

硕士学位论文

**基于微服务架构的线上围棋教学系统的设计与实现**

学位申请人：张威

指导教师： 赵银亮 教授

类别（领域）：工程硕士（软件工程）

2018 年 4月

**Design and Implementation of Online Computer Go of Teaching System Based On Microservice Architecture**

A thesis submitted to

Xi’an Jiaotong University

In partial fulfillment of the requirement

for the degree of

Master of Engineering

By

Wei Zhang

Supervisor: Assoc. Prof.Yinliang Zhao

(Software Engineering)

April 2018

**论文题目：基于微服务架构的线上围棋教学系统的设计与实现**

**类别（领域）：工程硕士（软件工程）**

**学位申请人：张威**

**指导教师：赵银亮 教授**

摘 要

围棋起源于中国，是中华民族传统文化的瑰宝，是集科学性、艺术性、竞技性于一体的智力运动，已经成为幼儿、青少年素质教育工作的重要手段。近年来随着网络技术的飞速发展，为我们提供了解决传统围棋教学弊端的方法，因此开拓了一个新的市场。借助线上围棋教学平台进行围棋推广具有重要的理论意义和实用价值。

本文集围棋对弈与教学功能于一体，采用移动客户端和服务端微服务化的设计方式开发了一个方便教师和学生的围棋教学系统。本文以“围棋教学”为主线，分析了线上围棋教学系统的所有业务流程，然后对系统的主要模块以及核心功能进行了详细设计和编码实现。本系统的服务端采用微服务架构模式，由学校管理、聊天管理和围棋引擎三个微服务构成，每个微服务都具有自身独立的技术架构与数据库存储，通过对外暴露服务接口进行模块之间信息与数据的传递与交流。服务端使用Docker容器部署所有的微服务。学校管理包括管理用户信息以及班级信息的功能，聊天管理服务主要包括用户的单聊、查看会话列表和查看历史聊天记录等功能，围棋引擎服务包括提供与客户端在围棋“人-人对弈”时的通信、查看或保存棋谱等功能。本文还提出基于厚势价值的影响模型，在围棋的形势判断上更加符合职业棋手的估值，这样用户可以实时查看对当前局面的局面分析，并动态调整对弈策略来获得围棋对弈的胜利。

经过对系统测试，该教学平台基本功能均已实现，基本满足应用需求。在理论上，本论文基于线上围棋教学所提出的设计方案为围棋教学提供了参考。在实践上，为围棋教学提供了一个方便快捷的平台，并且对围棋的推广进行了新的探索和尝试。

**关 键 词：**线上教学；微服务；围棋对弈；局面分析；

**论文类型：**应用研究

**Title:** **Design and Implementation of Online Computer Go of Teaching System Based On Microservice Architecture**

**Professional Fields: Software Engineering**

**Applicant: Wei Zhang**

**Supervisor: Assoc. Prof. Yinliang Zhao**

ABSTRACT

Go originated in China and is a treasure of traditional Chinese culture. It is an intellectual movement that integrates science, art, and athletics. It has become an important tool for the quality education of young children and adolescents. In recent years, with the rapid development of network technology, we have provided us with a solution to the disadvantages of traditional Go teaching. It is of great theoretical significance and practical value to use online chess teaching platform to carry out Go promotion.

This paper integrates chess and chess teaching functions, and uses a mobile client and server-side microservice design to develop a teaching system for teachers and students. This paper takes "Go teaching" as the main line,and then designs and codes the main modules and core functions of the system in detail. The server of this system adopts the micro service architecture model, which is composed of three micro services: school management, chat management, and Go engine. Each micro service has its own independent technical architecture and database storage, and is used to perform the interface between modules through externally exposed service interfaces. Information and data transmission and exchange. The server uses the Docker container to deploy all the microservices. The school management includes the functions of managing user information and class information. The chat management service mainly includes the user's single chat, viewing the conversation list, and viewing the history chat record. The Go engine service includes providing the client with the game of “human-to-human chess”. Communicate, view or save the game and other functions. This paper also proposes an influence model based on thick value, which is more in line with the value of professional chess players in the situation judgment of Go, so that the user can view the situation analysis of the current situation in real time, and dynamically adjust the game strategy to obtain the victory of Go chess.

After testing the system, the basic functions of the teaching platform have been realized and basically meet the application requirements. In theory, this paper provides a reference for the teaching of Go based on the design proposal proposed by the online Go teaching. In practice, it provides a convenient and quick platform for the teaching of Go.

**KEY WORDS**: Online Lessons; Microservice; Go chess playing; Situation analysis

**TYPE OF THESIS**: Application Research

# 

目 录

[1 绪论 1](#_Toc2626)

[1.1 选题意义及应用背景 1](#_Toc30641)

[1.2 国内外研究现状分析 2](#_Toc18177)

[1.3 论文的主要研究内容 3](#_Toc29353)

[1.4 论文的组织结构 5](#_Toc16792)

[2 相关技术 7](#_Toc17155)

[2.1 围棋概述 7](#_Toc2124)

[2.1.1 相关围棋术语 7](#_Toc24388)

[2.2 Android系统简介 10](#_Toc7061)

[2.2.1 Android系统概述 10](#_Toc16866)

[2.2.2 Android系统架构 10](#_Toc8303)

[2.3 微服务架构 12](#_Toc9951)

[2.3.1 微服务架构设计理念 12](#_Toc10743)

[2.3.2 微服务架构特征 13](#_Toc10083)

[2.3.3 微服务的设计原则 14](#_Toc32055)

[2.4 Docker容器技术 15](#_Toc10981)

[2.4.1 Docker技术简介 15](#_Toc2062)

[2.4.2 Docker实现的关键技术 17](#_Toc8935)

[2.4.3 Docker与微服务设计原则 18](#_Toc19604)

[2.5 本章小结 18](#_Toc27988)

[3 基于微服务架构的线上围棋教学系统分析 19](#_Toc19568)

[3.1 系统需求描述 19](#_Toc23290)

[3.2 系统功能分析模型 19](#_Toc7152)

[3.2.1 系统交互模型 19](#_Toc704)

[3.2.2 系统结构模型 23](#_Toc17133)

[3.2.3 系统行为模型 24](#_Toc3963)

[3.3 系统非功能需求分析 30](#_Toc447)

[3.3.1 移动客户端的兼容性需求 30](#_Toc26010)

[3.3.2 服务端的可扩展性需求 31](#_Toc709)

[3.3.3 服务端的稳定性需求 31](#_Toc9982)

[3.4 本章小结 31](#_Toc1367)

[4 基于微服务架构的线上围棋教学系统设计 32](#_Toc3762)

[4.1 系统概要设计 32](#_Toc13593)

[4.1.1 系统体系结构 32](#_Toc11118)

[4.1.2 系统功能模块结构 33](#_Toc14594)

[4.2 系统主要功能模块设计 34](#_Toc30459)

[4.2.1 系统总体架构 34](#_Toc8096)

[4.2.2 移动客户端 35](#_Toc31223)

[4.2.3 API网关系统 37](#_Toc14930)

[4.2.4 学校管理模块 38](#_Toc26254)

[4.2.5 聊天管理模块 40](#_Toc18331)

[4.2.6 围棋引擎模块 42](#_Toc10786)

[4.2.7 围棋局面分析的影响模型 44](#_Toc3253)

[4.3 基于REST的通信接口设计 53](#_Toc13554)

[4.4 系统的数据库设计 54](#_Toc16651)

[4.4.1 系统的概念数据模型 54](#_Toc28738)

[4.4.2 系统的物理数据模型 55](#_Toc585)

[4.5 本章小结 58](#_Toc24219)

[5 基于微服务架构的线上围棋教学系统实现与测试 59](#_Toc32289)

[5.1 系统开发环境 59](#_Toc32528)

[5.2 系统的实现 59](#_Toc9970)

[5.2.1 服务端微服务部署 59](#_Toc5040)

[5.2.2 API网关的实现 63](#_Toc13046)

[5.2.3 学校管理模块的实现 65](#_Toc11410)

[5.2.4 聊天管理模块的实现 69](#_Toc25001)

[5.2.5 围棋引擎模块的实现 71](#_Toc9234)

[5.2.6 围棋局面分析的实现 75](#_Toc8572)

[5.3 系统的测试 77](#_Toc24129)

[5.3.1 系统测试环境 77](#_Toc16407)

[5.3.2 功能测试 79](#_Toc27801)

[5.3.3 非功能测试 80](#_Toc25013)

[5.4 本章小结 81](#_Toc106)

[6 结论与展望 82](#_Toc8290)

[6.1 结论 82](#_Toc10657)

[6.2 展望 82](#_Toc31266)

