软件系统分析与设计

科技专家资源管理平台

系统架构设计说明书

|  |  |  |
| --- | --- | --- |
| **项目组成员信息** | | |
| **小组名称** | 加冰不放糖 | |
| **小组联系人** | 庄秉鑫 | |
| **学号** | **姓名** | **本文档中主要承担的工作内容** |
| 16211012 | 庄秉鑫 | 参与架构说明书的讨论，以及数据架构、运行架构、部署架构及非功能性设计中的高可用性模块的编写 |
| 16211019 | 李恒鑫 | 参与架构说明书的讨论，以及开发架构模块的编写 |
| 16211024 | 刘梓雄 | 参与架构说明书的讨论，以及非功能性设计模块的编写 |
| 16211030 | 李航 | 参与架构说明书的讨论，以及接口设计模块的编写，并对数据架构、逻辑架构模块提出意见 |
| 16211073 | 洛嘉航 | 参与架构说明书的讨论，以及逻辑架构模块的编写 |
| 16211074 | 唐彬瀚 | 参与架构说明书的讨论，以及文档概述、其他约束模块的编写 |
| 16211052 | 苏治杰 | 对架构说明书的编写提出意见 |

科技专家资源管理平台 1

一、文档概述 3

1.项目背景 3

2.项目目标 3

3.文档版本信息 3

4.目标读者 3

5.相关文档 3

6.术语说明 3

7.参考资料 4

二、逻辑架构 4

表示层 4

业务层 5

数据层 5

三、开发架构 5

Web框架：Spring 5

数据库：MySQL 6

前端：Bootstrap 6

数据库接口：Jdbc 7

四、接口设计 7

五、数据架构 12

数据表 12

六、运行架构 13

并发技术选型 13

线程同步 13

线程间的通信 14

七、部署架构 14

八、非功能性设计 16

安全性 16

可靠性 16

兼容性 16

高可用性 16

九、其他约束 17

时间约束 17

法律约束 17

质量约束 17

经费约束 17

工作方式约束 17

一、文档概述

1.项目背景

随着国家对科技创新的日益重视，科技专家的科研成果和资源正变成日益宝贵的企业和社会资源。如何有效地推动科研成果合作和科技资源共享正成为科学技术领域的热门话题。

作为科研活动的核心主体，管理和维护科技专家资源是科技成果共享重要的一环。在此背景下，建设一套面向全国乃至全世界的科技专家资源共享平台就变得非常有意义。科技专家资源共享平台由专门的管理机构运行和维护，在统一管理的基础上建立科技专家门户系统。其基本的运行和管理机制包括以下几个方面：

（1）通过标准数据库、网络抓取等多种技术获得专家资源基础数据，建立基础平台结构，并以专家为中心，搭建各个专家的门户系统。收集的基础数据包括但不限于：专家个人信息和专业技术领域等基本信息；以及所从事的科研项目、发表的科研论文和授权的科技专利等学术成果的动态管理。

（2）科技专家可以向平台申请管理和维护自己的门户系统，平台管理员统一审核，审核通过后科技专家可以完善和维护门户系统的各类信息。并可对自己的成果资源提供有偿增值服务，如科技成果转让、论文全文和专利转让等业务功能。

（3）普通用户可以检索和查看科技专家各类资源，获得各类免费资源，针对收费资源，普通用户可以通过在平台购买积分，并消耗一定的积分后获得资源。

（4）实现对各类专家资源进行有效的统计和分析功能，实现诸如专家关系网络、科研机构成果分类排名、科研领域热点分析等功能。

2.项目目标

建立一个面向全世界的科技专家资源共享平台，通过建立这样的一个平台，让专家们能更方便快捷的管理自己的科研成果，同时普通用户也能更全面更快捷地了解到科技最前沿的成果，专家和用户互相促进，构成一个良好的科研成果分享生态圈。

