# 文本复制检测报告单(全文标明引文)

ADBD2017R_20170531104203201705311150104312782785 28　　检测时间：2017-05-31 11:50:10

检测文献:　53130301_张意政_计算机科学与技术_基于vue和node的多人协作办公平台的研究和实现

作者:　张意政

检测范围:

中国学术期刊网络出版总库
中国博士学位论文全文数据库/中国优秀硕士学位论文全文数据库
中国重要会议论文全文数据库
中国重要报纸全文数据库
中国专利全文数据库
互联网资源(包含贴吧等论坛资源)
英文数据库(涵盖期刊、博硕、会议的英文数据以及德国Springer、英国Taylor&Francis 期刊数据库等)
港澳台学术文献库
优先出版文献库
互联网文档资源
图书资源
CNKI大成编客-原创作品库
大学生论文联合比对库
个人比对库

时间范围:　1900-01-01至2017-05-31

## 检测结果

总文字复制比：0.1%　　跨语言检测结果：0%

去除引用文献复制比：0.1%　　去除本人已发表文献复制比：0.1%

单篇最大文字复制比：0.1%

| | | | | | |
|---|---|---|---|---|---|
| 重复字数： | [34] | 总字数： | [26992] | 单篇最大重复字数： | [31] |
| 总段落数： | [6] | 前部重合字数： | [0] | 疑似段落最大重合字数： | [34] |
| 疑似段落数： | [1] | 后部重合字数： | [34] | 疑似段落最小重合字数： | [34] |

指　标：□疑似剽窃观点　□疑似剽窃文字表述　□疑似自我剽窃　□疑似整体剽窃　□过度引用

表格：0　　脚注与尾注：0

0%（0）　中英文摘要等（总1740字）
0%（0）　第1章绪论（总1579字）
0%（0）　第2章论文主体_第1部分（总9423字）
0%（0）　第2章论文主体_第2部分（总9260字）
1%（34）　第2章论文主体_第3部分（总3452字）
0%（0）　第3章总结（总1538字）

（注释：无问题部分　文字复制比部分　引用部分）

## 1. 中英文摘要等　　总字数：1740

相似文献列表　文字复制比：0%(0)　疑似剽窃观点：(0)

### 原文内容

吉林大学学士学位论文（设计）承诺书

本人郑重承诺：所呈交的学士学位毕业论文（设计），是本人在指导教师的指导下，独立进行实验、设计、调研等工作基础上取得的成果。除文中已经注明引用的内容外，本论文（设计）不包含任何其他个人或集体已经发表或撰写的作品成果。对本人实验或设计中做出重要贡献的个人或集体，均已在文中以明确的方式注明。本人完全意识到本承诺书的法律结果由本人承

担。

学士学位论文（设计）作者签名：

2017年5月20日


摘要

基于Vue和Node的多人协作办公平台的研究和实现

随着移动互联网和web2.0的发展，移动在线办公，多人协作办公逐渐成为企业办公的主流方式。如何提升办公效率，消除员工之间交流的障碍，高效合理的用于管理项目进度，发布需求成为这些产品的发展方向。Google DOCS和Microsoft Office，凭借他们的各自突出优点逐渐占领市场，越来越多的面向企业的产品涌入这个竞争趋于白热化的市场。本文将分析研究目前市场现有的办公产品的特点，然后基于Vue和Node实现一个在线B/S办公平台，其中会利用到Redis来解决数据库I/O的瓶颈，使用PM2来提高系统的健壮性，利用云服务器弹性伸缩来快速部署项目。在实际开发中，会对遇到的技术难点，提出有针对性的，合理的解决方案。最后总结该系统的功能特点和不足之处，以及对下一步迭代开发的计划。



关键词：Vue，Node，Redis，Express



Abstract

Research and implementation of multi person cooperative office platform based on Vue and Node

With the development of mobile Internet and Web2.0, mobile online office and multi person cooperative office have gradually become the mainstream way of enterprise office. How to improve office efficiency, eliminate barriers between employees, efficient and reasonable management of the project schedule, the release of demand has become the direction of these products. Google, Docs and Microsoft Office are gradually occupying the market with their respective outstanding advantages, and more and more enterprises oriented products are pouring into the market which is becoming more and more competitive. The paper will analyze the characteristics of existing office products, then Vue and Node to achieve an online office platform based on B/S, which will use Redis to solve the bottleneck of I/O database, using PM2 to improve the robustness of the system, the use of cloud server flexibility to quickly deploy the project. In the actual development, the technical difficulties encountered will be targeted and reasonable solutions. Finally, the functional features and shortcomings of the system are summarized, and the plan for the next iterative development is also given.



Keywords：Vue，Node，redis，Express


目录

## 2. 第1章绪论

总字数：1579

相似文献列表　文字复制比：0%(0)　疑似剽窃观点：(0)

原文内容

第1章绪论

1.1 当前类似产品分析

随着企业办公产品的逐渐发展，越来越多的产品融入这个竞争的激烈的市场。以Microsoft Office为传统代表的，功能全面，体验良好，跨多端移动办公的办公平台；以Google Docs为代表的轻量化，即插即用，多人在线实时编辑，高性能的办公平台；成为这一市场的领头羊。在进行产品设计和需求开发之前，我们首先分析市面上的一些优秀的产品，了解它们通过怎样的方式逐渐发展，并牢牢占据企业市场的。这些产品的功能特点以及发展轨迹将对对后期的产品设计带来启发，我们将会借鉴这些产品的优点，不断完善产品。

Microsoft Office是微软公司推出的办公套件 [1]，于1993年发布了Office3.0，当时的office包括Word，Excel，PowerPoint，Mail等套件。凭借其强大的功能逐渐占领市场，在现今的企业办公市场占有较大的份额。随着Office的发展，目前其已支持Microsoft Windows ，macOS，Windows Phone，iOS，Android等多个平台或操作系统，包含更丰富的功能套件，例如：OneNote，SharePoint，Visio，InfoPath等，随着“云”理念的逐渐深入人心，Microsoft Office也支持云端编辑保存文档。最新版的Office 2016，支持协同处理文档，可以多人在线编辑和查看文档，能够查看编辑的历史记录；并且简化了共享功能，可以从OneDrive中分享文档给其他的用户，并且可以配置相应的访问权限。

Google Docs是Google推出一款的在线办公套件[2]。其主要包括文档，表格，幻灯片，与Microsoft Office的三大著名组件相类似。Google Docs是基于B/S架构的，所以可以在各个操作系统的浏览器上打开，同时其还提供了移动端的APP，可以在智能手机上安装使用。用户既可以在浏览器或者移动应用中在线创建电子文档，也可以通过将本地文件导入Google文档中，进行编辑和分享。在线编辑的文档可以自动保存在服务器上，用户也可以将这些文件以多种格式下载到本地。比较方便的是，每次用户编辑之后，文档将会自动保存到服务器中，所以不用考虑正在编辑的文件没来得及保存而丢失。Google文档支持多用户协同工作，多个用户可以同时打开和编辑同一份文档，并且，在线编辑的文档可以以URL链接的方式分享出去让其他用户查看和编辑，这项功能十分强大。Google Docs默认启用HTTPS协议，在安全性上也有较好的表现。

1.2 当前相关产品特点总结

从早期的office，到现在的Google Docs，不难看出在线协作管理平台的发展规律：早期的产品，以微软的Office为代表，其主要的特点是功能齐全，能够满足办公中的各种需求，其诞生的主要原因是为了弥补办公平台的不足，不支持云端存储和在线编辑，不支持分享，主要的架构也是C/S。但是，随着移动互联网和web2.0的发展，pc不再是唯一的办公平台了，以Google Docs为代表的新兴产品横空出世，这个时候的一系列产品有一些共同的特点：支持多平台办公，包括Windows，mac OS，iOS, Android, web等。支持云端存储，支持在线实时编辑，支持分享，用户体验良好等特点。协作平台的演变，背后的动力包括：编程技术的发展解决了很多以前无法解决的技术难题，移动互联网的发展使办公人员增添了以移动办公为主的需求，企业的发展壮大，需要更加高效，更加人性化的办公工具来提升工作效率，消除壁垒等。