[致 谢 84](#_Toc3285)

[参考文献 85](#_Toc11591)

声明

CONTENTS

[1 Preface 1](#_Toc23309)

[1.1 Significance and Background 1](#_Toc7865)

[1.2 Glance of Current Research Status 2](#_Toc31715)

[1.3 Main Research Contents of Thesis 3](#_Toc13194)

[1.4 Organization of Thesis 5](#_Toc2069)

[2 Related Technologies 7](#_Toc7461)

[2.1 Go Overview 7](#_Toc22095)

[2.1.1 Related Go Terms 7](#_Toc28782)

[2.2 Android System Introduction 10](#_Toc1325)

[2.2.1 Related Android System 10](#_Toc16845)

[2.2.2 Android System Architecture 10](#_Toc28356)

[2.3 Microservice Architecture 12](#_Toc1293)

[2.3.1 Design Philosophy of Microservice Architecture 12](#_Toc22932)

[2.3.2 Features of the Microservice Architecture 13](#_Toc2311)

[2.3.3 Design Principles of Microservice 14](#_Toc23048)

[2.4 Docker container technology 15](#_Toc21923)

[2.4.1 Docker Technology Introduction 15](#_Toc12523)

[2.4.2 The key technology that Docker realizes 17](#_Toc9981)

[2.4.3 Docker and Microservice Design Principles 18](#_Toc5748)

[2.5 Chapter Summary 18](#_Toc22201)

[3 Requirement Analysis 19](#_Toc7902)

[3.1 Requirement Description 19](#_Toc19644)

[3.2 System Function Analytical Model 19](#_Toc3247)

[3.2.1 Interactive Model 19](#_Toc6884)

[3.2.2 Structural Model 23](#_Toc17927)

[3.2.3 Behavior Model 24](#_Toc17287)

[3.3 System Non-Functional Requirement Analysis 30](#_Toc3516)

[3.3.1 Compatibility Demand 30](#_Toc26799)

[3.3.2 Scalability Demand 31](#_Toc26127)

[3.3.3 Stability Demand 31](#_Toc21105)

[3.4 Chapter Summary 31](#_Toc29239)

[4 The Design of The System 32](#_Toc27862)

[4.1 High-Level System Design 32](#_Toc5399)

[4.1.1 Software Architecture 32](#_Toc8747)

[4.1.2 Function Module Architect 33](#_Toc16091)

[4.2 The Main Function Module Design 34](#_Toc23733)

[4.2.1 Overall system architecture 34](#_Toc10326)

[4.2.2 Mobile Client Module 35](#_Toc4890)

[4.2.3 API Gateway 37](#_Toc29256)

[4.2.4 School Management Module 38](#_Toc7423)

[4.2.5 Chat Management Module 40](#_Toc4099)

[4.2.6 Go Engine Module 42](#_Toc14764)

[4.2.7 Situation Analysis of Go 44](#_Toc24579)

[4.3 Communication Interface 53](#_Toc16563)

[4.4 Database of System 54](#_Toc10869)

[4.4.1 Conceptual Data Module 54](#_Toc29933)

[4.4.2 Physical Data Module 55](#_Toc15551)

[4.5 Chapter Summary 58](#_Toc29772)

[5 The Implementation and Test of The System 59](#_Toc21129)

[5.1 Development Environment 59](#_Toc21333)

[5.2 Implementation of The System 59](#_Toc16347)

[5.2.1 Server-side microservice deployment 59](#_Toc10553)

[5.2.2 Implementation of API Gateway 63](#_Toc11588)

[5.2.3 Implementation of School Management Module 65](#_Toc14820)

[5.2.4 Implementation of Chat Management Module 69](#_Toc32299)

[5.2.5 Implementation of Go Engine Module 71](#_Toc5516)

[5.2.6 Implementation of Situation Analysis 75](#_Toc31417)

[5.3 System Test 77](#_Toc370)

[5.3.1 System Environment Test 77](#_Toc25083)

[5.3.2 Test Procedure and Conclusion 79](#_Toc23542)

[5.3.3 Test Running Screenshot 80](#_Toc28205)

[5.4 Chapter Summary 81](#_Toc32320)

[6 Conclusions and Suggestions 82](#_Toc25681)

[6.1 Conclusions 82](#_Toc28659)

[6.2 Suggestions 82](#_Toc20858)

[Acknowledgements 84](#_Toc9682)

[References 85](#_Toc27507)

Declarations

# 绪论

## 选题意义及应用背景

围棋是一种智力游戏，起源于中国。围棋是中华民族传统文化中的瑰宝，它体现了中华民族对智慧的追求。由于它将科学、艺术和竞技三者融为一体，有着发展智力，培养意志品质和机动灵活的战略战术思想意识的特点。围棋的规则十分简单，却拥有十分广阔的落子空间，使得围棋变化多端。下围棋对人脑的智力开发很有帮助，可增强一个人的计算能力、创造能力、思维能力、判断能力，也能提高人的注意力和控制力。下围棋对儿童少年的智力发展起到积极作用，使他们能更好的分析事物。

在2016 年 3 月，AlphaGo 挑战韩国职业九段李世石，以 4：1 击败了对手。2017 年 5 月，AlphaGo 在中国乌镇围棋峰会的三局比赛中击败当时世界排名第一的中国棋手柯洁。这些比赛都引起了人们极大的关注，重新引发了人们关于围棋的兴趣，因此推广围棋教育有着重要的意义。

网络技术的飞速发展为我们提供了解决传统围棋教学弊端的办法，也因此开拓了一个全新的市场。国内一些网络平台纷纷开启网络对弈平台。但网络围棋教育仅局限于供求双方在围棋网站上发贴求师或下载些资料学习。而且他们的赢利模式也只是着眼于广告收入或举办竞猜等传统形式，却忽视了网络围棋教育这个大市场。而各地民间围棋培训学校建立的网站，主要以发布教学资讯和招生简章等，并无实质教学功能。围棋线下培训十分火爆，而线上围棋教育却发展缓慢，也引起了很多业内人士的思考。真正的网络课堂不应该是语音或者视频的简单传授，而应该有教学双方的互动等内容。

本文设计并实现了基于移动客户端和服务端微服务化的线上围棋教学平台，具有功能完备、扩展性良好的特点。并且该教学平台摆脱了传统围棋教学的限制，用户可以随时学习围棋技术。旨在以最有效的方式为学员与教师提供企业级平台化的专业在线教育服务，助力于中国围棋竞技水平的整体提升。该线上围棋教学系统有以下好处：

1）提供管理平台来整合学校教学资源，通过对平台各种教学工具的使用，围棋教学系统将获得各种教学资源的积累，可以解决学校资源流失，不易管理的问题，提升学校累计教学价值。便捷管理学校教师学生，以及发布围棋教学信息等。

2）不同于传统的单体式应用架构，该平台的服务端采用微服务化的设计模式，微服务架构中，各个服务松散、自治，强调服务的“独立性”—独立开发、独立部署和独立运行，减少应用开发的复杂性，从而具有灵活性大、实施性易和扩展性好等优点。

3）教师可以创建班级，并管理班级中的学生信息。学生也可以搜索自己感兴趣的班级，并申请加入该班级。教师可以向学生提供自己制作的作业等内容，制定自己的教学计划，并且在课后答疑解惑和批改作业，提升教师的教学质量。

4）该平台提供聊天管理服务，用户之间可以通过移动端的APP随时随地的沟通交流信息。用户也可以随时查看和其他用户之间的历史聊天记录。

5）该平台提供围棋对弈功能，学生可以线上进行围棋对弈，然后可以保存当前棋局、开始新的棋局、导入以前棋局等。该平台预存了经典名家对局棋谱。在围棋对弈时，用户可以查看当前局面的分析，动态调整围棋对弈策略来获得对弈胜利。

## 国内外研究现状分析

国内比较有名的在线教育系统比如中国大学MOOC网，是由网易与高教社联合推出的在线教育平台，向大众提供中国高校的MOOC课程，但是该网站并没有提供有关围棋的教学培训内容。同样国内比较有名的围棋对弈平台比如腾讯游戏平台，是由腾讯推出并且集各种网络对弈游戏与一体，但是该平台提供的围棋对弈平台，并没有提供围棋的教学培训等内容。国内正在开始对围棋教学系统的研究和实现。比较有名的新浪、TOM、弈城等网络平台纷纷开启网络对弈平台。但是这些平台提供的网络围棋教育的形式，还仅仅局限于供求双方在围棋网站上发贴求师、张贴招生，或者下载些免费资料学习而已。而作为平台运营方的新浪、TOM、弈城，他们的赢利模式也只是着眼于广告收入或举办竞猜等传统形式，却忽视了网络围棋教育这个大市场。

