全景 Cms 内容管理服务平台

架构设计说明书

2016 年 11 月 28 日

修订记录

版本	修订人	修订日期	修订内容
V1.0	张阳	2016/11/28	创建文档

1. 引言 5
1.1 编写目的 5
1.2 项目背景 5
2. 总体设计 7
2.1 需求规定 7
2.1.1 用户管理系统 7
2.1.2 权限管理系统 7
2.1.3 新闻管理［PC、IOS、ANDROID］ 7
2.1.4 图片管理系统［PC、IOS、ANDROID］ 7
2.1.5 视频管理系统［PC、IOS、ANDROID］ 7
2.1.6 模版管理系统 8
2.1.7 专题管理系统 8
2.1.8 频道管理 8
2.1.9 碎片管理 8
2.1.10 统计管理 8
2.2 运行环境 10
2.3 基础设计概念及处理流程 11
2.3.1 架构描述 11
2.3.2 物理架构－拓扑图 12
2.3.3 逻辑架构 13
2.3.5 数据库架构 15
2.3.6 开发结构 16
2.3.7 监控日志 19
2.3.8 数据库优化 20
2.4 需求与程序关系矩阵 21
2.5 人工处理过程 21
3. 接口设计 22
3.1 内部接口 22

4. 运行设计 23
4.1 运行模块组合 23
4.2 运行控制 24
4.3 运行时间 24
5. 系统数据结构设计 25
5.1 逻辑结构设计要点 25
5.2 物理结构设计要求 26
5.3 数据结构与程序关系 26
6. 系统出错处理设计 27
6.1 出错信息 27
6.2 补救措施 27
6.3 系统维护设计 27
7. 容我硬广一下 ..... 错误! 未定义书签。

1. 引言

1.1 编写目的

为了更好的规划、设计全景 CMS 系统，为本系统所有相关人员理解系统的架构特性和系统性能，为项目经理把控项目、制定工作计划提供支持，为开发人员提供技术方向，为测试人员提供黑盒、白盒测试依据，故编写此文档。

本文档为系统架构设计文档，文档内容及格式符合 GB-8567 国家标准的相关要求。为系统开发提供技术上的指导，主要读者为项目负责人、系统设计人员、系统开发人员、系统测试人员以及证券信息各业务部门、运维部门、二次开发人员。

1.2 项目背景

项目 / 系统名称：全景 CMS 内容管理服务平台

项目发起人：深圳市全景网络有限公司

项目实施团队：潍坊华盛信息科技有限公司

系统用户：编辑、开发

全景网是中国资本市场信息服务领域的龙头企业。公司构建起以数据库为基础、以信息披露为核心、集网站、电视、广播、杂志、新媒体终端为一体的多层次资本市场跨媒体信息传播体系，形成了巨潮喜讯网、全景网、新财富杂志、财富天下数字电视频道及系列财经电视节目、深证系列指数、巨潮系列指数等财经知名品牌。

目前，全景网 TRS 后台是多年前开发完成的，技术一直没有升级，很多新的技术跟功能都没实现。同时稳定性比较低、经常卡死、自主开发较差、功能也满足不了新媒体时代环境下编辑的业务需求、为了适应新媒体时代财经网站的发展，

全景网急需一个更全面的采编系统实现对网站资讯、图片、视频等公司资源的整合，实现移动端采编、全站搜索、模板化管理等功能。基于当前现状下，全景 CMS 内容管理服务平台应运而生。

全景 CMS 内容管理服务平台结合了当前大型财经类网站 CMS 的优点、比如：金融界、和讯网等，加入专题、碎片、模版管理、统计管理、新闻全文检索等功能。同时在此功能基础上，优化缓解当前数据库操作的压力，同时便于开发的快速迭代。

2. 总体设计

2.1 需求规定

2.1.1 用户管理系统

此模块系统实现对登陆用户的身份进行认证识别，对现有及新增用户进行综合管理。

2.1.2 权限管理系统

此模块系统提供了用户对各个模块的访问、使用的权限控制。可使用此模块对一个用户赋予访问某个模块的权限,未分配权限的用户无法访问该模块。

2.1.3 新闻管理［PC、IOS、ANDROID］

此模块系统主要用来对新闻进行采集、发布，对新闻进行管理、编辑、对现有新闻普通检索、全文检索等功能。

2.1.4 图片管理系统［PC、IOS、ANDROID］

此模块系统主要做为图片的上传、等比压缩、自定义压缩操作，对图片进行综合管理，属于图片资源库。所有人的采集的图片都会汇集到该处，便于资源共享。

2.1.5 视频管理系统［PC、IOS、ANDROID］

此模块系统主要用于视频的上传，视频的管理，视频的搜索。视频管理系统也是资源库的一类，统一整合所有视频资源，方便所有人使用。

2.1.6 模版管理系统

此模块系统是系统的一个核心.所有的新闻、代码片段等都依赖模板系统。模板系统提供了详情页、列表页、自定义小模块的模板管理操作。模块系统更是为了加快开发快速迭代产品而存在。采用模板语言来提取自己所需要的数据，会让后期页面开发更快。

2.1.7 专题管理系统

此模块系统是为了发布专题使用。第一版中暂时先支持数据提取和专题发布的功能，把做好的静态页使用模板语言标准【开发结束后提供】来提取相应的数据，粘贴到相应位置，然后填写上发布路径，编辑就可以直接发布专题

