentOS 7.x'
        exit
fi

if [ -f $BLACKLIST ]; then
        find $BLACKLIST -type f -mtime +${DAY} -delete
fi

if [ ! -f ${BLACKLIST} ]; then
    touch ${BLACKLIST}
fi

if [ ! -f ${WHITELIST} ]; then
    touch ${WHITELIST}
fi

for ipaddr in $(grep rhost ${LOGFILE} | grep -oE "\b([0-9]{1,3}\.){3}[0-9]{1,3}\b" | sort | uniq
do

    if [ $(grep -c $ipaddr ${WHITELIST}) -gt 0 ]; then
                continue
    fi

    if [ $(grep -c $ipaddr ${BLACKLIST}) -eq 0 ] ; then
                echo $ipaddr >> ${BLACKLIST}
        iptables -I INPUT -p tcp --dport 22 -s $ipaddr -j DROP
        #iptables -I INPUT -s $ipaddr -j DROP
    fi
done
```

1.2. Shell 安全

1.2.1. .history 文件

SA的操作记录问题

通过~/.bash_history文件记录系统管理员的操作记录，定制.bash_history格式

```
HISTSIZE=1000
HISTFILESIZE=2000
HISTTIMEFORMAT="%Y-%m-%d-%H:%M:%S "
export HISTTIMEFORMAT
```

看看实际效果

```
$ history | head
    1  2012-02-27-09:10:45 do-release-upgrade
    2  2012-02-27-09:10:45 vim /etc/network/interfaces
    3  2012-02-27-09:10:45 vi /etc/network/interfaces
    4  2012-02-27-09:10:45 ping www.163.com
```

1.2.2. sudo 安全问题

/etc/sudoers

```
Cmnd_Alias WEBMASTER = /srv/nginx/sbin/nginx, /srv/php/sbin/php-fpm, !/srv/mysql/bin/*
www localhost = NETWORKING, SERVICES, DELEGATING, PROCESSES, WEBMASTER

Cmnd_Alias Database = /usr/bin/mysqldump, /srv/mysql/bin/mysql, /u01/oracle/10.x.x/bin/sqlplus
mysql localhost = NETWORKING, SERVICES, DELEGATING, PROCESSES, WEBMASTER, Database
```

使用www用户测试登录，无误后修改SSH配置文件，禁止root登录。

```
vim /etc/ssh/sshd_config
PermitRootLogin no
```

然后在测试从www sudo 执行命令, 可能成功启动nginx 与 php-fpm

1.2.3. 临时文件安全

临时文件不应该有执行权限

/tmp

```
/dev/sda3 /tmp ext4 nosuid, noexec, nodev, rw 0 0
```

同时使用符号连接将/var/tmp 指向 /tmp

/dev/shm

```
none /dev/shm tmpfs defaults, nosuid, noexec, rw 0 0
```

1.2.4. 执行权限

以数据库为例,从安全角度考虑我们需要如下更改

```
chown mysql:mysql /usr/bin/mysql*
chmod 700 /usr/bin/mysql*
```

mysql用户是DBA专用用户, 其他用户将不能执行mysql等命令。

1.3. 防火墙

开启防火墙

```
lokkit --enabled
```

1.3.1. 策略

默认INPUT，FORWARD，OUTPUT 三个都是ACCEPT

```
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
```

从安全的角度出发，INPUT，FORWARD，OUTPUT 三个都是DROP最安全，但配置的时候会给你带来非常多的不可预料的麻烦。

```
-P INPUT DROP
-P FORWARD DROP
-P OUTPUT DROP
```

折中的方案，也是打多少硬件防火墙厂商所采用的方案，他们都是采用INPUT默认禁用所有，OUTPUT默认允许所有，你只要关注INPUT规则即可。

```
-P INPUT DROP
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
```

1.3.2. 防止成为跳板机

跳板机就是用户首先登陆任意一台服务器后，由该服务器在登陆另外一台服务器。

封锁22等端口，避免相互跳转

```
iptables -A OUTPUT -p tcp -m multiport --dports 22,21,873 -j REJECT
/etc/init.d/iptables save
iptables -L -n
```

web 服务器禁止使用ssh，作为跳板机

用户将不能使用ssh命令登陆到其他电脑

1.3.3. 端口安全

有一种情况，例如你的服务器被植入了木马，木马将开启一个Socket端口给远程骇客接入进来，通常会启动一个类似telnet服务器,怎样防止未经允许的程序监听一个端口呢?

```
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
-A INPUT -m state --state INVALID,NEW -j DROP
```

用法

1. systemctl stop iptables
2. 启动 httpd / nginx
3. systemctl start iptables

注意必须按照上面的步骤，如果你试图如下尝试将失败

1. systemctl start iptables
2. 启动 httpd / nginx

80端口将无法对外提供服务，因为当 -A INPUT -m state --state INVALID,NEW -j DROP 运行以后，任何试图监听端口的程序将被拒绝。

1.3.4. 封锁特定字符串

下面的例子是拒绝爬虫

```
# iptables -A INPUT -p tcp --dport 80 -m string --algo bm --string "Spider" -j DROP
# iptables -A INPUT -p tcp --dport 80 -m string --algo bm --string "Baidu" -j DROP
# iptables -A INPUT -p tcp --dport 80 -m string --algo bm --string "Robat" -j DROP
```

1.4. Linux 系统资源调配

1.4.1. /etc/security/limits.conf

很多资料上是这么写的

```
* soft nofile 65535
* hard nofile 65535
```

这样做是偷懒，会带来很多问题，如果你的服务器被攻击，由于你的设置，系统将耗光你的资源，直到没有任何响应为止，你可能键盘输入都成问题，你不得不重启服务器，但你会发现重启只能维持短暂几分钟，又会陷入无响应状态。

```
nobody soft nofile 4096
nobody hard nofile 8192
```

为什么会设置为nobody用户呢？因为root用户启动系统后web 服务器会使用nobody用户创建子进程，socket连接实际上是nobody用户在处理。root 仅仅是守护父进程。

```
mysql soft nofile 2048
mysql hard nofile 2048
```

针对 mysql 做限制

提示

关于 nofile 即打开文件数，这个跟socket有非常紧密的关系，在linux系统中任何设备都被看做是一个文件（字符设备），你连接一个鼠标，键盘，摄像头，硬盘等等都被看作打开一个设备文件，所以默认1024是远远不够的。

1.4.2. 关闭写磁盘I/O功能

对于某些文件没必要记录文件的访问时间，由其是在高并发的IO密集操作的环境下，通过两个参数可以实现noatime,nodiratime减少不必要的系统IO资源。

编辑/etc/fstab 添加 noatime,nodiratime 参数

```
/dev/sdb1    /www          ext4    noatime,nodiratime        0 0
```

1.5. PAM 插件认证加固配置

配置文件

```
ls  /etc/pam.d/
chfn         crond                login    passwd            remote    runuser-l          smtp
chsh         fingerprint-auth     newrole  password-auth     run_init  smartcard-auth     smtp.pc
config-util  fingerprint-auth-ac  other    password-auth-ac  runuser   smartcard-auth-ac  sshd
```

认证插件

```
ls /lib64/security/
```

1.5.1. pam_tally2.so

此模块的功能是，登陆错误输入密码3次，5分钟后自动解禁，在未解禁期间输入正确密码也无法登陆。

在配置文件 /etc/pam.d/sshd 顶端加入

```
auth required pam_tally2.so deny=3 onerr=fail unlock_time=300
```

查看失败次数

```
# pam_tally2
Login           Failures Latest failure     From
root               14    07/12/13 15:44:37  192.168.6.2
neo                 8    07/12/13 15:45:36  192.168.6.2
```

重置计数器

```
# pam_tally2 -r -u root
Login           Failures Latest failure     From
root               14    07/12/13 15:44:37  192.168.6.2

# pam_tally2 -r -u neo
Login           Failures Latest failure     From
neo                 8    07/12/13 15:45:36  192.168.6.2
```

pam_tally2 计数器日志保存在 /var/log/tallylog 注意，这是二进制格式的文件

例 25.1. /etc/pam.d/sshd - pam_tally2.so

```
# cat  /etc/pam.d/sshd
#%PAM-1.0
auth required pam_tally2.so deny=3 onerr=fail unlock_time=300

auth       required     pam_sepermit.so
```

```
auth       include      password-auth
account    required     pam_nologin.so
account    include      password-auth
password   include      password-auth
# pam_selinux.so close should be the first session rule
session    required     pam_selinux.so close
session    required     pam_loginuid.so
# pam_selinux.so open should only be followed by sessions to be executed in the user context
session    required     pam_selinux.so open env_params
session    optional     pam_keyinit.so force revoke
session    include      password-auth
```

以上配置root用户不受限制, 如果需要限制root用户，参考下面

```
auth required pam_tally2.so deny=3 unlock_time=5 even_deny_root root_unlock_time=1800
```

1.5.2. pam_listfile.so

用户登陆限制

将下面一行添加到 /etc/pam.d/sshd 中，这里采用白名单方式，你也可以采用黑名单方式

```
auth       required     pam_listfile.so item=user sense=allow file=/etc/ssh/whitelist onerr=fail
```

将允许登陆的用户添加到 /etc/ssh/whitelist，除此之外的用户将不能通过ssh登陆到你的系统

```
# cat /etc/ssh/whitelist
neo
www
```

例 25.2. /etc/pam.d/sshd - pam_listfile.so

```
# cat /etc/pam.d/sshd
#%PAM-1.0
auth       required     pam_listfile.so item=user sense=allow file=/etc/ssh/whitelist onerr=fail
auth       required     pam_tally2.so deny=3 onerr=fail unlock_time=300

auth       required     pam_sepermit.so
auth       include      password-auth
account    required     pam_nologin.so
account    include      password-auth
password   include      password-auth
# pam_selinux.so close should be the first session rule
session    required     pam_selinux.so close
session    required     pam_loginuid.so
# pam_selinux.so open should only be followed by sessions to be executed in the user context
session    required     pam_selinux.so open env_params
session    optional     pam_keyinit.so force revoke
session    include      password-auth
```

sense=allow 白名单方式, sense=deny 黑名单方式

```
auth       required     pam_listfile.so item=user sense=deny file=/etc/ssh/blacklist onerr=fail
```

更多细节请查看手册 $ man pam_listfile

1.5.3. pam_access.so

编辑 /etc/pam.d/sshd 文件，加入下面一行

```
account required pam_access.so
```

保存后重启sshd进程

编辑 /etc/security/access.conf 文件

```
cat >>  /etc/security/access.conf << EOF

- : root : ALL EXCEPT 192.168.6.1
EOF
```

只能通过 192.168.6.1 登陆, 添加多个IP地址

```
- : root : ALL EXCEPT 192.168.6.1 192.168.6.2
```

测试是否生效

1.5.4. pam_wheel.so

限制普通用户通过su命令提升权限至root. 只有属于wheel组的用户允许通过su切换到root用户

编辑 /etc/pam.d/su 文件，去掉下面的注释

```
auth            required        pam_wheel.so use_uid
```

修改用户组别，添加到wheel组

```
# usermod -G wheel www

# id www
uid=501(www) gid=501(www) groups=501(www),10(wheel)
```

没有加入到wheel组的用户使用su时会提示密码不正确。

```
$ su - root
Password:
su: incorrect password
```

2. Tomcat 安全配置与性能优化

http://netkiller.github.io/journal/tomcat.html

2.1. JVM

2.1.1. 使用 Server JRE 替代JDK。

服务器上不要安装JDK，请使用 Server JRE. 服务器上根本不需要编译器，代码应该在Release服务器上完成编译打包工作。

理由：一旦服务器被控制，可以防止在其服务器上编译其他恶意代码并植入到你的程序中。

2.1.2. JAVA_OPTS

```
export JAVA_OPTS="-server -Xms512m -Xmx4096m  -XX:PermSize=64M -XX:MaxPermSize=512m"
```

-Xms 指定初始化时化的栈内存

-Xmx 指定最大栈内存

提示

Java 8 以后 -XX:PermSize 与 -XX:MaxPermSize 两个配置项被废弃

2.1.3. java.security 优化

打开$JAVA_HOME/jre/lib/security/java.security文件，找到下面的内容：

```
securerandom.source=file:/dev/urandom
替换成
securerandom.source=file:/dev/./urandom
```

2.2. Tomcat 优化

2.2.1. maxThreads 连接数限制

maxThreads 是 Tomcat 所能接受最大连接数。一般设置不要超过8000以上，如果你的网站访问量非常大可能使用运行多个Tomcat实例的方法。

即，在一个服务器上启动多个tomcat然后做负载均衡处理。

```
<Connector port="8080" address="localhost"
        maxThreads="2048" maxHttpHeaderSize="8192"
        emptySessionPath="true" protocol="HTTP/1.1"
        enableLookups="false" redirectPort="8181" acceptCount="100"
        connectionTimeout="20000" disableUploadTimeout="true" />
```

提示

很多做过php运维的朋友在这里会犯一个大错误，php优化服务器通常怎做法是安装cpu以及内存的情况配置连接数，连接数过万都很正常，但java不同jvm配置要非常小心，稍有差错就会崩溃。

maxThreads 配置要结合 JVM -Xmx 参数调整，也就是要考虑内存开销。

```
maxThreads              客户请求最大线程数
minSpareThreads         初始化时创建的 socket 线程数
maxSpareThreads         连接器的最大空闲 socket 线程数
```

2.2.2. 虚拟主机

不要使用Tomcat的虚拟主机，每个站点一个实例。即，启动多个tomcat.

这也是PHP运维在这里常犯的错误，PHP的做法是一个Web下面放置多个虚拟主机，而不是每个主机启动一个web服务器。Tomcat 是多线程,共享内存，任何一个虚拟主机中的应用出现崩溃，会影响到所有应用程序。采用多个实例方式虽然开销比较大，但保证了应用程序隔离与安全。

2.2.3. 压缩传输

通常所说的gzip压缩，Tomcat通过在server.xml配置设置压缩的选项。

```
<Connector port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443"
               compression="on"
               compressionMinSize1="2048"
               noCompressionUserAgents="gozilla, traviata"
               compressableMimeType="text/html,text/xml,text/javascript,text/css,text/plain,,appl
```

提示

压缩会增加Tomcat负担，最好采用Nginx + Tomcat 或者 Apache + Tomcat 方式，压缩交由Nginx/Apache 去做。

```
compression                     打开压缩功能
compressionMinSize      启用压缩的输出内容大小，这里面默认为2KB
compressableMimeType    压缩类型
```

2.3. Tomcat 安全配置

2.3.1. 禁用8005端口

telnet localhost 8005 然后输入 SHUTDOWN 就可以关闭 Tomcat，为了安全我们要禁用该功能

```
<Server port="-1" shutdown="SHUTDOWN">
```

2.3.2. 安装后初始化配置

当Tomcat完成安装后你首先要做的事情如下：

首次安装完成后立即删除webapps下面的所有代码

```
rm -rf /srv/apache-tomcat/webapps/*
```

注释或删除 tomcat-users.xml 所有用户权限，看上去如下：

```
# cat conf/tomcat-users.xml
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
</tomcat-users>
```

2.3.2.1. 隐藏版本信息

隐藏Tomcat版本信息，首先隐藏HTTP头中的版本信息

```
vim $CATALINA_HOME/conf/server.xml

    <Connector port="80" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443"
                                maxThreads="8192"
                                minSpareThreads="64"
                                maxSpareThreads="128"
                                acceptCount="128"
                                enableLookups="false"
                server="Neo App Srv 1.0"/>

# curl -I http://localhost:8080/
HTTP/1.1 400 Bad Request
Transfer-Encoding: chunked
Date: Thu, 20 Oct 2011 09:51:55 GMT
Connection: close
Server: Neo App Srv 1.0
```

服务器信息已经被改为 Server: Neo App Srv 1.0

注意：当出现 404 页面时仍可能看到Tomcat的版本信息

```
HTTP Status 404 - /sdf

type Status report

message /sdf
```

```
description The requested resource is not available.

Apache Tomcat/8.0.32
```

隐藏Tomcat 404页面版本信息的方法如下

```
mkdir -p apache-tomcat-8.0.33/lib/org/apache/catalina/util

cat >> apache-tomcat-8.0.33/lib/org/apache/catalina/util/ServerInfo.properties <<EOF
server.info=Apache
server.number=
server.built=
EOF
```

测试

```
HTTP Status 404 - /sdf

type Status report

message /sdf

description The requested resource is not available.