国外比较有名的在线教育平台比如Coursera，它与全世界最顶尖的大学和机构合作，提供任何人可学习的免费在线课程。同样该平台并没有和围棋对弈结合起来。这种在线教育平台具有教学资源多元化、课程易于使用、课程受用面广，并且都具有自己的线上评测系统。国外开源的围棋软件GNU GO是免费软件，它的代码和文档都是公开的，用户可以在它的官网上查看相关信息。

Martin Fowler基于“高内聚、低耦合”的原则提出了微服务架构，把软件拆分成微服务[1]。在API Academy微服务峰会上，微服务专家们发表了他们对于微服务现状、相关架构和组织问题、过程和技术的看法。Miitra指出服务边界是微服务架构中最重要的概念，组织设计和文化在成功实施微服务方面扮演着重要的角色，微服务可以看作SOA的演化，并结合了持续部署、敏捷和DevOps的原则，更加有利于软件的开发。

围棋以棋盘上占领多少地域作为判断胜负的标准，通常目前的围棋程序常常根据空目数的多少作为最佳棋步的评判标准，但是有时棋子的领先数目并不能直观的观察。在对围棋当前局面进行分析时，通常采用影响模型来解决这个问题，然后将棋子向棋盘其他部分辐射的影响量化[2][3]。所以计算厚势价值同样非常重要，特别是在进行静态评估，然后量化获胜概率的时候，厚势价值计算的结果是否准确，将会影响获胜概率的最终量化结果。人类棋手在计算厚势价值的时候，通常源自对棋行的感觉，或者对局经验的反馈，这种判断缺少规律性和准确性，所以计算厚势价值通常难度比较高[4][5]。

综上所述，针对本文提出的移动端及时的围棋交互式学习是目前国内外围棋线上教学平台面临的普遍问题。研究解决目前所面临的问题能提高用户的学习体验和积极性，对线上围棋教学的发展具有促进作用。

## 论文的主要研究内容

围棋作为一种极具趣味的益智类游戏，其发展代表着文化的传承和人类智慧的发展。在科技发展迅速的当今，围棋运动依然长盛不衰。但围棋教育发展迟缓，形式单一，目前更多的是围棋爱好者聘请教师然后在线下进行学习。线上围棋教学急需改变当前围棋教育的现状[6]。

目前国内线上围棋教学没有比较好的方式，要么是单独的围棋对弈软件，要么是单独的课程教学。本平台集围棋对弈与教学功能于一体，采用移动客户端和服务端微服务化的设计方式开发了一个方便教师和学生的线上围棋教学平台。目前大多数的围棋对弈软件通常采用单体式应用架构，该软件实现比较简单，但是随着围棋软件功能的逐渐增加，软件规模越来越大，通常到后期就会难以维护。在分析了系统需求之后，本系统主要功能模块分为三个部分：学校管理、聊天管理和围棋引擎。本系统采用C/S架构模式，客户端使用Android平台开发移动客户端，然后服务端采用微服务化的设计模式，结合了微服务的功能特点，每个功能模块设计为一个独立的微服务，并且每个微服务部署在一个独立的Docker容器上。由于如果新增一个微服务，就要重新修改客户端模块，因此本系统采用API Gateway的微服务模式，API Gateway是系统的唯一入口，所有的客户端和服务端都要通过统一的网关接入微服务。

本文主要工作如下：

1）服务端采用微服务化的设计模式，根据功能划分了三个微服务模块，分别为学校管理、聊天管理和围棋引擎。每个微服务都具有自身独立的技术架构与数据库存储，通过对外暴露服务接口进行模块之间信息与数据的传递与交流。并且移动客户端通过API Gateway和服务器进行通信，它是进入后端服务器的唯一节点。所有来自客户端的请求都要先经过API Gateway，然后路由这些请求到对应的微服务。微服务部署在Docker容器上，每一个微服务都部署在一个独立的容器上。

2）学校管理：用户可以通过该模块管理用户信息和班级信息。登录成功后，用户可以维护个人信息。用户角色分为教师和学生，教师可以创建新的班级，然后批准学生的申请加入。学生可以搜索班级信息，并申请加入感兴趣的班级，并等待教师的批准。教师在班级中发布作业信息，班级中的学生会看到发布在该班级的作业信息。

3）聊天管理：由于本系统采用Android平台实现移动客户端，并实现即时聊天功能，在参考了一些市场IM软件之后，决定采用网易云信IM，该功能较为完善并且稳定。本系统移动客户端调用网易云信IM的SDK来实现即时聊天功能，并将聊天信息存储在本系统的聊天管理服务端的数据库中，方便用户下次直接查询聊天记录。学生与教师可以通过移动客户端的即时聊天功能来交流信息。该模块功能包括单聊和查看历史聊天记录等，聊天类型包括文字、表情、图片、语音、视频等。

4）围棋引擎：本系统提供对弈平台，用户可以“人-人对弈”，可供学生之间在课堂之后进行切磋交流。围棋对弈的基本功能如下子、提子和点目等，大部分围棋对弈软件都已实现该功能，在参考了一些对弈软件之后，决定采用开源GNU GO为基础[12]，一些功能都采用该软件的算法。客户端与围棋引擎通信采用目前主流的围棋对弈平台都默认的GTP协议[13]，这样方便以后服务端可以对接新的围棋引擎。本系统保存围棋棋谱采用通用的SGF文件。围棋对弈时，学生可以保存当前棋局、开始新的棋局、加载以前棋局等功能。用户也可以查看系统中预存的一些经典对弈棋谱。

5）围棋局面分析：在对弈时，局面分析为用户提供重要的辅助功能，用户可以查看局面形势分析，这样用户动态调整对弈策略来获取对弈的胜利，提升用户的围棋水平。围棋的局面分析通常采用影响模型，将棋子向棋盘其他部分的辐射影响进行量化。棋子的影响模型是战术决策的基础，在影响模型中，多个棋子的作用被看作是单个棋子的叠加，而单个棋子的作用根据相对位置不同进行一定的修改，黑白棋子产生相反的影响。目前已经提出的Zobrist的影响模型、Ryder的影响模型，陈克训的影响模型和GNU GO的影响模型都有各自的短板和缺陷。本系统提出的基于厚势价值的影响模型，更加符合职业棋手的估值，影响函数性能更高，会提高用户对弈判断的准确性[30][31]。

## 论文的组织结构

论文的组织结构如下

第一章 绪论

主要介绍了本文的选题背景与研究意义、国内外围棋教学的研究现状，并简要说明了本文的主要研究内容和组织结构。

第二章 相关技术的研究

简单描述了本文用到的相关技术，包括围棋概述、Android系统简介、微服务架构、Docker容器技术。

第三章 围棋教学系统分析

介绍了线上围棋教学系统的需求，然后分析了系统所要实现的主要功能和目标，分别详细的分析了系统的功能需求和非功能需求。

第四章 线上围棋教学系统设计

设计了线上围棋教学系统的总体功能结构和总体架构，分别对其主要的功能模块，即移动客户端模块、学校管理模块、聊天管理模块、围棋引擎模块、围棋局面分析的影响模型做了详细的设计。最后对系统的数据库进行了设计。

第五章 线上围棋教学系统实现与测试

根据系统设计的方案，对系统进行了具体实现，对主要功能模块的实现进行了详细介绍。最后对系统的主要功能点和系统性能进行测试，并对测试结果进行分析。

第六章 结论与展望

总结本文的研究工作，提出本文系统的优势，展望系统未来的研究和改进方向。

# 相关技术

在第一章中线上围棋教学系统的研究背景及意义的基础上，本章着重介绍线上围棋教学系统的相关技术，包括围棋概述，Android系统介绍，服务端的微服务开发和Docker容器技术。

