**北京邮电大学**

**本科毕业设计（论文）开题报告**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 学院 | | 信息与通信工程学院 | 专业 | | 信息工程 | | 班级 | 2014211128 |
| 学生姓名 | | 韩鹏昊 | 学号 | | 2014210743 | | 班内序号 | 03 |
| 指导教师姓名 | | 鄂海红 | 所在单位 | | 计算机学院 | | 职称 | 副教授 |
| 设计（论文）题目 | | （中文）基于Node.js的API网关组件设计与开发 | | | | | | |
| （英文）Design and Development of API Gateway Component Based on Node.js | | | | | | |
| 1. **选题的背景和意义**   随着软件应用领域的近些年的发展，公司项目的新需求不断增加，企业更新和修复大型整体式应用变得越来越困难。假如我们要开发一款全新的应用可能会使用Rails，Play，或者Maven来生成。应用的核心是商业逻辑，它由定义服务，域对象和事件各模块来完成，各个适配器围绕核心与外部交互。开发之后的应用可能会使用例如Maven来打包成一个单体的jar或者war文件，进而部署在Tomcat或者Jetty服务器上。这样风格的应用十分普遍，而这样的部署方式就会面临许多问题：如果只更改了项目的一小部分代码进行上线，则需要重新打包整个项目再进行部署，这无疑增加了部署的复杂程度；再之，如果存在多个开发团队，每个团队都会增加或修改一部分代码，久而久之项目就会显得臃肿复杂难易阅读。  许多公司诸如Amazon，eBay都使用了微服务架构范式来解决了上述的问题，微服务架构将应用拆分为许多小且互相关联的服务。一个微服务完成某个特定的任务，即专注做好一件事。每个微服务被独立的部署在一个云虚拟机或者Docker容器中，这种运行和部署方式使项目更方便于应对变化和快速交付。这也说明了当前和以后的项目部署趋势必将是零散化的服务，因为这类的服务便于维护和开发，适合于更庞大的项目的开发和部署。  近些年，微服务受业界关注的热度持续攀升（见图1）。这两张图首先体现了微服务概念受到的关注度正在持续稳步上升；其次，从一定程度上说明微服务架构的理念也正在为社会更为广泛的接受；最后也说明了微服务架构必将成为企业应用架构的大势所趋。    图1 “microservices”一词在Google Trend中的搜索热度  本课题拟使用的Node.js是运行在服务器端的JavaScript，是基于Chrome V8引擎建立的一个平台，而V8引擎执行JavaScript的优势就是速度快，效率高。在此之前的大部分项目很多都是使用Java Servlet将数据传输到jsp页面，或者使用PHP界面进行表单处理和数据库操作等等。现在有了Node.js这个很好的中间件后，就更方便我们在前端展示数据之前，进行额外的操作譬如数据分发预处理，路由服务。  综上所述，将Node.js的使用和API网关的设计结合起来即为本次毕业设计的主要方向，也说明了基于Node.js的API网关的设计和开发具有重要的创新和实用价值，是一个具有重大研究价值的课题。   1. **研究的基本内容和拟解决的主要问题**   **2.1 研究的基本内容**  由于本研究课题主要是在现有的一些其他框架的基础上进行学习，并在此基础上进行基于Node.js的API网关的设计和开发，所以研究内容流程大致如下：  第一部分，在现有的成熟的框架下研究微服务架构的实现，主要是依赖于Spring Cloud和Spring Boot的微服务。Spring Boot是一个内嵌了Web服务器(tomcat或jetty)的可执行程序的框架，我们开发的应用不需要作为war包部署到web服务器中，而是一个可执行程序，启动时把Web服务器配置好，加载起来即可。这也就说明了Spring Boot方便重启和更新，比较适合微服务的部署。  Spring Cloud是一套微服务开发和治理框架，提供了快速构建分布式系统中一些常见模式的工具(例如配置管理，服务发现，智能路由，微代理，控制总线)。分布式系统的协调导致了样板模式，这也就意味着使用Spring Cloud可以快速支持实现这些模式的服务和应用程序。Spring Cloud的主要成员有以下 Spring Cloud Netflix（与各种Netflix OSS组件集成，组成微服务的核心）、 Spring Cloud Config（配置中心，配置管理工具包）、 Spring Cloud Bus（事件、消息总线，用于在集群中传播状态变化）、 Spring Cloud Cluster（提供在分布式系统中的集群所需要的基础功能支持，如：选举、集群的状态一致性、全局锁、tokens等常见状态模式的抽象和实现）、 Spring Cloud Consul（支持健康检查,并允许 HTTP 和 DNS 协议调用 API 存储键值对）。  第二部分，在理解Spring Cloud的基础上，研究如何使用Node.js进行API网关的开发，基础方面的知识是JavaScript语言的编程习惯和Node.js的服务器端的开发。