2.1.8 频道管理

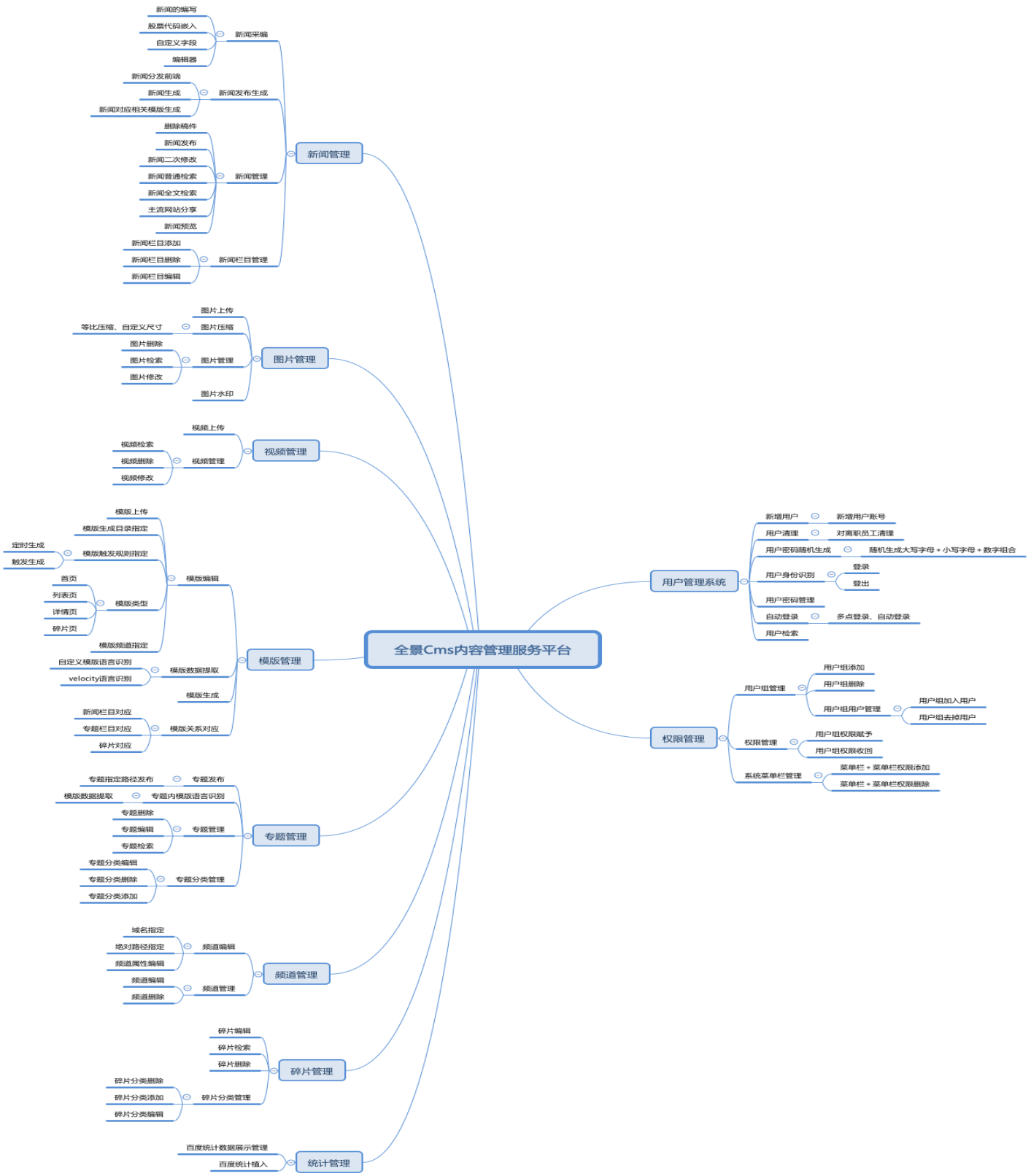
此模块系统是用于维护各个频道的网址和频道的单独目录，这个会自动在编辑选择新闻所属频道后自动应用相应频道网址和目录。