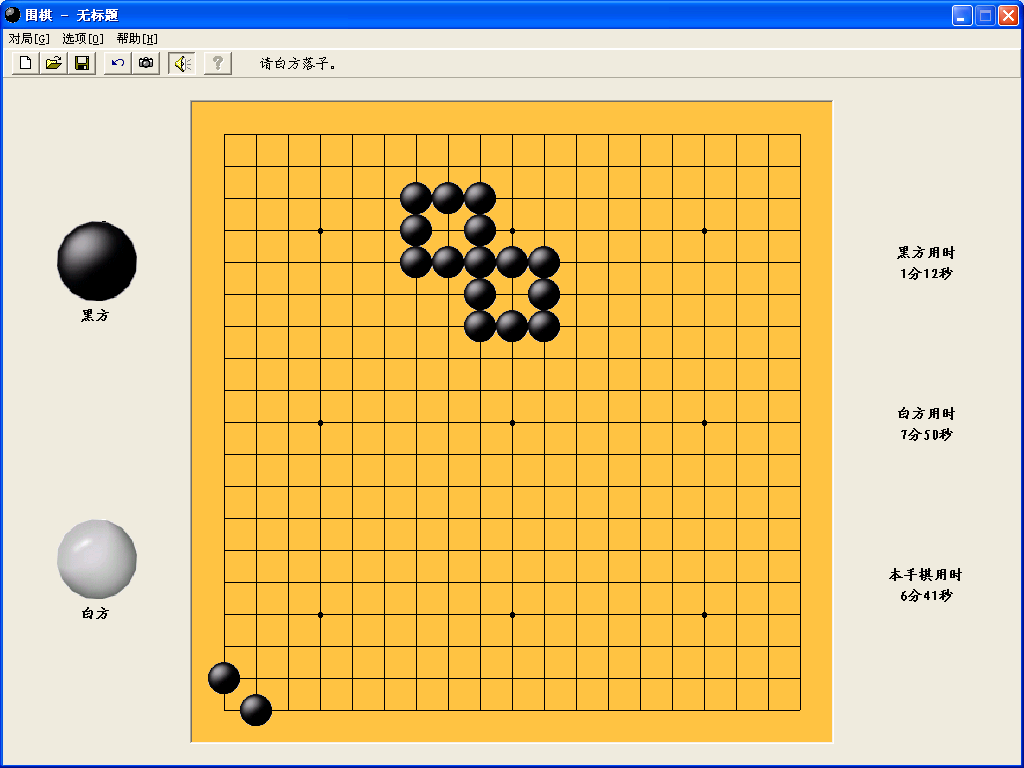
## 围棋概述

围棋，一种策略性两人棋类游戏，中国古时称“弈”，西方名称“Go”。流行于东亚国家（中、日、韩、朝），属琴棋书画四艺之一。围棋起源于中国，传为帝尧所作，春秋战国时期即有记载。隋唐时经朝鲜传入日本，流传到欧美各国。围棋蕴含着中华文化的丰富内涵，它是中国文化与文明的体现[7][8]。

### 相关围棋术语

1）块棋

连接就是相同颜色的棋子在经线或纬线上相邻。连接在一起的棋子组成了一块棋，或者单一的一颗棋子也称为一块棋。块棋的气是组成它的所有棋子的气总和。一般有两个及两个以上真眼的块棋是不能被提掉的，因为对方不可能同时落下两颗棋子填眼。



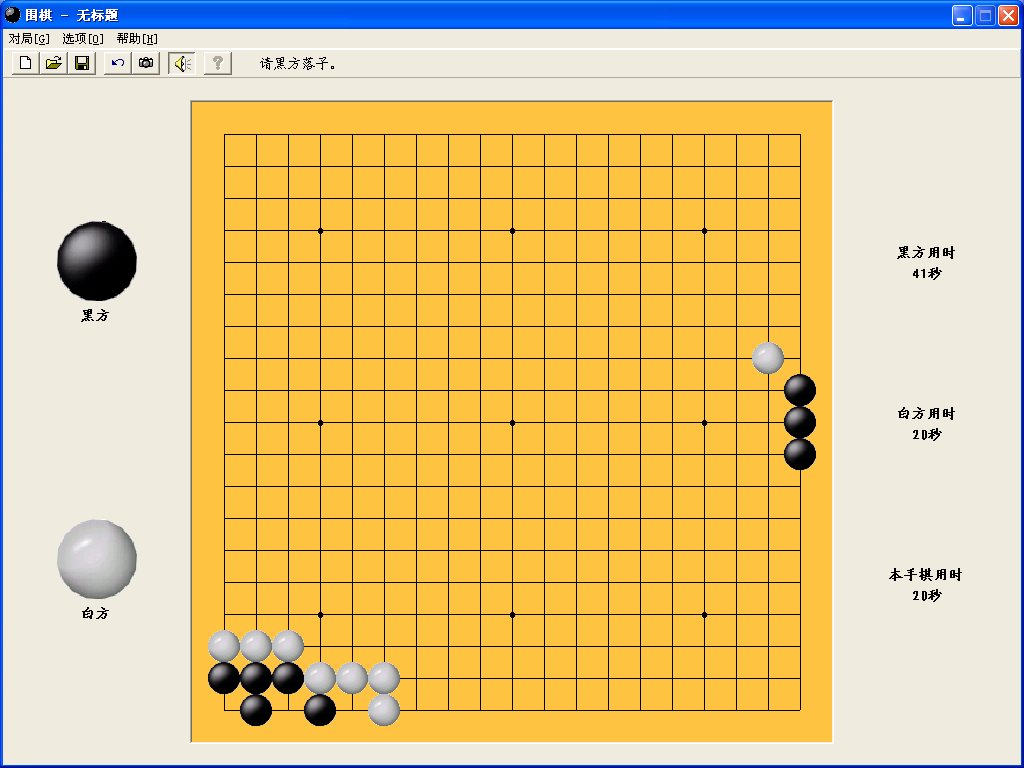
**eye**

**eye**

图2-1块棋

2）假眼

眼就是相同颜色的棋子围住的交叉点。真眼就是在棋块被围的情况下，还有两眼才能成活，而自己的两眼能使对方无法入内打吃取子。假眼即有时候眼形不完备，或者有被下子投入提吃的可能。一般一个完整的眼，在角上必须有三子，或者在边上必需有五子，或者在中腹最少占领周围七子。



a

b

图2-2眼

3）分断

分断就是把对方的棋子分割成两部分。此后双方将引起激烈战斗，所以要在对方两面薄弱的情况下选择断点。

4）死活

死棋就是棋盘上早晚都要被对方吃掉的棋子。常见的形式有两种：一种是没有气的棋子。另外一种是有些棋子虽然还有气，但本身已经没有条件来做成两个眼，最终会被对方吃掉。如图2-3所示的黑棋都是死棋。

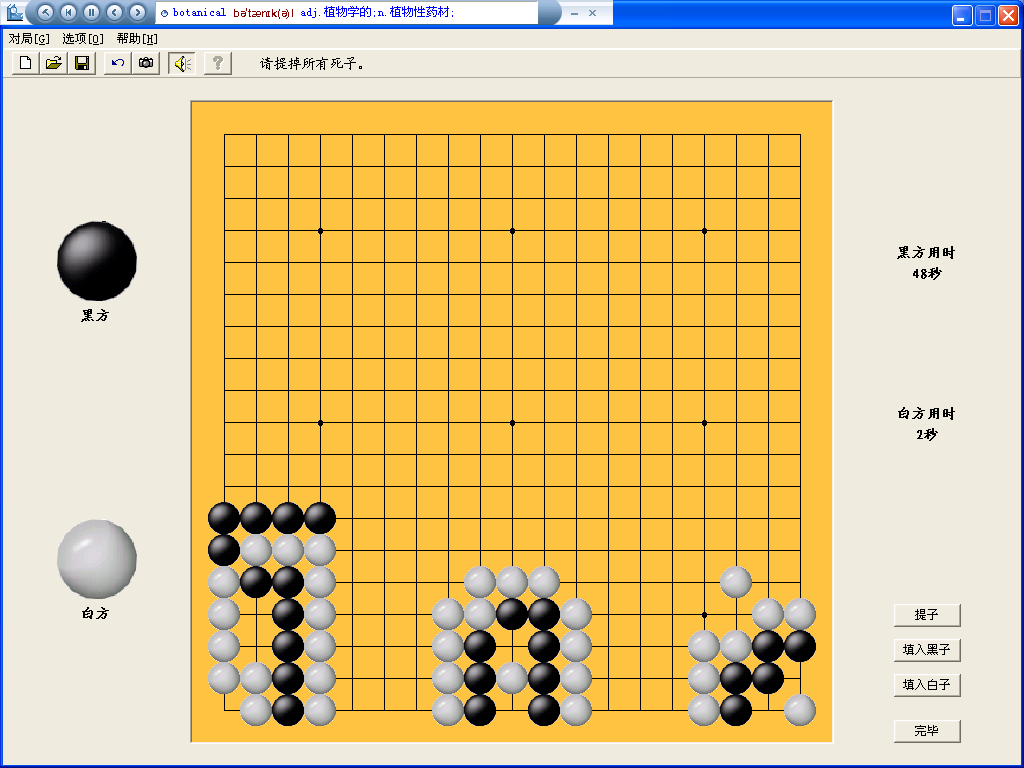
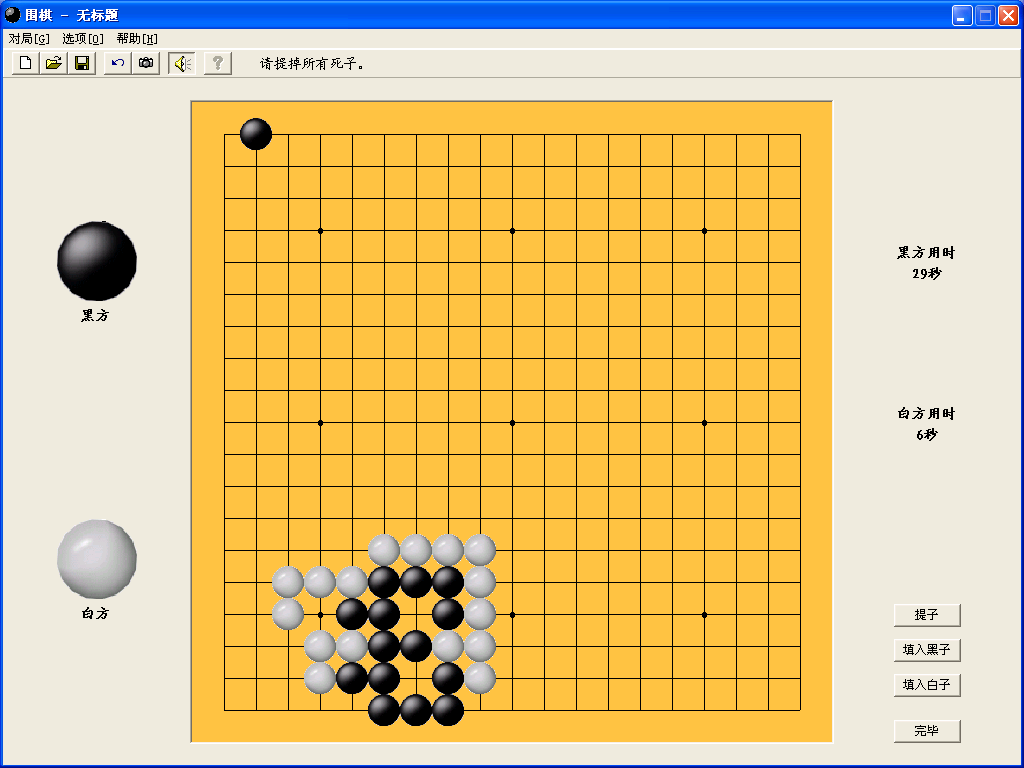