在此基础上，我将利用现有的技术体系，来构建一款轻量，高性能的在线办公平台，其基于B/S架构，提高可访问性，并通过相关的类库和框架解决开发中遇到的一系列技术难题。

## 3. 第2章论文主体_第1部分

总字数：9423

相似文献列表　文字复制比：0%(0)　疑似剽窃观点：(0)

原文内容

第2章论文主体

2.1 多人编辑技术研究

目前，大多数主流的办公产品仍然是单用户编辑的系统，若要多个用户参与其中，需要参与者事先进行沟通，划分各自的任务，这无疑会降低效率，增加工作量。因此，我们期待一种更好的工作方式：多个用户同时参入到一项任务中，一个用户能立即看到其他用户在这一时刻对工作内容做出的修改。这种工作方式无疑是一种革命，较少了很多与实际工作内容无关的工作量。协作编辑系统，通常分为协作感知系统和协作透明系统。协作感知系统以Prep，REDUCE 和SEPIA 等各种协作编辑系统为例，虽然提供了对协作的支持，但缺点是编辑功能比较弱，无法满足复杂的办公需求。协作透明系统使用中央服务器，为每个用户生成一个文档的副本来进行编辑，编辑的内容将会被合并广播到其他文档的副本实例上。多用户在线编辑的难点是并发控制，一个好的并发控制决定了在线编辑的质量。通常，并发控制机制包括交换，锁定，序列化，事务和可操作转换。这些并发机制通常划分为悲观机制（交换，锁定，悲观序列化）和乐观机制（乐观序列化和可操作转换）。悲观机制的优点是简单，易于实施，无需额外的操作，乐观机制的优点是灵活，复杂，支持无约束任意编辑的合作。

在多个用户编辑同一文档下，是否对编辑者进行约束也很重要。在这种情况下，编辑的次序被显示的转化，多个用户的编辑应该是快速的上下文切换，一个时候只有一个用户在编辑。并且这个操作可以被其他用户感知，这个编辑的次序由令牌控制，对令牌的访问是互斥的，当一个用户获取到令牌，则可以进行将文档编辑过的内容推送给服务服务器，由服务器推送给其他用户；没有得到令牌的用户也可以编辑，但是编辑的内容仅会被推送到服务器，在下次获取到令牌时进行由服务器同步给其他用户。

图2-1 Server-client架构

当用户在client窗口中键入字符时，为了提高性能，修改的内容将被buffer缓存起来，当用户得到令牌时将会被同步。因为有多个用户参与编辑，所以在server中需要为每一个接入的client保存状态信息，包括client的IP Address，port，文件信息和状态，并发控制等。

2.2 基础架构设计

我们知道，随着目前各大编程语言引入JIT，语言本身已经不再是web应用性能的瓶颈了，性能的主要瓶颈是对I/O的操作。并且要提高一个应用的性能，主要应该从瓶颈处入手。因此，在开发项目之前，我着力于从I/0的解读来提升整个后端系统的性能。为此，在基础的MySQL数据库层和后端逻辑层之间，引入Redis内存数据库[15]来突破性能的瓶颈。通常，对内存和硬盘执行随机访问，速度差距是非常明显的。就我们的应用而言，在用户发送消息时，如果每次用户发送消息都执行一次对硬盘的I/O操作，这会极大的增加性能的损耗。因此，针对用户频繁发送消息这一特点，考虑将用户发送的消息保存在Redis内存数据库中，在数据保存到一定量时，将消息推送给对应的用户，并且将其同步到MySQL中，这就在MySQL前建立了一道数据缓冲层，提高I/O性能。

图数据库-消息缓存

2.3 后端开发语言和框架选型思考

一门好的后端开发语言，其必不可少的优点是：稳定，高性能，开发效率高，生态好。常用的后端开发语言包括Java，PHP，Python等。以下，将这几门语言做一些对比。

Java作为一门优秀的企业级应用开发语言[3]，基于JVM的跨平台特性，让“一次编写，到处运行”成为可能。其最初由Sun公司开发用于嵌入式机顶盒，但由于其跨平台和良好的性能，让其逐渐在后端开发中占据重要位置。Java的出现，让大规模协作编程变得更加容易，应用的维护也更加的轻松。Java作为工业级语言的优秀代表，涌现了一大批优秀的框架，例如著名的“SSH”，Mybatis，Ibatis，JSF，vaasin等等。这一系列优秀的框架，使得Java开发具有很多可选的方案，不为受限于某个具体的框架，更加的灵活。这里我们不选择Java的理由是，其过于“重”，部署成本比较高。

PHP[4]作为“最好的语言”，使用非常广泛，基本上各大公司都有使用。PHP最大的特点就是部署方便，LAMP套件，使得不需要过多的配置，可以直接部署Web服务。同时，PHP语言集成了大量的插件功能，包括对数据库的支持，无需另做配置。PHP有众多web框架，包括Laravel，Symfony，Nette等。并且，PHP7的发布，使得PHP性能上的缺陷得到了补足。但是PHP对多线程支持，对异步编程的支持是其主要的缺点。

Python是一门优秀的编程语言【5】，作为后端web开发语言完全能胜任。Python最大的特点就是开发效率高，语法特性比较优美，能用较少的代码完成较多的工作。并且Python拥有一系列优秀的web框架，可以直接生成web服务器实例。但是Python最主要的特点就是性能较一般，虽然可以使用pypy或者是引入JIT等方式来提高性能，但是需要额外的配置工作，并且Python已经分化成了2版本和3版本，这两个版本不完全兼容，因此部分老旧应用升级比较麻烦。

谈谈我们为什么选择Node作为后端开发语言。Node是JavaScript服务器端运行时[6]，由C语言开发完成。在Node之前，也有许多的JavaScript服务器端实现，包括Mozilla的Rhino。拿Rhino来说，其设计的实现，参考了Java，更多采用了同步的逻辑。而Node的开发者最初的目的是创建一个web服务器，根据他多年的C/C++开发经验，觉得高性能的web服务器需要具有事件驱动，异步I/O的特点。其中备选的语言有很多，包括Scala，Ruby，Lua等。其中，与要求最为契合的就是JavaScript。JavaScript有许多有趣的特点，事件驱动，单线程，异步I/O，回调等。单线程的特点，让开发者无需关心死锁和同步的问题，并且不会具有多线程编程上下文切换带来的开销。在实际开发中，只需要关心主线程的执行流程。事件驱动，这个概念广泛的存在浏览器的JavaScript执行中，服务器端事件驱动带来的是一种编程思想的统一，使得前后端一致。异步值得Node天生具

有并行编程的理念，相较于同步阻塞的运行方式，异步使得关注点分离。利用JavaScript的事件和非阻塞特性，Node非常适合I/O密集型任务，同时，依赖于Chrome中V8引擎，在计算密集型任务中也有良好的表现，并且，还可以通过利用C/C++扩展进一步提高性能。并且，通过在操作系统和Node上层系统引入libuv这个中间层，使得Node具有跨平台的特性。Node的另一个优点就是其包管理工具NPM[7],一个优秀的包管理工具是非常重要的。对比其他语言，Python有pip，Php有Composer，Ruby有gem，，Node的包管理工具NPM无疑是非常优秀的，其发布和安装的机制都非常的方便，并且，Facebook还开源了新的包管理工具yarn[8]，新增锁机制。同时，NPM生态非常的繁荣，上面有数十万个开源的第三方模块。在后端开发中使用Node，可以减少与前端交互的负担，因为都是JavaScript语言，也省去了前后端开发语言不同带来的思维切换的负担。但是Node的语言特性，也导致它产生了一些缺点：由于当今的web应用为了提高性能，都是建立在多核服务器上，Node的单线程机制导致它无法利用多核CPU的优势，并且，单线程的Node服务，如果这个线程挂掉，那么整个应用就无法访问了。并且，Node受限于JavaScript的异步回调机制，在涉及到多个回调嵌套时，发生异常捕获，维护代码都十分费力。后面将详细阐述利用web workers和PM2来提高Node对多核CPU的利用以及程序的健壮性，利用Promise API来解决 “回调地狱”。