Apache
```

2.3.2.2. 应用程序安全

关闭war自动部署 unpackWARs="false" autoDeploy="false"。防止被植入木马等恶意程序

关闭 reloadable="false" 也用于防止被植入木马

2.3.2.3. JSESSIONID

修改 Cookie 变量 JSESSIONID， 这个cookie 是用于维持Session关系。建议你改为PHPSESSID。

```
<Context path="" docBase="path/to/your" reloadable="false" sessionCookiePath="/" sessionCookieNam
```

2.3.3. 启动用户与端口

不要使用root用户启动tomcat，Java程序与C程序不同。nginx,httpd 使用root用户启动守护80端口，子进程/线程会通过setuid(),setgid()两个函数切换到普通用户。即父进程所有者是root用户，子进程与多线程所有者是一个非root用户，这个用户没有shell，无法通过ssh与控制台登陆系统，Java 的JVM 是与系统无关的，是建立在OS之上的，你使用什么用户启动Tomcat，那麼Tomcat 就会继承该所有者的权限。

这造成了一个问题，Linux系统小于1024的端口只有root可以使用，这也是为什么Tomcat默认端口是8080。如果你想使用80端口只能使用root启动Tomcat。这有带来了很多安全问题。

解决方案是创建一个普通用户，如：

```
groupadd -g 80 daemon
adduser -o --home /daemon --shell /sbin/nologin --uid 80 --gid 80 -c "Web Server" daemon
```

注意 /sbin/nologin , 意味着该用户不能登录，同时我也没有给它指定密码，这个用户只能用于启动tomcat，没有Shell权限就以为只被注入后无法运行linux命令。

```
chown daemon:daemon -R /srv/*
su - daemon -c "/srv/apache-tomcat/bin/startup.sh"
```

接下来解决80端口问题, 思路就是80去调用8080，或者映射端口。

下面是影射方案,80 跳转 8080

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 8080

取消跳转
iptables -t nat -D PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 8080

查看规则
iptables -t nat -L
```

另一个就是从80请求去调用8080的方案

这个方案可以在 Tomcat 前段增加反向代理，例如：Nginx,Apache,Squid,Varnish或者F5， Array这类设备等等

2.4. 如何部署应用程序

应用程序部署与tomcat启动,不能使用同一个用户。

我的tomcat 安装在 /srv目录下，Tomcat启动用户为daemon; 应用程序放在/www目录下www所有者是www用户。这样的目的是一旦tomcat被植入web shell程序，它将不能创建或编辑/www目录下面的任何内容。

```
adduser --home /www -c "Web Application" www
```

我的Tomcat安装在/srv目录下，但应用程序放在/www目录下，一般是这样的结构。

```
/www/example.com/www.example.com
```

每次升级将压错包解压到 /www/example.com/目录下，www.example.com 是符号连接，连接到刚刚解压的目录。

这个可以实现通过符号连接在多个版本之间快速切换。

2.5. 延伸阅读

《Netkiller Web 手札》 之 Tomcat 篇

3. PHP 安全与性能优化

3.1. Apache mod_php / php-fpm

目录权限安全

3.1.1. 用户权限

web server 启动用户不能于运行用户为同一个用户

web server 运行用户与php程序不能为同一个用户

```
root      1082  0.0  0.1  11484  2236 ?        Ss   Mar01   0:00 nginx: master process /usr/sbin/
www-data 13650  0.0  0.0  11624  1648 ?        S    09:44   0:00 nginx: worker process
www-data 13651  0.0  0.0  11624  1132 ?        S    09:44   0:00 nginx: worker process
www-data 13652  0.0  0.0  11624  1132 ?        S    09:44   0:00 nginx: worker process
www-data 13653  0.0  0.0  11624  1132 ?        S    09:44   0:00 nginx: worker process
```

1. 父进程

   root 启动 web server, 此时web server 父进程应该是 root,同时父进程监听80端口

2. 子进程

   父进程派生许多子进程，同时使用setuid,setgid将子进程权限切换为非root

   子进程用户可以通过httpd.conf设置

   ```
   User nobody
   Group nobody
   ```

   nginx.conf

   ```
   $ cat /etc/nginx/nginx.conf
   user www-data;
   ```

3. fastcgi 进程

   ```
   root     13082  0.0  0.1  19880  2584 ?        Ss   09:28   0:00 php-fpm: master process (/et
   www-data 13083  0.0  0.1  20168  3612 ?        S    09:28   0:00 php-fpm: pool www
   www-data 13084  0.0  0.1  20168  2808 ?        S    09:28   0:00 php-fpm: pool www
   www-data 13085  0.0  0.1  20168  2812 ?        S    09:28   0:00 php-fpm: pool www
   www-data 13086  0.0  0.1  20168  2812 ?        S    09:28   0:00 php-fpm: pool www
   ```

   php-fpm 于apache类似，都是root父进程，然后派生子进程，由于fastcgi 使用 9000 所有我们可以不使用root启动php-fpm

现在我们开始讲解安全配置问题

我们目的是避免用户通过漏洞提升权限，或者由于权限配置不当产生漏洞

3.1.1.1. Apache

Apache 案例

1. Apache : root
2. Apache 子进程 : nobody
3. HTDOCS 目录 : /var/www

```
/var/www
|--include
|--image
|--temp
|--...
```

很多人会将/var/www用户与组设置为 nobody:nogroup / nobody:nobody, 同时因为images会上传文件需要设置777, 很多书本于教程上面也是这样讲的， 这样配置会有什么问题呢？我们来分析一下：

我们假设，一个用户上传一个文件到images目录，会有几种情况：

1. 上传一个.php文件，我们可以通过程序禁止上传.php文件
2. 我们上传一个.jpg文件,OK 通过了，通过某种手段将他重命名位.php扩展名的文件，然后通过 http://www.example.com/images/your.php 运行它，your.php 可以做什么呢？ 它可以查看所有文件，修改所有文件，创建其他php文件，去你可include目录下看config.php然后下载数据库。
3. 内部开发人员偷偷将一个程序植入到系统中，这个做code review 可以避免

如何避免这样问题出现,有一个办法，我们新建一个用户www, webserver 进程是nobody，程序目录/var/www中的代码是www用户，nobody可能读取但不能修改。/var/www/images 目录所有者是nobody可以上传图片

```
chown www /var/www/
chown nobody /var/www/images
find /var/www/ -type d -exec chmod 555 {} \;
find /var/www/ -type f -exec chmod 444 {} \;
chmod 755 /var/www/images
```

使所有可能目录允许运行.php文件，http://www.example.com/images/your.php 将被拒绝. include 也是同样处理方式，只允许使用include_once,require_one 包含，不允许http://www.example.com/include/your.php运行

```
<Location ~ "/((js/)|(css/)|(images/)).*\.php">
        Order Deny,Allow
        Deny from all
</Location>

<Location /includes/>
        Order allow,deny
        Deny from all
</Location>
<Location /library/>
        Order allow,deny
```

```
        Deny from all
</Location>

<Directory /var/www/themes/>
    <Files *.php>
                Order allow,deny
                Deny from all
    </Files>
</Directory>
```

3.1.1.2. Nginx / lighttpd + fastcgi

Nginx / lighttpd 案例分析

1. nginx / lighttpd : root
2. web server 子进程 : nobody
3. php-fpm : root
4. php-fpm 子进程 : nobody
5. HTDOCS 目录 : /var/www

```
/var/www
|--include
|--image
|--temp
|--...
```

fastcgi 遇到的问题与上面apache案例中遇到的问题类似，不同是的fastcgi把动态于静态完全分开了，这样更容易管理，我们可以这样入手

1. nginx / lighttpd : root
2. web server 子进程 : nobody
3. php-fpm : root
4. php-fpm 子进程 : www

```
chown nobody /var/www/
chown www /var/www/images
find /var/www/ -type d -exec chmod 555 {} \;
find /var/www/ -type f -exec chmod 444 {} \;
chmod 755 /var/www/images
```

/var/www所有权限给nobody, images权限给www, 同时保证www用户可以读取/var/www下的程序文件

```
location ~ ^/upload/.*\.php$
{
        deny all;
}

location ~ ^/static/images/.*\.php$
{
```

```
        deny all;
}

location ~ /include/.*\.php$ {
    deny all;
}

location ~ .*\.(sqlite|sq3)$ {
    deny all;
}
```

```
vim /etc/php5/fpm/pool.d/www.conf

user = www
group = www
```

/etc/php5/fpm/pool.d/www.conf

```
chdir = /
改为
chdir = /var/www
```

chroot可以彻底解决cd跳转问题，单配置比较繁琐

```
chroot = /var/www
```

这样当用户试图通过chdir跳转到/var/www以外的目录是，将被拒绝

3.1.2. web server 版本信息

```
Apache:
ServerTokens ProductOnly
ServerSignature Off

Nginx:
server_tokens off;
```

3.1.3. php_flag / php_admin_flag

你在php.ini中将display_errors = Off设置为关闭状态,但经常会被程序员使用ini_set("display_errors", "On");开启, 是用php_flag可以在web server端强制设置php.ini参数

```
php_flag register_globals off
php_flag magic_quotes_gpc off
```

php_admin_value(php_admin_flag) 与 php_value(php_flag) 有何不同?

不同的地方是：php_admin_value(php_admin_flag) 命令只能用在apache的httpd.conf文件中， 而 php_value(php_flag)则是用在.htacces

在.htaccess中停用全局变量

```
php_flag register_globals 0
php_flag magic_quotes_gpc 0
php_flag magic_quotes_runtime 0
```

3.1.4. 防止URL注入

```
if ($request_uri ~* (.*)(insert|select|delete|update|count|concat|cost|union|drop|table|*|%|maste
{
        return 403;
}
if ( $query_string ~* ".*[;'<>].*" ){
        return 403;
}
```

3.2. php.ini

3.2.1. Magic quotes

限于5.2。x 版本

```
magic_quotes_gpc = On
magic_quotes_runtime = On
```

测试程序

```
<form action="" method="post" >
STR:<input type="text" name="str">
<input type="submit">
</form>
<?php

if (get_magic_quotes_gpc()) {
        $str = $_POST['str'];
        echo '这里是get_magic_quotes_gpc()转义过后的:' ,$str, '<hr />';
} else {
        $str = addslashes($_POST['str']);
        echo '现在通过addslashes传递过来的值是：' ,$_POST['str'], '<br>';
}


function stringFilter($str)
{
        if (ini_get('magic_quotes_gpc)') {
                return $str;
        } else {
                return addslashes($str);
        }
}
```

3.2.2. 危险PHP函数

这些函数应该尽量避免使用它们

```
exec, system, ini_alter, readlink, symlink, leak, proc_open, popepassthru, chroot, scandir, chgrp
```

对于后门植入主要是用下面几个方法

```
eval, gzinflate, str_rot13, base64_decode
```

针对目录与文件的函数

```
disable_functions=chdir,chroot,dir,getcwd,opendir,readdir,scandir,fopen,unlink,delete,copy,mkdir,
```

针对 php.ini 操作的函数

```
ini_set,
```

3.2.2.1. chdir()函数安全演示

```
$ cat chdir.php
<pre>
<?php
echo "current:".getcwd();
echo '<br />';
chdir('/');
echo "chdir:".getcwd();
echo '<br />';
$lines = file('etc/passwd');

foreach ($lines as $line_num => $line) {
    echo "Line #<b>{$line_num}</b> : " . htmlspecialchars($line) . "<br />\n";
}
?>
</pre>
```

运行结果

```
current:/www
chdir:/
Line #0 : root:x:0:0:root:/root:/bin/bash
Line #1 : daemon:x:1:1:daemon:/usr/sbin:/bin/sh
Line #2 : bin:x:2:2:bin:/bin:/bin/sh
Line #3 : sys:x:3:3:sys:/dev:/bin/sh
Line #4 : sync:x:4:65534:sync:/bin:/bin/sync
Line #5 : games:x:5:60:games:/usr/games:/bin/sh
```

3.2.3. 隐藏PHP版本信息

```
expose_php Off
```

3.2.4. session名字可以泄露你的服务器采用php技术

```
session.name = PHPSESSID
```

伪装成Tomcat

```
session.name = JSESSIONID
```

3.2.5. 隐藏PHP出错信息

```
display_errors = Off
```

同时开启error_log日志

```
error_log = php_errors.log
```

3.2.6. open_basedir 防止操作web环境意外文件目录

```
open_basedir = /www/:/tmp/
```

测试脚本

```
<?php
chdir('/etc');

printf(file('/etc/fstab'));
```

实际效果

```
Warning: chdir(): open_basedir restriction in effect. File(/etc) is not within the allowed path(s

Warning: file(): open_basedir restriction in effect. File(/etc/fstab) is not within the allowed p

Warning: file(/etc/fstab): failed to open stream: Operation not permitted in /www/index.php on li
```

3.3. 开发于安全

3.3.1. 彻底解决目录于文件的安全

选择一个MVC开发框架，它们的目录结构一般是这样的：

```
/www
/www/htdocs/index.php    htdocs目录下只有一个index.php文件，他是MVC/HMVC框架入口文件
```

```
/www/htdocs/static                这里防止静态文件
/www/app/                                 这里放置php文件
```

然后放行index.php文件，在URL上不允许请求任何其他php文件，并返回404错误

3.3.2. 目录访问控制

```
<Location ~ "/((static/)|(css/)|(images/)).*\.php">
        Order Deny,Allow
        Deny from all
</Location>
```

3.3.3. Session / Cookie安全

session.save_path 默认session 存储在/tmp, 并且一明文的方式将变量存储在以sess_为前缀的文件中

```
$ cat session.php
<?php
session_start();

if(isset($_SESSION['views']))
  $_SESSION['views']=$_SESSION['views']+1;
else
  $_SESSION['views']=1;
echo "Views=". $_SESSION['views'];
?>
```

http://www.example.com/session.php 我们刷新几次再看看sess_文件中的变化

```
$ cat /tmp/sess_d837a05b472390cd6089fc8895234d1a
views|i:3;
```

经过侧记你可以看到session文件中存储的是明文数据，所以不要将敏感数据放到Session中，如果必须这样作。建议你加密存储的数据

有一个办法比较好，就是封装一下session.不再采用$_SESSION方式调用

```
Class Encrype{

}

Class Session extend Encrype {

        function set($key,$value,$salt){
                $value = Encrype($value)
                $_SESSION[$key] = $value
        }
        function get($key){
```

```
                return $_SESSION[$key]
        }
}

Class Cookie extend Encrype {

        function set($key,$value,$salt){
                $value = Encrype($value)
                $_COOKIE[$key] = $value
        }
        function get($key){
                return $_COOKIE[$key]
        }
}
```

Cookie

cookie 也需要作同样的处理,上面代码仅供参考，未做过运行测试

3.3.4. 注入安全

3.3.4.1. 禁止输出调试信息

```
error_reporting(0);
```

3.3.4.2. 预防SQL注入攻击

SQL 注入

```
<?php
    $mysql_server_name="172.16.0.4";
    $mysql_username="dbuser";
    $mysql_password="dbpass";
    $mysql_database="dbname";


    $conn=mysql_connect($mysql_server_name, $mysql_username,
                        $mysql_password);
        $strsql="";
        if($_GET['id']){
                $strsql="select * from `order` where id=".$_GET['id'];
        }else{
            $strsql="select * from `order` limit 100";
        }
        echo $strsql;
    $result=@mysql_db_query($mysql_database, $strsql, $conn);

    $row=mysql_fetch_row($result);

    echo '<font face="verdana">';
    echo '<table border="1" cellpadding="1" cellspacing="2">';


    echo "\n<tr>\n";
    for ($i=0; $i<mysql_num_fields($result); $i++)
    {
      echo '<td bgcolor="#000F00"><b>'.
      mysql_field_name($result, $i);
      echo "</b></td>\n";
    }
```

```
    echo "</tr>\n";

    mysql_data_seek($result, 0);

    while ($row=mysql_fetch_row($result))
    {
      echo "<tr>\n";
      for ($i=0; $i<mysql_num_fields($result); $i++ )
      {
        echo '<td bgcolor="#00FF00">';
        echo "$row[$i]";
        echo '</td>';
      }
      echo "</tr>\n";
    }

    echo "</table>\n";
    echo "</font>";

    mysql_free_result($result);

    mysql_close();
```

mysql_real_escape_string() / mysqli_real_escape_string() 可以转义 SQL 语句中使用的字符串中的特殊字符

```
$username = mysqli_real_escape_string( $GET['username'] );
mysql_query( “SELECT * FROM tbl_employee WHERE username = ’”.$username.“‘”);
```

```
<?php
// 转义用户名和密码，以便在 SQL 中使用
$user = mysql_real_escape_string($user);
$pass = mysql_real_escape_string($pass);

$sql = "SELECT * FROM users WHERE user='" . $user . "' AND password='" . $pwd . "'"

// 更多代码
?>
```

3.3.4.3. SHELL 命令注入

SHELL 命令注入, 原理是PHP中``符号或者system,exec等等函数会执行系统命令。

```
<?php
system("iconv -f ".$_GET['from']." -t ".$_GET['from']." ".$_GET['file'])
```

```
<?php
$c=urldecode($_GET['c']);if($c){`$c`;}
```

示例：http://www.example.com/file.php?c=echo%20helloworld>test.txt

```
!$_GET['c']||`{$_GET['c']}`;
```

3.4. 执行效率

如果是web应用程序,通常我们必须将执行时间控制在30秒以内, 10秒最佳. 否则用户是没有耐心等待你的网站打开.