图2-3死棋

活棋指棋盘上对方无法吃掉的棋子。常见的形式有两种：一种是一块棋至少有两个对方禁着点。另外一种是一块棋所围的空点比较多，已经具有做成两个眼的条件。如图2-4所示，黑棋有a和b两个白棋的禁着点，就是活棋。



a

b

图2-4活棋

5）棋块

棋块就是两块或者多块互相紧密联络的同色块棋。棋块是围棋对弈中棋手分析棋局形势的一个比较大的单位，它的最重要的属性是确定它是否是活的，或者可以与一个已有两个眼的活棋块联络，或者能否做出两个眼。

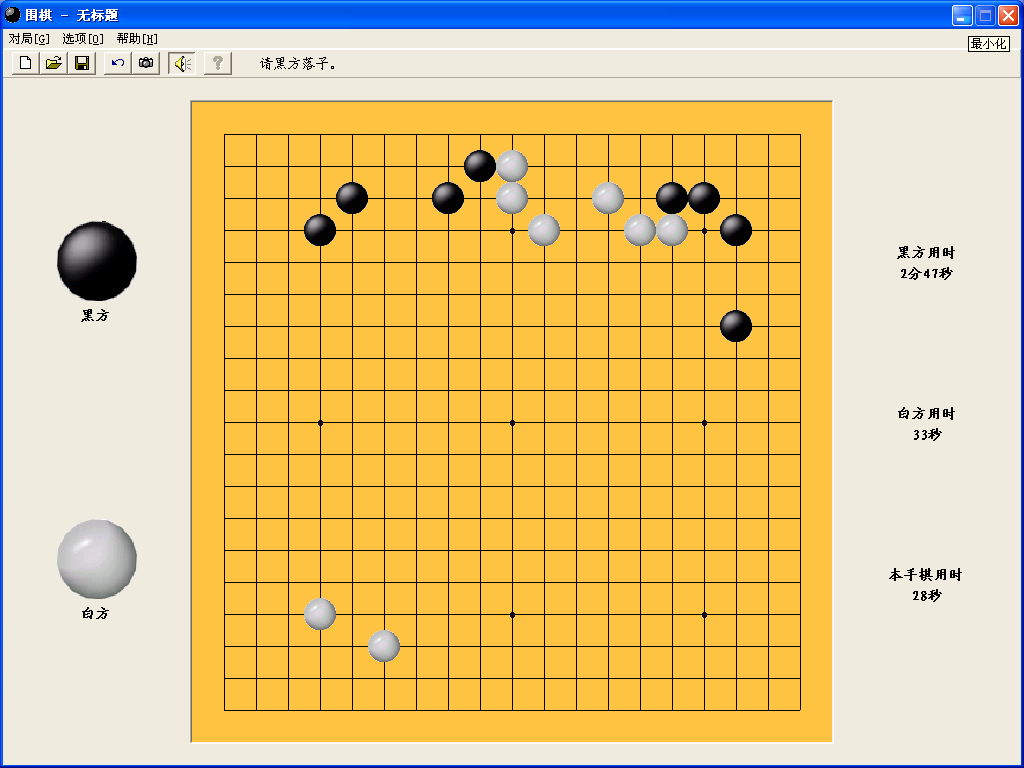


图2-5棋块

6）棋势

棋势就是两块或多块联络松散的同色块棋。对于棋手来说，将棋势转化为实地是至关重要的。而打入就是对方侵入这块地域[28][29]。

7）数棋

当围棋对弈结束后，从棋盘上提掉死子，然后分别数出双方的提子数和围空数之和，多的一方为胜者。

## Android系统简介

### Android系统概述

Android是以Linux内核为基础的开放源码移动操作平台。并且Android已经从最初的智能手机逐渐进入到人们生活的各个领域当中。Android平台由用户界面、应用软件、操作系统和中间件组成。Android平台是免费的，而且是开放的。

Android的出现不仅让Google成功迈出了移动互联网时代的第一步，也掀起了移动互联网行业新一轮的热潮。2008年20月21日，Android操作系统在Apache Software License(ASL)协议下开放源代码，同时Google在Apache协议下宣布了Android系统的全部源代码。Google将Android置于ASL许可证之下，来保证很多商业公司可以接收该平台，并利用自己独特的技术在平台上开发产品，为用户提供更多更好的服务。Android系统具备以下优点：

1）开放性

Google因为坚持”开放性“的理念，Android应用商店从而不断扩大规模。Android平台的开放性降低了许多开发人员的门槛，不仅为众多开发人员提供了更广阔的道路，而且还降低了开发成本，为运营商提供了更多开发创新应用程序的空间。

2）平等性

虽然Android系统有自己附带的程序，但是开发人员仍然可以开发新的应用程序来取代系统默认的程序，从而满足自身个性化的需求。Android平台是由一系列应用构成的平台，所有的应用程序都在一个Dalvik虚拟机上运行，该虚拟机提供了API来实现应用程序之间和硬件资源的通信访问。

3）无界限

Android平台的应用程序之间没有界限，通过开放的平台，开发人员可以自由的将自己的程序和其他相关程序结合扩展。例如将系统自带的音乐播放功能添加到自己的程序中或将GPS和网络进行交互。

4）便利性

在Android平台开发应用非常便利是因为Android平台为开发人员提供了很多库和工具，同时集成了Google Maps和Voice等强大功能，开发人员只需要调用相关代码就可将这些功能添加到自己的应用程序中。

### Android系统架构

Android的本质就是在标准的Linux系统上增加了Java虚拟机Dalvik，并且在Dalvik虚拟机上搭建了一个JAVA的application framework，所有的应用程序都是基于JAVA的application framework之上。Android主要应用于ARM平台，但不仅限于ARM，具有跨平台的特性，通过编译控制，Android在X86、MAC等体系结构的机器上也可以运行。Android系统架构如图2-6所示。



图2-6 Android系统架构图

从上图中可以看出，Android系统架构为四层结构，从上层到下层分别是应用程序层、应用程序框架层、系统运行库层以及Linux内核层，分别介绍如下：

1）应用程序层

Android平台不仅仅是操作系统，也包含了许多应用程序，诸如短信客户端程序、电话拨号程序、图片浏览器、Web浏览器等应用程序。这些应用程序都是用Java语言编写的，并且这些应用程序都是可以被开发人员开发的其他应用程序所替换，比其他手机操作系统固化在系统内部的系统软件更加灵活。

