新浪动态应用平台开发实践

2009年8月23日

sina 新浪

互联网公司技术架构系列资料

由  为您悉心整理

快简历

KuaiJianLi.com

/* 让工作重新关于成长和成就、关于快乐和分享、关于梦想和荣光 */

目录

- 平台介绍
  - 平台目的
  - 系统架构
  - 可扩展性
  - 多机房体系
  - 服务容量
  - 使用的开源软件
- 经验分享
  - 系统架构设计
  - 标准化配置
  - 约束和限制
  - 监控报警
  - 关于性能
  - 容量规划

平台介绍

sina 新浪

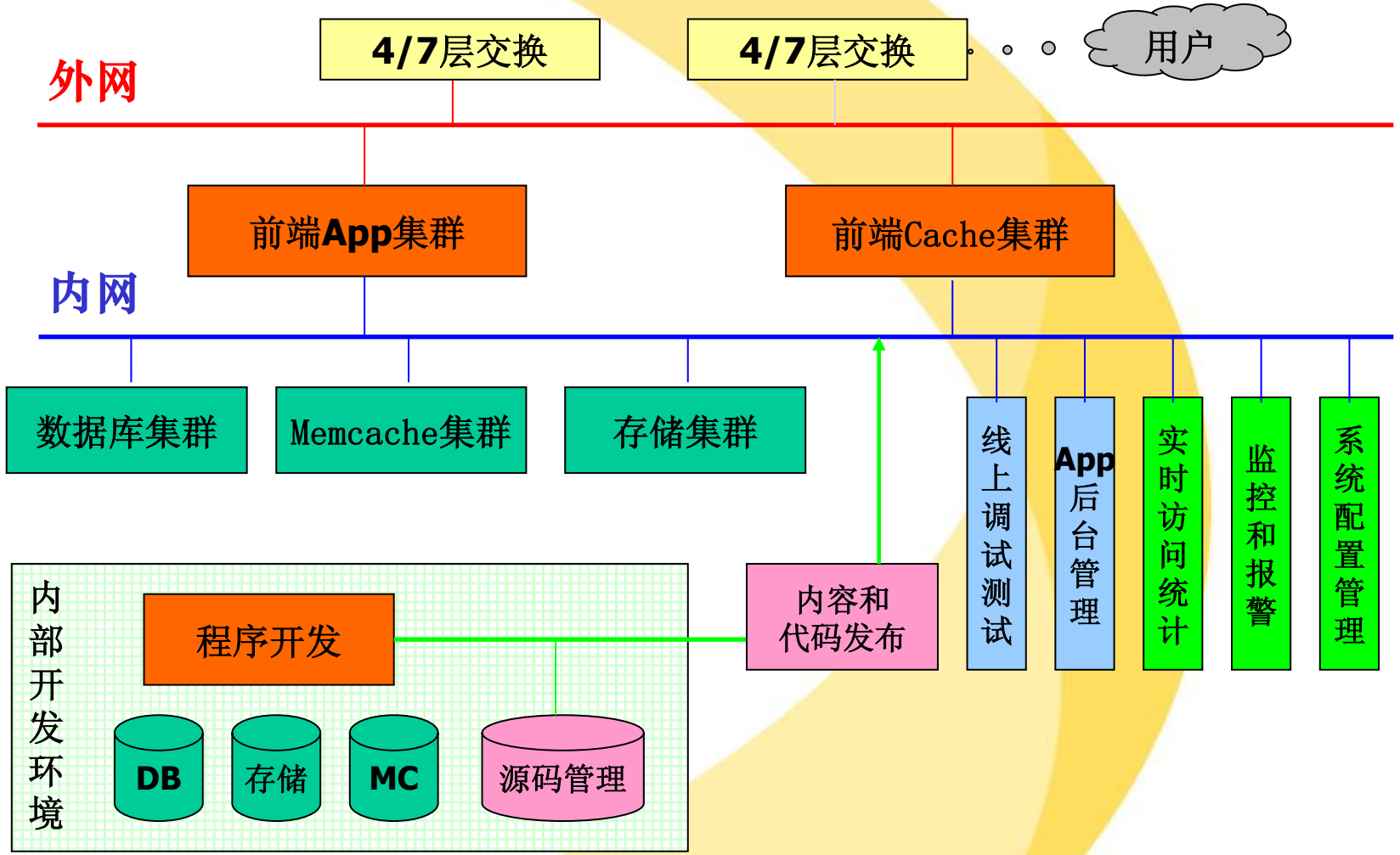
- **功能**
  - 提供：PHP程序托管服务的虚拟主机环境
  - 提供：数据库、存储、Memcached 等基础服务
  - 提供：代码分发、开发环境、线上调试、后台管理、Cron 配置功能
- **历史**
  - 1.0 版本
    - 2004年 由 CTO 李嵩波先生立项，9月份完成 1.0 环境搭建
    - 2005年 资源监控系统开发，实现项目资源的可用性监控和报警
    - 2005年 生产环境部分系统单元开始使用 Xen 虚拟机技术
    - 2006年 基于 MySQL 的数据库集群系统开发完成
  - 2.0 版本
    - 2006年 完成 2.0 环境的搭建
    - 2006年 托管项目数量突破 100 个，每日程序请求突破 3 亿次
    - 2006年 实时访问统计系统开发，实现恶意访问控制机制
    - 2007年 子系统或代码被其它业务使用：**代码分发、数据库集群、软件包仓库**
    - 2008年 基于数据库集群系统，发展为独立的数据库服务平台
    - 2008年 基于虚拟化技术经验，开发了虚拟机服务平台，并投入使用