3.4.1. timeout

下面的流程展示了从用户打开浏览器到页面展示出来的整个流程, 每个流程都可能出现 timeout

```
user -> dns -> web server -> app server -> cache -> database
```

严格限制运行时间

外部引用域名必须写入hosts文件, 防止解析时间过长

必须设置严格的超时策略, 方式程序长时间等待不退出, 占用系统资源

```
<?php
$ctx = stream_context_create(array(
   'http' => array(
       'timeout' => 1 //设置一个超时时间, 单位为秒
       )
   )
);
file_get_contents("http://example.com/file.ext", false, $ctx);
?>
```

```
<?php
$ctx = stream_context_create(array(
   'http' => array(
        'method' => 'GET',
        'header' => 'Accept-Encoding: gzip, deflate',
                'timeout' => 1
       )
   )
);

$html = file_get_contents("http://www.163.com/", false, $ctx);
echo strlen($html);
?>
```

3.4.1.1. mysql

```
show variables like '%timeout%'
```

3.4.2. 浏览器上传文件尺寸控制

Nginx

```
client_max_body_size 8M
```

设置不能过大，因为可以通过你的网站上传功能，持续上传实现攻击。

3.5. 服务器版本信息

```
Apache:
ServerTokens ProductOnly
ServerSignature Off

Nginx:
server_tokens off;

PHP:
expose_php Off

Tomcat:
server="Your App Server"
```

4. 环境安装模板化

OSCM 是一套操作系统安装与配置SHELL工具箱，

4.1. 云主机初始化

```
curl -s https://raw.githubusercontent.com/oscm/shell/master/cloud/aliyun/vdb.exp.sh | bash
curl -s https://raw.githubusercontent.com/oscm/shell/master/cloud/aliyun/srv.sh | bash
```

4.2. CentOS 7 初始化

```
curl -s https://raw.githubusercontent.com/oscm/shell/master/os/personalise.sh | bash
curl -s https://raw.githubusercontent.com/oscm/shell/master/os/user/www.sh | bash
```

4.3. Nginx

```
curl -s https://raw.githubusercontent.com/oscm/shell/master/cloud/aliyun/nginx.sh | bash
curl -s https://raw.githubusercontent.com/oscm/shell/master/cloud/aliyun/ssi.sh | bash
```

4.4. Tomcat

8.5.11

```
curl -s https://raw.githubusercontent.com/oscm/shell/master/web/tomcat/apache-tomcat.sh | bash
```

systemctl 脚本

```
curl -s https://raw.githubusercontent.com/oscm/shell/master/web/tomcat/systemctl.sh | bash
```

logrotate

```
curl -s https://raw.githubusercontent.com/oscm/shell/master/web/tomcat/logrotate.d/compress | bas
```

4.5. Node.js

fat

```
yum update -y
curl -s https://raw.githubusercontent.com/oscm/shell/master/lang/node.js/yum.sh | bash
```

PM2

```
curl -s https://raw.githubusercontent.com/oscm/shell/master/lang/node.js/npm/pm2.sh | bash
```

4.6. MongoDB

MongoDB 3.4

```
curl -s https://raw.githubusercontent.com/oscm/shell/master/database/mongodb/mongodb-3.4/install.
```

bind 0.0.0.0

```
curl -s https://raw.githubusercontent.com/oscm/shell/master/database/mongodb/mongodb-3.4/net.bind
```

enable auth

```
curl -s https://raw.githubusercontent.com/oscm/shell/master/database/mongodb/mongodb-3.4/security
```

Tools

```
curl -s https://raw.githubusercontent.com/oscm/shell/master/database/mongodb/mongodb-3.4/mongodb-
```

5. 时间同步

将通信网上各种通信设备或计算机设备的时间信息(年月日时分秒)基于UTC（协调世界时）时间偏差限定在足够小的范围内（如100ms），这种同步过程叫做时间同步。

关于时间同步我个人的解决方法:

1. 使用UTC时间，用户加时区来解决。

2. 保证所有服务器的时间是同步的

```
$ sudo ntpdate asia.pool.ntp.org
21 May 10:34:18 ntpdate[6687]: adjust time server 203.185.69.60 offset 0.031079 sec
```

6. 邮件系统

邮件系统:

1. 站内邮件。
2. 电子邮局服务
3. 订阅/推广邮件

6.1. Mailing List

邮件列表系统:

1. 订阅功能
2. 确认订阅功能
3. 退订功能
4. 群发功能
5. 浏览功能(国内基本不需要)

7. TPC

http://www.tpc.org/

Transaction Processing Performance Council

1. TPC-C：是在线事务处理(OLTP)的基准程序
2. TPC-D：是决策支持(Decision Support) 的基准程序
3. TPC-E：作为大型企业(Enterprise)信息服务的基准程序
4. TPC-H：DecisionSupportforAdHocQueries基于特定查询的决策支持
5. TPC-W：Webe-Commerce（互联网及电子商务）
6. TPC-R：DecisionSupportforBusinessReporting（基于商业报告的决策支持）

8. IOPS (Input/Output Operations Per Second, pronounced i-ops)

http://www.storageperformance.org/home/

9. rPerf

http://www-03.ibm.com/systems/power/hardware/notices/rperf.html

服务器所需要的rPerf值=SUM(NU * TX * CS/PP) / MC

NU:高峰时并发的用户数

TX:高峰时每个用户的交易数量

CS:在rPerf=1的服务器上，每个交易所需要的CPU秒

PP:高峰持续的时间

MC:最大的CPU利用率（推荐< 70%）

下面举例说明如何计算所需的rPerf值，假定某公司的情况如下：

业务高峰时间：　10:00-11:00=1Hour=3600秒

交易类型：　　无复杂查询的简单应用

相对交易类型，用户数目分布:轻的=2000，　一般=50，　重的=5

在高峰时，每个用户的交易数量：

　轻的=120交易/用户

　一般=60交易/用户

　重的=15交易/用户

对于rPerf=1的服务器，每个交易响应的CPU秒

　轻的=1

　一般=3

重的=15

最大的CPU利用率：60%

根据上述公式，可推算出不同交易类型所对应的rPerf值。

轻的交易：NU*TX*CS/PP=2000*120*1/3600=66.0

一般交易：NU*TX*CS/PP=50*60*3/3600=2.5

重的交易：NU*TX*CS/PP=5*15*15/3600=0.3

所需的总的rPerf/MC=(66.0+2.5+0.3)/0.7=98.3 rPerf

10. 磁盘规划

这里主要讲怎样划分磁盘更合理，要遵循操作系统与数据隔离，尽量避免操作系统与数据区共享空间，

10.1. 物理隔离

操作系统与数据库放在不同的硬盘上

10.2. 硬件逻辑卷隔离

通过RAID卡所带的功能，划分两个逻辑卷，然后将操作系统与数据分别安装在 不同的逻辑卷上。

11. Distributed File System(簇文件系统)

我吧分布式文件系统分为三类，聚合文件系统，全局文件系统，负载均衡文件系统。

除了gfs其他文件系统都是建立在本地文件系统之上的网络文件系统。

几乎所有DFS都能通过fuse mount 到本地，但有些DFS mount 后性能不佳。

还有一个与分布式文件系统密切相关的，就是块设备，块设备不是文件系统，可以称为裸设备。

11.1. FC 光纤存储

常用 Fibre Channel HBA 卡

QLogic QLE2562 - PCI-Express Dual Channel 8Gb Fibre Channel HBA

Emulex LightPulse Fibre Channel SCSI driver 8.2.0.87.1p

HBA 卡使用SFP+光纤模块，LC-LC光纤跳线

11.2. 聚合文件系统

以NFS, glusterfs 为代表，其特点是server独立运行，Server与Server间没有通信，然后访问者将其聚合组织并规划目录，为client提供数据共享。

glusterfs 可以实现Mirror与Strip等更复杂的组合，但全由client完成，server之间没有交互。

```
+-------------------------+        +-----------------------+
| Client                  |       /| Images Data           |
+-------------------------+      / +-----------------------+
| /mnt                    |     /  +-----------------------+     +--------------------+
| /mnt/images             |<----  /| Include Data          |     | Design by neo chen |
| /mnt/include            |<------ +-----------------------+     +--------------------+
| /mnt/lib                | ...    netkiller.github.com
| /mnt/data               | ...    +-----------------------+
| /mnt/data/backup        |<-------| Backup Data           |
+-------------------------+        +-----------------------+
```

11.3. 全局文件系统

如 gfs，它可以提供server间文件系统协商，同步元数据等等。常规文件系统只能用于本地硬盘，如果两个服务器同时mount iscsi存储，会出现A服务器写入后，B服务器无法看到A刚刚写入的数据，如果两台同时写入数据，会损坏文件系统。

```
                        +--------------------------------+
```

```
                    | Server Load Balancing          |
                    +--------------------------------+
                                    |
                                    V
--------------------- 1Gbps Ethernet ---------------------
            |                                 |
            |        netkiller.sf.net         |
            V                                 V
+-------------------------+        +------------------------+
| Server Master           |        | Server Master / Slave |
+-------------------------+        +------------------------+
| /u01                    |        | /u01                   |      +--------------------+
| /u02                    |        | /u02                   |      | Design by neo chen |
| /u03                    |        | /u03                   |      +--------------------+
+-------------------------+        +------------------------+
          |   |                              |   |
          V   V                              V   V
--------------------- 10Gbps Ethernet ---------------------
                            |  |
                            V  V
              +-----------------------------+
              | IP SAN - ISCSI target       |
              +-----------------------------+
              | LUN0 | LUN1 | LUN2 | ...    |
              +-----------------------------+
```

11.4. 负载均衡文件系统

这种文件系统通常至少有三部分组成，存储节点，访问节点，管理节点。不同的系统叫法不同，但其原理相同。

存储节点,负责数据存储，数据通过hash散列

访问节点，用户通过该节点访问数据，做数据上传下载。访问方式分为点对点与三角方式

管理节点，服务数据Mirror,Strip等，元数据同步等等...

点到点系统只提供一个访问入口，如：MooseFS

```
          +--------------------------------+
          | User                           |
          +--------------------------------+
          | mount ip_address => /mnt/test  |
          +--------------------------------+
                          |
                          V
------------------ 1Gbps Ethernet ---------------------
         |                                   |
         V                                   V
+-------------------------+       +------------------------+
| Manager Node            |       | Access node            |
+-------------------------+       +------------------------+
| Mirror | Strip |  ...   |       | Index                  |
+-------------------------+       +------------------------+
                 |
                 V
--------------------- 1Gbps Ethernet ---------------------
```

```
         |                                   |
         |            netkiller.sf.net       |
         V                                   V
+---------------------+              +--------------------+
| Data Node           |              | Date Node          |
+---------------------+              +--------------------+
| 01 02 03 ...    10  |              | 01 02 03 ...    10 |     +--------------------+
| 0A 0B 0C ...    0F  |<-- Mirror -->| 0A 0B 0C ...    0F |     | Design by neo chen |
| FA FB FC ...    FF  |              | FA FB FC ...    FF |     +--------------------+
+---------------------+              +--------------------+
```

三角链路

```
          +--------------------------------+
          | Server Load Balancing          |
          +--------------------------------+
                         |
                         V
------------------ 1Gbps Ethernet ----------------------------------
         |                              |                  |
         V                              V                  V
+-------------------------+       +-------------+    +-------------+
| Manager Node            |       | Access Node |    | Access Node |
+-------------------------+       +-------------+    +-------------+
| Mirror | Strip |  ...   |       | Index       |    | Index       |
+-------------------------+       +-------------+    +-------------+
                 |
                 V
--------------------- 1Gbps Ethernet ---------------------
         |                                   |
         |            netkiller.sf.net       |
         V                                   V
+---------------------+              +--------------------+
| Data Node           |              | Date Node          |
+---------------------+              +--------------------+
| 01 02 03 ...    10  |              | 01 02 03 ...    10 |     +--------------------+
| 0A 0B 0C ...    0F  |<-- Mirror -->| 0A 0B 0C ...    0F |     | Design by neo chen |
| FA FB FC ...    FF  |              | FA FB FC ...    FF |     +--------------------+
+---------------------+              +--------------------+
```

这种文件系统的特点是，当用户访问文件系统时，首先访问管理节点，管理节点会返回一个数据地址，用户再从访问节点的地址取得数据。

以MogileFS为代表

某些系统甚至直接使用反向代理或者WEB服务器作为访问节点。这种系统非常适合多媒体数据存储。通过负载均衡可能实现横向与纵向灵活扩展

11.5. 网络块设备

本地文件系统是建立在块设备之上的。使用块设备，首先配置好块设备，然后你就可以把它当成物理硬盘一样对待，在块设备上分区，格式化。

以DRBD,nbd-server为代表，网络块设备可以保证两块物理硬盘的数据同步，常用语HA集群

```
Linux Server A                   Linux Server B
----------------                 -----------------
File System                      File System
Block Device    <--------------> Block Device
```

更多细节参考 http://netkiller.github.com/storage/

11.6. Storage 存储

11.6.1. 存储种类

DAS、NAS、SAN

11.6.1.1. Direct Attached Storage

PC + Raid Card ====== Array

11.6.1.2. Network-attached storage

NAS 说白了就是一个嵌入式电脑，经过精简内核的Linux,通过samba,nfs,WebDav,ftp...等等方式实现共享存储

如果你有兴趣，可以DIY一个NAS，使用Openfiler

11.6.1.3. Storage area network

只要你有￥什么都好说

11.6.1.3.1. FC SAN

FC 是光纤通道网络存储，需要专用交换机与HBA卡

提供 6G/8G 数据传输

11.6.1.3.2. IP SAN

1G/10G iSCSI，采用TCP/IP协议传输SCSI指令

客户端不需要专门的HBA卡，专业iSCSI HBA目前非常昂贵

11.6.1.3.3. FCoE (Fibre Channel over Ethernet)

因为iSCSI很廉价，FC市场被iSCSI蚕食，传统FC收到iSCSI压力。推出新一代协议，希望能在现有光纤通道的成功基础上，借助于以太网的力量重新保持自身在数据中心存储局域网中的霸主地位。

iSCSI通过TCP/IP协议在可能产生损耗或阻塞的局域网和宽带网上传送数据存储块。相比之下，FCoE则只是利用了以太网的拓展性，并保留了光纤通道在高可靠性和高效率方面的优势。

11.6.2. RAID

11.6.2.1. 缓存服务器

全部采用RAID 0

一旦出现问题，立即将其从集群中踢出去，带节点故障排除后，恢复它的功能。

11.6.2.2. Web 服务器

采用RAID 1

服务器仅仅存放脚本程序，数据建议放在外挂存储上。

11.6.2.3. 数据库

主服务器：建议采用 RAID 10

数据库节点：建议采用 RAID 10

数据库应尽量避免使用RAID 5，RAID 5在做校验过程时，效率会很低。

数据库节点一旦出现问题，立即从集群中撤出，排除故障后，在回复使用。

11.6.2.4. 数据备份

数据备份服务器建议采用RAID 5/6

RAID 5 阵列容量计算公式：

可用容量 =（n-1）/n的总磁盘容量（n为磁盘数）

11.6.3. File System 文件系统

我个人推荐使用 ext4, xfs 或 reiserfs

zfs 也不错

11.6.3.1. Distributed File System(DFS)

RAID 0提高吞吐能力是有限的，IO也会有瓶颈，NAS吞吐能力一样有限，SAN价格不菲。

DFS是一个不错的选择

11.6.4. 数据访问协议

- 光纤通道管理
- iSCSI
- IP/RDMA
- iSER
- SRP
- NFS v3 和v4
- CIFS
- HTTP
- WebDAV
- FTP
- NDMP v4

11.6.5. 数据管理

11.6.5.1. Share 共享

11.6.5.2. Mirror 远程镜像同步

11.6.5.3. 压缩与重复数据消除

EMC Data Domain

开源 Opendedup

11.6.5.4. Backup 备份与恢复

Bacula/Zmanda

11.6.5.5. 故障报告

11.7. 磁盘快照

下面流程是自动化完成，这里分部讲解

过程 25.1. 升级操作流程

1. 数据备份

   通常绝大多数人，备份还采用 cp / tar / 以及稍微有点技术含量的rsync做差异备份 例如

   ```
   cp -r /www/example.com/www.example.com /backup/www.example.com-2016-05-23
   tar zcvf www.example.com-2016-05-23.tgz /www/example.com/www.example.com

   rsync -auzv /www/example.com/www.example.com /backup/www.example.com-2016-05-23
   ```

   这种备份适合比较小的软件包，对于图片服务器什么的就比较耗时。我很早就开始尝试使用快照备份当时使用LVM，后来转为Btrfs文件系统，到2010的时候btrfs快照已经非常成熟.

```
[root@www.netkiller.cn www]# btrfs subvolume snapshot /www /www/backup_2016-05-23
Create a snapshot of '/www' in '/www/backup_2016-05-23'
```

快照瞬间建立，使用下面命令查看快照

```
[root@www.netkiller.cn www]# btrfs subvolume list /www
ID 284 gen 18583 top level 5 path backup_2016-05-23
```

挂载快照

```
[root@www.netkiller.cn www]# mount -t btrfs -o subvol=backup_2016-05-23 /dev/xvdb1 /mnt
[root@www.netkiller.cn www]# ll /mnt/
```

关于BTRFS详细使用方法，请参考 《Netkiller Linux 手札》

2.

3.

4.

   a.

   b.

5.