其中偏重要的部分是Express.js的使用，它提供了丰富的HTTP快捷方法和任意排列组合的Connect中间件，而且Express不对Node.js已有的特性进行二次抽象，只是在Node.js基础上扩展了Web应用所需的基本功能。  Node.js作为一个后端开发的语言优势也是十分明显的：它采用事件驱动、异步编程，为网络服务而设计，其中JavaScript的匿名函数和闭包特性就十分适合事件驱动、异步编程；Node.js非阻塞模式的IO处理给Node.js带来在相对低系统资源耗用下的高性能与出众的负载能力，非常适合用作依赖其它IO资源的中间层服务；Node.js轻量高效，可以认为是数据密集型分布式部署环境下的实时应用系统的完美解决方案。Node.js在服务端工作的结构如图2所示。    图2 Node.js工作的结构  **2.2 拟解决的主要问题**  针对本课题研究的主要内容，我在设计开发的过程中主要解决以下几个问题：  其一，在学习并掌握Node.js的基础上，研究并实现：  ·不同的URL路径可以访问到不同的服务上，数据展示在相应的界面中；  ·基于Cookies和Session的用户身份认证，体现在由API实现的注册和登录上；  ·如何将需要的数据使用Html, CSS, JavaScript展示在前端界面。  其二，学习掌握Consul的配置和使用，研究如何将服务注册发布，实现客户端的成功调用。   1. **研究方法及措施**   **3.1 Node.js实现API网关部分功能**  在理解Node.js、API网关和微服务的基础上进行微服务的开发，主要分为三个方面：一是路由指定的服务；二是实现不同方式的验证登录。  **1、路由服务：**  路由是指如何定义应用的端点（URIs）以及如何响应客户端的请求。  将来自客户端的请求路由到特定的服务，有多种方法均可以实现，但最常用的的还是Express.js的app.METHOD方法，具体介绍如下：  Express.js是一个基于Node.js 的非常优秀的服务端开发框架。Express之所以可以快速地被工程师们接受，并得以广泛应用， 关键就在于其具备的一系列的特性：它可以快速进行开发，拥有灵活的扩展机制，使用简单方便，此外它还有着强大的路由、多模块支持等特性。  Express.js路由是由一个 URI、HTTP 请求（GET、POST等）和若干个句柄组成，它的结构如下： app.METHOD(path, [callback...], callback)， app是express对象的一个实例，METHOD 是一个 HTTP 请求方法，path 是服务器上的路径，callback 是当路由匹配时要执行的函数。  对于这一类逻辑，可以用一个BPMN流程图进行描述。网站服务端的API的逻辑处理如图3所示。    图3 路由服务逻辑  **2、身份验证：**  大部分的微服务基础设施都需要处理认证，将共享逻辑（如身份验证）添加到API网关中可以使服务保持小巧。微服务实现身份认证可以使用Cookies或者Token实现在API网关处的身份验证，从而控制用户可以或不可以使用的某些服务。  关于此类国际上已经有了许多的讨论， David Borsos 在伦敦的微服务大会上提出了四种方案：  ·单点登录（SSO）  这种方案意味着每个面向用户的服务都必须与认证服务交互，这会产生大量非常琐碎的网络流量和重复的工作，当动辄数十个微应用时，这种方案的弊端会更加明显。  ·分布式Session方案  分布式会话方案原理主要是将关于用户认证的信息存储在共享存储中，且通常由用户会话作为 key 来实现的简单分布式哈希映射。当用户访问微服务时，用户数据可以从共享存储中获取。在某些场景下，这种方案很不错，用户登录状态是不透明的。同时也是一个高可用且可扩展的解决方案。这种方案的缺点在于共享存储需要一定保护机制，因此需要通过安全链接来访问，这时解决方案的实现就通常具有相当高的复杂性了。  ·客户端Token方案  令牌在客户端生成，由身份验证服务进行签名，并且必须包含足够的信息，以便可以在所有微服务中建立用户身份。令牌会附加到每个请求上，为微服务提供用户身份验证，这种解决方案的安全性相对较好，但身份验证注销是一个大问题，缓解这种情况的方法可以使用短期令牌和频繁检查认证服务等。对于客户端令牌的编码方案，Borsos 更喜欢使用 JSON Web Tokens（JWT），它足够简单且库支持程度也比较好。  ·客户端Token与API网关结合  这个方案意味着所有请求都通过网关，从而有效地隐藏了微服务。 在请求时，网关将原始用户令牌转换为内部会话 ID 令牌。在这种情况下，注销就不是问题，因为网关可以在注销时撤销用户的令牌。  对比上述4种身份认证方案，都是自身缺点日益明显后从而促使了新的方案的出现，这里拿基于Token的认证为例。随着 Restful API、微服务的兴起，基于 Token 的认证现在已经越来越普遍。Token 和 Session ID 不同，并非只是一个 key。Token 一般会包含用户的相关信息，通过验证 Token 就可以完成身份校验。像 Twitter、微信、QQ、GitHub 等公有服务的 API 都是基于这种方式进行认证的，一些开发框架如 OpenStack、Kubernetes 内部 API 调用也是基于 Token 的认证。  