1 绪论

1.1 课题研究背景及意义

当今社会是一个“信息爆炸”的社会，尤其是进入互联网时代后，人们每天接触 的信息和互联网产品远远大于互联网发展初期的年代[2]。随着中国互联网技术的 不断发展以及 WEB 应用开发成本的不断降低，人们对于 WEB 应用的需求不再满 足于通过缓慢的加载和复杂的操作获取资讯[3]，许多的创业者希望通过开发一个 WEB 应用实现自己创业梦想的时代已经无法获得投资人的青睐。而且，由于近几 年 WEB 信息泄露的新闻不断进入消费者的视野，用户对于 WEB 应用的安全性体 验也提出了更高的要求。

随着互联网技术进一步普及，基于模型-视图-控制器 (Model-View-Controller, MVC) 模式的 WEB 应用程序[4] 被广大开发者所使用，目前主流的 WEB 应用程序 均在使用此框架。使用 MVC 框架，可以将 WEB 应用进行清晰的分层开发，前端 开发人员主要负责页面的呈现方式和用户体验，后台人员则主要负责应用的逻辑 实现以及数据结构的搭建，极大的降低了研发成本。

随着信息化的进一步普及，数据库的使用越来越广泛，数据成为一个应用甚 至也个企业最重要的价值体现，越来越多的企业发展离不开数据库。因此数据库 的稳定性和安全性也称为很多企业研发的重点。通过设计数据的管理和使用机制， 在提高数据库使用效率的同时，保障数据的完整性和安全性是企业运维人员的在 运维过程中的重要任务。

除此之外，如何实现WEB应用的高可用性和分布式服务也是保障用户体验的 一个很重要的部分，通过设计基于应用和服务器的不同层级不同纬度的优化策略， 提升应用的高可用性也是在 WEB 应用开发中必须要注意的。

1.2 国内外研究现状近几年，随着计算机技术和 Internet 的迅速发展，网络的信息量和访问量成几何数级增长，网络拥塞和服务超载日益成为网络及其服务器必须面对的严峻问 题[5]。2006 年新华网被黑事件、2010 年的百度域名劫持时间、2011 年的 CSDN 用户 数据泄漏事件以及 2015 年网易邮箱密码泄漏事件等事件无不说明系统优化的作用 和意义。1

基于 Spring MVC 架构的 WEB 应用的系统优化策略研究

目前大部分互联网产品在开始发展的阶段都重视了产品的设计和功能，然而 在产品的稳定性和安全性方面却考虑不足，这在一定程度上增加了互联网产品在 推广过程中潜在的风险。因此，在开发互联网产品的同时，通过一系列的优化策 略，提升应用的体验，增强应用的稳定性和安全性必须得到充分的认识。

1.3 本文主要研究内容 1.3.1 主要工作

论文在满足应用正常运行的基础上，通过不同维度的优化实践研究基于 Spring MVC 架构 WEB 应用的系统优化策略。首先通过搭建基于 Spring MVC 的 WEB 测 试应用，在实际用户使用的过程中通过不断开发和调整系统的优化策略，提升应 用的用户体验和应用的稳定性，探索行之有效的系统优化策略。

本文的研究对象主要有以下内容:

(1) 对 Jenkins 持续集成环境研究。自动化部署和代码检测是保证基于 WEB 的 应用产品质量的一个重要环节，通过研究部署 Jenkins 集成测试环境，对编写的代 码进行版本控制、自动化构建和代码测试，研究持续集成方案对于 WEB 应用系统 优化策略的影响。

(2) 对 Couchbase 缓存机制进行研究。目前大多数的 WEB 应用对于缓存性能 的优化还没有足够的重视，通过开发针对 WEB 应用的 Couchbase 缓存系统，研究 Couchbase 的缓存机制测试有效的缓存机制对于系统性能提升和用户体验的影响， 探索可用的系统优化策略。

(3) 对 Docker 容器编排技术进行研究。随着用户的不断增加，单一节点的 WEB 应用或数据库应用已经无法满足用户的需求，如何快速的部署新的应用节点提升 应用性能成为开发者关注的问题。通过构建基于Docker容器编排技术的应用容器， 在新的服务器快速部署新的应用，并加入到应用集群中去，探索 Docker 机制对于 系统优化策略的影响。