3.文档版本信息

此文档为第一版，版本号为1.0

4.目标读者

开发者和客户

5.相关文档

[1] 科技专家资源共享平台-背景陈述

[2]科技专家资源管理平台需求规格说明书

[3]科技专家资源管理平台分析模型

6.术语说明

B/S架构: 即浏览器和服务器架构模式

科技专家: 指在某一科学技术领域具有一定学术成就的专业技术人员，是国家科技创新的主要力量。科技专家一般来自于高等院校、科研院所或高新企业等不同的学术科研机构，其科研成果和资源一般隶属于其所在的科研机构。

专家门户系统：类似于个人主页的能让专家发布各种信息的页面。

注入：通过把SQL命令插入到Web表单提交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的SQL命令。

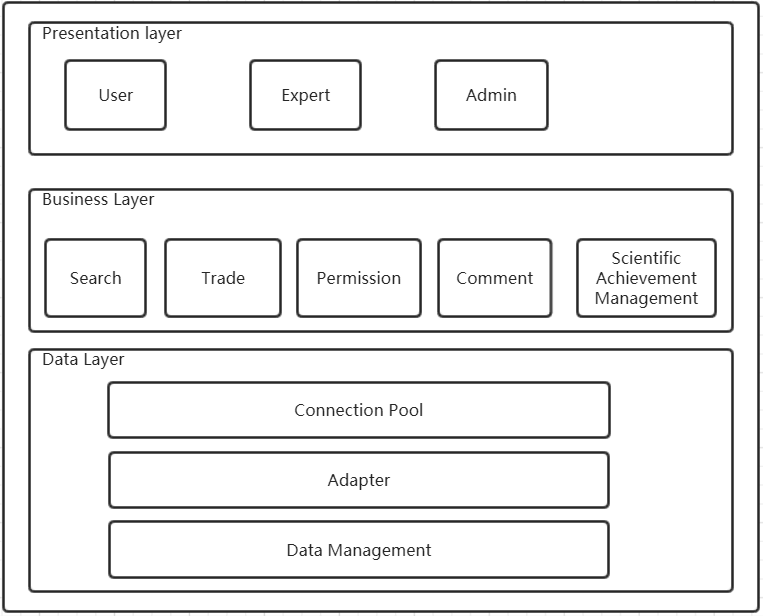
事务：是数据库管理系统执行过程中的一个逻辑单位，由一个有限的数据库操作序列构成。

死锁：两个或两个以上的进程在执行过程中，由于竞争资源或者由于彼此通信而造成的一种阻塞的现象，若无外力作用，它们都将无法推进下去。

Github：一个面向开源及私有软件项目的托管平台。

7.参考资料

[1].李智慧. (2013). 大型网站技术架构 : 核心原理与案例分析. 电子工业出版社.

二、逻辑架构

表示层

普通用户（User）：是平台最普通的用户群体，拥有检索以及购买资料等权限。

专家（Expert）：由用户升级而来，相比用户，专家可以建立自己的管理平台，以及出售自己的科技成果。

管理员（Admin）：系统管理员，拥有监控与管理系统中的资料与评论等的权限。

业务层

搜索（Search）：接受用户输入的搜索关键词，在数据库中进行搜索并返回给用户。

交易（Trade）：接受由用户提出的交易请求，获取交易双方的交易物品与确认信息，在数据库中对交易双方数据进行修改，并给用户返回交易结果。

权限升级（Permission）：接受由用户提出的权限升级请求，并获取用户提交的证明信息，核实无误之后在后端修改用户权限级别，并返回给用户结果。

评论系统（Comment）：接受用户提交的评论，审核后存储在后端数据库中，并在表示层刷新显示。接受管理员的修改、删除请求，在后端数据库中对相应评论数据做出修改，并在表示层刷新显示，同时要给评论者发回提示信息。

专家资源管理平台（Scientific Achievement Management）：接受由专家提出的对资源的增删改查请求，在数据库中完成相应操作，并向专家用户返回相应信息。