相比于简易的路由到指定的服务，加入共享逻辑后的服务只是在API网关处添加了一个Auth的身份验证。对于使用Token进行身份验证的API网关，其中核心的是请求获取Token和验证Token的正确性，使用Token的身份验证逻辑图如图4所示。      图4 Token的请求和验证  **3.2 整合其他开源组件**  结合Consul这个API网关工具进行开发，该网关已经比较成熟可以直接实现API的管理。在启动之后，我们可以在Consul服务端中进行服务的不限于注册、发现、注销等操作。Consul作为一个服务管理软件，具有以下的特点：   * 支持多数据中心下，分布式高可用的，服务发现和配置共享； * 支持健康检查，允许存储键值对； * 一致性协议采用 Raft 算法,用来保证服务的高可用； * 成员管理和消息广播 采用GOSSIP协议，支持ACL访问控制。   整合Node.js开发的部分功能组件可以减少部分的重复性工作，同时提高开发的效率。   1. **研究工作的步骤与进度** 2. 第一阶段(2017.1.1.-2017.1.30)：掌握Spring Cloud的使用，学习Node.js编程方法，完成其他基础性工具的使用学习。 3. 第二阶段(2017.2.1-2017.2.28)：完成Node.js下的API网关部分功能的设计和开发，同时进行Consul网关的学习和实践，为下一步两者的耦合做进一步准备。 4. 第三阶段(2017.3.1.-2017.4.30)：完成中期报告，集成Node.js网关和已有的其它API网关形成一个网关系统进行最后的部署和验证，并进行所有bug的排查。 5. 第四阶段(2017.5.1-2017.6.1)：完成毕业论文。   **五、主要参考文献**   1. 朴灵. 《深入浅出Node.js》. 人民邮电出版社.2013 2. 黄勇. 《轻量级微服务架构（上册）》. 电子工业出版社. 2016 3. 翟永超. 《Spring Cloud微服务实战》. 电子工业出版社.2017 4. Chris Richardson. 「Chris Richardson 微服务系列」微服务架构的优势与不足. http://blog.daocloud.io/microservices-1/ 5. PetterLiu. 微服务架构设计. http://www.cnblogs.com/wintersun/p/6219259.html 6. KKys. 分布式学习（1）----初识Zookeeper. https://zhuanlan.zhihu.com/p/24996631 7. 温晓丽,苏浩伟,陈欢,邹大毕. 基于SpringBoot微服务架构的城市一卡通手机充值支撑系统研究[J]. 电子产品世界,2017,24(10):59-62. 8. 刘丹. 一种微服务架构最佳实践[A]. 中国电机工程学会电力信息化专业委员会. 2017电力行业信息化年会论文集[C]. 中国电机工程学会电力信息化专业委员会:,2017:5. 9. 王健,李冬睿. 从单一模式系统架构往微服务架构迁移转化技术研究[J]. 科教导刊(下旬),2016(09):43-44. 10. 崔蔚,李春阳,刘迪,杨超,金逸. 面向微服务的统一应用开发平台[J]. 电力信息与通信技术,2016,14(09):12-17. 11. 郑彬彬. 基于微服务的OJ系统重构与优化[D]. 东华大学,2017. 12. 张允君,卫洁,付小攀,彭霄. 特定企业微服务架构落地的研究与实践[J]. 电力信息与通信技术,2017,15(11):77-82. 13. 邓杰文,曹彩凤. 微服务若干关键问题研究[J]. 五邑大学学报(自然科学版),2016,30(02):49-54. 14. 张晶,王琰洁,黄小锋. 一种微服务框架的实现[J]. 计算机系统应用,2017,26(04):82-86. 15. 郭栋,王伟,曾国荪. 一种基于微服务架构的新型云件PaaS平台[J]. 信息网络安全,2015(11):15-20. 16. 张晶,王琰洁,黄小锋. 一种微服务框架的实现[J]. 计算机系统应用,2017,26(04):82-86. 17. 张晶,黄小锋. 一种基于微服务的应用框架[J]. 计算机系统应用,2016,25(09):265-270. 18. M. F. Karagoez and C. Turgut, "Design and Implementation of RESTful Wireless Sensor Network Gateways Using Node.js Framework," European Wireless 2014; 20th European Wireless Conference, Barcelona, Spain, 2014, pp. 1-6. 19. Dickey, J. (2014). Write Modern Web Apps with the MEAN Stack: Mongo, Express, AngularJS, and Node.js. Place of publication not identified: Peachpit Press. | | | | | | | | |
| 指导教师签字 |  | | | 日期 | |  | | |