(4) 对数据库的主从复制进行研究。随着应用的发展，数据逐渐成为应用最有 价值的部分，如何更好的保证数据的完整性以及安全性在应用的维护过程中日益 重要。通过研究数据库的主从复制和延时复制方案，探索数据库优化对于系统安 全性优化的策略。

(5) 对基于 API 的监控和应急措施方案进行研究。随着服务节点的不断增加， 服务的高可用和服务的健康性检查时 WEB 应用的运维人员在维护过程中必须要 注意的方面。通过开发基于 API 的服务器、服务监控系统以及基于 API 的应急处 理系统研究 API 操作对于系统的快速检测和快速恢复的影响。2

基于 Spring MVC 架构的 WEB 应用的系统优化策略研究

(6) 其他方面，研究 WEB 应用的搭建过程、Tomcat 的相关配置、负载均衡以

及阿里云的相关配置研究 WEB 应用和服务器自身优化对于系统性能的影响。 1.3.2 论文目标

本论文致力于分析目前很多 WEB 应用开发过程中存在的问题和不完善的地 方，并探索 WEB 应用性能和服务器性能的优化方案。

在 Linux 平台上，使用 JAVA 语言和 MySQL 数据搭建一个基于 Spring MVC 架构的 WEB 应用，通过 Tomcat 实现应用的访问和测试，通过设计不同的优化方 案对 WEB 进行测试，研究在实际应用中有价值的优化策略。

系统性能优化策略研究主要包含应用层面、数据层面和服务器层面三个层面。 在应用层面，通过持续集成、代码审核实现应用的稳定性和安全性。在数据库层 面，通过多数据库主从复制、负载均衡和备份恢复等策略实现数据的高可用和稳 定性。在服务器层面，通过对服务的监控和服务器的监控、服务器间负载均衡的 配置、基于 API 的自动化 failover 等策略保障服务的正常运行和高负载应对。

1.4 论文组织结构

基于 Spring MVC 架构 WEB 应用的系统优化策略的研究是按照计划逐步完成 的，本人将此研究分为六个部分:

第一章:绪论。本章主要介绍 WEB 应用开发的现状和系统优化策略研究的意 义，明确了在 WEB 应用开发过程中对于应用和服务器进行优化的必要性和重要 性。介绍了本论文的主要工作内容、目标以及创新点。最后介绍了论文的编排结 构。

第二章:WEB 应用开发介绍。本章主要介绍了前端和后端的开发框架和开发 流程以及数据库的搭建过程。然后介绍了 Tomcat 的配置过程。之后介绍了 Jenkins 持续集成环境的搭建和使用过程。

第三章:应用性能优化介绍。本章主要介绍了在 WEB 应用性能优化方面册主 要方法和策略，主要包括 Couchbase 缓存优化，Tomcat 高性能 Apr 配置，Docker 分布式优化以及 SLB 负载均衡优化等方面。同时对应用的安全性和稳定性进行了 分析。

第四章:数据优化。本章主要介绍了 MySQL 数据库在设计和开发过程的优化 策略，主要包括基于数据稳定性的复制方案，基于高可用的负载均衡方案，基于 数据完整性的备份和恢复方案。同时总结了出现问题时的解决方案。 第五章:服务监控与应急措施优化。本章主要介绍了服务器层面的优化策略， 3

基于 Spring MVC 架构的 WEB 应用的系统优化策略研究

主要包括多服务器的心跳监听服务配置，服务性能的监控脚本实现，基于 API 的 应急措施处置以及服务器安全性配置等方面的优化策略。同时总结了多种方案整 合使用的策略。

第六章:总结与展望。本章主要总结了不同优化策略对于系统性能的影响以 及在研究过程中获得的经验，并对优化策略的进一步研究做了简单的展望。

2 WEB 应用开发及持续集成

本论文的研究目的是对于开发的 WEB 应用进行应用、数据库和服务器层级进 行优化，通过对比不同优化策略对于系统性能提升的影响研究一套可行的系统优 化策略。为了实现优化的目的，首先需要创建一个 WEB 应用或者对现有的应用进 行优化，本人在研究生期间，参与了海信集团智能商用公司的小微商户平台系统 的开发，主要工作是后台开发和服务器运维工作，因此本论文将基于参与的小微 商户 WEB 平台进行系统优化策略的研究。