2.1.9 碎片管理

此模块系统是用于手工维护页面上比较细碎的模块，比如首页的推荐文章，网站首页等比较关键的位置采取人工维护的模块，系统内统称为碎片。

2.1.10 统计管理

此模块系统是用于手工维护页面上比较细碎的模块，比如首页的推荐文章，网站首页等比较关键的位置采取人工维护的模块，系统内统称为碎片。

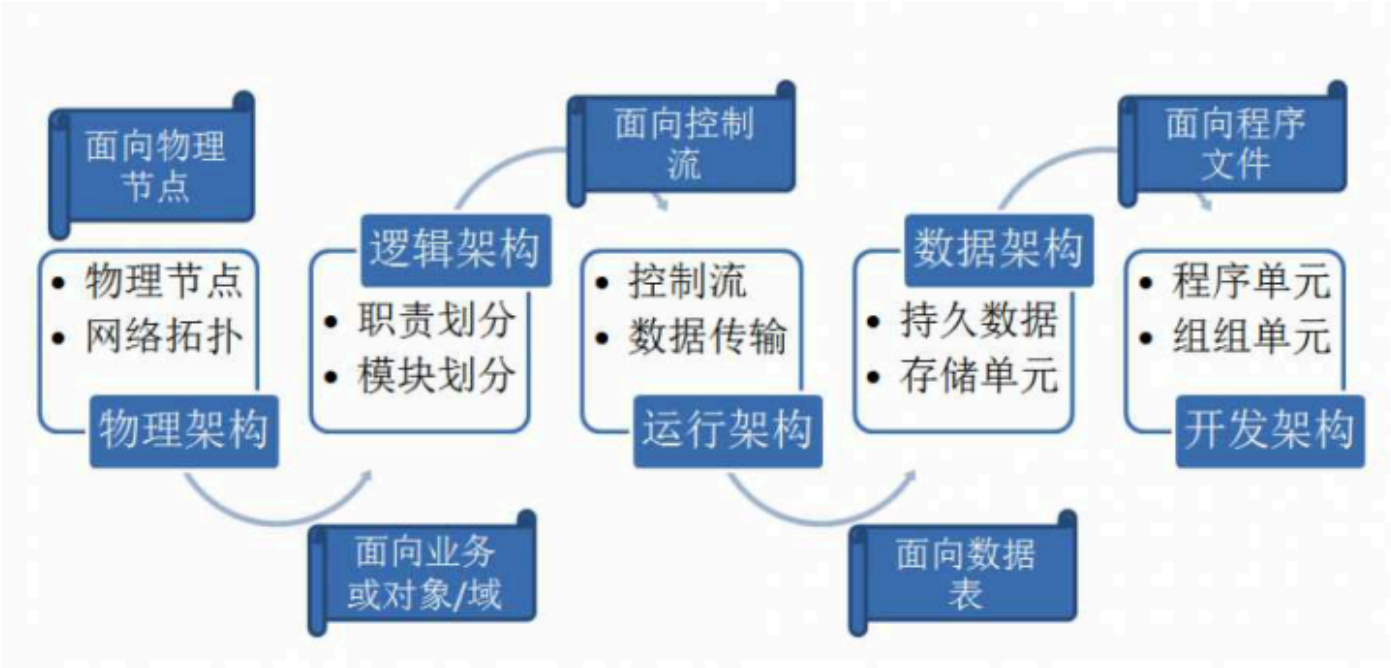


2.2 运行环境

名称	说明	备注
浏览器	IE8、IE8 以上的主流浏览器	
HTTP 负载均衡服务器	前端负载均衡：Tengine(当前) 开启模块：SSI 开启服务：RSYNC	前端和目前的策略保持一致。不改动
后端 HTTP 负载均衡服务器	后端操作系统：CentOS 64 bit 内核：建议 4 核 8 核心以上 内存：建议 8G 及以上 后端负载均衡：Nginx1.9.12 版本及以上 反向代理：HTTP 开启模块：SSI 开启服务：RSYNC	后端使用 nginx 做负载、代理、转发 两台同样配置、负载
应用服务器	应用容器：Tomcat8 运行环境：JDK8 内核：建议 4 核 8 核心以上 内存：建议 8G 及以上 操作系统：CentOS 64 bit 建议服务器：阿里云、 线路：建议内网←专线→阿里云	两台同样配置、负载
中间件服务器	运行环境：JDK8 操作系统：CentOS 64bit 缓存：Redis3.2.3 内核：建议 4 核 8 核心及以上 内存：建议 32G 及以上 缓存集群策略：Sentinel 消息服务：RabbitMQ3.6.5 搜索服务：ElasticSearch 5.0	两台同样配置、主备
分发服务器	运行环境：JDK8 操作系统：CentOS 64 bit 内核：建议 4 核 8 核心及以上 内存：建议 8G 及以上 服务：rsync 建议服务器：阿里云、 线路：建议内网←专线→阿里云	两个同样配置、负载
数据库服务器	保持线上一致	保持线上一致 索引策略优化 主从策略保持线上一致