2）应用程序框架层

应用程序框架层是我们从事Android开发的基础，很多核心应用程序也是通过这一层来实现其核心功能的，该层简化了组件的重用，开发人员可以直接使用其提供的组件来进行快速的应用程序开发，也可以通过继承而实现个性化的拓展。

3）系统运行库层

从图中可以看出，系统运行库层可以分成两部分，分别是系统库和Android运行时，分别介绍如下：

系统库是应用程序框架的支撑，是连接应用程序框架层与Linux内核层的重要纽带。其主要分为如下几个：Surface Manager、多媒体库、SQLite、FreeType、WebKit、SSL、Libc。

Android应用程序时采用Java语言编写，程序在Android运行时中执行，其运行时分为核心库和Dalvik虚拟机两部分。核心库提供了Java语言API中的大多数功能，同时也包含了Android的一些核心API，如android.os、android.net、android.media等。Dalvik虚拟机是一种基于寄存器的Java虚拟机，而不是传统的基于栈的虚拟机，并进行了内存资源使用的优化以及支持多个虚拟机的特点。需要注意的是，不同于J2me，Android程序在虚拟机中执行的并非编译后的字节码，而是通过转换工具dx将Java字节码转成dex格式的中间码。

4）Linux内核层

Android是基于Linux2.6内核，其核心系统服务如安全性、内存管理、进程管理、网路协议以及驱动模型都依赖于Linux内核。

## 微服务架构

微服务架构(Microservice Architect)是一种架构模式，它提倡将单块架构的应用划分成一组小的服务，然后服务之间互相协调和配合，从而为用户提供最终价值。每个服务都在其独立的进程中运行，并且服务和服务使用轻量级通信机制相互通信。 每项服务都是围绕特定业务构建的，可以独立部署到生产环境和类生产环境中等。

单体应用架构(Monolithic Architecture)是传统的应用架构。单体应用架构因为其庞大复杂，难以进行持续化部署，应用的扩展也很难，可靠性较差，让采用新框架和新语言变得极其困难。微服务架构被称为新一代的面向服务架构(Service Qriented Architecture SOA)，它实现了同时兼容多种技术，并且提高了应用迭代的效率，它是互联网应用的普适架构和最佳实践。

### 微服务架构设计理念

微服务架构起源于eBay、Netflix等世界著名互联网公司，是敏捷开发、虚拟化、DevOps等技术理念的产物。其核心设计理念是将复杂的应用系统以独立业务单元的形式分解为多个特定具体服务。微服务架构的设计理念继承了面向服务架构的特点，微服务架构中的各个服务松散和自治，强调服务的“独立性”——独立开发、独立部署和独立运行，从而具有易实施性和易扩展性等特点。微服务的应用系统将各种功能分开，并使用轻量级通信机制相互通信以实现完整的应用程序。系统中微服务体系结构服务之间的变化周期可以异步更新，并且可以更新局部过期的组件，从而完成整个系统升级并实现渐进式操作和维护。

单体式架构意味着部署的应用程序是一个整体，并且都在一个进程中运行。 然而在单体式架构应用中存在许多问题，如系统的复杂性，多个内部模块的紧密程度以及关联依赖杂。其次，运维比较困难，变更或升级的影响难以分析。 任何细微的修改都可能导致单体式应用程序的整体操作失败。同时对扩展性支持不好，部署也不能拆分。 如果出现性能瓶颈通常只能增加服务器或增加集群节点。

### 微服务架构特征

微服务体系架构借鉴了Unix的设计风格。 微服务不再强调传统SOA架构中相对较重的ESB企业服务总线，SOA理念被整合到单一业务系统中以实现真正的组件化。 因此微服务可以说是SOA的一种实践。微服务架构具有如下特征：

1）服务组件化

微服务体系结构将单个业务系统划分为几个可根据服务独立开发，运行和部署的组件。单体式应用程序中一个模块更改，会导致整个程序重新部署，如果使用微服务，则系统会合理划分为多个独立的服务。对每个服务的更改不会影响其他服务的部署。这样可以根据组网需求灵活扩展业务，并且可以根据实际情况动态添加和定制服务。

2）根据业务进行服务开发

传统的开发模式根据不同的技术进行分层，任何细微的变化都会带来跨层沟通。 微服务架构所倡导的开发模式是根据业务功能划分服务。每个微服务都需要整个业务功能所需的技术栈。团队必须拥有各方面业务需求的人才。微服务架构下的设计比单体式架构更畅顺。

3）去中心化

传统的应用开发中，不同的模块基本上是采用统一技术开发的，但是针对每个功能，可能会有更好的技术方案，这将大大提高开发效率。如果采用微服务体系结构，则单体式应用程序分为多个独立的服务。这使得可以针对不同服务服务的特性选择技术解决方案。开发团队可以选择不同的编程语言，不同的业务框架，服务不关心其实现细节。只需要使用统一的通信框架进行服务管理。开发人员可以自由选择当前的新技术，而不必考虑与先前采用的技术的兼容性。

4）数据管理去中心化

传统的单体式应用程序通常使用单个数据库来存储数据。微服务更倾向每个服务来管理自己的数据库。这些数据库可能是同一个数据库的不同实例，甚至可能是不同的数据库。这是多种类的持久化，可以降低数据耦合的程度。

5）基础设施自动化

当应用程序分为多个独立的微服务时，开发，测试，操作和维护的相应复杂性将得到改进，合适的自动化基础设施可以大大降低成本并确保软件的整体质量。

微服务的目的是有效地分割应用程序并实现敏捷开发和部署。微服务可以带来很多好处。例如微服务强调模块化结构。微服务支持独立部署。简单的服务当然比单个系统更容易部署。如果服务发生问题，不会导致整个系统失败，不同的微服务可以使用不同的编程语言和数据存储技术。

### 微服务的设计原则

微服务架构是一种软件开发模式，需要明确设计与开发微服务过程中遵循的原则，并且在符合微服务原则的情况下，来进行开发部署。这样就可以有利于发挥出该微服务架构所带来的优势。

1）单一功能原则

单一功能原则建议单一对象只应该承担单一责任。该模型中对象状态的变化只会由于外界单一因素的变化而产生。如果一个对象由于外部世界中的多种因素而发生变化，那么该对象在内部与多个功能职责相耦合。单个微服务所负责的业务逻辑是专注于具有设计良好的外部接口的特定功能，并且这些接口清楚地定义了功能之间的界限。 开发人员只需要指定功能内部的职责以及与外部交互的接口，就可以构建出具有良好的可重用性和可维护性的应用。

2）独立部署原则

单体式应用架构中，功能模块的代码集中在同一个发布包中。这种方法增加了功能代码之间相互影响的可能性。独立部署原则要求开发，测试，构建和部署每个微服务的整个过程都在物理上独立的环境中进行。该设计原则优势在于，单个微服务可以由一个独立团队开发，测试和部署，这缩短了微服务的交付周期。开发人员可以在完成自己的服务后及时发布，而不是像传统方式那样等待其他模块一起构建和发布。除了提高应用程序的部署效率之外，该原则还可以提高整个系统的容错能力。由于该服务是在单独的物理环境中部署和运行的，因此发生的故障可以限制在该服务范围内。

3）无状态原则

在使用微服务的过程中，微服务可以根据实际生产环境水平扩展或缩减。当微服务的多个实例被扩展或缩减时，如果服务内部保有先前请求处理的状态信息，那么当服务实例进行调整更改时，它涉及大量的状态信息保留和传输，对配置的更改或对应用程序代码的更改会导致复杂性增加。无状态原则意味着微服务在接受来自外部的请求时，不需要记录与外部用户有关的通信信息。当微服务实例出现问题，或者动态扩展缩减时，可以非常快速地添加和缩减无状态实例。

4）轻量级通信原则

传统的SOA应用架构中，不同的服务进程之间一般采用RPC(Remote Produce Call)远程过程调用的方式。该方式将请求程序作为客户端，然后提供服务的程序作为服务端。但是使用RPC会导致应用程序对开发语言和平台的依赖性太高，导致程序开发的灵活性下降。轻量级通信原则提倡开发者采用一些与开发语言和平台不相关的机制进行交互。常用的轻量级通信协议有XML和JSON，这两种协议具有对复杂信息的表达能力，并且能够实现跨平台的信息传输。现在比较适合微服务之间进行通信的是REST (Representational State Transfer)通信机制，在该格式中每个网络资源都被抽象成为一个地址，并且功能的实现通过对资源的操作来完成。