在优化之前，首先需要描述作为优化原型的小微商户WEB平台的平台的开发 框架以及应用部署流程。

2.1 Spring MVC 开发框架

Spring MVC 是一种基于 Web 开发的系统架构，目前主流的系统框架还有 Struts，但是 Spring MVC 架构引起优势依然处于不可替代的位置。在 Spring MVC 架构中，系统的终极目标被解聚、分块，并在不同的模块中表达出来，如此系统 的耦合度得到大幅降低。同时，该框架的整体运行也是在模块驱动的基础之上的， 用户首先发出请求，然后系统予以相应，即常规的请求-相应模型。

2.1.1 Spring MVC 架构的特点

Spring MVC 架构不同于传统的体系架构，在其发展的过程中吸收其他系统的 特点，如支持 Userful 风格，灵活的本地化等。这些特征保证了 Spring MVC 框架在 发展过程中，更加受到开发者的青睐。同时，吸收了前人的经验以后，Spring MVC 框架也具备专属的特点，比如用 Spring MVC 框架设计的 WEB 系统其耦合度较 低，系统总体表现出简洁明了的风格，这样用户就能更快捷的掌握数据库系统的 操作[2]。

Spring MVC 框架是在 Spring 框架的基础上发展起来的，因此 Spring MVC 框 架集成了 Spring 框架的所有功能，而且两者在相互继承方面表现出很好的融合性。 这一特点给系统开发带来了很多方便。除此之外，Spring MVC 模型在图形兼容性 和数据验证的灵活性上都有先天的优势。正式因为这些特点，使 Spring MVC 模型 在实际应用中发挥出巨大的优势，同时也在程序员中受到青睐。本文的 WEB 应用 就是建立在该框架之上的。5

基于 Spring MVC 架构的 WEB 应用的系统优化策略研究 2.1.2 Spring MVC 框架处理流程

Spring MVC 架构是一个基因用户请求的系统架构，其过程也要求有 Web 驱动 的参加。具体过程现有用户发送请求到前端控制器，这时候前端控制器将请求委 托给处理器;于是处理器就开始处理请求，即调用业务对象进行分析，分析完毕后 将模型数据返回给处理器;然后处理器再将处理结果通过模型和视图返回给前端 控制器;此时前端控制器显示的数据用户还无法处理，还需要进行视图渲染，最 终在前端控制器显示出来，即用户发送的请求产生了响应[6]。整个过程参见图 2-1， 此图反映了用户请求-响应的过程。

2.1.2.1 体系的三层架构

MVC(Model View Controller), 即视图、模型和控制器三个英文单词的首字母缩 写，按照应用的流程本系统可以在 Spring MVC 架构下分为三层，具体的架构为显 示层、控制层和数据访问层。

1. 显示层。主要有 Spring 和 JSP 经过页面 REQUEST 请求，Spring 的 URL Mapping 对应到响应的控制器，通过对页面发送的请求，Spring 自动的去寻 找响应的控制类，并将结构展示在页面上。

2. 控制层。也就是业务层，是系统中最重要的一层，这层主要是接受URL发送 过来的业务请求，然后去访问 Service 层，把需要的数据装入缓存中，任何通 知 Action 进行跳转。

3. 数据访问层。这层最重要的作用是对数据库中数据以及 XML 文件中的数据 进行访问及操作，并把操作的结果反馈到业务逻辑层，将相应的结果交由下 一层去处理。这个三层架构的体系的应用很好的解决了系统资源的分配问题，而且将系统的耦 合度降到了可以接受的最低水平。同时，关于异步处理问题上，本系统使用了 FDP 框架，FDP 框架是对 AJAX 的一种轻量级的封装，它的目标是使 AJAX 和 XML 6

基于 Spring MVC 架构的 WEB 应用的系统优化策略研究

Http Request 更简单。通过 FDP 框架，我们只需要在相应的配置对应的 JAVA 方法 以及相对于的实体类，我们很容易的就可以实现许多异步操作，使一些操作不用 刷新页面即可达到相应的效果。可以认为 FDP 框架是对 MVC 三层框架的一种拓 展，其本质还是在 MVC 框架内的。由此可见，笔者在系统体系的选择上更倾向于 架构简单、可移植性好的系统框架模型。