数据层

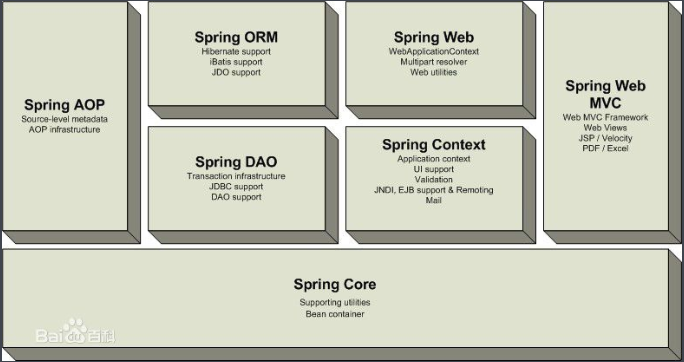
连接池（Connection Pool）：连接池是创建和管理一个连接的缓冲池的技术，用于业务层和数据层建立连接。

适配器（Adapter）：业务层调用数据库sql语句的适配，定义了接口标准。

数据管理平台（Data Management）：由业务层调用，对数据库进行操作。

三、开发架构

Web框架：Spring

Spring是一个轻量级的控制反转（loC）和面向切面（AOP）的容器框架。由如下七个部分构成。

使用 Spring 框架主要的好处是：

* Spring 在一个单元模式中是有组织的。即使包和类的数量非常大，你只要担心你需要的，而其它的就可以忽略了。
* Spring 不会让你白费力气做重复工作，它真正的利用了一些现有的技术，像ORM 框架、日志框架、JEE、Quartz 和 JDK 计时器，其他视图技术。
* 测试一个用 Spring 编写的应用程序很容易，因为环境相关的代码被移动到这个框架中。此外，通过使用 JavaBean-style POJOs，它在使用依赖注入注入测试数据时变得更容易。
* Spring 的 web 框架是一个设计良好的 web MVC 框架，它为比如 Structs 或者其他工程上的或者不怎么受欢迎的 web 框架提供了一个很好的供替代的选择。MVC模式导致应用程序的不同方面(输入逻辑，业务逻辑和UI逻辑)分离，同时提供这些元素之间的松散耦合。
  + 模型(Model)封装了应用程序数据，通常它们将由POJO类组成。
  + 视图(View)负责渲染模型数据，一般来说它生成客户端浏览器可以解释HTML输出。
  + 控制器(Controller)负责处理用户请求并构建适当的模型，并将其传递给视图进行渲染。
* Spring 对JavaEE开发中非常难用的一些API（JDBC、JavaMail、远程调用等），都提供了封装，使这些API应用难度大大降低。
* 轻量级的 IOC 容器往往是轻量级的，例如，特别是当与 EJB 容器相比的时候。这有利于在内存和 CPU 资源有限的计算机上开发和部署应用程序。
* Spring提供了一致的事务管理接口，可向下扩展到（使用一个单一的数据库，例如）本地事务并扩展到全局事务（例如，使用 JTA）。

数据库：MySQL

MySQL是一个开源的关系型数据库，不宜存储图结构等数据，同时其功能没有付费数据库（如SQLServer）强大，稳定性不足，后期技术支持几乎没有。但胜在其社区较为丰富，安装使用较为简洁。

前端：Bootstrap

目前主流的前端框架有BootStrap，React和Vue，其各自的特点如下：

BootStrap:

特点是栅格系统, 使用简单, 上手容易.  专为响应式页面而生. 一套代码就可以自适应平板电脑和PC.

缺点:

缺少一套有力的成体系的组件(当时调查的时候还没有, 现在据说有了), 我在实际使用时, 发生了作用域冲突的问题,  如果没有整理好一整套组件, 开发很累.