## Docker容器技术

根据上面所述开发微服务的设计原则，需要一种适合微服务特点的实现方法和工具，并且需要使得开发过程变得便捷而快速。随着云计算技术的普及，硬件设施虚拟化技术成为了支撑云计算应用部署和运行的关键技术。Docker容器的Linux平台容器虚拟化技术，受到产业界的关注和发展。Docker容器的出现使得软件开发可以更好地利用云计算带来的高效资源利用的优势[9]。

### Docker技术简介

Docker的核心理念为“构建一次，到处运行”，它是开源的容器引擎，在Linux容器技术的基础上，进一步封装了容器的一些操作接口，并且提供给开发者管理和使用容器，而不需要关注Linux容器底层较为复杂的运行机制。传统的虚拟化技术在宿主操作系统之上还需要提供一层针对虚拟机的管理层和虚拟机本身的操作系统，这样的做法虽然可以模拟出一套完整的硬件环境给虚拟机，但是因为对资源过度的封装使得虚拟机对系统的资源需求很大，而且额外的虚拟机管理层也会占用系统资源，从而导致了一台物理机上只能够运行数量较少的虚拟机[10]。

Docker容器技术去除了针对虚拟机的管理层和每个虚拟机中的操作系统，直接在宿主操作系统上利用LXC(Linux Container)技术来进行资源的管理，直接运行在操作系统内核之上的用户空间，从而使得Docker容器基本不消耗额外的系统资源，同时还可以实现多个独立的进程运行于同一宿主机上。正是由于Docker的这种基于原生操作系统进行虚拟化的方式，使得在实际生产环境中我们可以同时在一台物理机上部署运行上千个Docker容器[27]。



图2-7 Docker和传统虚拟化方式对比图

从应用程序开发人员的角度来看，我们可以将Docker容器环境视为沙盒，然后打包并部署应用程序的源代码，函数库和工具从而在沙箱中运行。不需要关心容器之间的资源隔离、独立运行和通信问题。Docker容器有着如下的优点：

1）持续部署与测试

Docker消除了线上和线下的环境差异，从而保证了应用生命周期的环境一致性和标准化。开发人员使用镜像实现标准开发环境的构建，开发完成后通过分装着完整环境的应用的镜像进行迁移。从而测试和运维人员可以合一，然后通过直接部署软件镜像来进行测试和开发，大大简化了持续集成、测试和发布的过程。

2）跨云平台支持

Docker带来的好处之一就是其适配性。随着越来越多的云平台支持Docker，用户没有必要担心受到云平台的捆绑，同时也让应用多平台混合部署成为可能。

3）环境标准化和版本控制

基于Docker提供的环境一致性和标准化，可以使用git等工具对Docker镜像进行版本控制，并能够对整个应用运行环境实现版本控制，一旦出现故障可以快速回滚。相比以前的虚拟机镜像，Docker压缩和备份速度更快，镜像启动也如启动一个普通进程一样快速。

4）高资源利用率与隔离

Docker容器没有管理程序的额外开销，与底层共享操作系统，性能更加优良，系统负载更低，在同等条件下可运行更多的应用实例，可以更充分地利用系统资源。Docker拥有良好的资源隔离与限制能力，可以精确地为应用分配CPU、内存等资源，保证了应用间不会相互影响。

5）容器便携性与镜像

Linux容器虽然早在Linux2.6版本内核已经存在，但是缺少容器的便携性，难以推广。Docker在原有Linux容器的基础上进行革新，为容器设定了一整套标准化的配置方法，将应用及其依赖的运行环境打包成镜像，真正实现了“构建一次，到处运行”的理念，从而大大提高了容器的便携性。

6）应用镜像仓库

Docker官方构建了一个镜像仓库，并已累积了非常多的镜像。由于Docker的跨平台适配性，相当于为用户提供了一个非常有用的应用商店，因此所有人都可以自由地下载微服务组件，从而为开发者提供了巨大的便利。

### Docker实现的关键技术

随着Docker容器技术的发展，一些问题也随之而来。比如确保用户的资源隔离性问题，用户的资源使用情况统计问题，用户的Docker实例进行迁移或者横向扩展问题，确保Docker容器进程的操作权限的安全性问题等等。我们描述Docker通过解决这些问题来介绍Docker实现的原理。

1）Linux内核命名空间

命名空间技术是由Linux内核创建的资源隔离方案，以更好地支持容器级的虚拟化。它是为了使每个容器在使用不同的硬件资源时都有不同的名称空间，比如CPU、内核、内存、网络和进程等，以确保每个容器的资源在命名空间中对其他命名空间不可见。该技术解决了如何确保容器使用者资源的隔离问题。目前Linux提供的命名空间如表2-1所示。

表2-1 Linux内核命名空间

|  |  |  |
| --- | --- | --- |
| 命名空间 | 系统调用参数 | 作用 |
| 主机域名命名空间 | CLONE\_NEWUTS | 提供独立主机名和域名，模拟网络上独立的主机 |
| 进程交互命名空间 | CLONE\_NEWIPC | 确定进程交互权限 |
| 进程命名空间 | CLONE\_NEWPID | 管理不同进程的进程号 |
| 网络命名空间 | CLONE\_NEWNET | 隔离网络端口 |
| 挂载命名空间 | CLONE\_NEWNS | 针对进程隔离文件目录 |
| 用户命名空间 | CLONE\_NEWUSER | 划分用户和组 |

2）Linux内核控制组技术

Linux内核控制组技术主要用于限制用户进程所占用的资源，并且可以提供相对准确的度量，并且还可以提供报告来衡量实际的系统资源分配情况。控制组技术的主要实现机制是为不同资源创建一系列子系统。每个子系统可以描述和控制不同类型的资源。设备子系统通过不同的组来划分对特定设备的访问。

3）Union文件系统

构建Docker镜像的基础就是Union文件系统。其特点是文件系统的操作被抽象为一种提交形式。在进程运行过程中，多个操作将被修改。这些操作形成一系列提交，并且用户对文件的操作由提交的叠加来描述。在该文件系统中，该文件可以具有三个操作属性，包括写入、读写和只读。

### Docker与微服务设计原则

在完成上面对微服务设计原则的阐述和Docker实现技术分析后，本节将结合Docker容器优势和微服务的设计原则，然后通过Docker容器技术来部署微服务。通过命名空间技术提供的Docker容器间的隔离性和控制组技术提供的资源分配和监控功能可以满足微服务的独立部署原则。独立部署是微服务区别于其他传统的SOA架构服务和一体化架构的主要特点。该原则主要是从物理的角度规定了微服务的部署环境必须在物理上相对的独立，而Docker容器这种资源隔离特性借助Linux系统提供的强大的资源隔离措施，从而完全可以达到独立部署的需要。Docker可以通过在宿主机上通过网桥模式便捷的虚拟出独立的地址对外进行网络访问，而且容器不仅可以和宿主机进行通信，Docker实现了同一宿主机内部不同容器之间的通信机制。便捷的网络配置和通信也给实施轻量级通信原则带来便捷。

综上所述，微服务架构可以为开发者提供独立部署、应用解耦、快速迭代等优势，Docker容器为微服务提供了便捷的实现方式和部署运维方案。

## 本章小结

本章对线上围棋教学系统中的相关技术进行了详细的说明。首先介绍了围棋相关的术语，了解了围棋相关的知识。然后分析 Android 的系统概述与平台架构。最后，分析了微服务架构的设计理念和Docker容器技术，然后介绍了结合Docker容器的优势与微服务的设计原则的优势，这些基础性技术是本文系统设计的基础。

# 基于微服务架构的线上围棋教学系统分析

本章从总体上对线上围棋教学系统进行了分析，分析了线上围棋教学系统的需求，包括其功能性需求和非功能性需求，并对系统实现模型即功能模型、结构模型和行为模型对系统的功能模块进行了详细的分析，使用UML图形来分析介绍整个系统。

## 系统需求描述

随着移动互联网的发展越来越成熟，对移动应用的多元化和生活化有了更多的需求。移动用户只需要通过手机就能获取所需信息，并发布相关信息。本系统的功能是集围棋对弈与围棋教学于一体，采用移动客户端和服务端微服务化的设计方式开发了一个方便教师和学生的围棋教学系统。本系统需要在设计完成后，能实现以下功能：

1）学校管理：该功能主要管理用户的个人信息和班级信息。用户可以在客户端实现普通账号和第三方平台账号的注册和登录功能。用户角色分为教师和学生。教师可以创建班级，批准学生的加入，并维护班级学生信息。学生可以搜索班级信息，并选择自己感兴趣的班级，并申请加入和等待批准加入。教师可以在管理的班级中发布作业，该班级中的学生可以查看到该作业信息；