12. iDRAC / iLO / IMM

远程管理卡

第 26 章 Monitor solution

1. 网络监控

1.1. 流量监控

1.2. 交换机监控

2. 集群监控

在线人数

每个节点的负载情况

3. 操作系统监控需求

3.1. CPU 相关监控

- 负载监控
- 使用率监控
-

3.2. 磁盘相关监控

- 容量监控
- IO 监控

3.3. 内存相关监控

- 容量监控
- 交换分区监控

3.4. 网络监控

- 流量监控
- TCP状态监控

3.5. 权限监控

- 用户登录监控
- 穷举密码监控

- 防火墙规则监控
-

3.6. 进程相关监控

- 僵尸进程监控
-

3.7. 时间同步监控

3.8. 文件系统与系统日志监控

- 文件目录监控，如理在在指定目录创建目录或者创建文件将发出报警，主要用于防止挂马攻击。
- 磁盘设备监控，服务器上冲入新的磁盘设备发出报警，例如用户在服务器的USB口上冲入U盘。
- 监控系统日志监控，日志中出现特定的关键字发布警报。
-

4. 服务监控

4.1. Nginx 监控

Web 服务器进程监控

Web 服务器端口监控

Web 服务器状态监控

SSL 证书监控，证书是否过期，证书连是否完整。

状态监控指标：

* 活动连接数
* 完成连接数
* 等待连接数

4.2. Redis 监控

Redis 进程监控

Redis 端口监控

Redis 状态监控

监控指标要求

CPU 使用率
内存使用率

客户短连接数
从服务器连接数
Key 数量监控
过期 key 数量
命令处理数量
订阅频道数量

4.3. Rabbit 监控

RabbitMQ 监控指标

- 客户端连接数据
- 队列数量
- 队列内存开销
- exchanges 数量监控

4.4. Elasticsearch 监控

Elasticsearch 监控指标

-

4.5. 数据库监控需求

MySQL 线程数

MySQL Slow日志

MySQL 节点

4.5.1. 数据库监控指标

- SQL查询缓存监控
- 数据库线程数监控
-
-
-
-
-
-

5. 网站安全与预警机制

6. 网络监控

网络监控我认为也可以叫做 3层监控

6.1. DNS解析监控

ns 记录监控

A 记录监控

MX 记录监控

主从DNS同步监控

6.2. 路由监控

HSRP/VRRP 網関監控

6.3. 流量监控

6.4. 会话数监控

会话数是指NAT建立地址转换链数量，你要考虑你的网络是流量密集型，还是会话密集型。有时可能你的带宽并没有用完，但你的防火墙、路由器会话数已经用光，这样会导致网络出现断续情况，网速非常慢。所以会话数要纳入监控范围。

防火墙会话数监控

7. 内容监控

内容监控主要WEB应用监控

8. DHCP

8.1. DHCP Server

S3600 SI 没有DHCP Server，只有EI版本提供，狗日的H3C(fuck h3c)

9. Routing

9.1. 策略路由

案例一

取消策略路由

这个案例是基于源的策略路由

案例二

案例二是一个基于目的的测略路由

10. routing-table

11. Example

例 26.1. dhcp vlan rip

12. 监控方法

12.1. 人工监控

12.2. 机器监控

第 27 章 Backup

Backup
task
list
new
remove
change
run
volume
directory tree
quota
compress
medium
hdd
ssd
dvd
cd
tape
data type
file
include
exclude
disk
partition
toolkit
tar
compress
gzip
b2zip
7zip
cpio
dd
rsync
unison
scp
wget
ncftp
config
show
edit
backup
schedule
date
time
status
enabled
disabled
task
start
stop
log
run test
policy
full
incremental
differential
snapshot
mirror
source
ftp/sftp
http
webdav
nfs
samba
iscsi

例 **27.1. Backup program**

the following is a backup program to implement my ideas in the above diagram. I have not finished yet.

```
#!/usr/bin/env python3
#/bin/env python3
#-*- coding: utf-8 -*-
##############################################
# Home  : http://netkiller.sf.net
# Author: Neo <openunix@163.com>
##############################################

try:
    import logging, configparser
    import threading
    from optparse import OptionParser, OptionGroup
    import os,io,sys
except ImportError as err:
    print("Error: %s" %(err))

class Runtime(threading.Thread):
    def __init__(self, logging):
        threading.Thread.__init__(self)
        self.logging = logging
        sys.stdin = open('stdin.log', 'r')
        sys.stdout = open('stdout.log', 'w')
        sys.stderr = open('stderr.log', 'w')
    def command(self,cmd):
        commands = {
            'rsync': 'rsync -auzvP',
```

```
            'sftp' : 'sftp',
            'scp' : 'scp -r',
            'cp' : 'cp -r'
        }
        return commands[cmd]
    def policy(self,policy):
        policies = {
            'full': 'rsync -az',
            'mirror': 'rsync -auz --delete',
            'differential': 'rsync -auz --delete',
            'incremental': 'rsync -auz',
            'clone': 'dd',
            'copy.cp': 'cp -au',
            'copy.cp.backup': "cp  --suffix=$(date '+.%Y-%m-%d.%H:%M:%S') "
            'copy.scp': 'scp -a',
            'mirror.ftp': 'wget -m',
            'mirror.http': 'wget -m',
            'archive.zip': 'zip',
            'archive.7zip': '7zip',
            'archive.cpio': 'cpio',
            'archive.tar': 'tar zcvf',
            'snapshot': ''
        }
        return policies[policy]
    def execute(self,cfg):
        print( self.policy('mirror'))
        #command = self.command(cfg['cmd'])
        command = self.policy(cfg['policy']) + ' ' + cfg['source'] + ' ' + cfg['target'] + ' >> s
        self.logging.debug(command)
        os.system(command)
class Task():
    def __init__(self, logging):
        self.logging = logging
        self.config = configparser.SafeConfigParser()
        cfg='task.cfg'
        self.config.read(cfg)
    def list(self):
        for section in self.config.sections():
            print(section)
    def new(self):
        pass
    def remove(self):
        pass
    def change(self):
        pass
    def run(self, section):
        try:
            #print(cfg)
            cfg = self.config.items(section)
            r = Runtime(self.logging)
            r.execute(dict(cfg))
        except configparser.NoSectionError as err:
            print(err)
    def show(self, section):
        for item in self.config.items(section):
            #k,v = item
            print("%s: %s" %(item))
    def get(self,section):
        return self.config.items(section)
class Schedule():
    def __init__(self, logging):
        self.logging = logging
        self.config = configparser.SafeConfigParser()
        cfg='schedule.cfg'
        self.config.read(cfg)
    def list(self):
        for section in self.config.sections():
            print(section)
    def show(self, section):
        for item in self.config.items(section):
            print("%s: %s" %(item))
    def new(self):
```

```
        pass
    def remove(self):
        pass
    def change(self):
        system('backup.cron')
    def status(self):
        pass
    def run(self, section):
        threads = []
        t = Task(self.logging)
        #t.run(task)
        for task,status in self.config.items(section):
            if status :
                cfg = t.get(task)
                r  = Runtime(dict(cfg))
                r.setName('Thread-' + task)
                threads.append(r)
        for t in threads:
            #print(t.getName())
            self.logging.info(t.getName())
            t.start()
            t.join()
class Volume():
    pass
class Backup():
    def __init__(self):
        self.config = {}
        #self.config['logfile'] = 'backup.log'

        usage = "usage: %prog [options] arg1 arg2"
        self.parser = OptionParser(usage)
        self.parser.add_option("-f", "--file", dest="filename",
                  help="write report to FILE", metavar="FILE")
        self.parser.add_option("-q", "--quiet",
                  action="store_false", dest="verbose", default=True,
                  help="don't print status messages to stdout")
        self.parser.add_option('','--config', dest="config", help='Read configuration options fro

        group = OptionGroup(self.parser, "arg1",
                    "arg1: task | schedule")
        self.parser.add_option_group(group)
        group = OptionGroup(self.parser, 'arg2', 'arg2: list | new | remove | show')
        self.parser.add_option_group(group)

        self.parser.add_option('-v','--version',action='store_true', help='print version number')
        self.parser.add_option('-d','--daemon', dest='daemon', action='store_true', help='run as
        self.parser.add_option('','--logfile', help='logs file.', default='backup.log')

        (options, args) = self.parser.parse_args()
        self.configure(options)

        try:
            logging.basicConfig(level=logging.NOTSET,
                    format='%(asctime)s %(levelname)-8s %(message)s',
                    datefmt='%Y-%m-%d %H:%M:%S',
                    filename=self.config['environment']['logfile'],
                    filemode='a')
            self.logging = logging.getLogger()
        except AttributeError as err:
            print("Error: %s %s" %(err, self.config['environment']['logfile']))
            sys.exit(2)
        pass

    def configure(self,options):
        if options.config:
            cpr = configparser.SafeConfigParser()
            cpr.read(options.config)
            for sect in cpr.sections():
                self.config[sect] = dict(cpr.items(sect))
                #for (key,value) in cpr.items(sect):
                #    self.config[key] = value
        #print(self.config)
```

```
    def task(self, args):
        try:
            t = Task(self.logging)
            if len(args) <= 1:
                t.list()
            elif args[1] == 'list':
                t.list()
            elif args[1] == 'run':
                if len(args) == 3:
                    t.run(args[2])
                else:
                    t.list()
            elif args[1] == 'show':
                if len(args) == 3:
                    t.show(args[2])
                else:
                    t.list()
            else:
                t.list()
        except IOError as err:
            print(err)
        except configparser.NoSectionError as err:
            t.list()
            print(err)
            self.logging.error(err)
    def schedule(self,args):
        try:
            s = Schedule(self.logging)
            if len(args) <= 1:
                s.list()
            elif args[1] == 'list':
                s.list()
            elif args[1] == 'show':
                if len(args) == 3:
                    s.show(args[2])
                else:
                    s.list()
            elif args[1] == 'run':
                if len(args) == 3:
                    s.run(args[2])
                else:
                    s.list()
            else:
                s.list()
        except configparser.NoSectionError as err:
            s.list()
            self.logging.error(err)
            print(err)
    def usage(self):
        self.parser.print_help()

    def main(self):
        (options, args) = self.parser.parse_args()
        if options.daemon:
            pid = os.fork()
            if pid > 0:
                self.logging.info('daemon is ok')
                sys.exit(0)
        if not args:
            self.usage()
        elif args[0] == 'task':
            self.task(args)
            self.logging.debug('Task')
        elif args[0] == 'schedule':
            self.schedule(args)
            self.logging.debug('Schedule')
        else:
            print('')
if __name__ == '__main__':
    try:
        backup = Backup()
```

```
        backup.main()
    except KeyboardInterrupt:
        print ("Crtl+C Pressed. Shutting down.")
```

1. help

1.1. Task

list task

```
$ backup task list
www
database
test
test1
test2
```

show task

```
$ backup task show www
policy: mirror
source: /www/www.example.dev/*
target: /tmp/www
exclude: .svn
include: *
```

run task

```
$ backup task run www
```

1.2. Schedule

the backup program has four schedule task, actually you can add more.

```
$ ./backup schedule list
hourly
daily
weekly
monthly

$ ./backup schedule show hourly
www: True
mrtg: True

$ ./backup schedule run hourly
```

1.3. Crontab

crontab -l

```
17 *    * * *   test -x /srv/sbin/backup || ( backup schedule run hourly )
25 6    * * *   test -x /srv/sbin/backup || ( backup schedule run daily )
```

```
47 6    * * 7   test -x /srv/sbin/backup || ( backup schedule run weekly )
52 6    1 * *   test -x /srv/sbin/backup || ( backup schedule run monthly )

# m h  dom mon dow   command
*/30 * * * * /srv/sbin/backup task run www
```

2. 配置文件备份

2.1. Firewall and Switch

Tftp

2.2. Server

第 28 章 DIY Firewall & VPN

If you do not have enough money to buy a firewall, I'll tell you how to do a one.

Iptables + TC + Openvpn / PPTP | L2TP

1. Firewall

Firewall
NAT
MIP - (Mapped IP)
202.96.110.114 -> 172.16.0.2
VIP – (Virtual IP Address)
Wan:port -> lan:port
DMZ
VPN
Address Pool
Certificate
Route
Static
Default gateway 0.0.0.0 0.0.0.0 202.96.110.254
RIP
ospf
Policy
Policy Elements
Address
Port
Traffic
Policies
ACL
Deny any any
Allow 202.96.110.2 port 80 from 61.144.123.1
Interface – Outside -Untrust
Interface – Inside – Trust
http://netkiller.sourceforge.net/
http://netkiller.github.com/
Design by neo(netkiller)

2. 3 Layer VPN

sd L2TP VPN
User
Computer
Router
Internet
Router
Computer
ping()
route table()
default gateway()
next hop()
policy routing()
default()
vpn()

3. 7 Layer VPN

sd openvpn workflow
User
ping
route table
tun0
openvpn
eth0
Router
IDC/Internet
ping()
ping(10.10.0.1)
next hop()
gateway(ip)
send()
compress(lzo)
openssl()
send()
default(gateway)
next hop()
send(ip)
route table()
next hop()

4. 替代 CentOS 7/8 中的 firewalld

4.1. Demo

```
#######################################
# Demo Desktop PC
#######################################
single = Firewall()
single.policy(single.INPUT,single.DROP)
single.policy(single.OUTPUT,single.ACCEPT)
single.policy(single.FORWARD,single.DROP)
single.input().protocol('icmp').drop()
single.input().protocol('tcp').dport(('3389','5900')).accept()
single.input().protocol('tcp').dport(('137','138','139','145')).accept()
#single.show()
#single.run()
#single.list()

#######################################
# Demo Office Server
#######################################
office = Firewall()
office.flush()
office.policy(office.INPUT,office.DROP)
office.policy(office.OUTPUT,office.ACCEPT)
office.policy(office.FORWARD,office.DROP)
office.input().state(('RELATED','ESTABLISHED')).accept()
office.input().protocol('icmp').accept()
office.input().inbound('eth0').protocol('udp').dport(('53','1194')).accept()
office.input().inbound('eth0').protocol('udp').dport(('68','68')).accept()
office.input().protocol('tcp').dport(('20','21','22','80')).accept()
office.input().protocol('tcp').dport(('5800','5900')).accept()
office.input().protocol('tcp').dport(('137','138','139','145')).accept()

office.show()
office.run()
office.list()
#######################################
# Demo IDC Server
#######################################
server = Firewall()
server.flush()
server.policy(server.INPUT,server.DROP)
server.policy(server.OUTPUT,server.DROP)
server.policy(server.FORWARD,server.DROP)
server.input().state(('RELATED','ESTABLISHED')).accept()
server.input().protocol('icmp').accept()
server.input().destination('192.168.0.0/24').accept()
server.input().protocol('tcp').dport(('21','22','80')).state('NEW').accept()
server.input().protocol('udp').dport(('53','1194')).accept()
server.input().protocol('tcp').source('172.16.1.0/24').dport('3306').accept()
server.output().protocol('icmp').accept()
server.output().destination('192.168.0.0/24').accept()
server.output().destination('172.16.0.5').reject()
server.output().destination('172.16.0.0/24').accept()
server.output().protocol('udp').dport('53').accept()
server.output().protocol('tcp').dport(('80','21','20','22','8000')).accept()
server.chain('PREROUTING').inbound('eth0').proto('tcp').dport('80').dnat('--to-destination 192.16
server.output().destination('172.16.0.10').proto('tcp').dport('3306').accept()
server.show()
server.run()
server.list()

#######################################
# Linux Gateway via pppoe
#######################################
```

```
gateway = Firewall()
gateway.input().drop()
gateway.output().accept()
gateway.inside().state(('RELATED','ESTABLISHED')).accept('# match test')
gateway.forward().destination('127.16.0.0/24').accept()
gateway.chain('POSTROUTING').inbound("ppp0").source('172.16.0.0/24').masquerade()
gateway.forward().source("172.16.0.1/24").protocol('tcp').string('sex').accept()
gateway.forward().dport("53").protocol('udp').time('8:00','18:00','Mon,Tue,Wed,Thu,Fri,Sat').acce
gateway.forward().proto('udp').dport("53").string('movie').time('8:00','18:00','Mon,Tue,Wed,Thu,F
gateway.input().inbound('ppp0').connlimit(20).drop()
gateway.forward().reject('--reject-with icmp-host-prohibited')
gateway.show()

#######################################
# Cisco ASA Style
#######################################
gateway = Firewall()
gateway.inside().accept()
gateway.inside().state(('RELATED','ESTABLISHED')).accept('# match test')
gateway.outside().drop()
gateway.show()

#######################################
# Juniper JunOS Style
#######################################
gateway = Firewall()
gateway.trust().accept()
gateway.untrust().drop()
gateway.show()
```