虽然Node提供了HTTP以及一大批优秀的第三方模块供我们构建web服务器应用，但是仅仅有HTTP服务还不能构建一个大型的web服务，我们还需要控制路由，缓存，校验权限，Cookies的管理，跨域的处理等等。为了快速的搭建应用，我们需要使用web框架。对于Node web框架，较为著名的有Express[9]和Koa[10]。Express最大的有点就是可扩展性极强，它的强扩展性主要体现在其丰富的中间件上。中间件就像一个函数清单一样，用户的访问经过路由系统再流经这些中间件，经处理后将访问传递给下一个中间件，这样就能完成很多操作。并且Express模板渲染引擎，方便进行UI模块的渲染和展示。Koa是Express原班人马打造的另一个及其优秀的NodeJS web开发框架。Koa与Express的不同主要有三点：一是，中间件的级联方式，Express的中间件是顺序执行的，用户的请求会一个接一个的流过中间件函数，而koa会有一个会穿的过程，先向下执行，在向上执行。其次，Express使用callback作为请求的处理方式，在涉及到多个异步的处理时，需要手动组织相关逻辑，而koa使用generator，新版的koa甚至使用了最新的Ayn/Await新特性来解决calllback的问题。最后是处理函数参数的不同，Express的处理函数主要有request对象和response对象，这两个来自于Node，分别代表HTTP中的请求对象和响应对象；而koa中的参数为context，context封装了request对象和response对象，在更加底层的基础上去定义这个对象，在编写中间件的时候就无需考虑这里面的逻辑了，更加的友好。Koa总体说来，更加的定制化，不在框架层作任何的限定，基本上所有的功能点都是通过第三方的中间件来进行提供。又因为Koa来源于Express，Express中大多数的中间件都可以经过一些小幅度的修改直接用在Koa上。比较之下，我们在这里选择了Express作为后端开发框架。主要原因是，Express经过长时间的发展，生态相对Koa来说更加的稳定成熟，其丰富的中间件是Koa无法进行比拟的；同时，Express的上手难度相对于Koa来说要更加的简单，并且，Express拥有更长时间的应用实践，相关问题的解决方案都比较成熟。

2.4 前端框架的选型思考

在前端框架的选择上，我们着眼于选择一个灵活，轻量，插件丰富的框架。目前，前端开发领域发展非常迅速，大量的框架层出不穷。无论是以前常用的jQuery，backbone，YUI，还是目前比较流行的Angular，React和Vue，都有一大批忠实的用户。为了选择一个适合当前场景的前端框架，我们需要深入了解各个框架的特点，然后做出选择。

常规的早期框架例如jQuery[11]，其出现主要的目的是为了兼容早期不同的浏览器，其提供了功能强大的选择器以及丰富的DOM操作。但是这一类型的框架有一些缺点，手动进行DOM操作，在复杂的业务场景下，项目代码的复杂度会急剧上升。

Angular是Google开发的一个前端框架[12]，其主要特点是MVC，模块化，双向数据绑定，依赖注入等。Angular的出现，对前端开发来说是一次革命，相对出传统框架手动操作DOM来说，Angular能让开发人员更加专注于当前的业务需求，双向数据绑定让我们不再需要手动操作DOM。但是Angular为前端工程师所诟病的就是其”脏检查”机制，并且Angular的升级到2.0后，使用了Micorsoft开发的TypeScript框架的复杂度和上手难度都上升了一个数量级。

React是Facebook开发的一个前端框架[13]，主要特点是模块化，MVVM，JSX，单向数据流，虚拟DOM。React刚在GitHub上开源就获得一大批拥趸，其思想也是创新性的。React独创了JSX语法，作为JavaScript的语法扩展，我们可以像调用JavaScript的表达式一样来调用JSX，其支持嵌套，条件判断等方式，具有非常强大的表达力和灵活的易懂性。并且，React将其组件定义了state和props两个可用的属性，对于如何维护自身状态以及组件通信带来了良好的时间。并且，来自于React社区的Redux关于前端状态管理的创新观念，使得大规模前端应用的组织和管理变得更加井然有序。

Vue是尤雨溪主导开发的一个渐进式前端开发框架[14]。其具有双向数据流，组件化，虚拟DOM，MVVM，服务器端渲染等特点。Vue结合了Angular和React的优点，提供了丰富的指令，降低开发难度。并且Vue非常灵活的框架，对具体的开发技术选型没有特别的限制，但是又提供了大量的插件与其配合，供开发者灵活的选择。Vue提供了组件化的开发方式，使得多个开发者只需要关注自己的组件功能特点，降低了耦合性。

通过对比了以上的各个类型的框架，我们最终选择了Vue作为前端开发的框架。理由有很多，其文档友好，非常灵活，组件化的开发方式使得关注点分离，同时，其生态优秀，有大量可使用的插件供在各种场景下使用，并且十分的轻量化。

2.5 数据库选型和思考

数据库选型上面，主要考虑稳定和成熟的数据库。现在的数据库主要分为两类，一类是关系型数据库，一类是文档型数据库。以下对一些常见的数据库进行对比分析：

MySQL是一款流行的关系型数据库，其具有性能高，可靠性好的特点，得益于广大中小型网站的崛起以及LAMP套装的部署方便简单的特点，MySQL被广泛安装使用，其被誉为最流行的关系型数据库，在数据库市场占有大量的份额。MySQL支持多种操作系统以及多种编程语言，在实际的使用过程中，只需要进行简单的配置就可以使用，而且随着MySQL的发展，它的各种文档都很完善，遇到问题能够很快解决。一些公司利用MySQL进行二次开发，构建自己的数据库集群。

PostgreSQL是发源于UC Berkeley的关系型数据库。Postgre-SQL对SQL标准的支持非常完善。PG默认为Innodb引擎，在一些极端场景下非常可靠。PG支持高并发读写，在负载逼近极限的情况下，PG的性能指标仍能维持双曲线和对数双曲线，到顶峰后不再下降。PostgreSQL有多种索引可以使用，包括条件索引和函数索引，并具有无锁定特性。另外，PostgreSQL支持trigram索引，可以帮助改进全文搜索结果，实现高效的正则搜索；能灵活高效的处理图结构；PostgreSQL可以直接组织返回JSON的结果。PG支持继承，其数据表的结构及属性可以从一个“父”表中继承，数据可以在两者之间进行共享。多一个表中的数据进行操作也将会把结果反映到另一个数据表中。

Miscrosoft SQL Server，是Microsoft开发的关系型数据库，运行在windows操作系统下。通常在做linux系统下的web开发时，有很多的选择，包括操作系统发行版，后端编程语言，数据库等等。而微软提供了成熟的软件解决方案，不需要纠结解决方案的选择，包括windows server操作系统，C#/VC++等都有成熟可靠的产品。该数据库基本的数据机构是纵向划分，分为协议层，关系引擎，存储引擎，SQLOS。SQL执行的过程就是逐层解析，一层一层的向下解析SQL。

MongoDB是一款流行的文档型数据库，支持bson类型的数据，非常适宜做文档的存储。它提供了对多种语言的支持，并且能跨平台使用。能够提供更加灵活的操作，以及更加复杂的数据结构。MongoDB中文档的key和value无需定义具体的大小，因此在实际的开发中不需要定义传统关系型数据库中的“表”，更加的方便。

HBase是一个运行在Hadoop上的分布式存储系统，具有可伸缩，高性能等优秀特点。其设计的思路来自于Google Big Table。HBase对环境要求特别高，需要运行在HDFS上，但具有高度扩展性。HBase可以通过键进行随机访问，也可以通过map-reduce进行脱机或批访问。正是由于这两种访问方式，让HBase既可以支持实时访问，又可以支持离线访问。

2.6 基于Sequelize的ORM数据库操作

在操作MySQL数据库时，考虑到常规SQL数据表现力不足，无法真实反映真实对象之间的关系，并且为了提升数据库I/O的性能，需要引入数据库连接池。在这里我们使用Sequelize，Sequelize[16]是一个基于Pormise的NodeJS数据库连接驱动，它提供了对PostgreSQL， MySQL, MariaDB和MSSQL的连接支持。并且其提供对事务，关系，读复制等特性的支持。在我们应用数据库设计上，主要有User，Group，Document，Message四个实体对象，他们的设计大致如下：