2.2 应用开发工具

在开发基于 Spring MVC 的 WEB 应用过程中，需要用到的基础编程语言是 JAVA，系统的架构采用的 MVC 三层架构。这些都属于手段，但是想要达到最终目 的则需要用到一些基础的软件工具，这些工具的使用将加快工作效率。

2.2.1 IntelliJ IDEA

IntelliJ IDEA 是一款 Java 集中开发环境工具软件，由捷克软件公司 JetBrains 在 2001 年 1 月时推出最初版[7]。该软件被认为是当前 Java 开发效率最快的 IDE 工 具。它集成了开发过程中实用的众多功能，几乎可以不用鼠标可以方便的完成你 要做的任何事情，最大程度的加快开发的速度。简单而又功能强大。与其他的一 些繁冗而复杂的 IDE 工具有鲜明的对比[8]。

2.2.2 Maven

Apache Maven 是由 Apache 软件基金会所提供德一个项目管理及自动构建工 具[9]。它基于项目对象模型概念、能够实现一个 Java 项目的构建和依赖管理。本 论文所涉及的 WEB 应用就是使用 Maven 来构建 Java Web 项目。

2.2.3 Tomcat

WEB 应用的 Web 服务器采用的是有 Apache 软件基金会 (Apache Software Foundation) 下属的 Jakarta 项目开发的一个 Servlet 容器，由于其本身也包含一个 HTTP 服务器，因此也常常被用作一个单独的 Web 服务器[10]。Tomcat7 支持最新 的 Servlet 3.0 规范，而且技术先进、稳定性强一级免费的特点，深受 Java 爱好者 的喜爱并得到了部分软件开发商的认可，成为目前比较流行的 Web 应用服务器。

2.2.4 MySQLMySQl 是个小型关系型数据库管理系统，之所以使用 MySQL 是因为 MySQL 是一款免费的数据库管理系统，而且其建议的操作以及其兼容性都是其优点[11]。

MySQL 的特点主要有:

• 为多种編程语言提供了API。这些編程语言包括C、C++、C♯、Java、Perl、PHP、

Python、Ruby 等。

• 支持多线程，充分利用 CPU 资源，支持多用户。

• 提供 TCP/IP、ODBC 和 JDBC 等多种数据库连接途径。

• 提供用于管理、检查、優化数据库操作的管理工具。

• 可以处理拥有上千万条记录的大型数据库。

2.2.5 Couchbase

在互联网时代，我们面对的是更多的客户端，更低的请求延迟，这就需要对数 据做大量的 Cache 以提高读写速度[12]。目前主流的 Cache 系统 memcached 和 redis 虽然有很熟的解决方案，但是也都有其局限性 [13] :

• Cluster 支持不够。在扩容、负载均衡、高可用等方面存在明显不足。

• 持久化支持不好，出现问题后恢复的代价大。memcached完全不支持持久化，

redis 的持久化会造成系统间歇性的负载很高。

Couchbase 是一个 NoSQL 数据库，它是世界各国的开发者在 2011 年推出的，由于 它有良好的 cluster 支持、异步持久化的支持得到了众多开发者的青睐，他的特点 主要有:

• Couchbase 的对等网设计，smart client 直接获取整的集群的信息，在客户端 实现负载均衡，整个集群没有单点失效，并且完全支持平行扩展。

• vBucket 的引入，完全实现了 auto sharding，可以方便灵活的把数据的子集在 不同节点上移动，以实现集群动态管理。

• Couchbase 有一个非常专业的 web 管理界面，并且支持通过 RESTful API 管 理，这也是 memcached, redis 不能企及的。

2.3 应用开发流程WEB 应用在开发的时候设计为前后端分离，通过 FDP 平台实现前端到后端 的请求，所以本论文测试 WEB 应用的开发主要分为前端开发、后端开发以及数据 库搭建三个方面。2.3.1 前端开发应用的前端相当于 MVC 架构中的视图层，主要实现用户的交互，包括页面信息的展示以及用户请求的转发和响应，在前端开发中，通过 Html 和 JavaScript 来 设计实现应用的页面展示，通过 FDP 的 Ajax 请求将前端的用户请求转发到应用

的后端。

2.3.2 后端开发

应用的后端主要分成了 Controller、Service、Pst 三层，其中 Controller 层负责 处理用户的请求然后将业务转发给 Service 层，Service 层负责实现用户的请求，通 过设计不同的业务逻辑将用户需要的数据返回，涉及到数据的读取则通过 Pst 层， Pst 层主要负责对数据库的增删改查，将结果返回到 Service 层。