React:

特点主要有2个:

1. 数据的单向响应.当数据发生变化的时候, React将会自动的管理UI界面更新

2.  VisualDom(虚拟Dom). 由React来管理Dom树的变化, 开发者只需使用React的虚拟Dom进行开发, 页面的呈现会更快. 这个引申出了一个更大的优点:   虚拟Dom到真实Dom的映射, 是由React来控制的. 那么如果虚拟Dom映射为IOS或者Android呢? 于是React Native诞生了, 约80%的Reactive Native代码是可以同时支持IOS和Android的.

 缺点:

最近React的GPL协议有变动, 容易引起商业问题...

Vue:

特点在于, 站在了React的肩膀上, 许多方面更出色了:

1. 数据的双向绑定. 数据改, 页面改. 页面改, 数据也改.

2. 页面的呈现据说比React还快.

3. 组件化(实际上React也有组件化). 这和BootStrap这种组件化的概念还有点不同, Vue的组件化, 已经把作用域冲突的问题给搞定了.另外, js测试也变得更专业化了(有点类似junit).

4. 单html开发. 它的开发模式, 是一个html, 然后不停替换组件对应的js来切换显示效果. html中的共通js和css只需要download一次, 理论上比Iframe页面的呈现更快.

5. 结合node.js提供的webpack等, 可以进行成体系的打包发布.

 缺点:

上手难度比较高, 并且大规模开发的化, 需要有体系化开发的积累. Vue的组件化开发推荐使用

数据库接口：Jdbc

数据库的访问接口，开发简单，提供统一的操作接口。弊端在于安全性差容易产生SQL注入，代码美观度不够，不易维护。

四、接口设计

1. 登录

|  |  |  |
| --- | --- | --- |
| loginIn : String | | |
| username | Post | 用户名 |
| password | Post | 密码 |
| 返回值 | String | session id |

2. 密码找回

|  |  |  |
| --- | --- | --- |
| findPasswd : String | | |
| username | Post | 用户名 |
| email | Post | 注册邮箱 |
| 返回值 | String | session id |

3. 密码找回输入新密码

|  |  |  |
| --- | --- | --- |
| changePasswd : Boolean | | |
| sessionid | Post | session id |
| newpasswd | Post | 新密码 |
| newpasswd2 | Post | 第二次输入密码 |
| 返回值 | Boolean | 是否成功 |

4. 有密码更改密码

|  |  |  |
| --- | --- | --- |
| changePasswd : Boolean | | |
| username | Post | 用户名 |
| oldpasswd | Post | 旧密码 |
| newpasswd | Post | 新密码 |
| newpasswd2 | Post | 第二次输入密码 |
| 返回值 | Boolean | 是否成功 |

5. 普通用户注册

|  |  |  |
| --- | --- | --- |
| normalSign : Boolean | | |
| username | Post | 用户名 |
| passwd | Post | 密码 |
| passwd2 | Post | 第二次输入密码 |
| email | Post | 用户邮箱 |
| 返回值 | Boolean | 是否成功 |

6. 科技专家注册

|  |  |  |
| --- | --- | --- |
| expertSign : Boolean | | |
| username | Post | 用户名 |
| passwd | Post | 密码 |
| passwd2 | Post | 第二次输入密码 |
| email | Post | 用户邮箱 |
| otherinfo | Post(JSON) | 其他需要的信息 |
| 返回值 | Boolean | 是否成功 |

7. 更新普通用户信息

|  |  |  |
| --- | --- | --- |
| newUserInfo :Boolean | | |
| sessionid | Post | session id |
| newinfo | Post(JSON) | 新信息 |
| 返回值 | Boolean | 是否成功 |

8. 普通用户升级为专家

|  |  |  |
| --- | --- | --- |
| becomeExpert : Boolean | | |
| sessionid | Post | session id |
| expertinfo | Post(JSON) | 科技专家需要的信息 |
| 返回值 | Boolean | 是否成功 |

9. 充值

|  |  |  |
| --- | --- | --- |
| recharge : Boolean | | |
| sessionid | Post | session id |
| source | Post | 充值源 |
| money | Post | 钱数 |
| needed | Post(JSON) | 充值源需要的认证数据 |
| 返回值 | Boolean | 是否成功 |