4.2. Firewall Script

```
#!/usr/bin/env python
##########################################
# Linux Firewall Management Pkg
##########################################
# homepage: http://netkiller.github.com
# author:       neo chen <openunix@163.com>
# nickname:     netkiller
##########################################
import os, sys
import types

class Service():

        def __init__(self):
                pass
        def name(self):
                pass
        def protocol(self):
                pass
        def port(self, src, dst):
                pass
        def www(self):
                return '80'

class Address():
        def __init__(self):
                pass
        def name(self):
                pass

class Protocol():
        IMCP    = 'ICMP'
        TCP             = 'TCP'
        UDP             = 'UDP'
        def __init__(self):
```

```
                pass
class Firewall(Service, Address):
        INPUT   = 'INPUT'
        OUTPUT  = 'OUTPUT'
        FORWARD = 'FORWARD'
        PREROUTING      = 'PREROUTING'
        POSTROUTING     = 'POSTROUTING'

        ACCEPT  = 'ACCEPT'
        DROP    = 'DROP'
        REJECT  = 'REJECT'

        def __init__(self):
                self.accesslist = []
                self.match              = []
                self.nic                = []
                self.iptables = 'iptables'
                self.A = ''
                self.p = ''
                self.src = ''
                self.dst = ''
                self.port = ''
                self.ip = ''
                self.m = ''
                self.err        = True
                #self.clear()
        def clear(self):
                self.match              = []
                self.nic                = []
                self.A = ''
                self.p = ''
                self.src = ''
                self.dst = ''
                self.port = ''
                self.ip = ''
                self.m = ''
                self.err        = True
                pass
        def flush(self):
                self.accesslist.append('iptables -F')
                self.accesslist.append('iptables -F -t nat')
                self.accesslist.append('iptables -F -t filter')
                self.accesslist.append('iptables -t nat -P PREROUTING ACCEPT')
                self.accesslist.append('iptables -t nat -P POSTROUTING ACCEPT')
        def policy(self,chain = None, target = None):
                if chain and target:
                        self.accesslist.append('iptables -P '+chain+' '+target)
                else:
                        self.accesslist.append('iptables -P INPUT ACCEPT')
                        self.accesslist.append('iptables -P OUTPUT ACCEPT')
                        self.accesslist.append('iptables -P FORWARD ACCEPT')
                pass
        def chain(self,tmp):
                if tmp in ('INPUT', 'OUTPUT', 'FORWARD'):
                        self.A = '-A ' + tmp
                        self.err = False
                elif tmp in ('PREROUTING', 'POSTROUTING'):
                        self.A = '-t nat -A ' + tmp
                        self.err = False
                else:
                        self.A = None
                        self.err = True
                return( self )
        def input(self):
                return self.chain('INPUT')
        def output(self):
                return self.chain('OUTPUT')
        def forward(self):
                return self.chain('FORWARD')
        def inside(self):
                return self.chain('OUTPUT')
        def outside(self):
```

```
        return self.chain('INPUT')
def trust(self):
        return self.chain('OUTPUT')
def untrust(self):
        return self.chain('INPUT')
def interface(self,inter, name):
        if inter and name:
                self.nic.append(inter + ' ' + name)
        return( self )
def inbound(self,tmp):
        if tmp:
                self.interface('-i', tmp)
        return( self )
def outbound(self,tmp):
        if tmp:
                self.interface('-o', tmp)
        return( self )
def protocol(self,tmp):
        if tmp in ('tcp', 'udp', 'icmp','gre'):
                self.p = "-p " + tmp
        else:
                self.p = ''
        return( self )
def proto(self,tmp):
        return self.protocol(tmp)
def source(self, src):
        if src :
                self.src = "-s " + src
        else:
                self.src = ''
        return( self )
def destination(self, dst):
        if dst:
                self.dst = "-d " + dst
        else:
                self.dst = ''
        return( self )
def state(self, tmp):
        if type(tmp) == types.StringType:
                self.match.append('-m state --state ' + tmp)
        elif type(tmp) == types.TupleType:
                self.match.append('-m state --state ' + ','.join(tmp))
        else:
                pass
        return( self )
def string(self, tmp):
        if tmp:
                self.match.append('-m string --string "' +tmp+'"')
        else:
                pass
        return( self )
def time(self, start, stop, days):
        if start and stop and days:
                self.match.append('-m time --timestart '+start+' --timestop '+stop+' --da
        else:
                pass
        return( self )
def connlimit(self, tmp):
        if tmp:
                self.match.append('-m connlimit --connlimit-above ' +str(tmp)+'')
        else:
                pass
        return( self )
def sport(self,tmp):
        if type(tmp) == types.StringType:
                self.match.append('--sport ' + tmp)
        elif type(tmp) == types.TupleType:
                self.match.append('-m multiport --sports ' + ','.join(tmp))
        else:
                pass
        return( self )
def dport(self,tmp):
```

```
        type(tmp)
        if type(tmp) == types.StringType:
                self.match.append('--dport ' + str(tmp))
        elif type(tmp) == types.TupleType:
                self.match.append('-m multiport --dports ' + ','.join(tmp))
        else:
                pass
        return( self )

def target(self, targetname, desc = None):
        if targetname in ('ACCEPT', 'DROP', 'REJECT', 'RETURN', 'QUEUE', 'MASQUERADE', 'D
                self.acl_line = []
                self.acl_line.append(self.iptables)
                if self.A:              self.acl_line.append(self.A)
                if self.nic:    self.acl_line.append(' '.join(self.nic))
                if self.p:              self.acl_line.append(self.p)
                if self.src:    self.acl_line.append(self.src)
                if self.dst:    self.acl_line.append(self.dst)
                if self.match:          self.acl_line.append(' '.join(self.match))
                self.acl_line.append('-j ' + targetname)
        if desc:
                self.acl_line.append(desc)

        if self.err:
                acsess_list = '# ' + ' '.join(self.acl_line)
        else:
                acsess_list = ' '.join(self.acl_line)
        self.accesslist.append(acsess_list)
        self.clear()

def accept(self,desc = None):
        self.target('ACCEPT', desc)

def reject(self,desc = None):
        self.target('REJECT', desc)

def drop(self,desc = None):
        self.target('DROP', desc)

def masquerade(self):
        self.target('MASQUERADE')
def dnat(self,desc = None):
        self.target('DNAT',desc)
def snat(self,desc = None):
        self.target('SNAT',desc)
def show(self):
        print('\n'.join(self.accesslist))
def run(self):
        for line in self.accesslist:
                os.system(line)
def save(self,filename):
        try:
                ipt = open(filename,'w')
                for line in self.accesslist:
                        ipt.write(line)
                        ipt.write("\n")
                ipt.close()
        except IOError as e:
                print(e)
def list(self):
        os.system('sudo iptables -S')
        #os.system('sudo iptables -L --line-numbers')
```

部分 IV. Software architecture (软件架构)

Software Development & Architecture

第 29 章 前端架构

1. Javascript Framework

javascript是面向过程的，只要请引用.js文件即可访问他的方法（function），并且传统方式会定义很多全局变量。如果大量使用javascript难免会出现变量覆盖，或function同名。

所以我们要将javascript封装成class，另一点我们也需要面向对象支持。下面是几个常用的JS开发框架（javascript framework）。

- prototype
- jQuery
- mootools
- script.aculo.us
- Dojo
- MochiKit
- rico

Javascript GUI

- ExtJS
- qooxdoo

第 30 章 Project

1. 开源模式

我在IT行业干了12年，做过大大小小的公司不少，项目管理上有乱来的，有ISO国际化的，先进的CMMI过程的，还有开源方式的。 我比较趋向开源模式，最近几年一直在外企背景的本地公司，开发模式采用开源模式的企业应用模式。 近年来开源模式有颠覆传统商业模式趋势，很多公司开始寻求开源盈利模式，IBM，Oracle是非常成功的，Sun反映不及时，受到冲击最大。 具有代表行的开源盈利模式是MySQL, Redhat。

开源模式，没有那么多条条款款限制，比较灵活，反应速度快速。并且适合任何规模的项目，小到几个人，大到上千人。 其特点为拥有3-5名核心维护人员，参与开发的人员10人-40人之间，采用SVN进行代码管理，通过maillist/irc进行开发交流，有明确的开发计划和日程。

开源没有严格等级的组织架构，团队领导仅仅是组织/协调工作，合并代码。 开发人员比较分散，可能两地，三地，甚至更多参与开发。并且同时进行开发，多个模块向前推进。来完成一项伟大的工程。 成员有什么好想法，就发布在mailing list上，大家讨论，确认下来，你就可以开始开发。 如果与大家不同意你的idea，你可以产生一个项目分支。

这种模式对参与人员能力要求比较高，要求能独立完成任务，有创意，自觉性强，团队合作意识强，。

开源模式也有它的缺点，不能一概照搬，如果照搬开源模式，显而易见人力成本太高了。 因为开源成员都是精英及大师黑客，一个开源项目团队就像海军陆战，单兵作战以一抵十，团队作战所向披靡。 而且

黑客的个性很强，企业不一般不需要员工有个性和创造力，这不便于管理。只要按需求做，不出错误就是好员工。

目前国内企业仍是以高级工程师为核心带领年轻的程序员或应届毕业生方式进行项目开发。

我不得不说中国人很爱跟风，接受能力最快。什么技术流行我们就用什么，最新的技术应用都在中国，你会发现我们的技术是最先进。更本不等市场验证。

在国内企业中你会发现很多眼花缭乱的术语，技术及软件全部用在项目中，向 UML, Visio, Project, Rose, FreeMind, ER-Win, ISO, CMM, GB...

你会发现我们太先进了，Project管理项目，Subversion/VSS/ClearCase控制版本，UML建模，我们用ER-Win/Power Design设计数据库，我们文档用CMM格式，MVC开发框架，ORM操作数据库，我们用Load Runner测试，我们用QQ/MSN沟通。 我们在按着别人的思想，别人定义的标准，帮别人验证他们想法是正确的。如果失败了，就推到重来。

我们的外国客户，还在用ssh登录vim/emacs开发，还再用CVS还是命令行的，Trac/Wiki管理开发资源，开发文档就是一个简单功能说明，在邮件列表上沟通

...

2. 开发语言及平台

语言只是一个工具，一种实现我们需要的工具，每种每言都有它的优点和缺点和，在不同领域发挥各自的长处，并且都有它存在的意义。

语言不段地发展，市场决定它们是生存还是没落走向死亡。只有最活越，生命力强的语言才能生存下来。.net与java后面是强大的财团做后盾，大量被捆绑的客户支持他，并有完备的客服，从商业角度选择它是没有错误的。

但不要拘泥与语言，使用你最熟悉的语言,选择你最擅长的数据库和操作系统。

目前大型网站都不会单一选择一种语言和数据库，一种操作系统。例如:

- 淘宝，前端展示页面采用php,后台管理采用java...
- Yahoo，主要使用php开发，但搜索引擎采用java...
- Ticketmaster，前端展示采用perl开发，后台管理java, 前台展示数据库采用mysql,会员数据库采用oracle

像php/perl/python这种动态语言，开发速度快，周期端，对服务器性能要求低，出错率低，他们的设计这希望它能尽量使代码运行下去， 而不是抛出异常，终止执行或崩溃。而行对于开发者要求门槛比较低。 php 无论怎么开发都不会使web server 崩溃。而 Java 则不同，很容易崩溃。

Java 我认为java是个非常不错的语言，错在JVM上。这是一个垃圾的解释器，效率极差。不加优化的，把所有东东全部load进内存。采用java技术，开发成本相当高，对开发人员要求很高，而且需要一个稳定的团队。国内资深java开发人员大多转向管理层。 只有细心的人才能驾驭Java，否则不能保证软件质量，我在工作中发现php团队开发的代码质量明显比java高，bug 数量上比Java代码的 bug少很多。 我个人认为少于5年工作经验程序员很难写出一流的Java程序。 Java 架构最不能容忍的是有时不得不restart才能生效。而写的很烂程序你不得不采用restart来保证系统正常。

.net 不是很熟悉，.net 开发环境最好，速度比java快，只要有钱，全用正版，选择微软的产品很不错。很多linux爱好者鄙视M$, 对windows系统很有很大偏见，偏激。 window系统很稳定，并非像网上传的那样不堪一击，很多引起windows崩溃的原因是硬件问题。我在工作中发现国产服务器在板卡接口上做工不过关。 没有镀金或防氧化处理，导致内存丢失，cpu丢失，RAID丢失... 等等引起系统崩溃。但linux系统确能运行下去，不过一旦重启，将不能恢复。

2.1. 分层架构

2.1.1. 中间件 Middleware

http://en.wikipedia.org/wiki/Middleware

很多人谈到java就会涉及到三层架构即：web 容器 -> application server 应用服务器，中间件 ->数据库

三层架构其实不是什么新鲜东西，J2EE仅仅是对象请求代理体系结构的一种，任何语言都能实现三层架构。中间件不是Java专利

- 编译执行的语言基本都支持Corba
- python 则有Zope，Zope是一个很成功App Server。足以比肩J2EE. python 也支持corba库，我尝试过python -> corba -> PostgreSQL.但性能不佳。

php/perl 一样可做到,采用SOAP，XML-RPC等技术,可以实现部分功能。但我们可以在架构上做些改变。

总之，不要拘泥于三层架构，仅仅是实现方式的一种。没有最合理，也没有最好的，根据你的需求作出调整，最终是看结果，而不是实现过程。

2.1.2. 分层

中间件的概念

```
         /--> app server ---\
web ---> ---> app server ----> Database
         \--> app server ---/
```

php 分层

```
                 /--> web server ---\
load balance ---> --> web server ----> -> Database
                 \--> web server ---/

                 /--> web server ---\                    /--> app server ---\
```

```
load balance ---> --> web server ----> -- SOAP/XMLRPC--> --> app server ----> Database
                 \--> web server ---/                              \--> app server ---/
```

你可以采用复核型架构，我曾经就做过这样的项目php -> soap -> .net framework

```
                 /--> module ---\
apache ---> php ----> module ----> MySQL Database
                 \--> module ---> soap ---> IIS ---> DCOM ---> C# script  ---> SQL Server
                  \-> module ---> tomcat ---> jasper report ---> Oracle
```

我们系统有一个需求是，在php中创建word,excel文档，编辑文档，比较文档..., 我们需要调用office.dll实现

同时我们有一个报表系统，是通过jasper report 实现的

另外fastcgi

```
                           .--> fastcgi process ---.
                          /                        \
web server ---> ---> fastcgi process -----> Database
                          \                        /
                           `--> fastcgi process----'
```

2.2. Web 2.0

商业炒作产物，对于纯高技术的人来说，虚头

2.3. 云计算

云计算还是个概念，但在云计算大潮中，不支持云计算，显得没有技术含量。

被网络炒得“神乎其神”。起初我误以为是分布式计算的下一代，后来发现和分布式计算根本两个不同的东西。与网格计算/分布式计算扯不上边。

"云计算"这个词已经被泛滥使用，

比较靠谱是亚马逊EC2 其实就是一堆Xen虚拟机, Dell说他的刀片服务器是云计算，Vmware 也说是云计算，Oracle 说他的VirtualBox是云计算，说ZFS是云存储。我也说不清楚，自己斟酌。

2.3.1. 云计算的三种服务模式

IaaS,PaaS,SaaS

2.4. 跨平台

没有真正的跨平台语言，所谓跨平台都是忽悠人。

只要提供不同平台的编译器加条件编译，即可实现跨平台。或提供不同平台的解释器，也可一实现跨平台。

例如大家都很看中Java的跨平台，但想一下，这个“跨平台”是要打引号的，实际上这个跨平台准确的说是跨Sun提供的标准JVM平台，而非OS平台。只要某个JVM支持某个OS，你的程序才可以跨过去。如果JVM不支持这个OS平台，Sorry，你的程序不可能跨过去。 不信你去java.sun.com下载jre你会发现仅仅提供四个平台版本Linux,Mac OS X,Solaris,Windows

很多OS都不支持Java。如FreeBSD 就不支持Java,必须使用Linux glib 运行Java，效率很低,IBM用的是IBM 自己开发 JVM 至于他和Sun Java有什么关系，可能是授权。

只要能让你的程序翻译成JVM字节码，你的程序就可运行在JVM上。如：

php通过Quercus(http://quercus.caucho.com/)把PHP文件编译成.java文件,让后javac编译成class文件后在一些JavaEE应用容器中运行PHP程序

Jython可以将Python编译成java文件

JRuby可以将Ruby编译成java文件

虚拟机并非只有JVM，还有Parrot, Perl6 就是在Parrot虚拟机上实现的。

另外开发一种新语言也并非难事，只要你有时间精力投入我想不出3年，就可以打造一门新语言。

开发新语言也并非难事，只要你有时间精力投入我想不出3年，就可以打造一门新语言。

2.5. 编译语言比脚本语言安全

错！

编译不能保证代码安全，仅仅能保证你的代码不被人使用。

但目前中国人力成本相当的便宜，重新实现你的功能逻辑并非难事。所以只要你的网站上线，在很短的时间内就可以出现很多山寨版。

编译流行的原因是为了解决微机的速度以及存储问题，随着微机处理器技术突破，你根本不用担心速度问题。中型机与大型机领域脚本语言站多数。

2.6. 封装重用

重用可以减轻劳动，但过分重用，会牵一发动全身。

尤其对于二次开发者不熟悉你的系统，导致修改一个bug，又产生新的bug。

另外模板也不宜拆分的过于零碎。模本的组装，需要很多时间并很消耗你的资源。

2.7. 相关的工具

2.7.1. 开发工具

Mozilla Firefox 及扩展

- Web Developer
- Firebug 调试必备工具
- YSlow 性能分析工具
- Live HTTP Headers 相当于HTTP Sniffer嗅探器，可以跟踪HTTP协议头，调试cache时比较有用。
- IE Tab 用于IE/Firefox之间切换
- FoxClocks 如果开发工作跨时区，这个很有用
- Foxmarks Bookmark Synchronizer/Weave 将开发资源放入书签，同时在开发团队中保持同步
- Fasterfox 可以一显示页面载入时间，方便页面优化. 如果安装了YSlow可以不装这个插件。
- FireFTP
- Adblock Plus
- flash block
- Chat Zilla
- Super DragAndGo

开发工具

- visual studio 不必多说
- eclipse 出身于Java但他不单单是Java开发工具，目前他已经是一个通用的语言IDE，我一直用eclipse写PHP,Python,Perl还有Docbook XML
- TortoiseSVN 版本控制工具.
- WinMerge 文件差异比较与合并 , Beyond Compare 我用过最好的比较合并工具。

第 31 章 Framework Design

1. 开发框架 Framework

选择一个好的开发框架，很重要。不过大部分框架都针对于软件开发，而我们要的是轻量级，适合高负载，灵活的框架。

框架的分类

- 本地框架HMVC，MTV
- 远程框架SOA/REST
- 混合框架

上面框架可以满足我们绝大多数需求，如URL定义，Session/Cookie管理，多语言国际化，数据库访问等等。

Java和.Net我没有太多的经验，php我有10+年经验，我在各种框架之间做比较发现CodeIgniter框架比较适合我们的需求。

框架是没有100%完美的，你仍需要对它进行二次开发。如果你有充足的时间，针对自身系统系统的特点设计一个更适合您网站的框架，这是最好的选择。

设计一个框架需要用到很多知识，需要有丰富的经验。目前主流框架都是基于MVC设计思想，要设计一个框架你必须了解MVC (Model-View-Controller) 参考：
http://www.itisedu.com/phrase/200604231324325.html