图分组实体数据类型

表示分组的实体Group共包含6个字段，分别是groupId, groupName, ownerId, members, documents, headPicture, 其中groupId为主键，ownerId为外键，指向User中的userId;

图文档实体数据类型

表示文档的实体Documents共包含documentId，documentName，groupId，ownerId，content五个字段，其中documentId为主键，groupId和ownerId为外键，分别指向Group的groupId和User中的userId

图消息实体数据类型

表示消息的实体Messages共包含senderId，receiverId，message，date四个字段，其中senderId和recieverId既是主键也是外键，指向User中的userId。

图用户实体数据类型

表示用户的实体User共包含userId, username, password, phone, email, headPicture, friends, groups，其中userId为主键

因此，我们在同一个Model.js创建数据库连接池，在连接到连接到数据库时会自动创建五个数据库连接，并进行管理。声明四个实体对象的属性以及对象之间的关联关系，然后将其实例化，这就完成数据库表的创建。另外，在涉及到对数据库表的CRUD操作的时候，我们为每一个对象创建一个单独的CRUD操作的JavaScript文件，在这四个文件中导入Model.js,在这四个文件中可以利用Sequelize为实体对象定义的一系列方法，例如create, destroy, update, findOne, findAll等进行数据库操作，免去频繁编写SQL语句的烦恼。在执行对应的方法后，结果会Promise对象的形式返回以供调用操作数据库。

图2-3 数据库系统设计

2.7 基于WebSocket的服务器端推送

在应用的需求中，当一个用户给另一个用户发送消息时，需要将用户发送的消息，通过服务端推送给接收该消息的用户，所以需要提供服务端推送的能力。常规的服务器端推送技术包括Comet，Ajax轮询，Ajax长轮询，WebSocket等。这里因为后端使用了Node作为开发语言，WebSocket又有成熟的Node库Socket.io[17],因此选择使用了WebSocket方案。

在搭建服务器端推送时，还有一个棘手的问题需要解决。众所周知，HTTP协议是无状态的，这里的具体含义是，客户端是否在线，服务器端是不知道的。如果客户端不在线，那么服务器端向客户端推送消息之后，客户端是无法查看到的。因此，为了在服务器端侦测客户端是否在线，需要模拟连接的情况，这就需要使用到心跳系统。心跳系统，就像心跳一样，定时发送短小的数据包，用于证明客户端在线，如果客户端持续发送心跳包，服务器端就判定用户在线。为了搭建心跳系统和服务器推送系统，我们在服务器端引入socket.io这个Node模块，在客户端引入socket.io-client这个JavaScript类库。

在客户端，当页面的Vue实例初始化后，我们在“mounted”这个回调函数中，初始化心跳包的发送和响应，Vue实例相关内

容将在后序章节详细介绍。客户端心跳包的发送，我们定义为2分钟发送一次，服务器端接收这个心跳包，并且计数，当每接收到10个心跳包，也就是每隔二十分钟，会在redis读取对应的用户消息，然后推送至客户端，这就完成了服务端推送的搭建。

图2-4 消息推送系统设计

2.8 前端图片上传和后端图片管理

在用户注册时，用户可以选择自己上传自己的头像。通常前端提交表单有常规的方式和使用Ajax异步提交的方式。常规模式下，我们无需做过多的配置。但是，如果使用Ajax异步提交的话，需要对图片上传提交做一些修改，在这里，我们使用FormData[18]这个HTML5的新特性，它主要的特点是自动对表单数据进行序列化。在后端处理上传的图片时，我们使用multer[19]这个Express框架的中间件，它可以拦截请求，并且自动读取提交的数据。另外，在管理和存储用户上传的图片的时候，这也是一个比较棘手的问题。因为，我们不能把用户上传的图片直接存到数据库中，如果直接存到数据库中，每次获取图片的时候，都需要操作一次数据库，那么这将会对I/O造成很大的负担。因此，在处理图片文件管理和存储时，我们根据linux文件系统的特性，选择将部分linux的部分文件路径划分出来存储图片，然后在数据库中保存对应文件的路径名。并且在express的路由中设置专门用户处理图片请求的路由，根据请求的路径名返回请求的图片。

2.9 前端构建和辅助开发相关准备

前端发展发展非常迅速，每年都有很多新的轮子被造出来，如何选择也成为了一到难题。在进行前端开发相关工作之前，需要一定的时间进行相关的技术准备。

在代码开发过程中，为了提升代码的编写质量，需要一些代码检查和格式化工具来控制代码编写的风格，避免因为代码编写过程中的疏忽引入一些低级的错误。在这里，我们使用Eslint[20]来进行代码检查。使用Eslint时，需要进行配置，其主要的配置方式有两种：第一种，使用JavaScript注释直接把配置嵌入到JS文件中；第二种，使用配置文件来为全部的目录和它的子目录指定配置信息。在这里我们使用.eslintrc文件配合WebStorm的Code Quality Tools工具来进行代码格式的定义和检查。只需要在WebStorm的settings中，将Code Quality Tools中对应的Eslint选项开启，并且找到Eslint的文件位置，然后在项目根目录中引入.eslintrc文件进行配置即可。

并且随着代码的复杂度越来越高，资源的种类越来越多，常规的通过文件目录来组织管理代码的方式就无法适用了。我们需要引入构建工具来打包，分割，管理资源。在这里我们使用Webpack[21]来作为模块打包构建工具

## 4. 第2章论文主体_第2部分

总字数：9260

相似文献列表　文字复制比：0%(0)　疑似剽窃观点：(0)

原文内容

Webpack相较于grunt，gulp，browserify等构建工具，提出四个概念，入口，出口，插件，加载器。具体的，使用webpack.config.js配置文件将前端工程定义成的开发模式和构建模式。在平时开发调试的时候，使用开发模式，并且依赖于Webpack Hot Loader插件，我们的每次修改，会使浏览器自动刷新，立即反应代码的变化。在开发调试完成后，切换为构建模式，将开发工程目录下的所有文件，通过Webpack的加载器加载打包成HTML，CSS, JavaScripp文件。

同时，为了使用JavaScript语言的新特性，前端开发使用ES6[22] 进行开发。ES6具有一些JavaScript的新特性，包括但不限于适用let定义块级作用域变量，使用cost定义常量；使用”=>”符号创建格式更加简洁的函数；使用import/export引入或导出代码模块；使用解构赋值分割对象和数组；使用“`”和“${}”定义模板字符串;使用“class”关键字定义类;使用Promise关键字提供原生Promise对象解决异步回调等。由于不是所有浏览器都能支持新特性，因此我们引入babel这个代码转换工具，将ES6转换成ES5版本的JavaScript。由于使用了Webpack，这里使用babel-loader这个加载器加载JavaScript文件即可。

2.10 网站部署与稳定性建设

在启动服务实例时，因为我们使用Express框架，它可以自动启动一个服务器，但是，由于Express过于“薄”，我们需要引入其他的web服务器。在这里综合性能和稳定性，选择使用了Nginx反向代理服务器，Nginx与Node一样，都是基于事件机制，配合自然非常好。这里需要区分正向代理和反向代理。正向代理，通常代理服务器代理客户端去向服务器发出请求，服务器并不知道真正的客户请求来自哪里。反向代理，利用类似于Nginx反向代理服务器，用户发出请求到反向代理服务器上，用户不知道也无需知道处理用户请求的服务器具体是一台，而是通过反向代理服务器将用户请求转发到背后的服务器中，待处理完毕之后，再把请求的结果通过反向代理服务器发送给用户。

另外，在前面讲到了Node的优缺点，Node特点是单线程，，单线程带来了无需关心同步和锁的优点，也没有创建线程和线程切换的负担；但也带来了单线程程序健壮性不够，无法充分利用多核CPU来提高性能的缺点。为了提高健壮性，可以利用Node本身的特性，比如监听Process对象的exit事件，如果这个事件发生了，我们可以重启整个服务，但这只解决了健壮性的问题。为了提高程序性能，可以利用Node的child_process模块。child_process的机制是，通过JavaScript创建一个自包含的环境，这个环境开启一个独立的线程，可以在这个环境执行一些计算密集型任务，在执行完成后，通过消息传递机制将计算的结果返回给主线程。这种机制有一些限制，比如无法同主线程共享状态，只能通过消息传递机制进行通信等。同时也带来了一些好处，首先，这个单独的线程是无法访问主线程的变量的，这就不会带来同步的问题，并且，在这个线程执行长时间计算时