10. 付费并阅览、免费或已付费资源阅览

|  |  |  |
| --- | --- | --- |
| payRead : Url | | |
| sessionid | Post | session id |
| resourceid | Post | 阅览资源id |
| 返回值 | Url | 全文阅览Url |

11. 上传资源

|  |  |  |
| --- | --- | --- |
| uploadResource ： Boolean | | |
| sessionid | Post | session id |
| uri | Post | 本地文件 |
| others | Post(JSON) | 资源相关属性 |
| 返回值 | Boolean | 是否成功 |

12. 删除资源

|  |  |  |
| --- | --- | --- |
| deleteResource : Boolean | | |
| sessionid | Post | session id |
| resourceid | Post | 资源id |
| 返回值 | Boolean | 是否成功 |

13. 转让资源

|  |  |  |
| --- | --- | --- |
| transResource | | |
| sessionid | Post | session id |
| resourceid | Post | 资源id |
| to | Post | 转让目标用户id |
| 返回值 | Boolean | 是否成功 |

14. 检索资源

|  |  |  |
| --- | --- | --- |
| findResource : List<Int> | | |
| fields | Post(JSON) | 所有查询字段 |
| 返回值 | List<Int> | 资源id列表 |

15. 获取资源预览（付费资源未付费）（免费资源可预览量100%）

|  |  |  |
| --- | --- | --- |
| preRead : Url | | |
| resourceid | Post | 资源id |
| 返回值 | Url | 预览页面 |

16. 发起关注用户请求

|  |  |  |
| --- | --- | --- |
| followUser : Boolean | | |
| sessionid | Post | session id |
| userid | Post | 关注的用户id |
| 返回值 | Boolean | 是否成功 |

17. 取消关注

|  |  |  |
| --- | --- | --- |
| removeFollow : Boolean | | |
| sessionid | Post | session id |
| userid | Post | 删除关注的用户id |
| 返回值 | Boolean | 是否成功 |

18. 设置管理员

|  |  |  |
| --- | --- | --- |
| setManager : Boolean | | |
| sessionid | Post | session id |
| userid | Post | 新管理员id |
| 返回值 | Boolean | 是否成功 |

19. 解除管理员

|  |  |  |
| --- | --- | --- |
| deleteManager : Boolean | | |
| sessionid | Post | session id |
| userid | Post | 删除管理员id |
| 返回值 | Boolean | 是否成功 |

20. 评论

|  |  |  |
| --- | --- | --- |
| makeComment : Boolean | | |
| sessionid | Post | session id |
| commentto | Post | 发起评论的资源id |
| comments | Post | 评论内容 |
| 返回值 | Boolean | 是否成功 |

21. 删除评论

|  |  |  |
| --- | --- | --- |
| deleteComment : Boolean | | |
| sessionid | Post | session id |
| commentid | Post | 删除的评论id |
| 返回值 | Boolean | 是否成功 |

22. 举报资源

|  |  |  |
| --- | --- | --- |
| reportResource : Boolean | | |
| sessionid | Post | session id |
| resourceid | Post | 举报资源id |
| why | Post | 举报理由描述 |
| 返回值 | Boolean | 是否成功 |

23. 用户反馈

|  |  |  |
| --- | --- | --- |
| reportBugs : Boolean | | |
| sessionid | Post | session id |
| bugs | Post | 反馈内容 |
| 返回值 | Boolean | 是否成功 |

24. 发布公告

|  |  |  |
| --- | --- | --- |
| pushNotice : Boolean | | |
| sessionid | Post | session id |
| notice | Post | 公告内容 |
| 返回值 | Boolean | 是否成功 |

25. 发送私信

|  |  |  |
| --- | --- | --- |
| pushMail : Boolean | | |
| sessionid | Post | session id |
| mail | Post | 私信内容 |
| userid | Post | 发送私信的对象 |
| 返回值 | Boolean | 是否成功 |