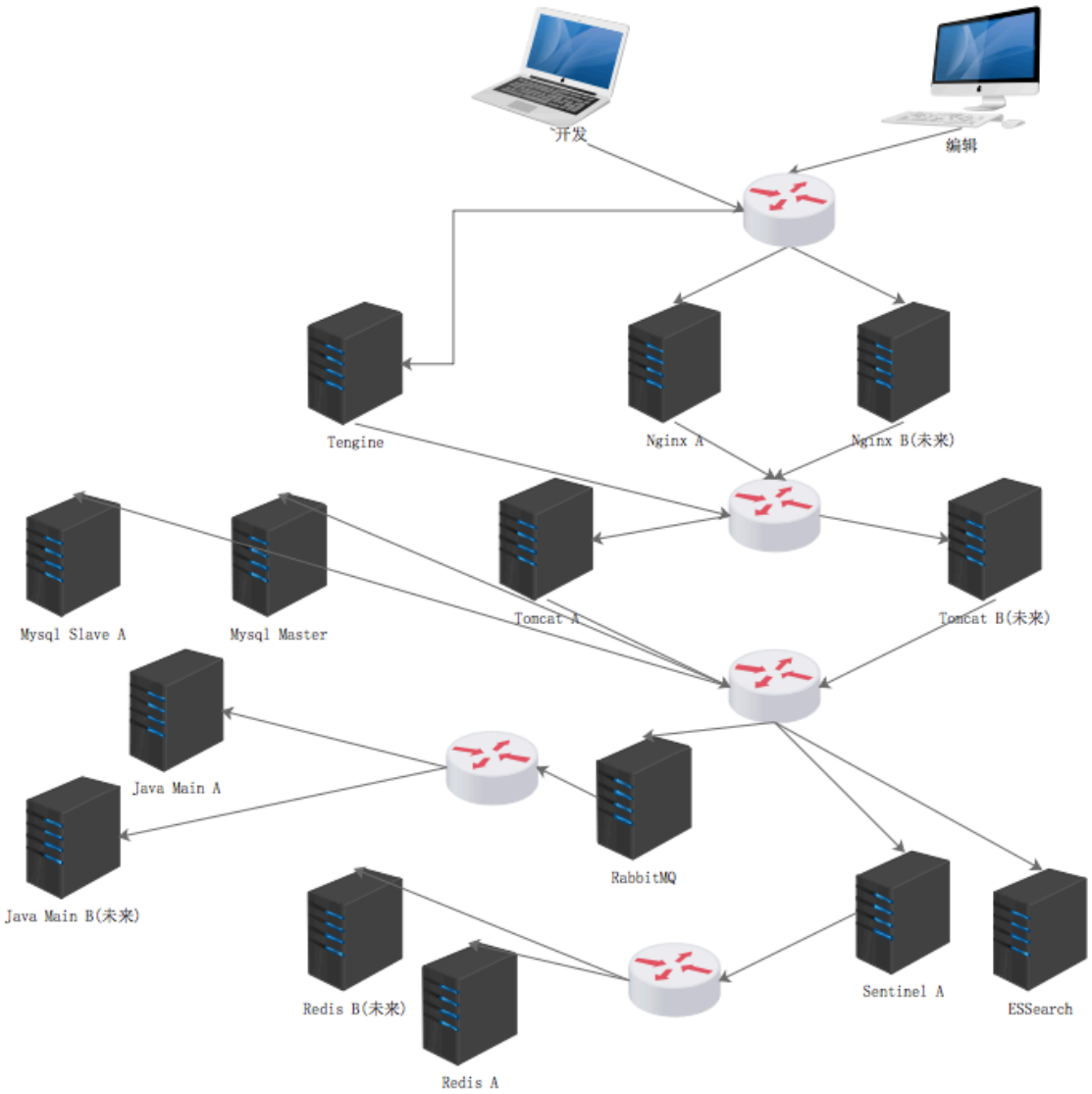
2.3 基础设计概念及处理流程

2.3.1 架构描述



名称	说明	备注
物理架构	描述系统的物理、网络拓扑结构。展现各个节点的互相关系	
逻辑架构	描述系统的逻辑分层结构，说明各个层级的作用与依赖、构建完整的系统结构	
运行架构	描述整个程序运行的结构	
数据架构	描述数据持久化、存储方案	
开发架构	描述系统程序组织结构、文件结构、命名规则	