，并不会阻塞主线程，这就使得主线程和child_process单独执行不同的任务。

除此之外，还需要考虑负载均衡的问题，当一个应用得到大规模的扩展之后，负载均衡就不能忽视了。如果一个应用被部署到多个服务器上，面对大量用户的请求，怎样将这些请求均匀的分发到各个服务器上，这是后续需要考虑的问题。常规的负载均衡的方式大致有如下几种：

链路层负载均衡。DNS负载均衡是较早的负载均衡解决方案，通过DNS域名解析协议，将不同地理位置的用户请求解析到不同的IP,这些IP都对应独立的服务器，这样，就可以实现不同地方的用户访问不同的服务器，实现均衡负载。DNS负载均衡非常的简单，只需要修改DNS解析，不需要花费额外的精力去维护。但是DNS也具有生效时间较长，扩展性和维护性较差的特点。

IP负载均衡：IP负载均衡服务器的主要原理是，设置一台负载均衡服务器，当用户的请求到达该服务器时，服务器拦截用户发出的数据包，然后通过负载均衡算法得到负责处理用户请求的真实服务器IP地址，然后将请求需要访问的IP地址改为这个处理请求的服务器IP地址进行转发。IP负载均衡的主要优点是生效及时，性能比DNS负载均衡更好，但是，由于所有的请求都需要流经负载均衡服务器，如果负载均衡器服务器配置不够好，或者负载均衡算法不够好，那么负载均衡系统的性能将大打折扣。

链路层负载均衡：相较于IP负载均衡工作在网络层，链路层负载均衡工作在链路层上，更加的靠近底层。在这里，同样需要设置负载均衡服务器，这个服务器拦截所有用户请求，然后通过负载均衡算法得出一个服务器的MAC地址，然后根据MAC地址，直接将用户请求转发到这个负责处理用户请求的服务器上。链路层负载均衡相较于IP负载均衡，性能更好，省去了IP地址到MAC地址转换的过程。但是，链路层负载均衡在架构上更加的复杂，可维护性也要差一些。

综上，为了处理如上三个问题，需要将多种方案集合起来使用，为了能优雅快速的解决上面的问题，我们引入了PM2这个NodeJS的框架。PM2[23]是一个用NodeJS编写的框架，其提供负载均衡，后台运行，不停机重载，控制台检测，详细日志系统等功能。PM2利用Node内建的cluster模块，可以开启多个进程，达到集群化的目的。同时，为了方便管理，我在package.json文件中，针对PM2提供的start，restart，stop等功能配置了相关的npm script，可以实现命令行输入简单命令完成开启服务，重启服务，停止服务等功能。

2.11 文档编辑相关功能和数据架构的实现

在项目中，最主要的功能就是实现文档编辑的功能，而且还需要实现多人编辑的功能，这里的技术难度较大，目前Google Docs的产品体验很优秀，但是相关的技术资料比较少，所以需要探索实现这一功能。

为了实现文档编辑的功能，我们首先需要定义相关的数据库结构。这里我们为文档单独创建一个数据表documents，其包含documentId，documentName，content，ownerId，groupId等字段，我们将对应文档内容的conent字段定义为string类型，因为常规的富文本编辑插件获取真实的内容就是string类型的。

其次，我们使用了wangEditor这个富文本插件来实现文档的编辑功能。但这里有个问题，wangEditor提供的引入方式是通过script标签，而在前端开发的时候是使用webpack模块打包的方式，这样在Vue的组件中，就无法访问wangEditor暴露出来的API。查阅webpack的文档，我们找到了对应的解决方案，通过在webpack.config.js的externals引入对应的字段，既可以在Vue的组件中通过require的方式访问wangEdtior暴露出来的API。

在实现文档编辑功能时，在用户登陆时，当用户要访问某个文档时，会为用户初始化一个文档编辑的实例，并且，通过WebSocket在后端维护一个数据结构，这个数据结构用于统计某个文档当前访问的用户数，当用户编辑这个文档或者离开时会自动的增加或减少响应的文档对应的用户数。

在用户编辑文档时，利用WebSocket通知后端系统将上次编辑完成后的文档内容发送到客户端，客户端将上次的内容与此次用户编辑的文档内容进行比对，得出最佳的内容，并将该内容发送到客户端作为匹配的副本。在这里对两个文档进行比对的时候，比对的算法是提升性能的关键。因为，富文本编辑器中的内容其实就是HTML的DOM tree，如果从根节点往下一个接一个的进行DOM Node的对比，这里的时间复杂度约为O(n^3),因此我们引入了React里面著名的diff[23]算法来提升对比的性能，其时间复杂度将维持在O(n).这里详细阐述一下diff算法的思路， 在比较新旧两棵DOM tree的时候，首先比较根节点，如果两棵DOM tree的根节点不同时，我们判断这是两棵不同的DOM tree，这是只需要简单的移除老的DOM tree，引入新的DOM tree，如果根节点相同，这个时候我们使用类似树的层次遍历的方式来详细比较其他的DOM Node，如果DOM Node类型不同，则移除老的DOM Node，插入新的DOM Node，如果DOM Node的类型相同，但是attributes不同，只需要修改老的DOM Node上的attributes即可。同时，针对一些列表类型的DOM Node，引入key来区分不同的列表子项，如果在新的DOM tree的列表中发现与老的DOM tree的列表有相同的列表子项，即使出现的顺序不同，我们仍然保留老的列表子项，仅针对未出现在新的DOM tree中的列表子项进行diff。

在对比了本地和服务器的文档之后，需要将对比后的文档内容进行持久化。在这里使用了三层数据架构，主要的思路如下图：

这里使用了三层数据架构。当用户需要编辑文档需要同步内容时，会通过WebSocket的方式向服务器发起请求，服务器接收到请求时，会首先从内存中的一个名为lastDocContent的NodeJS Object数据结构来获取文档的内容，如果没有，则在Redis缓存中获取对应的文档内容，如果Reids缓存中也没有，则向MySQL数据库中查询文档的内容。通过三层数据架构设计，能够很好避免I/O性能的瓶颈，最上层的内存Object数据结构，容量最小，访问速度最快，最下层的Mysql容量最大，访问速

度最慢，这样一层一层的缓解数据库的压力。

同时，在数据一致性方面，我们在设计lastDocContent这个Object数据结构时，以文档的ID为key，value为文档的内容，操作的次数，上一次操作时间，当操作达到一定的次数时，会将该部分文档的内容同步到Redis中，并且，隔一段时间我们会遍历这个数据结构，如果某个文档的上次操作时间达到一定的时长后，也会同步至Redis中；同理，在Reids中也会维持同样的数据结构用于同步到MySQL中。通过上述的同步机制，保存的文档内容的一致性。

使用内存的好处就是速度快，性能好。但是，内存容量是有限的，如果不能及时释放，那么web服务就无法接收其他的请求。因此，从稳定性上考虑，如果一个文档长时间没有被访问，那么lastDocContent和Redis中该文档对应的数据字段将会被清理，以便其他文档内容被缓存。

2.12 使用HTTPS提升安全性能

应用安全是一个不可忽视的话题，在当前复杂的网络环境下，HTTP劫持的问题略见不鲜，对用户的信息安全发起了严峻的挑战。常规的HTTP协议没有提供对数据包进行加密，网络包在有线和无线的网络环境中都可能被窃听，未加密的数据包可以直接通过抓包工具抓取查看。为了提升应用的安全，引入HTTPS，HTTPS是在HTTP协议上引入TLS或SSL协议对连接进行加密。在加密方面，通常分为对称加密和非对称加密，启用HTTPS协议时，首先会使用非对称加密建立与客户端的连接，在非对称加密连接的基础上传送对称加密的公钥，用于建立对称加密，然后在该对称加密的连接上进行数据的传送。启用这种机制的主要原因是：1，非对称加密的加解密过程相较于对称加密更加消耗计算资源 2，对称机密的公钥分发无法确保绝对安全通常，启用HTTPS需要web服务器的支持，因为选择了Nginx作为代理服务器，所以能够对HTTPS提供较好的支持。通常启用HTTPS还需要使用HTTPS证书。因此，为了满足上述条件，我在Let`s Encr-ypt 上面申请了HTTPS证书，然后在Nginx中配置HTTPS连接相关的信息。