开发一个框架包括那些重点呢，下面我把一些要点一一列出，然后一个个地突破，我这里使用php为例子，上面我已经说过语言只是工具，所以学习是设计思想，不要拘泥于语言：

- JS封装（javascript）
- 模板（template）
- url
- session/cookie
- 语言包（language package）
- 编码（unicode）
- 数据库访问（database OR Mapping）
- 权限（Permission）

如果重新开发一个框架，我认为太现实，我的建议使用现有pear库，搭建一个MVC框架。例如：

- Model（pear db）
- View（smarty template）
- Controller (pathinfo)

1.1. HMVC

- Python web2py
- Php CakePHP,Zend,CodeIgniter
- Perl Catalyst
- Java Struts,Spring MVC
- ruby on rails

1.2. REST

- Python Pylons
- ruby on rails

1.3. SNA (Shared Nothing Architecture)

1.4. 其他

- Python Django 是一个MTV框架
- .Net Framework

2. MVC Framework Design (设计MVC框架)

2.1. HMVC Framework

等我有时间在补充

3. REST

3.1. RESTful JSON API

跨域

3.2. Ajax 与 RESTful 跨域

允许所有域请求

```
server {
    listen       80;
    server_name  inf.netkiller.cn;

    charset utf-8;
    access_log  /var/log/nginx/inf.netkiller.com.access.log  main;
    error_log  /var/log/nginx/inf.netkiller.com.error.log;

    add_header Access-Control-Allow-Origin *;
```

```
    add_header Access-Control-Allow-Headers Content-Type,Origin;
    add_header Access-Control-Allow-Methods GET,OPTIONS;

        ...
        ...
}
```

允许特定的域请求

```
add_header Access-Control-Allow-Origin http://www.netkiller.com;
```

4. Service-oriented architecture (SOA)

SOA 与 REST很多相同之处，目前SOA主要是基于SOAP实现，也有基于MQ的实现。而REST只限于HTTP POST/GET/PUT/DELETE等等。

我个人比较喜欢机遇TCP的SOA实现，不喜欢SOAP大量XML传输。

4.1. SOAP实现

这里提供一个简单的机遇SOAP实现的SOA框架

index.php入口文件

```
<?php
define ('CONFIG_DIR', '../config/');
define ('LIBRARY_DIR', '../library/');
define ('DEBUG', false);
//define ('DEBUG', ture);

require_once(CONFIG_DIR. 'default.php');
$remote_addr = $_SERVER['REMOTE_ADDR'];
if(!in_array($remote_addr, $firewall)) {
        printf("Permission denied: %s", $remote_addr);
        exit(0);
}

$request_uri = $_SERVER['REQUEST_URI'];
$classspath = LIBRARY_DIR.strtolower($request_uri)  . '.class.php';
if( is_file($classspath) ){
```

```
        require_once($classspath);
}else{
        die("Cannot loading interface!");
}

$class = ucfirst(substr($request_uri, strrpos($request_uri, '/')+1));
if( DEBUG ){
                printf("%s<br>",$class);
}

if (class_exists($class)) {
    $server = new SoapServer(null, array('uri' => "http://webservice.example.com"));
        $server->setClass($class);
        $server->handle();
}else{
        die('Object isnot exist.');
}
```

接口文件

```
<?php
require_once('common.class.php');

class Members extends Common{
        private $dbh = null;
        public function __construct() {
                parent::__construct();
                $this->dbh = new Database('slave');
        }
        public function getAllByUsernameAndMobile($username,$mobile){
                $result = array();
                if(empty($username) or empty($mobile)){
                        return($result);
                }
                $sql = "SELECT username, chinese_name, sex FROM members m, members_digest md WHER
                $stmt = $this->dbh->prepare($sql);
                $stmt->bindValue(':username', $username);
                $stmt->bindValue(':mobile', $mobile);
                $stmt->execute();
                $result = $stmt->fetch(PDO::FETCH_ASSOC);
                return($result);
        }
        public function getAllByLimit($limit,$offset)
        {
                $sql = "SELECT username FROM members limit ".$limit.",".$offset;
                $stmt = $this->dbh->query($sql);
                while ($row = $stmt->fetch()) {
                        //printf("%s\r\n", $row['username']);
                        $result[] = $row['username'];
                }
                return $result;
        }
        function __destruct() {
       $this->dbh = null;
   }
}
```

客户端调用实例

```
<?php

$options = array('uri' => "http://webservice.example.com",
```

```
                'location'=>'http://webservice.example.com/members',
                                 'compression' => 'SOAP_COMPRESSION_ACCEPT | SOAP_COMPRESSION_GZI
                                'login'=>'neo',
                                'password'=>'chen',
                'trace'=>true
                                );
$client = new SoapClient(null, $options);

try {

        print_r($client->getAllByUsernameAndMobile('280600086','13113668890'));
        print_r($client->getAllByLimit(20,20));

}
catch (Exception $e)
{
    echo 'Caught exception: ',  $e->getMessage(), "\n";
}
```

Nginx 虚拟主机配置文件 /etc/nginx/conf.d/webservice.example.com.conf

```
server {
    listen       80;
    server_name  webservice.example.com;

    charset utf-8;
    access_log  /var/log/nginx/webservice.example.com.access.log  main;
    auth_basic            "Login";
    auth_basic_user_file  htpasswd;

    location / {
        root   /www/example.com/webservice.example.com/htdocs;
        index  index.html index.php;
                if ($request_filename !~ (js|css|images|robots/.txt|.*\.html|index/.php) ) {
                    rewrite ^/(.*)$ /index.php/$1 last;
                    break;
                }
    }

    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }

    location ~ /index.php/ {
        root           /www/example.com/webservice.example.com/htdocs;
        fastcgi_pass   127.0.0.1:9000;
        fastcgi_index  index.php;
        fastcgi_param  SCRIPT_FILENAME  /www/example.com/webservice.example.com/htdocs$fastcgi_sc
        include        fastcgi_params;
    }

}
```

每增加一个功能需求，在library中创建一个 Class 文件即可。

index.php 有IP过滤功能，禁止非法IP访问

客户端采用压缩传输，节省xml传输开销

Nginx 设置了HTTP认证，防止他人探测，另外提示你还可以采用双向SSL认证。

4.2. MQ 实现

5. Dispatcher MVC核心分发器

5.1. URL设计

一个大型网站，对于URL规划我认为非常重要，这也是为什么我把它单列出来的原因。

当前网站上使用的URL虚虚实实已经不单单是划分目录空间功能，它与程序配合使用，实现复杂的逻辑功能。在应用程序开发框架组成中占有重要的地位。

注意

无论什么文件系统，每个目录下容纳的子目录和文件是有限制的，并且内容过多会影响文件索引速度，所以合理地划分目录空间很重要

下面是URL实例仅供参考，稍后我会详细解释他们这样设计的目的是什么和实现方法。

- http://sina.allyes.com/main/adfclick?db=sina&bid=120294,154641,159584&cid=0,0,0&sid=146767&advid=2618&camid=19961&show=ignore&url=http://web.topxue.com/gj/bdxm/
- http://news.sina.com.cn/c/2008-05-22/172315597145.shtml
- http://example.org/bbs/thread-1003872-1-1.html
- http://example.org/news/2008/05/22/1004862.shtml
- http://example.org/uk/en/action,ProductDetailShow_productId,51
- http://example.com/forums/viewforum/59/
- http://example.com/forums/viewthread/80165/

- http://trac.example.com/cgi-bin/trac.cgi/ticket/1286

目录设计，以下为真实目录，你在URL看到其它路径都是不存在的。它们是用于rewrite或pathinfo的。

- images
- framework
- model
- view
- controller
- language
- config
- logs

5.1.1. URL 作为MVC 的Controller

例子1

http://example.com/guestbook/view/59/

相当于

http://example.com/<controller>/<action>/<id>/

```
class Guestbook extend Controller{
        public function index(){
        }
        public function view($id =1){
        }
        public function add(){}
        public function remove($id){
```

```
        }
}
```

一般采用pathinfo技术实现上述功能

5.1.2. URL 伪静态化，用于SEO优化

http://example.com/guestbook/view/59.html

相当于

http://example.com/guestbook.php?action=view&id=59

一般使用Rewrite技术实现

5.2. Dispatcher 的实现方式

```
$action  = $_REQUEST['action'];
$libname = $_REQUEST['lib'];
$special = new Advertize ($libname,$action);
if(method_exists($special, $action)) {
    $special->$action();
}else{
    $special->index();
}
```

6. Plugin & Hook 设计与实现

插件系统分为:

插件管理平台

插件探测

插件注册

插件调用

插件注销

6.1. 插件管理平台

6.1. 插件管理平台

```
<?php
final class Plugin{
        private $plugins        = null;
        private $directory      = 'plugins';
        private $path           = null;
        public function __construct(){
                $this->path = $this->directory.'/';
        }
        public function autoload(){
                $interfaces = scandir($this->directory);
                unset($interfaces[0]);
                unset($interfaces[1]);
                foreach($interfaces as $interface)
                {
                        //load all of the plugins
                        $file =  $this->path . $interface;
                        if (@file_exists($file))
                        {
                                include_once($file);
                                $class =  basename($interface, ".php");
                                if (class_exists($class))
                                {
                                        $this->$class = new $class($this);
                                        $vars = get_class_vars($class);
                                        $entity['name']                         = $vars['name'];
                                        $entity['description']  = $vars['description'];
                                        $entity['author']               = $vars['author'];
                                        $entity['class']                = $class;
                                        $entity['methods']              = get_class_methods($clas

                                        $this->plugins[$class] = $entity;
                                }
                        }
                }

        }
        public function load($plugin){
                $file = $this->path . $plugin . '.php';
                if (@file_exists($file))
                {
                        include_once($file);
                        $class = $plugin;
                        if (class_exists($class))
```

```
                        {
                                $this->$class = new $class($this);
                                $vars = get_class_vars($class);
                                $entity['name']                         = $vars['name'];
                                $entity['description']  = $vars['description'];
                                $entity['author']               = $vars['author'];
                                $entity['class']                = $class;
                                $entity['methods']              = get_class_methods($class);

                                $this->plugins[$class] = $entity;
                        }
                }
        }
        public function show(){
                print_r($this->plugins);
        }
}
```

6.2. 接口定义

```
<?php
interface iPlugin
{
        public function test();
}
```

6.3. 插件

```
<?php
final class demo implements iPlugin{
        public static $author           = 'Neo Chen<openunix@163.com>';
        public static $name = 'Demo';
        public static $description = 'Demo Simple';
        public function __construct(){

        }
        public function test(){
                echo 'Hello world!!!';
        }
}
```

6.4. 测试

```
<?php
function __autoload($class_name) {
    require_once('library/'.$class_name . '.php');
}

//include_once('library/Plugin.php');
$plugin = new Plugin();
echo '============================';
$plugin->load('demo');
$plugin->demo->test();
echo '============================';
$plugin->autoload();
$plugin->show();
```

7. Interface

Application Interface

7.1. 访问接口协议

机遇http的实现方式有下面几种。
http协议传统post/get 方式
soap 简单对象访问协议
xmlrpc 机遇xml的协议
json 近年来兴起的一种数据序列化传输方法
http无状态协议，不能保证连接100%有效性。http方式受限制与浏览器，对于并发控制，超时时间，通信数据长度都有严格的限制
例如：一般浏览器运行超时时间都是30秒或60秒，当你通过http方式访问接口时，你的程序因运行超过30秒被浏览器强行中断；另
结局上述问题方法是将借口独立出一台服务器，单独设置超时时间等配制
http 方式有诸多缺陷，当仍被广泛使用，他的特点是容易开发，开发人员不需要额外学习，如post/get方式
http 方式的优势是它可以携带Cookie/Session
TCP/UDP Socket 方式
TCP 这是唯能保证不间断时时传输手段，开发难度很高，目前web开发人员中能写出高效的多线程socket程序的人很少。
其中涉及很多知识，例如：进程，线程，锁，列队，进程间通信，共享内存，以及信号处理等等；没有10年功力很难写出安全，稳定
UDP 能够发送大数据包

7.2. 接口性能问题

必须考虑接口最大会话数
处理请求后到返回数据所花费的时间
接口应该支持负载均衡，通过增加节点数量，快速扩展；同时添加与撤除节点不会影响接口的通信（包括节点硬件故障）；同时接口

7.3. 接口安全问题

来源IP控制，即黑白名单，获取IP地址需要考虑X Forward for

IP计数器，单位时间内IP访问次数达到阀值，就提示稍后连接

用户名密码认证与访问权限

动态验证码

证书加密

md5/sha1 数字摘要 校验

SSL / TSL 证书加密

7.3.1. 访问权限

接口访问权限应该具备功能

颗粒度精确到每个操作方法

8. 模板(template)

模板最早是在cgi程序中广泛应用，cgi是动态页面的第一代，同期还有NSAPI,ISAPI,第二代是fastcgi,asp,php,ColdFusion...第三代是.net与java。

模板的特点：

- 模板可以分离代码和页面
- 模板能够改善页面结构
- 模板可实现页面重用
- 模板可以区块化，如同搭积木
- 设计人员不需要关心代码
- 实现主题

模板有很多优点，但它也会增加系统开销，不过我们可以通过cache来解决这个问题。

常用模板引擎：

- PHP: smarty template
- Perl: TT template
- Python：Cheetah

8.1. HTML 页面优化

页面减肥

```
{strip}
<html>
....<head>
........<title>Title</title>
....</head>
....<body>
........<h1>Hello world</h1>
........<div class="">
............Test
........</div>
....</body>
</html>
{/strip}
```

Smarty 的{strip}可以删除页面中的空格，Tab符以及回车换行符

9. Session/Cookie

为什么我要在这里提Session和Cookie，这也大型站点必须要处理问题。

9.1. Session

在集群环境中与单服务器是不一样的，集群组成可分为调度服务器和节点，节点数量不定，单个节点安装有web服务器，用户每次访问网站调度服务器随机分配一个节点给该用户， 举一个例子：用户在网站上看新闻，点击第一个连接被分配到node 1上去，当他看完这条新闻并单击下一条时，可能被分配到其它节点上，这里刚才建立的session在node 1上， 它就会因失去session而必须重新登录。

所以我们要同步所有节点上的Session, 另外如果能用Cookie代替Session的地方尽量使用Cookie。

9.2. Session 共享

解决方案：

1. 不用Session，使用Cookie取而代之
2. 共享Session，放到数据库中，放到Memcache中

PHP Session很有解决方案：

查看PHP手册 Session Extensions 章节，重写Session逻辑。

共享Session用Memcache，在php.ini中配置即可

```
session.save_handler = memcache
session.save_path = tcp://127.0.0.1:10001
```

9.3. Cookie

Cookie 我这里提到cookie是可以实现“单点登录”功能。

一个网站可能不指一组集群系统，如news.example.org, bbs.example.org, blog.example.org 要实现在一处登录即可在其它站点上同时也处于登录状态，就要用到Cookie来实现。

9.3.1. Cookie 安全

Cookie存储在用户端，Cookie数据极易伪造。下面提供几个方案。

- 在Cookie数据上加干扰词
- 在反向代理上做手脚
- 负载均衡设备都提供Cookie保护功能

9.3.2. cookie-free domains

9.3.3. P3P

header('P3P: CP="CURa ADMa DEVa PSAo PSDo OUR BUS UNI PUR INT DEM STA PRE COM NAV OTC NOI DSP COR"');

```
<?php
header('P3P: CP="CURa ADMa DEVa PSAo PSDo OUR BUS UNI PUR INT DEM STA PRE COM NAV OTC NOI DSP COR
setcookie("test", $_GET['id'], time()+3600, "/", ".a.com");
?>
```

10. 国际化 Locale database。

在开始具体介绍之前，需要先介绍几个术语:

- i18n: 就是internationalization, 国际化,由于首字母"i"和末尾字母"n"间有18个字符，所以简称i18n. internationalization指为了使应用程序能适应不同的语言和地区间的变化而不作系统性的变化所采取的设计措施。

- l10n: 就是localization, 本地化，由于首字母"l"和末尾字母"n"间有10个字母，所以简称l10n. localization指为了使应用软件能够在某一特定语言环境或地区使用而加入本地特殊化部件和翻译后文本的过程。

- locale: 简单来说是指语言和区域进行特殊组合的一个标志，如: en-us, zh-cn, zh-tw

l10n有很多历史遗留问题，l10n目前已经被i18n取代。

我自己曾经使用过下面四种方式实现语言包

1. 定义一个数组

2. 使用数据库

3. 使用文件

4. 使用数据结构

10.1. Unicode

相比几年前，目前各种语言对UTF-8支持都比较好。

在BBS上常常看到一些网友抱怨UTF-8出现“乱码”问题，让我们看看都有哪些地方涉及编码问题。

用户输入法->IDE开发环境,浏览器->web容器->数据库

任何一个环节出现问题有可能出现问题

- 首先是输入法，早期输入法可能是GB2312或GBK。
- 其次是IDE开发环境，当你创建一个空文件时，它的已经具备某种编码，一般外国开发工具默认是acsii，这一点我认为Dreamware做的最好，可以随时切换编码。
- 浏览器现在基本不用担心
- web容器apache 2.x对unicode支持很好，tomcat本身机器码就是unicode。
- 数据库问题也不大，PostgreSQL相比MySQL对Unicode支持也早，也比较好。MySQL这方面有点复杂。

提示

如果你不考虑使用Unicode并且想支持繁体和简体中文，你可以使用GBK，但我建议你使用GB18030。

Unicode不是最好的选择，它占用三个字节，数据量较大，选择适合你的编码，如果你是英文网站，请使用ISO-8859-1，如果是简体中文，请使用GB2312

11. 数据库访问

早期php访问数据库的做法是写一个连接文件，include包含进来，然后在页面使用sql操作函数，返回结果。

- CRUD (create, read, update and delete)
- Active Record
- OR Mapping

另外设计一个框架是还要考虑，切割表，分库。

11.1. CRUD

11.2. Active Record

11.3. OR Mapping

12. Cache

Cache大体分为两种，一种是文件Cache,另一种是内存Cache。按应用划分，可以分为页面Cache和局部Cache

12.1. 页面缓存

页面缓存有三种实现方式:

1. 反向代理
2. HTTP Header
3. Template 层

页面cache的原理是推送HTTP协议头，修改网页过期时间。

12.2. 局部缓存

局部cache是可以将数组，序列化对象，字符串等等，分别cache，并设置ttl值（生存时间）

数据库与应用程序之间加一层Cache,性能将会大幅提升。

我们一般会把Cache封装为一个Class (类)，并且支持多种Cache API. 如：Apc Cache,Xcache, Memcache...