2.3.3 数据库搭建

应用的数据主要分为基础数据也业务数据，其中基础数据主要包括页面的功 能板块、系统的定时任务、数据的权限设置、页面的功能逻辑等数据，业务数据主 要包括用户的信息、用户操作记录、用户的商店、商品、销售信息等。

2.4 基于 Jenkins 的持续集成方案开发

在每一次 WEB 应用的开发、上线过程中，不可避免的要将本地环境打包上传 到生产环境或者是测试环境进行解压，每一次人工的干预无疑增加了时间成本和 错误率，通过 Jenkis 设计实现应用的持续集成，这在很大程度上能够帮助开发着 实现快速的应用部署和错误重现 [14] 。

Jenkins 是一个用 Java 编写的开源的持续集成工具，它提供了软件开发的持续 集成服务，运行在 Servlet 容器中，它可以执行基于 Apache Ant 和 Apache Maven 的项目，以及任意的 Shell 脚本和 Windows 批处理命令。

2.5 本章总结

本章主要介绍了本人参与的小微项目的 WEB 平台所使用的开发框架、开发工 具、开发过程以及系统上线的持续集成方案，通过持续集成方案增强了代码上线 的稳定性，在本文后面的三章将会对本章开发的平台进行不通层级的优化，来实 现本项目的安全性和高性能。

3 应用性能优化

3.1 Couchbase 缓存优化

目前大多数的 WEB 应用产品在设计开发的过程中对数据的读取还依旧通过 直接对数据库进行增、删、改、查来实现。然而随着用户数量的增加，用户的请求 也不断的增多，这对于数据库的压力是非常大的。之前的数据操作流程参见图 3-1。

为了缓解数据库的压力，在数据的读取过程中将系统的基础数据读取到 Couchbase 缓存中，这样只需要对数据的基础数据进行一次读取即可完成数据的 加载，降低了数据库的压力。通过缓存的数据操作流程参见图 3-2。

本论文中的WEB应用的数据包含基础数据和应用数据两部分，其中基础数据 主要包括系统的基本配置数据、系统的常用变量数据、用户权限数据、枚举类型等 相对固定的数据，这些数据在用户访问应用的时候只需要加载一次即可，不需要重复的从数据中读取，而应用数据则主要包含应用内的用户信息、商户信息、交

易数据、库存数据等等不固定的数据，这些数据随着用户的时候会不断的发生变 化，而且新的数据从数据库重新读取，这些数据在用户使用时需要多次加载。针对 于以上不同数据的特点，将基础数据在用户首次登陆时加载到 Couchbase 缓存中， 以后再读取时直接从缓存中读取。

直接安装[12]

首先，通过浏览器访问https://www.couchbase.com/nosql-databases/downloads ， 在页面中选择 4.1.0 版本、支持 Red Hat 7 平台的 Couchbase Server 安装文件 couchbase-server-enterprise-4.1.0-centos7.x86\_64.rpm, 将 rpm 文件复制到服务 器中。或者直接在 shell 中通过

命令将 rpm 安装文件下载到制定路径中。

然后，通过 root 用户访问 Linux 服务器，在终端中进入到 couchbase 安装文 件所在路径，通过如下命令安装 couchbase:

安装完成后自动启动 couchbase，并且会有如下提示信息:

couchbase 默认安装位置为“/opt/couchbase/”

2. Docker 安装

Couchbase 安装完成后，需要在阿里云的安全策略中配置外网访问的端又和应用访 问的端又，相关配置参见5.4

3.1.2 Couchbase 集群配置Couchbase 服务器及可以单独运行，也可以将多个服务器组成一个集群，作为集群来运行。通过 Couchbase 集群，可以实现缓存数据的分布式存储及负载均衡，提升缓存的高可用以及系统的性能。

Couchbase 数据分布是按计算分配到多个节点上，每个节点都储存两部分数据 有效数据和副本数据，客户端对数据的操作主要是按照节点中对应的有效数据进 行操作，执行压力会部分到不同的节点，如 3-3所示:

Couchbase 的集群管理是由 erlang/otp 进行集群通信管理，集群之间使用心跳 机制进行监测服务器节点健康监测，配置参数信息是同步到每一个节点上进行储 存。整个集群以 vbucket 为单位划分映射到不同服务器节点中进行储存，划分规则 如下:

1. 均匀的分配有效 vbucket 和副本 vbucket 到不同服务器节点中; 2. 把有效数据与副本数据划分到不同物理节点中;

3. 在复制多份数据时，尽量有其它节点进行数据传播;

4. 扩展时，以最小化数据迁移量进行复制。

在 Couchbase 负载均衡中，我们所操作的每一个 bucket 会逻辑划分为 1024 个 vbucket，其数据的储存基于每个 vbucket 储存并且每个 vbucket 都会映射到相 对应的服务器节点，这种储存结构的方式叫做集群映射。如 3-4所示，当应用与 Couchbase 服务器交互时，会通过 SDK 的与服务器数据进行交互，当应用操作某 一个的 bucket 的 key 值时，在 SDK 中会通过哈希的方式计算，使用公式 crc32(key)

在设置标签页中的集群标签页下新建和配置集群信息

在服务器节点标签页下新增服务器节点

增加完服务器节点后可以看到集群的基本信息

3.1.3 Couchbase 缓存对系统性能影响

通过 ab 命令对使用应用进行压力测试，在开启 couchbase 和关闭 couchbase 的 情况下分别模拟 5 万个请求，每次并发 10 个请求来对比缓存对系统性能的影响:

Tomcat 对用户请求的处理方式有 BIO、NIO 以及 APR 三种方式，其中 BIO 模 式为 Tomcat 的默认处理方式[16]。

BIO 模式是一种阻塞式 I/O 操作，这种模式表示 Tomcat 使用的是传统 Java I/O 操作。在这种模式下对于每个请求都要创建一个线程来处理，线程开销较大，不 能处理高并发的场景，在三种模式中性能也最低。启动 tomcat 看到如图 3-8所示日 志，表示使用的是 BIO 模式:

NIO 模式是 Java SE 1.4 及后续版本提供的一种新的 I/O 操作方式。该模式是一 个基于缓冲区、并能提供非阻塞 I/O 操作的 Java API，它拥有比传统 I/O 操作 (bio) 更好的并发运行性能。启动 tomcat 看到如图 3-9所示日志，表示使用的是 NIO 模 式:

APR模式是从操作系统级别解决异步IO问题，大幅度的提高服务器的处理和 响应性能，也是 Tomcat 运行高并发应用的首选模式。启动 tomcat 看到如图 3-10所 示日志，表示使用的是 NIO 模式:

考虑到应用在大量用户访问的情况下必然会出现高并发的现象，因此调整

Tomcat 的请求处理方式为 APR 模式对于提升系统的高并发处理能力时非常必要 的[17]。

APR(Apache Portable Runtime) 是一个高可移植库，它是 Apache HTTP Server 2.x 的核心。APR 有很多用途，包括访问高级 IO 功能 (例如 sendfile,epoll 和 OpenSSL)， OS 级别功能 (随机数生成，系统状态等等)，本地进程管理 (共享内存，NT 管道和 UNIX sockets)。这些功能可以使 Tomcat 作为一个通常的前台 WEB 服务器，能更 好地和其它本地 web 技术集成，总体上让 Java 更有效率作为一个高性能 web 服务 器平台而不是简单作为后台容器。

在产品环境中，特别是直接使用 Tomcat 做 WEB 服务器的时候，应该使用 Tomcat Native 来提高其性能，要测 APR 给 tomcat 带来的好处最好的方法是在慢速 网络上(模拟 Internet)，将 Tomcat 线程数开到 300 以上的水平，然后模拟一大堆 并发请求。

如果不配 APR，基本上 300 个线程狠快就会用满，以后的请求就只好等待。但 是配上 APR 之后，并发的线程数量明显下降，从原来的 300 可能会马上下降到只 有几十，新的请求会毫无阻塞的进来。在局域网环境测，就算是 400 个并发，也是 一瞬间就处理/传输完毕，但是在真实的 Internet 环境下，页面处理时间只占 0.1% 都不到，绝大部分时间都用来页面传输。如果不用 APR，一个线程同一时间只能 处理一个用户，势必会造成阻塞。所以生产环境下用 apr 是非常必要的。