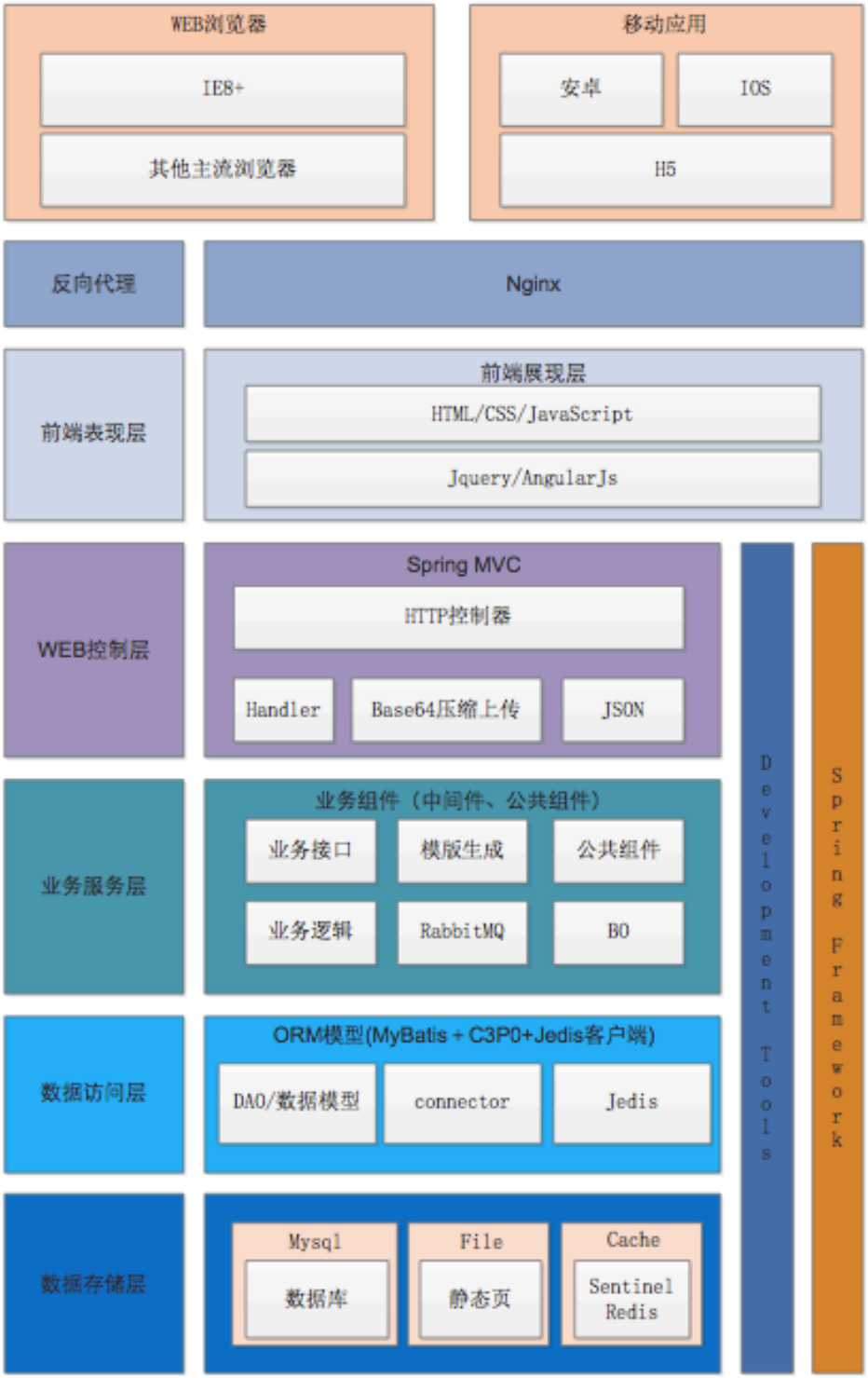
2.3.2 物理架构－拓扑图



名称	说明	备注
Nginx	直接面向终端用户，分流 HTTP 请求。分发请求到不同的业务服务器节点、Nginx 负责对 WEB 请求的分发。对静态资源的管理。前端页面使用 angularJS MVC 框架。	

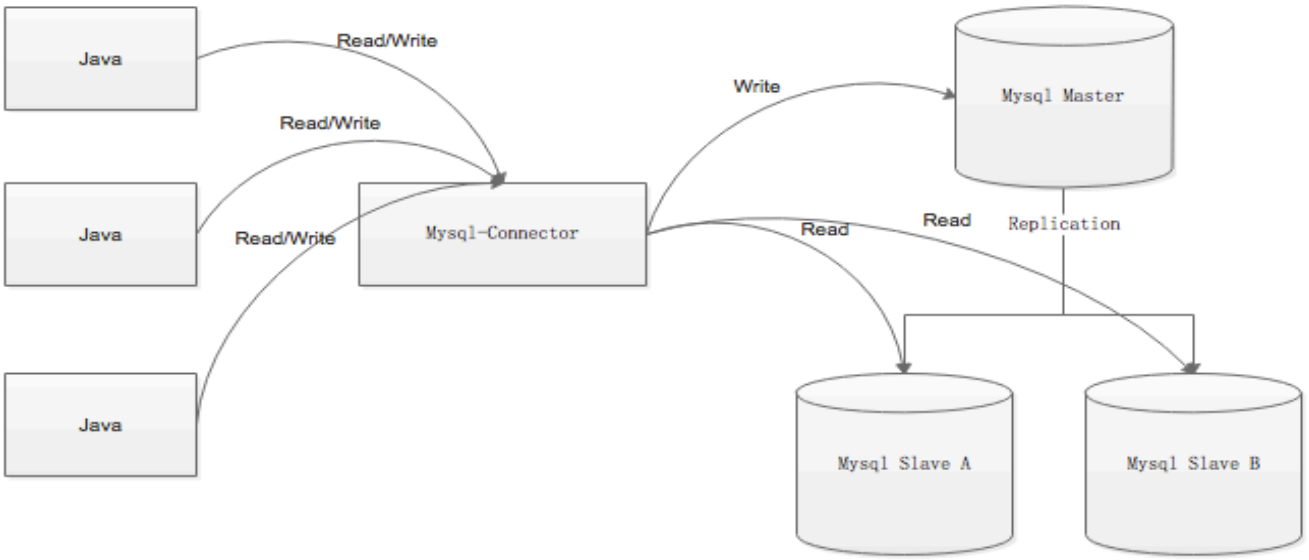
Tomcat	动态页面容器，采用的是 tomcat8 版本，多节点部署。部署的是 CSM 管理服务后台
Tengine	内网使用的前端解释机，所有静态页、分发的内容会发布到外网以外，也会发布一份到 Tengine 上，用作预览。
缓存	Sentinel＋Redis 的模式。Sentinel 用作集群策略。Redis 集群机挂到 Sentinel，当出现 Redis 集群内的一台机器挂起的时候。Sentinel 会自动切换
RabbitMQ	用作应用之间的异步消息服务机，采用 1 对 1 消息。多点应用只要有一台消费了消息后，另一台就不会再收到同一条消息
EsSearch	全文检索服务。当前比较主流、好用的检索引擎。用作新闻的检索，可多点
Java Main	分发系统，主要用作处理模版生成，发布。可多点。采用 RabbitMq 与 CMS 后台交互。
数据库	Mysql 数据库使用当前的 Mysql 版本。机制也和当前保持一致(已知 主从机制)。添加新库的表索引优化机制。

2.3.3 逻辑架构



名称	说明	备注
反向代理	Nginx 做负载均衡，以实现物理节点的横向扩展	
前端表现层	以前端展示为主，包含用户数据接收、系统数据展示、页面布局、排版、Build 美化，所有用到的框架：Jquery、AngularJS MVC 等	
WEB 控制层	提交数据进行封装，控制转发，调用后台业务服务器，与服务器进行通讯 接收服务器业务处理结果进行数据封装，返回结果	
服务实现层	面向服务的接口定义、服务端与 WEB 端交互点，服务接口的实现，业务逻辑、第三方中间件调用等。	第三方中间件调用等作为服务层绘制。如果细分起来是可以改变框架格局。
数据访问层	数据持久化层、	Redis 和 Mysql 是相互独立的数据层。Redis 无事务操作。Mysql 是事务。都属于数据访问层。
数据存储层	存储业务产生的业务数据和系统数据、静态文件、缓存等。	日志不列入其中。