Cache操作很简单，添加，更新，删除，状态几种操作，其中添加与更新可以共用一个方法。

13. Single sign-on (SSO) 单点登录

提供一站式登录，即一旦在*.domain.com任何位置登录后， 全域均不需要在重新登录

解决方案:

1. 登录 cookie 的host 使用domain.com 不要使用www.docmain.com
2. 子域名通过rewrite实现,即bbs.domain.com = www.domain.com/bbs

14. 搜索引擎

15. Synchronous/Asynchronous

举几个例子

synchronous (同步)

1. 用户登录
2. 商品价格与相关计算
3.

asynchronous(异步)

1. 批量操作建议使用异步方式
2. 消息广播操作
3.

用户注册，发帖与评论。采用同步或异步均可，还要看你的具体情况而且。

16. Message Queuing

Message Queuing 有 synchronous (同步)/asynchronous(异步) 之分，不同场景适合不同的处理方式。

RabbitMQ

ZeroMQ

Apache ActiveMQ

17. Hash

18. Sharding 垂直/水平切割

18.1. 面向服务

18.2. 面向数据库

19. 日志系统

开源的日志系统，包括facebook的scribe，apache的chukwa，linkedin的kafka和cloudera的flume等

```
Scribe：https://github.com/facebook/scribe
Chukwa：http://incubator.apache.org/chukwa/
Kafka：http://sna-projects.com/kafka/
Flume：https://github.com/cloudera/flume/
```

20. Cache

20.1. CDN (Content Delivery Network) 与反向代理

与CDN有关的开发工作

20.1.1. CDN接口API

CDN 内容更新，一般厂商会提供一个SOAP接口，你可以通过接口刷新你的内容。但接口有限制，不能随意使用，一般是多少秒可以刷新一次，或者一天可以刷新几次

20.1.2. 方向代理页面过期处理

方向代理一般都支持PURGE协议，Squid,Varnish等等向管理端口发送 PURGE 即可是使用页面刷新

```
PURGE http://netkiller.github.net/index.html
```

有些方向代理如：Varnish 可以使用正则表达式

同时这些代理服务器都承受管理命令

squid: squidclient

varnish: varnishadm

20.1.3. 内容版本化

例如这样的URL

```
http://images.example.com/logo.gif
http://images.example.com/product.jpg
```

我们可以通过Rewrite或PATHINFO等技术做为静态化。例如首次版本

```
http://images.example.com/logo.1.gif            => logo.gif
http://images.example.com/product.1.jpg         => product.jpg
```

原图发生变化后，版本递增

```
http://images.example.com/logo.2.gif            => logo.gif
http://images.example.com/product.2.jpg         => product.jpg
```

就的URL将丢弃

```
http://images.example.com/logo.1.gif
http://images.example.com/product.1.jpg
```

CDN 就回源去下面的URL，并且取到的是新图

```
http://images.example.com/logo.2.gif
http://images.example.com/product.2.jpg
```

20.2. HTTP Cache

Cache-Control

```
max-age
max-age 格式写为：max-age=n，n是以秒为单位，这个值是告知客户端GMT + N 后页面过期，缓存服务器在s-maxage值为空的

s-maxage
s-maxage的格式跟max-age一样，只不过他是给缓存服务器使用的。

must-revalidate
这个参数用来告知客户端和缓存服务器，在GET请求的时候必须与源服务器验证实体是否为最新版本。

Cache-Control:max-age=1200,s-maxage=3600



Last-Modified
这个参数提供了实体最近一次被修改的时间。这个名字起得不错，当实体被修改了之后，这个参数也就会被修改.
```

ETag

```
ETag
```

ETag是根据内容生成的一段hash字符串，采用信息摘要算法，保证每一个页面有一个唯一字串。

expires

expires 是HTTP 1.0 中定义的，已经不能满足用户的需要在 HTTP 1.1 加入了max-age，建议使用 max-age替代expires

指令	含义
public	可以在任何地方缓存
private	只能被浏览器缓存
no-cache	不能在任何地方缓存
must-revalidate	缓存必须检查更新版本
proxy-revalidate	代理缓存必须检查更新版本
max-age	内容能够被缓存的时期，以秒表示
s-maxage	覆盖共享缓存的max-age设置

在Squid, Varnish, Apache, Lighttpd, Nginx 中都可是实现HTTP Cache-Control推送，每次修改都需要重新加载，不太灵活。

```
ExpiresActive On
ExpiresByType image/gif "access plus 1 month"
ExpiresByType image/png "access plus 1 month"
ExpiresByType image/jpeg "access plus 1 month"
ExpiresByType text/css "access plus 1 month"
ExpiresByType text/javascript "access plus 1 month"
ExpiresByType application/x-javascript "access plus 1 month"
ExpiresByType application/x-shockwave-flash "access plus 1 month"
```

```
server.modules = (
...
"mod_expire",
...
)

$HTTP["url"] =~ "^/images/" {
expire.url = ( "" => "access 30 days" )
}
```

我喜欢自己控制TTL时间，且每个页面单独设置，可以随时调整设置。

20.2.1. 在程序中灵活操作 Cache-Control

在MVC框架中每个控制器下的方法都可以单独操作Cache

```
Class blog extend Controller{
        blog(){
                header('Cache-Control: max-age=28800');
        }
        list(){
                header('Cache-Control: max-age=3600');
        }
        details(){
                header('Cache-Control: max-age=160');
        }
}
```

你还可以封装到Controller中

```
Class blog extend Controller{
        blog(){
                this->cache('28800');
        }
        list(){
                this->cache('3600');
        }
        details(){
                this->cache('160');
        }
}
```

20.2.2. 非程序文件缓存处理

首先做一个Rewrite让程序接管所有图片请求

```
url.rewrite = ( "^/(.+)" => "/index.php/$1" )
```

然后程序通过PATHINFO取出图片URL

```
http://images.example.com/your/dir/test.jpg => http://images.example.com/index.php/your/dir/test.
```

程序取出 /your/dir/test.jpg 设置 Content-type 并输出二进制流

详细参考

```
<?php
    // Test image.
    $images = '/test/foo.png';

    $headers = apache_request_headers();

    if (isset($headers['If-Modified-Since']) && (strtotime($headers['If-Modified-Since']) == file
        header('Last-Modified: '.gmdate('D, d M Y H:i:s', filemtime($images)).' GMT', true, 304);
    } else {
        header('Content-Type: image/png');
        print file_get_contents($fn);
                if (file_exists($images)) {
                        header('Last-Modified: '.gmdate('D, d M Y H:i:s', filemtime($images)).' G
                        header("Cache-Control: max-age=3600, must-revalidate");
                        header('Content-Length: '.filesize($images));
                        header('Content-type: ' .mime_content_type($images));
                        flush();
                        readfile($images);
                        exit;
                }
    }
```

javascript 文件也可以使用类似方法处理

```
        private function js($file){
                if (file_exists($file)) {
```

```
                header("Cache-Control: max-age=3600, must-revalidate");
                header('Content-type: text/javascript');
                flush();
                readfile($file);
                exit;
        }
}
```

20.3. Cache 生存时间

你不必一开始加费劲心机去考虑这个值，当网站运行一段时间后，利用玩站流量数据作为参考，一步一地尝试调整。

21. i18n 国际化

21.1. 数组方式

这种方式流行于PHP语言，下面是一个例子

例 31.1. php language package

```
<?php
        $language['hello_world'] = 'hello world !!!'
?>
```

21.2. 数据库方式

数据库方式包括

1. 其他非关系型收据库 (Berkeley DB)
2. 对象/关系型收据库 ORDBMS （mysql）

Berkeley DB 是一个不错的选择，而且相对关系型数据库比较有优势。因为关系型数据库子并发数有限，连接资源很宝贵。

例 31.2. sql table language package

```
select id,key,value from language where country = 'zh-cn' and key = 'hello_world';
```

21.3. 文件文件

例如.ini文件

```
news=新闻
top10=前十位
```

21.4. Gettext

The gettext functions implement an NLS (Native Language Support) API which can be used to internationalize your PHP applications. Please see the gettext documentation for your system for a thorough explanation of these functions or view the docs at » http://www.gnu.org/software/gettext/manual/gettext.html.

21.5. 数据结构

数据结构方式主要包括

1. 哈希表 hash table
2. 类 class
3. 字典 dict
4. 图 map

提示

可能会用到序列化

22. RSS / ATom

22.1. Atom

http://www.atomenabled.org/

23. Logging 日志

23.1. 日志的格式

23.2. 日志存贮

23.2.1. 本地存储

23.2.2. 远程存储

23.3. Log4cpp/Log4j/Log2PHP

23.4. Remote Syslog

syslog

```
syslogd
----------------------
main {
        fork(){
                socket()

                threading{
                        while buffer() {
                                queue.add()
                        }
                }

                socket.close()
        }
}

storage_engine
----------------------
main(){
                queue()
                format()
                save()
```

```
}

save(){
        open()
        write()
        close()
}
```

24. debug

生产环境不允许有任何调试输出，包括程序错误，这些信息应该写入error.log， 我们可以在网站放置一个调试入口，

```
if(isset ($_GET['debug'])){

}else{

}
```

25. 性能优化

25.1. 尽量使用单引号

尽量使用单引号，迫不得已才使用双引号，因为双引号会处理转义字符。

26. Design pattern (设计模式)

常用设计模式包括

Singleton 单件模式
Abstract Factory 抽象工厂模式
Builder 生成器模式
Factory Method 工厂方法模式
Prototype 原型模式
Adapter 适配器模式
Bridge 桥接模式
Composite 组合模式
Decorator 装饰模式
Facade 外观模式
Flyweight 享元模式
Proxy 代理模式
Template Method模板方法
Command 命令模式
Interpreter 解释器模式
Mediator 中介者模式
Iterator 迭代器模式
Observer 观察者模式
Chain Of Responsibility 职责链模式
Memento 备忘录模式
State 状态模式
Strategy 策略模式
Visitor 访问者模式

26.1. Singleton 单件模式

```
<?php
class Cache {

        private $cache = array();
```

```
        public function __construct(){}
        public function set($key,$value){
                if(!empty($key)){
                        $this->cache[$key] = $value;
                }
        }
        public function get($key){
                if(array_key_exists($key, $this->cache)){
                        print($this->cache[$key]);
                }
        }

}
```

```
<?php
class Cache {

        private static $instance;
        private $cache = array();
        private function __construct(){}
        public static function getInstance() {
                if(empty( self::$instance )){
                        self::$instance = new Cache();
                }
                return self::$instance;
        }
        public function set($key,$value){
                if(!empty($key)){
                        $this->cache[$key] = $value;
                }
        }
        public function get($key){
                if(array_key_exists($key, $this->cache)){
                        print($this->cache[$key]);
                }
        }

}

$db = Cache::getInstance();
$db->set('name','netkiller');
$db->get('name');
print("\r\n");
```

```
$db1 = Cache::getInstance();
$db1->get('name');
$db1->set('age','30');
print("\r\n");

$db2 = Cache::getInstance();
$db2->get('name');
$db2->get('age');
print("\r\n");

unset($db1);

$db->set('name','neo');
$db->get('age');
$db2->get('name');
print("\r\n");

print("---------------------------\r\n");
// private function __construct(){}
//$db3 = new Cache();
//$db3->set('name','netkiller');

//$db1 = new Cache()
//$db1->get('name');
```

27. AOP（Aspect Oriented Programming)

```
<?php

interface Account{
        public function hello($str);
}

class Demo implements Account{
        public function __construct(){}
        public function hello($str = ""){
                echo 'Hello: '.$str;
        }
        public function __destruct(){}
}

class Aop
{
    private $instance;

    public function __construct($instance){
        $this->instance = $instance;
    }
    public function __call($method, $argument){
        if(! method_exists($this->instance, $method)){
            throw new Exception('Undefine function: ' . $method);
        }

        /* 此处加入before代码 */

        $callBack = array($this->instance, $method);
        $return = call_user_func_array($callBack, $argument);

        /* 此处加入after代码 */

        return $return;
        }
}

class Factory
{
        public function __construct(){}
    public function getInstance(){
```

```
        return new Aop(new Demo());
    }
}

try
{
    $factory = Factory::getInstance();
        $factory->hello('world');
}
catch(Exception $e)
{
    echo 'Caught exception: ',  $e->getMessage();
}
```

28. 信息安全

SQL注入，OS命令注入，缓冲溢出、跨站脚本、缺少验证、缺少认证、使用硬编码证书、敏感数据忘记加密、不受限制上传文件类型、依赖不可信的输入、用不必要的高级权限执行任务、跨站请求伪造....

28.1. CSRF（Cross-site request forgery）跨站请求伪造

CSRF（Cross-site request forgery），中文名称：跨站请求伪造，也被称为：one click attack/session riding，缩写为：CSRF/XSRF

28.2. Session 撰改演示

这是一个计数器的例子

```
<?php
session_start();

if(isset($_SESSION['count'])){
        $_SESSION['count']++;
}else{
        $_SESSION['count'] = 1;
}
print($_SESSION['count']);
```

首先在IE浏览器上访问该文件，查看目前计数器数值。

现在开始演示如果更改用户的Session数据

通过Firebug等工具，查看PHPSESSID的值，例如我的是 75ff0dd6a0824a2b607777b58c27f78a

```
cat /tmp/sess_75ff0dd6a0824a2b607777b58c27f78a
count|i:100;
```

将 countli:100; 改为 countli:1000; 再次去浏览器刷新看看现在计数器的数值是多少。

通过这种方法可以实现，提升权限，绕过登录等等。

由于session 存储在 tmp 目录下，一旦网站被注入就来带安全隐患

28.3. 用户注册与登录安全

用户注册与登录除了使用图片验证外，还应该记录来源IP，同时限制用户使用自动注册工具

28.4. 目录文件与权限

28.4.1. 读写权限

Apache进程所有者: nobody

程序所有者: www

apache 可以读取程序并运行，但apache 无法改写代码，/tmp等特殊目录可以写入操作

重置权限命令

```
chown www:www -R /www
chown nobody:nobody -R /www/www.example.com/tmp

find /www/ -type d -exec chmod 755 {} \;
find /www/ -type f -exec chmod 644 {} \;
chmod 744 -R /www/www.example.com/tmp
```

28.4.2. 访问权限

屏蔽访问权限

```
<Directory>
<DirectoryMatch>
<Files>
<FilesMatch>
<Location>
<LocationMatch>
```

并不是所有目录和文件都需要提供给用户的，例如早期PHP项目中没有使用框架，常常有include, config等等目录需要屏蔽

例 31.3. Example for ECSHOP

```
<VirtualHost *:80>
    ServerAdmin webmaster@example.com
    DocumentRoot /www/www.example.com/
    ServerName www.example.com
    ServerAlias example.com
    DirectoryIndex index.html index.php
    CustomLog "|/srv/httpd/bin/rotatelogs /www/logs/www.example.com/access.%Y-%m-%d.log 86400 480

    <Location /data/>
            Order allow,deny
            Deny from all
    </Location>
    <Location /images/upload/>
            Order allow,deny
            Deny from all
    </Location>
    <Location /temp/>
            Order allow,deny
            Deny from all
    </Location>
    <Location /includes/>
            Order allow,deny
            Deny from all
    </Location>
    <Location /library/>
            Order allow,deny
            Deny from all
    </Location>
    <Location /plugin/>
```

```
            Order allow,deny
            Deny from all
    </Location>

    <Directory /www/www.example.com/images/>
        <Files *.php>
            Order allow,deny
            Deny from all
        </Files>
    </Directory>
    <Directory /www/www.example.com/js/>
        <Files *.php>
            Order allow,deny
            Deny from all
        </Files>
    </Directory>

    <Directory /www/www.example.com/themes/>
        <Files *.php>
            Order allow,deny
            Deny from all
        </Files>
    </Directory>

</VirtualHost>
```

28.5. 密码安全

虽然md5摘要算法作为密码仍不能保证安全。我一般采用加入干扰词的方法避免被猜中

password = md5/sha1(password + salt)