平台目的

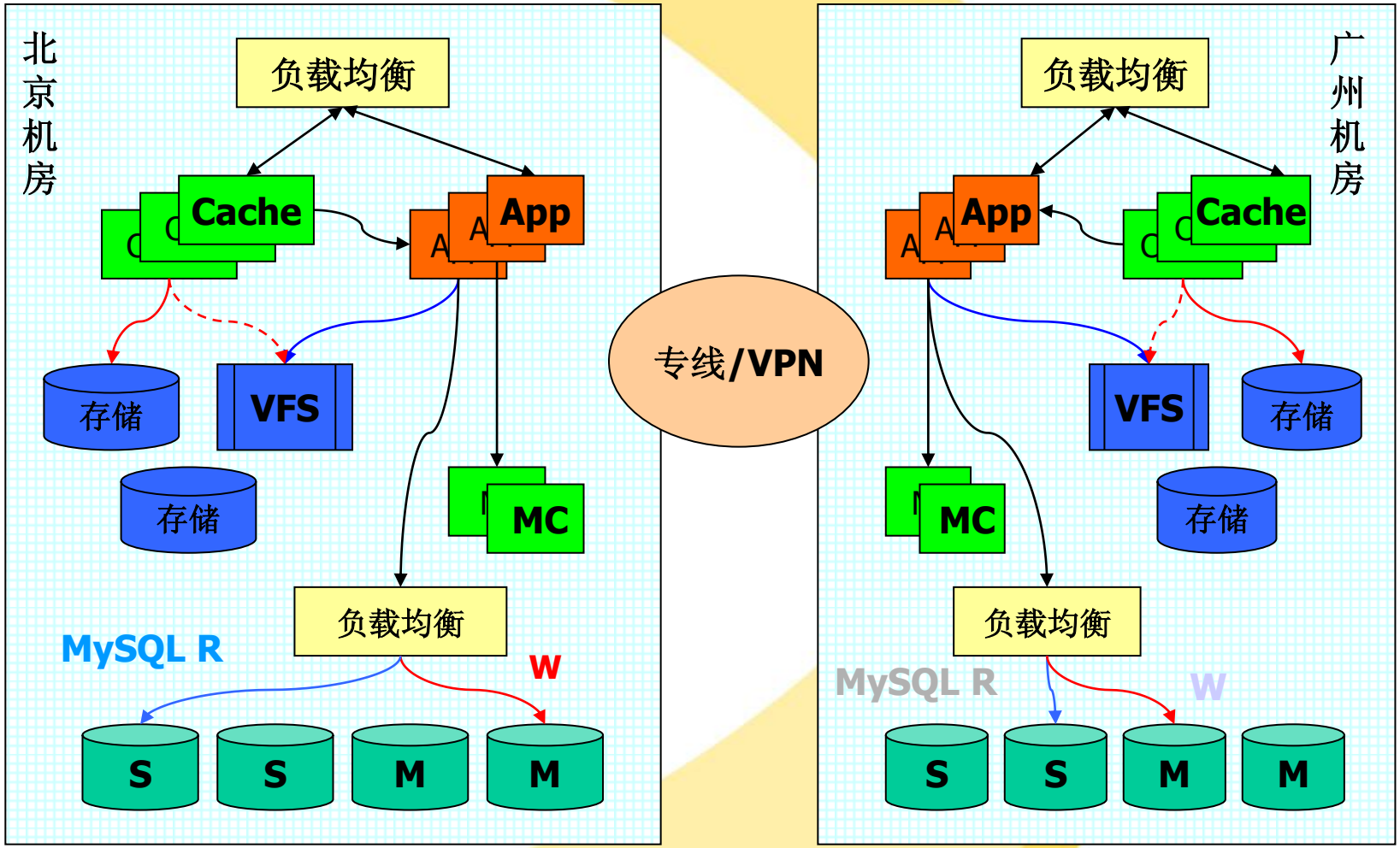
sina 新浪

- **高可用 + 低成本 + 可扩展**
  - 没有单点故障
  - 基于LAMP架构和其它开源软件技术
  - 平台资源被多个业务共享从而提高资产利用率
  - 增加服务器即可提升服务容量并且对应用透明
- **统一的运行环境**
  - 统一标准的系统环境配置，方便开发，方便运维
  - 开发环境发布代码到生产环境即可正常运行
- **简单映像的体系架构**
  - 让开发人员只看到一个Web前端、一个DB、一个存储、一个Cache
  - 底层系统集群技术、节点多机房分布等技术对开发人员透明
- **应用开发和平台开发分离**
  - 技术团队专业分工，各自作擅长的事情
  - 开发人员在已有系统平台迅速开发应用，缩短开发周期，提高效率