2.13 利用Promise解决回调嵌套和可信任问题

NodeJS代码编写过程中，无可避免的会需要写回调函数，尤其是在事件处理上面。JavaScript的事件循环机制为我们提供了简洁的处理事件方法，但异步回调的嵌套在代码的书写，调试，维护上面带来了额外的负担。当多个异步处理需要按照一定的顺序执行的时候，我们只能采用回调嵌套的方式，但是如果有许多的异步处理，那么这个嵌套的层数将非常的深，如果代码运行时出现了错误，查找错误将变得非常的困难，所以顺序的问题亟待解决。除此之外，当我们需要调用第三方API，使用方式是编写回调函数供第三方API作为参数来进行调用。这里就会暴露出可信任的问题，因为第三方API的具体执行情况我们并不清楚。Promise[25]很早就被社区创建并使用，主要用于解决可信任性和顺序性。Promise对象的三个状态不可逆，并且通过一系列对promise对象的限制来解决可信任的问题。其次，ES6中原生支持Promise，其能将类thenable对象转换成Promise对象进行调用，还提供了Promise.all和Promise.race等新的API来满足不同场景下的需要。通过这些机制来确保可信任的问题。另外，当一个Promise被决议之后，其then方法中会自动返回一个新的Promise对象，而且在then之后的处理函数中返回的值将作为下一个对象的决议值，通过这种方式来实现链式调用，用于解决异步回调嵌套的问题。具体的在应用，当前端向后端发起Ajax请求时，得到的返回对象就是一个Promise对象，通过调用该对象的。随着JavaScript原因的发展，ES6中引入了generator，可以通过在gerator中yield promise对象的方式来实现顺序性编程；更进一步的，ES7发布了新的特性Async/Await,能够实现更加优雅的解决异步回调的问题，可以使用同步的方式来编写异步代码

2.14 使用Vuex管理前端状态

前端状态管理是一件非常有挑战的事情，因为通常将异步跟状态混杂在一起，无法明确的区分开。在前端状态管理工具出现之前，通常的做法是，当父组件需要向子组件传递消息时，通过为子组件设置属性的方式实现。当子组件需要为父组件传递消息时，通过在父组件中定义一个事件处理函数，然后将该事件处理函数作为子组件的属性传递该子组件，在子组件中执行特点的操作时，可以触发这个事件处理函数，完成子组件向父组件传递消息的功能。以上的两种方式是在父子组件之间传递消息，但是如果两个组件需要传递消息，但是他们并没有确定的父子关系时，这个时候就需要另外考虑。我们可以通过向上追溯这两个组件，当发现共同的祖先组件时，在这个组件中传递消息，但是这样就会将整个应用的结构变得十分的混乱。因此，为管理复杂的前端状态，我们使用Vuex[27]这个前端状态管理工具。Vuex的理念来自于redux，redux提出了三个原则：单一状态树，状态是只读的，使用纯函数(reducer)等原则。具体的，在我们的应用中，我们将redux需要使用到的state，mutation，action分别编写在独立的JavaScript文件中，然后在一个单独的store文件中导入这三个文件中的变量，然后组装完成后就形成了全局单一的store并导出。这个store就是整个应用的状态仓库，我们在根组件注入这个store对象，然后在子组件中需要的地方即可自动拿到这个store对象。在使用store对象的时候，可以在Vue组件的methods对象中定义一个方法，当用户触发了某种操作时，可以执行这个方法，在该方法中，可以通过store.dipatch()方法来dispatch一个action，action中可以commit mutation，mutation中可以返回新的状态。反过来，在Vue组件中，定义一个数据，在UI视图中会需要这个数据，这个数据由store的state来提供，前面触发返回新的状态，将会自动修改视图，完成一个状态循环。

2.15 使用Vue-router管理前端路由

在SPA十分流行的今天，许多的应用设定成单页面应用。单页面应用的特点是，一个网站或者应用，所有的用户体验集中在一个页面中。当用户进入第一次进入网站时，页面的组件立即加载完成，当需要执行其他操作时，不需要再不加载组件，而是获取用户数据组件中，并进行展示。因为在最初加载时就加载了所有的页面，因此页面的逻辑就变得十分的复杂，所有的组件融合在一次，管理前端路由就需要解决。常规的前端路由管理方式包括Hash和History API的方式。这两种方式，通过阻止

默认事件或者其他的方式，都可以让浏览器页面不刷新。在我们的网站中，由于使用了Vue，因此借助开源的vue-router[28]来构建单页面逻辑，我们可以定义多个组件，然后在页面中设定多个链接，当用户点击这些链接时，会展示对应的组件，当然，点击这些链接时不会发生跳转的，只是单纯的改变页面的URL和组件的展示。

2.16 用户数据验证和校验

对于一个类似包含大量表单的应用，需要用户输入数据然后校验并提交到后端处理。由于Vue在处理表单数据时，具有双向数据绑定，我们只需要用“v-model”将一个data中的属性中绑定到一个表单元素中，当用户输入数据时，可以自动更新data中的属性，我们可以拿到这个数据，然后进行校验。在这里，为了简单，我们针对不同的数据类型编写了对应的正则表达式匹配组件，当用户点击提交按钮时，如果用户输入的数据与预期的数据类型不合理，将会阻止用户提交，并给出相应的提示。

2.17 单元测试

一个项目，如果要想保持项目代码的可靠性和质量，编写单元测试是最好方案。通常在一个项目代码编写完成后，就需要进行到测试阶段。测试阶段主要对程序进行功能性的验证，可能不断模拟用户操作，也会验证各种数据边界的情况。这里我们选用了karma[29]+jasmine[30]作为单元测试工具，因为整个项目的前后端部分都是用JavaScript语言编写的，因此使用JavaScript单元测试框架来进行前后端的单元测试就是顺理成章了。Karma提供了很多好用的特性，包括模拟在不同设备上面运行的效果；通过命令行或者IDE的方式提供对远程调试的支持；使用jasmin对大型项目框架进行测试等。其使用方式大致如下：

图2-5 karma的使用

而jasmine是一个行为驱动的JavaScript的开发测试框架，其使用的方法非常的语义化，只需要定义一个单元的功能描述，然后在该描述下进行功能的验证。其使用方式如下，非常的语义化：

图2-6 jasmine的使用

具体的，在我们的项目中，主要使用karma+jasmine对后端的路由文件进行单元测试，以及对前端的Vue组件和一些公共的方法进行单元测试。

2.18 提升安全性

一般在开发中，安全威胁很多，其中就包括SQL注入，XSS，CSRF等。SQL注入攻击比较常见，通常产生这种安全威胁的直接原因就是拼接SQL语句。一般的，当用户在前端提交数据时，后端可以通过各种方式拿到这个提交的数据，例如用户名和密码等，如果不经任何处理，直接将这一部分提交的数据用于拼接SQL语句，如果直接执行，那么极有可能发生SQL注入。常规的应对SQL注入，通常使用占位符的形式，在SQL语句中需要插入用户数据的地方使用“？”作为占位符，然后在将用户输入的数据传入到语句中。在我们的语句中，由于使用了Squelize作为数据库操作框架，其会自动针对用户输入的数据进行编译，避免SQL注入的危害。

XSS也称为跨站脚本攻击，常见的有三类：DOM型，反射型，存储型。DOM型和反射型比较类似，用户输入的数据不经判断和过滤直接展示或者反映在当前页面中。存储型是用户输入的非法数据经由后端存储后，再次返回到前端来用于展示。具体的，针对用户的输入，我们编写了一个公共组件，在这个公共组件中，用户输入的内容如果包含有不合法的字符，将会被编码转义。