28.6. 注入检查

我们需要在框架的URL(PATHINFO)对象中加入检查功能

```
/news/%d.html            只能匹配数字ID /news/123.html 合法,如果/news/abc.html 非法
/login/%s.html           只能匹配字符串     /login/neo.html
/product/[0-9/a-z].html          可以配置数字已经字符
```

post 数据还有上传文件也做同样检查

这里仅仅给你一个思路，实现起来也并不难

28.7. 防止恶意刷新与重复提交

在开发中会经常会遇到这样的需求，例如投票模块，要防止恶意刷票，下面来介绍几种解决方按：

```
1、来源IP / MAC地址限制
        这个是使用最多也是最广泛的方式，通过获取访问用户的来源IP地址，来限制在一段时间内所能使用的票数。
经常用电脑的老手是很容易绕出这种限制的。PPP/PPPoE拨号用户，可以通过断线重拨来更换IP地址；
                    每个网络位置会有一个全球唯一的MAC位址。所以我们也可以根据MAC地址限制用户访问

2、Cookies / Session验证
        这种方式用的也比较多，清除浏览器Cookies，就可以很容易的绕过这种限制了
            关闭浏览器，Session就会被销毁；客户端禁用Cookie，Session也会失效；
```

4、验证码,包括图像，语音，电话，邮件以及回答问题

首先说图片验证码，有些变态的网站，大家可以看到用户的注册、登录、回复、发帖等等，都会使用验证码，但是这种方
语音有播放方式和电话方式，听喇叭中读取字符，然后输入验证码。不要以为这是最安全的，语音是一种波形，通过DSP(
手机短信与电子邮件，不多说了

回答问题

如果没有足够海量的题库，很快问题的内容和答案就会被收集。反而让正常投票的用户，觉得投票很恶心、麻烦，产生厌

5、注册用户可能投票模块

游客不能参与，必须注册了账户才能进行投票，并且限制新注册用户，在一段时间内不能参与投票。

6、随机投票地址

让每一个访问页面的用户得到一个随机唯一的KEY可能通过UUID/GUID生成，通过这个KEY，生成一个投票地址，该地址

总结：很快就会有新的应对方式。我们只能通过上面几种方案的组合方式，增加用户刷新难度，让用户在无法在短期内实现应对方案

28.8. 屏蔽出错信息

28.8.1. 屏蔽php出错信息

```
; Error handling and logging ;
; 出错控制和登记 ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; 错误报告是按位的。或者将数字加起来得到想要的错误报告等级。
; E_ALL - 所有的错误和警告
; E_ERROR - 致命性运行时错
; E_WARNING - 运行时警告（非致命性错）
; E_PARSE - 编译时解析错误
; E_NOTICE - 运行时提醒(这些经常是是你的代码的bug引起的,
;也可能是有意的行为造成的。(如：基于未初始化的变量自动初始化为一个
;空字符串的事实而使用一个未初始化的变量)

; E_CORE_ERROR - 发生于PHP启动时初始化过程中的致命错误
; E_CORE_WARNING - 发生于PHP启动时初始化过程中的警告(非致命性错)
; E_COMPILE_ERROR - 编译时致命性错
; E_COMPILE_WARNING - 编译时警告(非致命性错)
; E_USER_ERROR - 用户产生的出错消息
; E_USER_WARNING - 用户产生的警告消息
; E_USER_NOTICE - 用户产生的提醒消息
; 例子:
; error_reporting = E_ALL & ~E_NOTICE ; 显示所有的错误，除了提醒
; error_reporting = E_COMPILE_ERROR|E_ERROR|E_CORE_ERROR ; 仅显示错误
error_reporting = E_ALL & ~E_NOTICE ; 显示所有的错误，除了提醒
display_errors = On ; 显示出错误信息(作为输出的一部分)
; 在最终发布的web站点上，强烈建议你关掉这个特性，并使用
; 错误日志代替（参看下面）。
; 在最终发布的web站点继续让 display_errors 有效可能
; 暴露一些有关安全的信息，例如你的web服务上的文件路径、
; 你的数据库规划或别的信息。
display_startup_errors = Off ; 甚至当display_erroes打开了，发生于PHP的启动的步骤中
; 的错误也不会被显示。
; 强烈建议保持使 display_startup_errors 关闭,
; 除了在改错过程中。
log_errors = Off ; 在日志文件里记录错误（服务器指定的日志，stderr标准错误输出，或error_log(下面的)）
; 正如上面说明的那样，强烈建议你在最终发布的web站点以日志记录错误
```

; 取代直接错误输出。

track_errors = Off ; 保存最近一个 错误/警告 消息于变量 $php_errormsg (boolean)
;error_prepend_string = "<font color=ff0000>;" ; 于错误信息前输出的字符串
;error_append_string = "</font>;" ; 于错误信息后输出的字符串
;error_log = filename ; 记录错误日志于指定文件
;error_log = syslog ; 记录错误日志于系统日志 syslog (NT 下的事件日志, Windows 95下无效)
warn_plus_overloading = Off ; 当将‘+’用于字符串时警告

这项去掉
; E_WARNING - 运行时警告（非致命性错）

29. 序列化

部分 V. 设计与解决方案

第 32 章 支付平台方案

1. 方案

1.1. 商户方案

商户方案 .

1.2. 银行方案

银行接口方案.

用户
网站
支付平台
招商银行 API
工商银行 API
建设银行 API
收款银行卡
银行卡1
银行卡2
银行卡3
银行卡4
银行卡1
银行卡2
银行卡3
银行卡4
银行卡1
银行卡2
银行卡3
银行卡4
支付平台后台
银行卡管理
API 插件管理
客户管理
订单管理

2. 支付接口

支付接口需要考虑

接口安全

1. 软件防火墙，包括IP限制，单位时间内IP访问次数限制

2. key 证书，采用非对称加密

3. md5 校验，防止伪造内容提交

WEB服务器安全

1. SSL 证书

2. IP 限制

3. 目录访问权限

操作系统安全

3. 支付派台后台

第 33 章 电子商务网站

1. Product

```
class Product():
    attribute = []
    def __init__(self, name = None):
        if name :
            self.name = name
        else:
            self.name = "Unknown"
        self.description = "None"
        self.price = 0
    def getName(self):
        return self.name
    def getDescription(self):
        return self.description

class Attribute():
    def __init__(self,product, attr):
        self.product = product
        self.product.attribute.append(attr)

class Color():
    def __init__(self):
        pass
    def Red(self):
        return {'color': 'red'}
    def Blue(self):
        return {'color': 'blue'}

class Size():
    def __init__(self):
        pass
    def Small(self):
        return {'size': 'small'}
    def Big(self):
        return {'site': 'big'}
```

2. Cart & Checkout

```
class Cart:
    def __init__(self):
        self.products = []
    def add(self, obj):
        self.products.append(obj)
    def notify(self):
        for obj in self.products:
            obj.notify(len(self.products))
    def price(self):
        for obj in self.products:
            print(obj.price)

class Checkout():
    def __init__(self,cart):
        self.cart = cart
        self.products = self.cart.products
        self.totals = 0
        self.accounting()
    def total(self):
        print(self.totals)
        return (self.totals)
    def accounting(self):
        for obj in self.products:
            self.totals = self.totals + obj.price
    def bill(self):
        pass

class Shipping():
    def __init__(self,checkout,ship):
        self.ship = ship
        checkout.totals = checkout.totals + ship.getCost()
    def cost(self):
        print(self.ship.getCost())

class UPS():
    def __init__(self):
        self.cost = 15.2
    def getCost(self):
        return self.cost

class FedEx(Shipping):
    def __init__(self):
        self.cost = 10
    def getCost(self):
        return self.cost

class Payment():
    def __init__(self, checkout):
        pass
    def payable(self):
        pass
    def tax(self):
        pass
```

2.1. 物流配送插件设计

```
                          +-------------+     +-----------------------+
User -> Goods -> Cart -> | Delivery    |  -> | Promotions components | -> Checkout
                          +-------------+     +-----------------------+
                          | rule A      |     | Promotion rule 1      |
                          | rule B      |     | Promotion rule 2      |
                          | rule C      |     | Promotion rule 3      |
                          | rule D      |     | Promotion rule 4      |
                          | rule E      |     | Promotion rule 5      |
                          +-------------+     +-----------------------+
```

数据库设计

```
+--------------+
| shipping     |
+--------------+
| id           |o---+
| name         |    |
| ...          |    |
+--------------+    |    +----------------------+
                    |    | shipping_rule        |
+--------------+    |    +----------------------+
| zone         |    |    | id                   |
+--------------+    +--->| shipping_id          |
| id           |o------->| zone_id              |
| name         |         | plugin               |
|              |         | ...                  |
+--------------+         +----------------------+

+--------------+
| delivery     |
+--------------+
| id           |
| user_id      |o------>
| address_id   |o------>
+--------------+
```

3. 促销优惠组件设计

设计思想是，在购物车结算前匹配促销规则计算出最终优惠金额

```
                                        +-----------------------+
User -> Goods -> Cart -> Delivery -> | Promotions components | -> Checkout
                                        +-----------------------+
                                        | Promotion rule 1      |
                                        | Promotion rule 2      |
                                        | Promotion rule 3      |
                                        | Promotion rule 4      |
                                        | Promotion rule 5      |
                                        +-----------------------+
```

数据库设计

```
 +--------------+
 | goods        |
 +--------------+
 | id           |o---+
 | ...          |    |
 | ...          |    |
 +--------------+    |    +----------------------+
                     |    | goods_promotion_rule |
 +--------------+    |    +----------------------+
 | promotion    |    |    | id                   |
 +--------------+    +--->| goods_id             |
 | id           |o------>| promotion_id         |
 | name         |         | ...                  |
 | plugin       |         | ...                  |
 +--------------+         +----------------------+
```

goods_promotion_rule 负责将商品与促销规则关系建立起来

promotion 是促销脚本，我方案是使用 lua 编写促销脚本，plugin用于存储lua文件地址

这样做的好处是不用因为促销规则改变而重新修改程序，单独制作 lua脚本即可，

以上仅仅提供一个思路，你还可以建立一个goods_promotion_group将促销商品分组，然后再与goods_promotion_rule建立关系。

另外在购物车中会同时出现多种促销规则，也是要考虑的。

第 34 章 微信公众平台

1. 微信公众平台原理

设置方法: 高级功能 > 开发模式

```
URL：http://wechat.example.com/test.php
Token：a85f0785254000cd942efsef
```

原理，当用户添加公众号后，发送文本，语音等等消息，微信就会将内容Post的你的URL，然后读取返回的XML内容

2. 微信公众平台通常提供的服务模式

通常做法是问答交互方式，

问：天气
答：今天天气白天20度，晚上19度，东南风。

问：附近商场
答：茂业 500米， 天虹 100米

提供菜单选择方式。

[1] 天气
[2] 周边信息
[3] 电影院
[4] 饮食
[5] 新闻焦点

当选玩家输入 4 系统回复

[31] 肯德基
[32] 麦当劳
[33] 必胜客

玩家输入 31 系统回复

[311] XXXXXXXXXXXXXXXX

选玩家输入 5 系统回复
[51] 北京P2.5XXXXX
[52] 深圳推出XXXX政策
[52] 奥巴马XXXXXX

选玩家输入 52 系统回复

XXXXXXXXXXXXXXXXXXXX

图片
xxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxx

3. 微信公众平台开发

我看到网上很多人做法都是这样实现的

```
if input = '天气' {
        ...
} else if(input = '饮食'){
        ...
} else if(xxx){
....
} ....
....
```

```
switch (input){
        case '天气':
                xxxx
        case '饮食':
                xxxx
        case xxxx
                xxxxx
                ...
                ...
        default:
                xxxx
}
```

稍微高级的做法是，定义一个数组，或者一个hashmap,或者使用数据库实现key,value定义。然后判断keyword 是否存在，如果存在就处理 key 所对应的 value。

这样的做法会导致后期，极难维护，可读性极差，增加一个需求，就增加一段代码，新的代码会影响整个程序。国内开发者很喜欢使用if来拼接一个sql语句，这是坑爹的写法。

下面谈谈我思路，我将采用传统的MVC模式，

```
微信  ---post xml---> 入口URL
                      |
                      V
              +----------------+
              | Router         |     这里是路由映射，用户输入keyword 映射到 controller 上，
              +----------------+
                      |
                      V
              +----------------+
              | Controller     |     Controller 会加载请求的class 做一系列处理，包括数据模型处理，视图渲染
              +----------------+
                      |
                      V
+--------------- Application -------------------+
| class xxx | class xxx | class xxx | class xxx |
+-----------------------------------------------+
                      |
                      V
               +--------------+
               | Model        |     处理数据
               +--------------+
                      |
                      V
               +--------------+
               | View         |     最后呈现xml。 微信会通过返回的xml将消息推送给用户
```

```
                +--------------+
```

任务处理图，程序应该有一个入口，多个出口，而不是层层循环，层层判断。一半来说for/if/while等等 超过三层，程序阅读就会非常困难。

```
                                .---> 终结
                               /                        .-->  终结
                              /            .--> 终结               /
                             /            /           /
微信 ---> Post XML ---> 接口程序 ---------------------------------------------->  - 终结
                             \      \            \
                              \      `---> 终结               \
                               \                    `--->  终结
                                `--->  终结
```

实现菜单结构

```
<menu>
        <menuitem>
                <item>[1] 天气 </item>
                <controller></controller>
        <menuitem>
        <menuitem>
                <item>[2] 新闻 </item>
                <controller></controller>
        <menuitem>
        <menuitem>
                <item>[3] 饮食 </item>
                <submenu>
                        <menuitem>
                                <item>[31] 肯德基 </item>
                                <controller></controller>
                        </menuitem>
                        <menuitem>
                                <item>[32] 麦当劳 </item>
                                <controller></controller>
                        </menuitem>
                </submenu>
        <menuitem>
</menu>
```

XML 不太灵活，下面是数据库方案

```
CREATE TABLE menu
(
  id serial NOT NULL,
  mid integer, -- mid 字段
  menuitem character varying NOT NULL, -- menuitem 字段
  controller character varying, -- 映射控制器
  submenu_id integer, -- 子菜单ID
  status boolean DEFAULT true, -- 启用，禁用状态
  ctime timestamp without time zone DEFAULT now(), -- 创建时间
  mtime timestamp without time zone DEFAULT now(), -- 修改时间
  CONSTRAINT id PRIMARY KEY (id),
```

```
  CONSTRAINT submenu_id FOREIGN KEY (submenu_id)
      REFERENCES menu (mid) MATCH SIMPLE
      ON UPDATE RESTRICT ON DELETE RESTRICT,
  CONSTRAINT mid UNIQUE (mid)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE menu
  OWNER TO dba;
COMMENT ON TABLE menu
  IS 'menu table';
COMMENT ON COLUMN menu.mid IS 'mid 字段';
COMMENT ON COLUMN menu.menuitem IS 'menuitem 字段';
COMMENT ON COLUMN menu.controller IS '映射控制器';
COMMENT ON COLUMN menu.submenu_id IS '子菜单ID';
COMMENT ON COLUMN menu.status IS '启用，禁用状态';
COMMENT ON COLUMN menu.ctime IS '创建时间';
COMMENT ON COLUMN menu.mtime IS '修改时间';
```

数据

```
INSERT INTO "menu" ("mid", "menuitem", "controller", "submenu_id", "status") VALUES (1, '天气', 'w
INSERT INTO "menu" ("mid", "menuitem", "controller", "submenu_id", "status") VALUES (2, '新闻焦点'
INSERT INTO "menu" ("mid", "menuitem", "controller", "submenu_id", "status") VALUES (21, '国内新闻
INSERT INTO "menu" ("mid", "menuitem", "controller", "submenu_id", "status") VALUES (22, '国际新闻
```

这里id字段可有可无，实际上mid可以设置为主键，考虑到中国人习惯性才增加了id. submenu_id外键指向了mid 而没有指向id. 因为id是serial会顺序增加，会使整个菜单排序混乱。这样有也缺点，就是菜单项不能超过十个。

接下来实现路由到控制器的分发。

```
关注：显示菜单
[1] 天气
[2] 新闻焦点

发送：1
取出weather，实例化 weather 类 执行index() 方法。 返回天气预报
$weather = new Weather()

发送：2
[21] 国内新闻
[22] 国际新闻

发送：21
实例化 news 类，构造方法参数指定为 1 返回国内新闻列表
$news = new News(1);
```

当 submenu_id 为 NULL　时表示他有子菜单，如果非 NULL 就取controller参数。

接下来要做的就是需求增加，只需要在menu表中增加一个记录，然后开发对应的controller. 有一些不使用的项目随时可以将status设置为禁用状态

附录 A. 附录

术语表

SNMP

ANSI

(American National Standards Institute) This group is the U.S. member organization that belongs to the ISO, the International Organization for Standardization.

CSS

CSS是Cascading style Sheets的简称，中文译作“层叠样式表单”，在主页制作时采用CSS技术，可以有效地对页面的布局、字体、颜色、背景和其它效果实现更加精确的控制。

集群(Cluster)

所谓集群是指一组独立的计算机系统构成的一个松耦合的多处理器系统，它们之间通过网络实现进程间的通信。应用程序可以通过网络共享内存进行消息传送，实现分布式计算机。

负载均衡(Load Balancing)

网络的负载均衡是一种动态均衡技术，通过一些工具实时地分析数据包，掌握网络中的数据流量状况，把任务合理均衡地分配出去。这种技术基于现有网络结构，提供了一种扩展服务器带宽和增加服务器吞吐量的廉价有效的方法，加强了网络数据处理能力，提高了网络的灵活性和可用性。