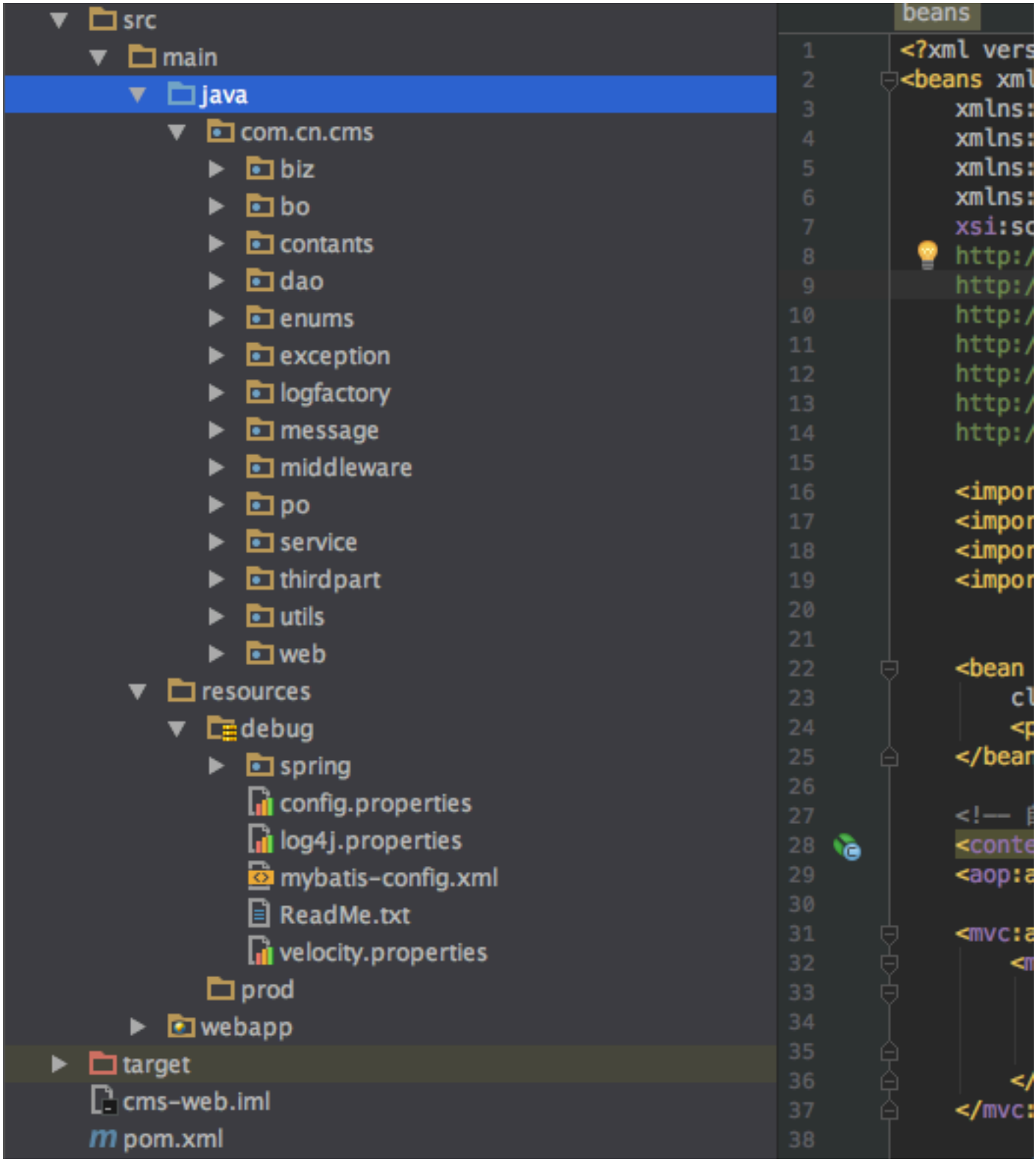
2.3.5 数据库架构



数据库采用当前现有的数据库结构。使用 Mysql-Connector 作为读写分离的客户端。

2.3.6 开发结构

项目使用的 Moven 构建。简单方便。一个 CMS-Model 基础项目。和三个客户端项目分别包含 CMS-WEB、CMS-API、CMS-PUBLISH，如下方图所示：



以单个项目为例子，举例说明所有包的用途，及包的定义及说明。建议二次开发沿用此规则。尤其是 java 包下的所有子包的定义及使用规则

Java	包含了所有类的定义。构建主要包
Resources	内包含了 debug、prod 两套环境的资源配置。在下面的包说明里 就不提资源包了，一看就比较清晰、构建主要包
Webapp	Web 的各种文件、配置等。构建主要包。
Target	输出编辑包。这个包内存的是编辑后的 class、resource 配置、webapp 等。构建主要包
Pom.xml	Moven 构建配置。里面包含了所有 jar 的引入及 resource 指定。和构建信息。主要文件
com.cn.cms.biz	业务封装包，这里面可以定义一些 biz 类 用来处理业务逻辑。聚合一些数据等。不做任何涉及事务的操作。
com.cn.cms.bo	Bo 类。可以定义一些业务 bean。剔除掉 PO 类里的敏感字段后的 bean 包。可用作接口数据外放。
com.cn.cms.contants	常量包，主要存放一些常量类。比如 Redis Key、一些 static final 类的变量等。
com.cn.cms.dao	主要用来存放 Dao 类。和 mybatis 映射 mapper 文件。主要的数据操作都在这个包内。
com.cn.cms.enums	枚举包，存放所有公共的枚举类定义。枚举使用主要是存储的文字与数字的对应。数据库存储数字。用枚举转换数字成文字展示。加快数据库的查询速度
com.cn.cms.exception	自定义异常包，主要用来抛出自定义异常。在 Exceptionhandler 内使用。