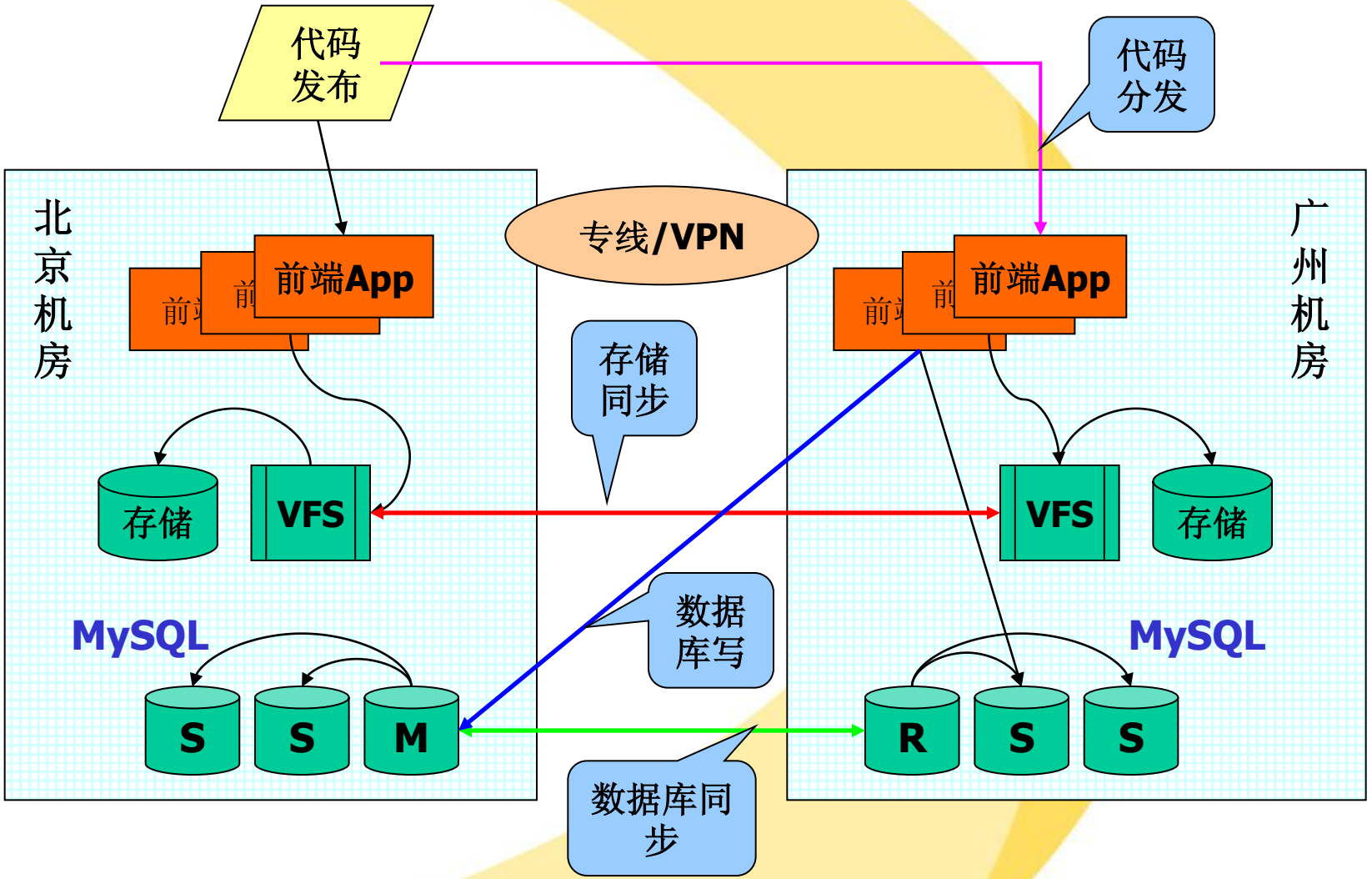
系统架构



性能和可扩展性

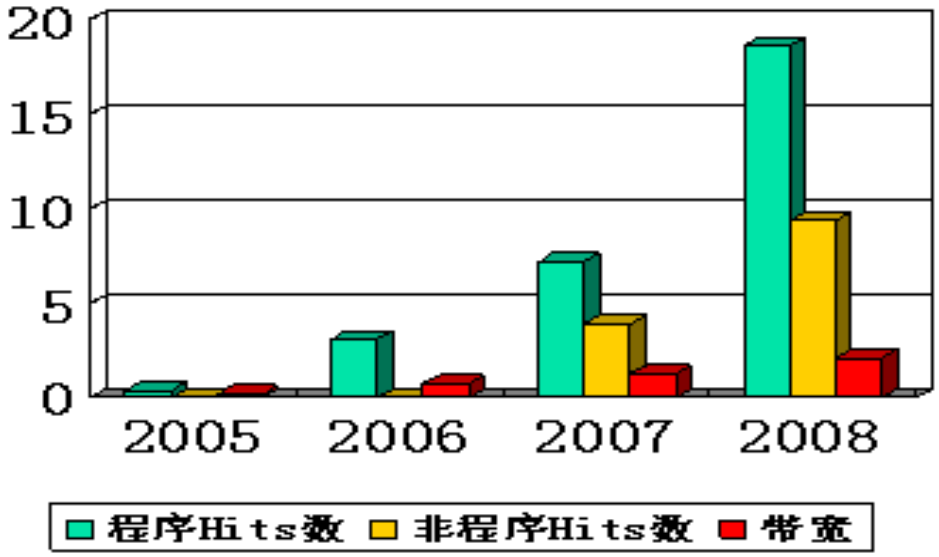


多机房分布式体系



历年访问量

- **程序Hits量每年都翻倍**
  - 每年增加一定数量的机器即可满足需求



使用的开源软件

- LAMP
- Memcached
- Squid-Cache
- Heartbeat + DRBD
- Xen
- Cfengine
- Bind DNS + Mon
- Haproxy

经验分享－系统架构设计

1. **借鉴成功的经验**
   1. LiveJournal (Memcached、MogileFS)
   2. Yahoo!、Google、Youtube、Facebook

2. **使用成熟的软件**
   1. 除非别无选择才去自己开发代码
   2. 自己开发代码也许容易，但长期维护很不容易
   3. 深入了解你所使用的成熟的软件，用好它！

3. **一定程度的底层封装和隔离**
   1. 目的：使系统具有可扩展性

4. **简单 = 高效**
   1. 不要过于追求完美，够用就好！
   2. 循序渐进的开发过程，分阶段达成目标
   3. 简单 (结构设计 + 实现方法) = 高效 (开发过程 + 故障排查)
   4. 简单是指：简单的体系结构、更少的系统单元、更少的处理流程、更少的代码编写