26. 专家申请自己的门户系统

|  |  |  |
| --- | --- | --- |
| createHomepage : Url | | |
| sessionid | Post | session id |
| 返回值 | Url | 门户主页 |

五、数据架构

数据表

user

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **id** | name | identification | cellphoneNumber | password | points | email |

resource

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **id** | downloadPrice | transferPrice | title | path |

report

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | userId | content | resouceId |

project

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **expertId** | approvingDepartment | funds | startDate | endDate | id |

patent

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | type | applicationDate | authorizedDate |

paper

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | brief | from | author | ownerId | issuedTime |

order

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | customerId | sellerId | resourceId | time | state |

notice

|  |  |  |
| --- | --- | --- |
| **id** | content | releasedTime |

message

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **id** | content | sentTime | senderId | receiverId |

keyword

|  |  |  |
| --- | --- | --- |
| **resourceId** | words | id |

follow

|  |  |
| --- | --- |
| **userid** | followedId |

expert

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **userId** | englishName | academicId | technicalField | homepageURL |

appeal

|  |  |  |
| --- | --- | --- |
| **id** | reason | state |

administrator

|  |
| --- |
| userId |

academicInstitution

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | name | englishName | type |

六、运行架构

并发技术选型

使用分布式消息队列Kafka，Kafka是一种高吞吐量的分布式发布订阅消息系统，它可以处理消费者规模的网站中的所有动作流数据。

Kafka 有如下特性：

1.以时间复杂度为O(1)的方式提供消息持久化能力，即使对TB级以上数据也能保证常数时间复杂度的访问性能。

2.高吞吐率。即使在非常廉价的商用机器上也能做到单机支持每秒100K条以上消息的传输。

3.支持Kafka Server间的消息分区，及分布式消费，同时保证每个Partition内的消息顺序传输。

4.同时支持离线数据处理和实时数据处理。

5.支持在线水平扩展。

Kafka的模块：

Broker：Kafka集群包含一个或多个服务器，这种服务器被称为broker。

Topic：每条发布到Kafka集群的消息都有一个类别，这个类别被称为Topic。（物理上不同Topic的消息分开存储，逻辑上一个Topic的消息虽然保存于一个或多个broker上但用户只需指定消息的Topic即可生产或消费数据而不必关心数据存于何处）

Partition：Partition是物理上的概念，每个Topic包含一个或多个Partition。

Producer：负责发布消息到Kafka broker。

Consumer：消息消费者，向Kafka broker读取消息的客户端。

Consumer Group:每个Consumer属于一个特定的Consumer Group（可为每个Consumer指定group name，若不指定group name则属于默认的group）。

线程同步

使用synchronized代码块来实现线程同步，由于同一进程的多个线程共享同一片存储空间，在带来方便的同时，也带来了访问冲突这个严重的问题。Java语言提供了专门机制以解决这种冲突，有效避免了同一个数据对象被多个线程同时访问。