com.cn.cms.logfactory	日志工厂包，包内主要定义了自定义的日志类。方便未来扩展。
com.cn.cms.message	消息包，包含 common 消息体、自定义消息类型等的定义。也存储消息的提供、消费类等。
com.cn.cms.middleware	中间件的调用服务包、主要用来存放第三方中间件调用的 Client 类。比如 JedisClient 等。
com.cn.cms.po	Po 包。ORM 映射持久化对象包。跟 java 类跟 Table 表、表字段一一对应。
com.cn.cms.service	主要存放事务类及事务类的接口定义、项目内的所有关于事务操作的操作都放到这个包内。统一管理。避免出现事务内掺杂无关的第三方调用、业务逻辑等。防止出现连接数过多。
com.cn.cms.thirdpart	第三方接口调用包、此包用于封装调用第三方接口的服务。方便共用。和使用公共策略。比如 connect、read timeout 等等
com.cn.cms.utils	共用工具包。主要存放一些共用的工具类等。比如加密、字符串操作、时间操作类似的其他相关工具类
com.cn.cms.web	存放 Controller、handler、共用返回类和与此相关的 ann 注解定义等等。这个包是所有接口的入口包。

2.3.7 监控日志

监控系统如果细化做起来是一个很庞大的一个系统。包括 zabbix 运维监控等。不过如果要搭建这么庞大的系统不是一朝一夕的事情。咱们初期这里主要实现:可以查看当前系统的运行情况。是否有异常、是否正常等日志的纪录工作、和业务上的任何操作的业务日志纪录。

下面咱们根据系统运行日志和业务日志两个方面介绍以下当前系统的日志模块：

1. 系统运行日志：指系统在运行过程中纪录的程序运行信息。以文件的形式存储到当前系统运行的服务器内，未来日志统一化管理并监控的话可以采用 shell、python 等做日志同步工作。咱们系统日志使用 log4j 纪录。每日生成一个新的日志文件。方便未来使用和避免日志文件过大造成查看和性能的不便。日志分四个级别 ERROR、WARNING、INFO、DEBUG。可以根据不同情况设置不同的级别。目前 ERROR、WARNING、INFO 都写入到一个文件.方便查看上下文信息。

2. 业务日志：指用户在使用系统过程中产生的行为日志。系统回把这些日志纪录到数据库中。所有日志纪录均是用户操作成功后的操作纪录。如果用户在过程中出现了中断请求的情况.那么不纪录无效行为日志。我们可以通过数据库内存储的行为日志。分析用户每天的操作情况。和操作的所有内容。

3. 日志查看方式：

   a) 系统运行日志通过 SZ 的方式 down 到本机查看。或者直接在服务器内通过命令查看

   b) 业务日志通过连接数据库，查看数据库表数据的方式进行查看

2.3.8 数据库优化

根据沟通结果得出，当前数据库在抽取数据生成模版的情况下。数据库的压力过大。操作系统缓慢。所以本系统会对当前的情况进行适当优化。

分析：按照当前所知,模版采用数据库存储的方式进行存储。每个模版的大小至少需要 text 类型才可以存储。每次生成模版都要去数据库查询模版。造成大字段查询过多。过频繁。同时在查询中会伴随数据抽取的情况发生。

优化：新 CMS 建成以后。模版采用静态页＋模版语言的方式进行解析。主要使用硬盘读写。降低数据库压力。所有常用的表字段查询均添加必要索引。包含唯一、组合等。加快查询速度。在当前 mysql 下建立新的库。降低对其他业务的影响，大字段的一些内容字段均拆出单独的表做存储。降低大字段查询频率。Mysql 策略会把常用查询缓存到内存里。Size 有限。如果大字段过多就会影响其他数据的查询速度。同时程序内采用 Redis 缓存、事务独立、读写分离、等策略极大的降低数据库的查询压力。这些策略加在一块足可以满足当前业务的支持。

2.4 需求与程序关系矩阵

程序 需求								

2.5 人工处理过程

暂无

2.6 尚未解决的问题

暂无

3. 接口设计

3.1 内部接口

本系统后台管理系统是前后端分离的架构。所有的功能模块均采用 REST 接口形式提供服务。

3.2 外部接口

3.2.1 股票代码接口

股票代码接口，用于嵌入新闻详情内使用。

3.2.2 行业分类接口

行业分类接口，用于嵌入新闻详情内使用。

3.2.3 评论

评论使用畅言模块组件。组装到公共模块内。

3.2.4 用户

直接嵌入当前全景网用户组件 JS。

3.2.5 视频上传接口

使用当前第三方视频上传接口。

4. 运行设计

4.1 运行模块组合

名称	说明	备注
用户发起请求	前端用户通过浏览器发起用户请求，后台接收到请求以后，处理当前请求,将相应数据通过 JSON 形式或者页面返回到浏览器。 浏览器接收到响应后，解析 JSON、页面展现	
Nginx	Nginx 接收用户请求，接收到请求后转发到后台应用服务器，或者在 Nginx 端直接处理当前请求后响应返回	
应用服务器接收	应用服务器接收到 Nginx 转发过来的请求后，开始处理当前请求。 组合业务数据后，开始返回响应的 JSON 数据给 Nginx 端，Nginx 端接收到应用服务器的响应数据，响应给用户	
业务逻辑处理	根据请求开始业务逻辑的处理。业务逻辑的处理包含：业务数据计算、组装（外部接口调用、第三方中间件、数据）等	
数据持久化	对于业务逻辑中的内容进行持久化存储，包含数据库、文件、页面、缓存服务等。	