CSRF被称为跨站请求伪造。触发的条件是，当用户登陆成功了一个网站后，这个网站会生成并保存cookie，在不退出这个网站的前提下，访问其他非法网站，非法网站诱导用户执行一些的操作，这些操作会对登陆成功的网站发起一定的请求，这个时候因为用户还没有退出登陆成功的网站，该网站会以为这是用户自己发出的请求。常见的预防手段有很多，通常可以通过referrer来判断请求的来源，对于非法来源的请求拒绝处理；也可以通过验证码的形式强制用户进行验证，但是这样会降低用户体验；比较好的做法是使用随机token的方式，当用户每次提交表单的时候，会随着用户的请求附带一个token参数，并且这个token也会被保存到cookie中，用户提交表单时，会对表单中的token和cookie中的token验证是否相同，如果不同或没有则默认为非法，拒绝处理请求。

2.19 服务器端性能优化

为了提高应用性能，首先需要优化服务器端的性能。首先，考虑到租用的云服务器是1核CPU，1G内存，所以资源变得非常宝贵，为了提升性能。我准备从I/0，内存和CPU三个方面进行优化。

I/0通常是应用性能的瓶颈，因为硬盘和内存访问速度的差异，执行一次硬盘访问要比内存消耗更多的时间。如果一个应用中，需要频繁的执行I/0操作，那么这将会对数据库访问带来巨大的负担。针对该问题，我在服务器端引入了redis内存数据库作为缓存使用。通常提升I/0的性能，常规中的做法就是引入中间层，包括但不限于HBase，kafka，redis等方案。目前，一些公司会引入redis作为分布式集群，来处理海量的数据，也有很多针对redis主从消息同步，稳定性方面的探索。在我们的应用中，引入服务器资源有限，所以引入了单实例的redis，作为消息和文档的缓存使用。

在优化了I/O之后，下一步就需要优化内存的使用了。由于后端系统使用了Node作为开发语言，而Node相对于Java和PHP来说，性能分析和监控工具还不太完善，并且，犹如Node的语言特性，比较容易发生内存泄露，所以这一块需要更多的探索和实践。我对应用内存的使用做了一些排查，发现了如下几个问题，闭包的使用，缓存的使用。闭包的使用，因为Node基于JavaScript的语言特性，其会有大量的闭包存在，例如定时器，事件处理函数等，因此这一块的内存GC不会自动释放，因此，排查完相关的闭包后，对一些闭包在运行完后进行手动的释放，包括清除定时器，解绑事件处理函数

解决了闭包的问题，接下来就是缓存的使用，因为使用redis来保存了用户发送的消息，以及对文档的处理。如果包含了大量的用户消息以及对文档的访问操作，那么内存将会很快被占满。因此，针对这种情况，我们做出了一些改变。针对消息系统，如果用户发送的消息已经被推送给其他用户了，我们会立即将已经推送给用户的消息同步到MySQL数据库中，然后在内存中删除这一部分内容。如果一个用户的消息长期没有被得到推送，我们也会将这一部分内容率先同步到MySQL数据库中，然后在内存中删除。待下一次用户登录系统时，会针对MySQL数据库做一次查询，将之前同步到数据库中，但尚未被推送的消息推送给用户。这里需要对一个未经推送的消息是否同步到数据库中进行权衡，如果频繁的进行同步，而不是在消息积攒到一定量之后进行同步，那么I/O的瓶颈仍然无法得到解决。因此，我们这里设定了当用户的消息达到10条，并且在十分钟内无法得到推送后，做一次数据库的消息内部。

接下来是CPU性能的优化。CPU的使用通常客户端并发数量太大，伴随着垃圾回收频率过高，执行计算密集型任务等问题。这里首先解决并发数量的问题，因为在消息推送和心跳系统上会使用到WebSocket，为了提升性能，我们会在客户端针对同一用户进行socket的复用。具体的，在心跳系统，服务器消息推送，文档编辑三个功能上会复用同一个socket连接，通过在Vue的顶层组件中初始化一个socket实例，然后将其注入到组件树中，这样在需要使用到socket的子组件中自然就可以拿到这个实例，进行复用。并且，当用户退出系统时，我们会立即释放这个socket连接，避免闲置的资源被占用。然后是避免计算密集型任务长时间执行占用主线程，因为Node是单线程的，如果CPU长时间执行计算任务，那么后续的异步I/O将无法进行。我们可以将较长时间的计算任务划分成几段任务并行执行，同时还可以利用C/C++编写的原生模块来提高性能。为了分析应用的执行效率，内存使用情况等，查看V8引擎的日志。

2.20 前端性能优化

前端性能优化是一个老生常谈的话题，比较久远的是雅虎的“三十五条军规”。这些规则虽然比较久远，不一定适宜今天的开发环境，但是对我们优化性能有一定的启发作用。其中，例如需要合并请求这一块，如今HTTP2已经相对成熟，其连接复用的特性使得合并请求不是那么的必要了。在这里仅仅针对我在前端上面做的一些优化进行阐述。首先，将所有的静态资源压缩丑化，针对CSS文件和JavaScript文件，可以通过使用插件来去掉文件中的空格，注释，将所有的代码压缩成一行，替换其中的变量名等来缩小文件的大小，以达到减小文件的加载时间，提升页面性能的目的。然后将静态资源和动态资源分开，这里的设计思路是，浏览器针对同一个域名能发起的连接是有限的，因此，我们将一些静态文件放在单独的静态域名或者静态文件服务器下，提升连接的使用率。将一些小的图片使用sprit图合并到一起。其次，充分利用好缓存，将页面一些不需要立即展示的内容延迟加载。合并一些对后台接口的请求。将部分的内容从客户端渲染改用服务端渲染，这里主要考虑的是，由于使用了打包工具，页面中的大量的文件被打包到一些目标文件中，这其中包括框架，插件的代码，所以单个文件的体积将会非常的大，当浏览器加载执行这些大体积的代码时，会造成加载阻塞，在较长时间内，页面都无法响应用户的操作。因此这里可以将一部分的内容转移到服务器端来进行渲染，恰好Vue2提供了服务器的渲染的功能。Vue服务器端渲染的设计理念是，在服务端初始化一个Vue的实例，将其执行的结果插入到页面中，这样当浏览器加载页面时，其拿到的页面可以立即显示执行过后的内容，无需进行长时间执行JavaScript，可以大幅提高用户首次进入到网站时页面的加载速度，并且，因为是执行过后的内容，也提高了SEO。同时VUE还支持流式渲染，在服务器端通过Node流读取Vue实例，然后一点点的返回到客户端，进一步提高页面首次可见的加载速度。

2.21 主要功能设计

登陆页提供常规的用户注册，登陆功能，用户密码找回功能。着这里将三个功能几种到一起，通过HTML 的hashbang的方式，当用户点击不同的功能时，展示不同的模块，并且页面不发生跳转。

用户注册功能，为了能让用户提供详细的信息，这里需要用户注册的时候填写用户名，密码，邮箱，电话号，并且提供头像。用户名作为用户的身份识别，是唯一的，用于区分不同的用户；邮箱和电话号用于用户之间相互联系以及找回用户密码；用户头像可以作为个人的个人信息的一部分。

找回密码功能，在实际使用中，用户可能遗忘个人的密码。这种现象是十分常见的，在前期的用户调查时，受访的大部分用户都会遇到这样的情况。为了保证安全性和易用性，我们设计了用户密码找回功能。首先，用户的密码会通过sha256加密保存在数据库中，所以当用户丢失密码之后，无法直接找回原来的密码，因此，这里我们使用了重置密码的方式。当用户需要找回密码时，用户需要提供手机号或者邮箱，然后后台在数据库里检索对应的手机号或者邮箱是否存在，然后使用第三方的发

送接口向用户的手机或者邮箱发送验证码，当用户提供的验证码通过审核后，用户将可以重置自己的用户密码。

登陆功能，应用在设计之初，就需要用户登陆系统。这样设计的初衷是，企业内部交流需要真实的个人信息，在分组以及文档的编辑时，需要验证某个用户是否在分组内，以及是否具有编辑文档的权限。具体的，登陆时，用户可以使用自己的用户名和个人密码进行登陆。