synchronized关键字的作用范围：

1. 修饰一个代码块，被修饰的代码块称为同步语句块，其作用的范围是大括号{}括起来的代码，作用的对象是调用这个代码块的对象；

2. 修饰一个方法，被修饰的方法称为同步方法，其作用的范围是整个方法，作用的对象是调用这个方法的对象；

3. 修改一个静态的方法，其作用的范围是整个静态方法，作用的对象是这个类的所有对象；

4. 修改一个类，其作用的范围是synchronized后面括号括起来的部分，作用主的对象是这个类的所有对象。

线程间的通信

1.同步

2.while轮询的方式

3.wait/notify机制

4.管道通信就是使用java.io.PipedInputStream 和 java.io.PipedOutputStream进行通信

七、部署架构

服务器：在阿里云、腾讯云上租用的服务器

操作系统：CentOS 7.2 64位

前端服务器

CPU:10 Cores

内存:128G

硬盘:1TB SSD x4 RAID10

网络:GE端口

mysql数据库服务器：

CPU:20 Cores

内存:64G

硬盘:1TB SSD

网络:GE端口，FC端口

具体分为：

CDN服务器

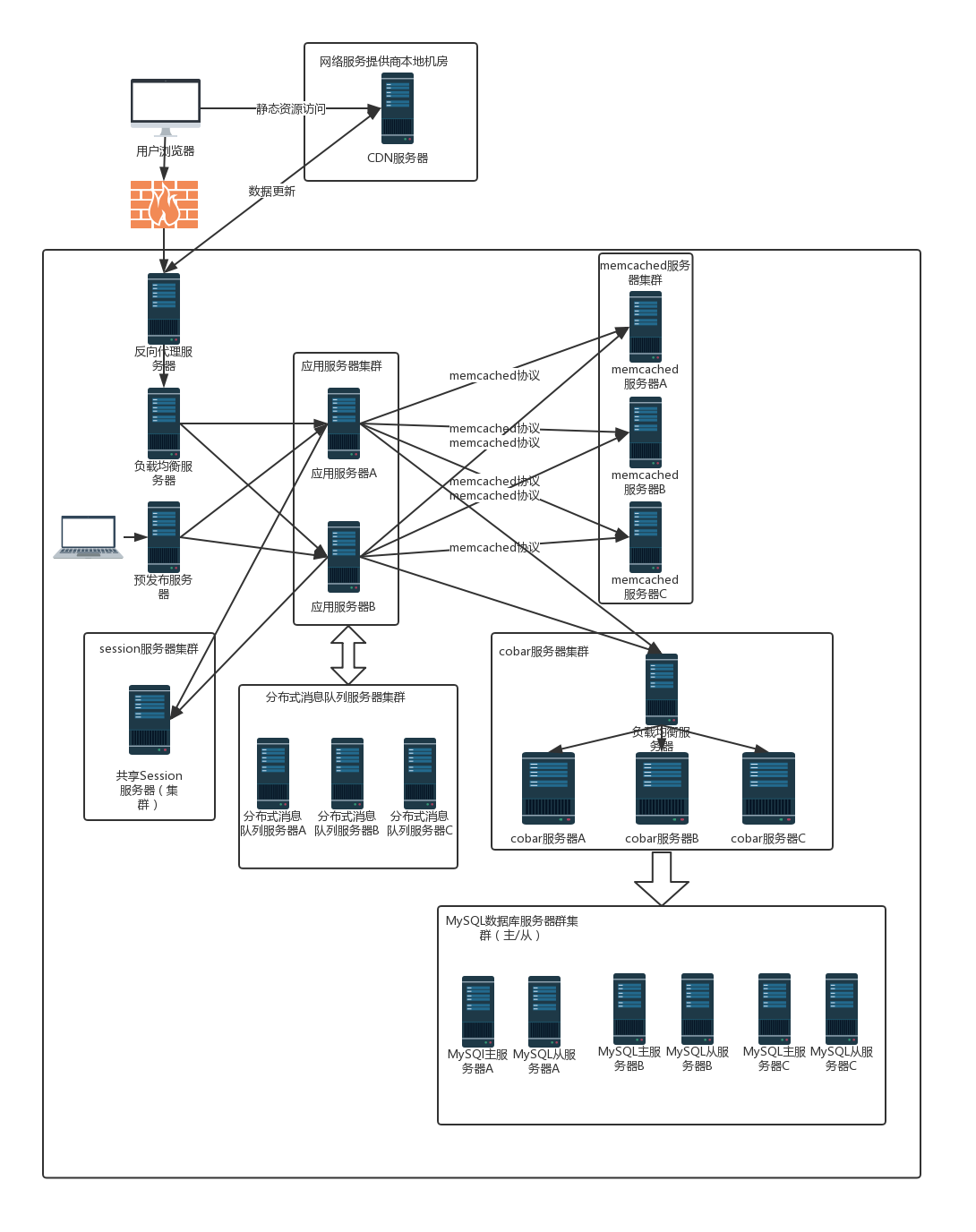
反向代理服务器

负载均衡服务器

应用程序服务器集群

session服务器集群

Cobar服务器集群

mysql主从数据库服务器集群（选用不同地域的服务器）

分布式消息队列服务器集群

memcached服务器集群

预发布服务器

具体部署情况如下图：

八、非功能性设计

安全性

1.把数据库的关键操作写成事务

2.对敏感数据加锁，使用时需要获得锁权限

3.对账，每日交易完成后进行对账

4.防注入

可靠性

1.对输入数据进行检查，防止数据异常

2.能处理系统运行过程中出现的各种异常情况，如：

(1)人为操作失误时不执行错误操作，且可正确提示错误操作并引导正确操作

(2)输入数据异常时返回错误提示

兼容性

1. 支持chrome，safari，edge，firefox等常见浏览器
2. 支持不同分辨率，包括手机端网页

高可用性

1.分级管理

在运维上对服务器采用分级管理策略，核心应用和服务优先使用配置更高的服务器，如用户注册、用户登录、用户充值、用户付款、用户转让资源、用户阅览资源，而非核心应用和服务，如用户反馈、用户评价等就使用配置较低的服务器。