4.2 运行控制

系统运行期间，不需要人为控制各逻辑层的流程，每一层之间的流转由程序根据上一节点的处理结果参数进行控制，系统的数据源来自业务提交的数据和初始化数据及页面，其他的数据皆为中间数据。

4.3 运行时间

1. 服务器确保 7*24 小时稳定可靠的运行，提供系统崩溃时的快速恢复机制，确保系统出现故障时候能快速恢复系统正常运行。加入负载机制。多机部署。出现单台机器故障的情况下。保证系统可正常切换，系统正常可用。
2. 操作系统的稳定性。非本系统相关的程序不可部署到本系统所部署的机器上。保证不受外在因素影响
3. 连接的稳定性。内网使用，建议阿里云和内网交互采用专线的方式(建议) 进行通信。可以保证连接的可靠性。
4. 数据层采用连接池技术，并使用连接池提高系统性能，对于每个连接的申请和归还都有明确的规则处理，严格保证每个系统所开启的最大连接数在可控范围。

5. 系统数据结构设计

5.1 逻辑结构设计要点

本系统由以下几点考虑设计：

1. 动态配置：对于内部使用的一些内容，均采用可动态配置或静态配置的方式进行，极大的保证系统的可操作性，便捷可用性也比较高。

2. 性能：

   a) 模版采用静态页模版读取、模版识别的方式。抛掉以前的数据库存储模版的方式。极大的降低了数据库操作的频率、降低数据库大字段查询的频率。同时提高了数据库其他查询资源的精简。间接加快数据库内存中可缓存数据表的数量。加快数据库操作。

   b) 程序分层明确，事务中不掺杂任何业务逻辑、第三方调用、中间件调用等。使线程事务的操作大大加快。这样避免连接数太高的几率。连接数过高是很多公司都会碰到的问题。咱们在前期就规避这个问题。会让系统稳定性大大加强。同时对数据库造成的压力会更小

   c) 使用缓存机制。对一些常用的数据。一些变化不大的业务进行缓存存储。降低数据库的操作频率。

   d) 对于一些时效要求不高的内容采用先缓存、后入库的机制。降低数据库的操作频率 对于数据库的操作控制在可控的范围内。

   e) 数据库操作采用读写分离的形式。读操作走读库、写数据走写库。分散数据库压力。读是最多的。写是最有压力的(更新索引等)。同样如果走一个库压力也会很大。读写分离降低单个数据库的压力。压力分流。

   f) 使用 EsSearch 做全文检索。代替 like 操作。Like 操作对数据库的压力很大。因为 like 操作是没有索引的。同时全文检索会使用大字段的 like 操作。现在用搜索引擎做全文检索。降低数据库的操作。替代 like 指令也可以降低数据库计算压力。

g) 使用 MQ 异步消息。一些不需要立刻产生结果的计算、操作等均使用 MQ 异步消息给任务处理机（CMS-PUBLISH）取处理。提高后台服务的稳定性。降低后台服务压力。

h)

5.2 物理结构设计要求

根据用户非功能需求约束，本系统物理结构设计满足以下几点：

1. 应用之间互相独立部署。
2. 中间件均可扩展集群||主从等机制。
3. 静态资源共享。
4. 系统可集群部署。
5. 系统可扩展性强。

5.3 数据结构与程序关系

略

6. 系统出错处理设计

6.1 出错信息

故障现象	用户展示	处理机制
页面不存在	404 友好页面	跳转到 404 错误页。进行友好提示
服务 500+错误	友好提示，不影响之前页面展现	采用“天塌了地陷了小花猫不见了“等有意思的展示。降低用户的抵触信息。纪录服务器错误日志。方便查看。
数据库宕机	前端无影响	后端自动切换机器（当前）
网络中断	友好提示	

6.2 补救措施

说明故障出现后可能采取的变通措施，包含：

1. 备机策略，建议多机多机房部署。使用灾备策略。如果有一台机器、一个机房出现问题。可以随时切换到另一台机器或者机房。

2. 一键部署策略，建议未来可以使用一键部署。我们公司有完整的一键部署策略。我们在做当前系统的时候也预留了一键部署的扩展。方便未来如果出现不可抗拒因素。可以随时一键部署所有的应用。降低出现不可抗拒情况下的长时间不可访问的情况发生。也加快系统代码升级迭代的效率。

6.3 系统维护设计

系统维护是一个软件系统生命周期中重要的一环，本系统也不例外：

1. 运行维护，通常包含制定运维方案及设计，主要包括系统监控、备份机制、审计、优化等。

2. 升级维护，通常涉及到功能变化，一般都需要修改源程序，一般这类的升级都会影响到业务的正常操作(区别是：时间长短的问题)，因此每次升级程序，应该得到组织的审核和风险评估、升级之前全量备份旧系统，升级完成后投入使用之前，应该做好健康性检查，并更新相关文档.