添加好友功，为了方便的不同的用户进行交流，用户可以使用添加好友的功能。首先，用户可以在搜索页搜索对应的用户，搜索成功之后，可以向选定的好友发送添加好友的邀请。邀请信息将以服务器推送消息的形式向指定的用户发送，待邀请同意后，两个用户正式成为好友。

发送消息功能，为了方便企业的用户进行交流和沟通，用户成为好友之后，可以互相发送消息。在好友列表，选中指定的好友，点击发送消息按钮，即可进入消息发送界面。消息发送界面主要展示当前发送消息的好友名称和头像，还包括消息列表。在对应的消息编辑界面编辑消息完成后，就可以点击发送按钮发送消息。消息发送完成后，会将发送成功的消息展示在消息列表中。并且，发送成功的消息先会被保存在Redis的缓存数据库中，待消息达到一定量之后，这些消息通过消息推送系统发送给客户端，客户端接受消息成功并展示，然后向服务器端发送消息说明接受成功，服务器端将会把发送成功的消息同步至MySQL数据库中。当用户需要查看历史消息时，这些消息用户被读取出来。

创建和添加分组功能，在实际的企业沟通中，分组功能是十分重要的。具体的应用场景是，当创建某个项目时，这个项目的成员需要共享和功能编辑文档。因此，就需要成立一个分组。具体的，一个用户可以创建一个分组，这个分组需要设定分组名称和头像。其他的用户可以在分组搜索页面搜索这个分组，搜索到后，就可以点击加入分组的功能，这个时候用户就自动加入到这个分组中，而无需审核。在分组详情页，用户可以看到对应的分组的成员以及属于分组的文档。

文档创建和编辑功能，为了便于用户协作办公，就引入了文档创建和编辑的功能。这里设计思路是，文档属于分组共有的，当一个用户在某个分组中创建了一个文档，这个文档就对这个分组的所有用户可见，这个分组的所有成员都有编辑文档的权限。为了提供优秀的编辑文档的体验，这里引入了开源的富文本编辑插件。文档支持富文本编辑功能，除了提供丰富的文字编辑功能之外，还包括图片上传与编辑，地理位置插入，文档分享等功能，十分强大。并且，当用户点开一个文档进行编辑时，保存机制设计了两种方式，一种是每隔一段时间通知服务器进行文档内容的同步，并且也提供了手动同步的按钮供用户点击进行手动同步文档的内容。

2.22 性能测试

在进行前端性能测试时，我们使用了Chrome自带的dev tools工具。在不适用缓存和使用缓存的情况下得到了如下结果：

可以看出在禁用缓存时，DOMContentLoaded时间为755ms，在不禁用缓存时，DOMContentLoaded时间为694ms。

同时我们使用wetest对页面负载进行压力测试，测试结果如下：

从测试结果来看，其平均TPS约为207.8/s ,在1G 1核的服务器环境下，这种性能表现比较满意。

## 6. 第3章总结

总字数：1538

相似文献列表　文字复制比：0%(0)　疑似剽窃观点：(0)

原文内容

第3章总结

3.1 设计与开发过程思考

在项目设计阶段，收集了功能类似的相关产品资料，例如Microsoft Office和Google Docs，研究了他们流行的原因，并确定了要完成的项目的主要功能特点。

在项目开发之前，对比了不同技术栈的的特点，其中包括后端编程语言选型，前端开发框架选型，服务器类型的选型等。通过查阅相关资料，技术文档等，深入研究了不同类型后端编程语言，数据库，前端开发框架的优缺点，然后分析开发项目的特点，针对该种特点，选择使用了MySQL + Redis + Node + Vue的技术栈。因为上述的技术选型，也会后面的开发带来了快捷，稳定的开发体验。

在项目的开发过程中，首先完成了数据库表相关的设计工作，然后使用Express搭建了后端开发框架。在此基础上，使用Sequelize编写了MySQL数据库操作相关的API，以及Redis数据库操作的API，然后基于WebSocket技术，使用Socket.io搭建了心跳系统以及服务器端推送系统。完成数据库操作API接口和Socket.io系统的编写后，就开始了express的路由系统相关接口的设计和实现。以上，后端系统大致结构开发完成，然后就开始搭建前端开发系统。这里我们使用了vue-cli脚手架来创建主目录，在此基础上，引入了vuex进行前端状态管理，vue-router进行路由管理，然后编写页面的各个vue组件实例，在编写开发的过程中，与后端的数据接口进行调试，确保功能正常。其中，在处理多人在线编辑的功能时，由于前期技术选型铺垫得比较好，也避免了一些常见的问题，例如死锁和同步等。同时也在多人在线编辑文档同步的时候，根据以前学过的react而引入了diff算法来提高效率，为了缓解数据库I/0的压力，创新性的引入了三层数据架构等。

开发完成后，需要将项目部署到服务器上。在这里我们使用了centos的服务器。然后进行了一些配置工作，首先安装MySQL和Redis，然后安装NodeJS相关运行环境，这之后就能正常启动Node的服务了，然后安装Nginx，配置端口转发相关

功能，最后在Nginx配置文件中配置HTTPS相关内容。

3.2 反思与总结

目前已完成协作办公平台的主要功能的设计开发工作。主要包括：用户登陆，注册，找回密码；分组创建，加入；添加好友，消息发送；文档创建与编辑等功能。基本完成相关点的开发工作。

通过这次开发，个人能力得到了很大的提升。首先从产品角度，在设计相关产品之前，首先需要分析类似产品，目的是确定好要开发的项目应该具有怎样的特点。因为，一个好的办公平台，应该具有良好的用户体验，在实际的使用中，该平台是否符合操作习惯都是需要考虑的。所以在设计该平台的时候，我会更多的站在用户的角度来思考这个平台的体验。其次，从技术的角度来看，为了完成上述的各个功能，需要掌握各种技能，从数据库的设计，后端平台的开发，前端界面的开发等等，都需要独立完成，比较考验个人的编程能力。在此过程中，我也学会了通过查阅资料，快速学习和掌握一门新的编程语言，新的开发框架；同时，也需要从性能，安全性的角度去考虑，解决缓存，跨域，安全验证，服务器推送的相关功能。当然，由于时间有限，该平台也会存在着一些缺陷，后续将会持续改进。

参考文献

[1]. Microsoft Office详细资料
[2]. Google Docs详细资料
[3]. Java编程语言
[4]. PHP编程语言
[5]. Python编程语言
[6]. NodeJS-JavaScripot服务器端运行时
[7]. NPM-Node包管理工具
[8]. Yarn 快速，可靠，安全的依赖管理
[9]. Express 基于NodeJS平台，快速，开放，极简的web开发框架
[10]. Koa 基于NodeJS的下一代web开发框架
[11]. Jquery-wirte less,do more
[12]. AngularJS-Superheroic JavaScript MVW Framework
[13]. React–A JavaScript library for building user interfaces
[14]. Vue–渐进式JavaScript框架
[15]. Redis is an open source ,in-memory data structure store
[16]. Sequelize is a promise-based ORM for NodeJS
[17]. Socket.io enables real-time bidirectional event-based communication
[18]. FormData
[19]. Multer- Node.js middleware for handling `multipart/form-data`.
[20]. Eslint-Pluggable JavaScript linter
[21]. Webpack – wepback is a module bundler
[22]. ES6
[23]. PM2 advanced,production process manager for NodeJS
[24]. Diff
[25]. Promise- The Promise object represents the eventual completion .
[26]. Vuex
[27]. Vue-router
[28]. Jasmine – behavior-driven development framework for JavaScript code
[29]. Karma-brings a productive testing environment to developers

致谢

在整个项目的设计开发过程中，非常感谢我的指导老师冯铁老师。对于项目中一些不确定的技术点，我都会向老师咨询，冯老师总能给出一些比较有帮助的建议。由于某些技术没有详细的技术文档和参考资料，冯老师都会力所能及的用他丰富的经验指导我。同时，在论文的撰写过程中，论文中用词，语言组织等方面，冯老师也给我了很多的改进意见，再次感谢冯铁老师。