同时在服务器部署上也进行必要的隔离，避免故障的连锁反应。由于我们使用的是腾讯云、阿里云的服务器，所以会租用不同地域、不同配置的服务器来确保安全性。

2.超时设置

由于服务器可能会宕机、线程可能会发生死锁，会导致应用程序对服务端的调用失去响应，进而导致用户请求长时间得不到响应、同时会占用应用程序的资源，不利于及时将访问请求转移到正常的服务器上。

因此我们会在应用程序中设置服务调用的超时时间，超时时间将根据不同服务的具体情况来设置，当发生超时情况，通信框架就会抛出异常，应用程序根据服务调度策略，我们会设置最大重试次数为5次，会先重试，达到最大次数后会将请求转移到提供相同服务的其他服务器上

3.异步调用

应用对部分服务的调用通过分布式消息队列完成，避免一个服务失败导致整个应用请求失败的情况。例如用户注册服务，提交一个新用户的注册请求，需要调用三个服务，将用户信息写入数据库，将注册成功的邮件发送至用户邮箱，开通用户账户的对应权限。不采用消息队列时，如果邮件队列阻塞不能发送邮件，则会导致其他两个服务也无法进行，最终导致注册失败。而采用消息队列，将立即返回用户注册成功，即使邮件队列阻塞也不影响其他两个服务的进行，用户只是晚一点收到注册成功的邮件而已。

4.服务降级

在网站访问高峰期，服务可能因为大量的并发调用而性能下降，严重时可能会使服务器宕机。采用服务降级的策略，保证核心应用和功能的正常运行，低优先级的服务采取拒绝或关闭的方式。

拒绝服务：拒绝低优先级应用的调用，减少服务调用并发数，确保核心应用正成使用。

关闭功能：关闭部分不重要的服务，如评价系统

5.幂等性设计

应用调用服务失败后，会将调用请求重新发送到其他服务器，但是这个失败可能是由于网络原因或其他原因造成的虚假的失败，导致服务重复调用。

有些服务不需要特殊设计，如设置性别，即使多调用一次结果也相同，但是转账等业务就需要就行特殊处理，避免重复服务。

九、其他约束

时间约束

整体工作必须在交作业之前完成

法律约束

以合法的方式调用专利局的数据，以合法的方式爬取网络资源，如论文，科研项目等。

质量约束

需要能在高峰期正常使用，在高并发的情况下也能正常运行，不出现无响应或响应时间异常缓慢的情况

经费约束

由于团队中的大部分人还没有收入来源，所以初期尽量租用廉价的服务器，合理部署。

工作方式约束

由于不是在固定地点进行集中开发，开发团队中的每个人通过github协同工作，会定期对各个分支进行合并、消除冲突，便于开发的进行。