经验分享－标准化配置

1. 标准化配置的好处

1. 使开发环境和生产环境分离
2. 系统环境配置变更后，系统管理员更新资源配置文件即可

2. 运行环境配置

1. 软件包、版本、路径、用户
2. 常用配置文件统一管理：httpd.conf, php.ini, hosts
3. Apache 和 Squid 日志记录每个请求执行时长，MySQL打开慢日志

3. 资源配置

1. 资源包括：数据库账户、存储路径、Memcached、数据目录等
2. 如数据库的IP和端口的资源变量名:
    1. SINASRV_DB_HOST
    2. SINASRV_DB_PORT
3. **通过 Apache 环境变量取值: $dbPort = $_SERVER[“SINASRV_DB_PORT”]**
4. **通过解析配置文件获取: $SINASRV_CFG = parse_ini_file($_SERVER[“DOCUMENT_ROOT“] . “/system/SINASRV_CONFIG”)**

经验分享－约定和限制

1. 基本的约定

1. 数据库读写分离
2. 程序不允许写磁盘文件 Cache，尽量使用 Memcached
3. 程序不允许使用绝对路径
4. 程序目录可执行不可写，数据目录可写不可执行
5. 访问 MySQL 和 Memcached 不使用长连接模式
6. 数据库查询时间大于 1 秒的次数每分钟不超过 1 次
7. 更多。。。

2. 必要的限制

1. PHP 中禁用一些危险函数
2. 程序的执行时长不能超过 30 秒
3. 程序调用外部网络服务超时小于 30 秒
4. 程序可用的内存限制在 18M 字节以内
5. 项目的域名虚拟主机有 MaxClient 限制
6. 项目的数据库账户有 MaxConnection 限制
7. 更多。。。

经验分享－监控报警

sina新浪

1. 原则：详细的感知运行状况，一定程度的自动控制和修复能力

2. 统计分析
   1. WWW日志 (每5分钟)
   2. MySQL查询 (每5分钟)、MySQL慢日志 (每日)
   3. 错误日志 (WWW、系统日志)
   4. 应用程序日志
3. 监控
   1. 可用性 (ping/port/磁盘…)
   2. 容　量 (带宽/磁盘/CPU…)
   3. 质　量 (响应速度/成功率…)
   4. 异　常 (进程/访问量…)
4. 主动处理尝试
   1. 拒绝恶意访问的IP地址
   2. 暂停或重启故障服务、Kill异常进程等等
5. 报警
   1. 仪表板、邮件、短信、 IM

经验分享－关于性能

1. **在保证高可用的前提下追求高性能**

2. **程序开发语言使用**
   1. 如果系统中的性能瓶颈不在CPU上，那就选择解释性语言吧！

3. **服务器软件的合理使用**
   1. 每种软件都有最适合的环境，如 Nginx/Lighttpd 适合高并发高IO的服务

4. **影响性能的软硬件因素**
   1. 硬件：各种应用场景中常见的硬件性能瓶颈和导致的后果现象
   2. 软件：事件处理模型、请求处理时长、内存使用、CPU使用、磁盘使用

5. **网络服务的性能指标**
   1. 每秒请求数、并发度、网络吞吐量

6. **性能分析**
   1. **不要只看表面现象，深入挖掘根本问题所在，对症下药解决主要矛盾**
   2. **给系统把脉的方法：负载->CPU(us+sy+io)/磁盘IO/网络流量/内存使用**

经验分享－容量规划

1. **日常情况合理的使用率**
    1. 负载小于CPU核心数，如单CPU 4核应该小于4
    2. 各项资源使用率在40%以内，峰值情况下资源使用率不超过80%
2. **使用率超出 60% 时就应该引起注意**
    1. 分析是正常的业务增长还是不合理的使用？
        1. 如果是正常的增长应该考虑扩容
    2. 分析是否为异常的用户访问，是否遭到攻击？
    3. 分析是否可用对业务逻辑进行优化？
        1. 性能分析和优化
3. **保留 20% CPU 资源给系统管理和监控使用**
    1. 如果业务系统把资源全部耗尽，那么用户的访问体验也不会太好