2）聊天管理：支持学生与教师之间的实时交流功能，用户之间可以点对点的进行即时聊天，聊天的内容包括文本、图片等常见的格式。用户也可以查看会话列表信息和历史聊天记录；

3）围棋对弈：用户可以在移动客户端实现对弈功能，并且支持落子、自动提子等功能。并且用户可以查看系统中预存的一些经典名家对局棋谱，用户还可以在进行对弈时，保存新的棋谱，方便下次查看；

4）围棋局面分析：该功能主要为在围棋对弈过程中，可以实时查看围棋形势分析情况。棋局是动态发展的，形势判断要求棋手对未来的棋局发展做出预期，用户可以根据形势状况，实时调整自己的围棋对弈策略，来获得围棋对弈的胜利；

## 系统功能分析模型

结合系统的需求，对系统进行分析，为了更加清晰的分析系统的功能需求，根据面向对象的相关理论和技术，从三个维度对系统建立相应的模型进行分析，包括系统的交互模型、系统的结构模型和系统的行为模型。

### 系统交互模型

通过对系统进行功能需求分析，该系统包括学生和教师两种用户。教师用户在系统中主要是管理班级信息和发布围棋作业信息。学生用户在系统中主要是加入班级和查看班级中发布的作业信息，并且在移动客户端实现围棋对弈功能。

围棋教学用例图如图3-1所示。



图3-1 围棋教学用例图

学生可以进行如下操作：学校管理、聊天管理和围棋对弈。学校管理主要为管理用户信息和申请加入班级，加入班级后就可以查看该班级中其他学生信息和作业信息。聊天管理主要为单聊、查看聊天会话和查看历史聊天记录。围棋对弈主要为学生可以和其他用户进行围棋对弈、保存棋谱、查看棋谱和查看当前局面的形势分析。

学生围棋教学用例的详细设计如表3-1所示。

表3-1 学生围棋教学用例描述

|  |  |
| --- | --- |
| 用例 | 描述 |
| 用例名称 | 围棋教学 |
| 主要参与者 | 学生 |
| 前置条件 | 系统运行正常，用户成功登录围棋教学系统 |
| 基本事件流 | 1、学生登陆系统  2、学生点击学校管理，修改用户的个人信息  3、学生点击学校管理，搜索班级信息，并申请加入班级  4、学生点击聊天，选择其他用户进行单聊  5、学生点击聊天，查看聊天会话列表  6、学生点击聊天，查看历史聊天记录  7、学生点击围棋对弈，选择其他用户进行对弈  8、学生点击围棋对弈，保存棋谱或者查看棋谱  9、学生点击围棋对弈，查看当前局面的形势分析 |
| 结论 | 学生围棋教学成功 |

教师可以进行如下操作：学校管理、聊天管理和围棋对弈。学校管理主要为管理用户信息和创建班级并批准学生加入班级，教师在班级中发布作业信息。聊天管理主要为单聊、查看聊天会话和查看历史聊天记录。

教师围棋教学用例的详细设计如表3-2所示。

表3-2 教师围棋教学用例描述

|  |  |
| --- | --- |
| 用例 | 描述 |
| 用例名称 | 围棋教学 |
| 主要参与者 | 教师 |
| 前置条件 | 系统运行正常，用户成功登录围棋教学系统 |
| 基本事件流 | 1、教师登陆系统  2、教师点击学校管理，修改用户的个人信息  3、教师点击学校管理，创建班级，批准学生加入，发布作业信息  4、教师点击聊天，选择其他用户进行单聊  5、教师点击聊天，查看聊天会话列表  6、教师点击聊天，查看历史聊天记录 |
| 结论 | 教师围棋教学成功 |

1. 教师用户角色

在线上围棋教学系统中，教师负责班级管理，并给学生答疑解惑，在系统中是不可或缺的一类角色。教师用户的注册和登录功能是教师用户进入到系统中的入口，注册和登录功能又分为普通账号注册登录和第三方平台新浪微博账号注册登录。这类用户独有的功能是创建班级和发布作业。该角色的用例图如图3-2所示。



图3-2 教师角色用例图

教师可以进行如下操作：单聊、管理班级信息和管理作业信息。

（1）单聊用例的详细设计如表3-3所示。

表3-3 单聊用例描述

|  |  |
| --- | --- |
| 用例 | 描述 |
| 用例名称 | 与学生进行即时在线聊天 |
| 主要参与者 | 教师 |
| 前置条件 | 用户成功登录教学系统 |
| 约束条件 | 聊天内容格式仅限文字、语音、图片 |
| 基本事件流 | 1、用户登陆系统，进入用户主页面  2、可以查看通讯录中的联系人信息  3、点击选择一个联系人信息，输入想要发送的聊天内容  4、点击发送按钮，发送聊天内容 |
| 结论 | 发送聊天内容成功 |

（2）管理班级信息用例的详细设计如表3-4所示。

表3-4 管理班级信息用例描述

|  |  |
| --- | --- |
| 用例 | 描述 |
| 用例名称 | 管理班级信息 |
| 主要参与者 | 教师 |
| 前置条件 | 用户成功登录教学系统 |
| 基本事件流 | 1、用户登陆系统，进入班级界面  2、创建新的班级，等待学生的申请  3、教师看到申请信息，并批准学生的加入  4、教师管理该班级信息 |
| 结论 | 管理班级信息成功 |

（3）管理作业信息用例的详细设计如表3-5所示。

表3-5 管理作业信息用例描述

|  |  |
| --- | --- |
| 用例 | 描述 |
| 用例名称 | 上传并管理作业信息 |
| 主要参与者 | 教师 |
| 前置条件 | 用户成功登录教学系统 |
| 基本事件流 | 1、用户登陆系统，进入用户主页面  2、可以上传作业信息的资料  3、点击上传按钮，可以查看上传的详细信息  4、用户可以管理自己上传的作业信息，例如修改、删除等  5、教师管理发布作业信息 |
| 结论 | 上传或管理作业信息成功 |

1. 学生用户角色

在线上围棋教学系统中，学生可以咨询教师围棋信息，并在围棋对弈中提升自己的围棋水平。学生可以和其他用户进行单聊，聊天内容可以是文字、语音和图片。学生用户可以搜索班级，并申请加入自己感兴趣的班级，也可以查看到该班级中发布的作业信息。学生可以其他用户进行对弈，查看棋谱，以及实时查看当前对弈局面的形势评估，动态调整对弈策略。该角色的用例图如图3-3所示。



图3-3 学生角色用例图

学生可以进行如下操作：单聊、班级管理和围棋对弈。

（1）单聊用例的详细设计如表3-6所示。

表3-6 单聊用例描述

|  |  |
| --- | --- |
| 用例 | 描述 |
| 用例名称 | 与教师进行即时在线聊天 |
| 主要参与者 | 学生 |
| 前置条件 | 用户成功登录教学系统 |
| 约束条件 | 聊天内容格式仅限文字、语音、图片 |
| 基本事件流 | 1、用户登陆系统，进入用户主页面  2、可以查看通讯录中的联系人信息  3、点击选择一个联系人信息，输入想要发送的聊天内容  4、点击发送按钮，发送聊天内容 |
| 结论 | 发送聊天内容成功 |

（2）班级管理用例的详细设计如表3-7所示。

表3-7 班级管理用例描述

|  |  |
| --- | --- |
| 用例 | 描述 |
| 用例名称 | 班级管理 |
| 主要参与者 | 学生 |
| 前置条件 | 用户成功登录教学系统 |
| 基本事件流 | 1、用户登陆系统，进入班级页面  2、可以查看并搜索自己感兴趣的班级信息  3、选择某个班级，并申请加入  4、教师如果批准，则该学生成为班级中的一员  5、学生可以查看班级中的作业信息 |
| 结论 | 班级管理成功 |

（3）围棋对弈用例的详细设计如表3-8所示。

表3-8 围棋对弈用例描述

|  |  |
| --- | --- |
| 用例 | 描述 |
| 用例名称 | 与其他用户进行围棋对弈 |
| 主要参与者 | 学生 |
| 前置条件 | 用户成功登录教学系统 |
| 基本事件流 | 1、用户登陆系统，进入对弈界面  2、可以选择某个用户进行对弈  3、点击新的棋局或者加载保存的棋局，进行对弈  4、用户可以查看系统中预存的一些经典对局棋谱，也可以在对弈时，选择保存当前对局，成为新的棋谱  5、对弈时，用户可以实时查看对当前局面的形势评估  6、用户可以申请点目，来结束围棋对弈  7、用户可以申请和棋，来结束围棋对弈  8、用户可以认输，来结束围棋对弈  9、围棋对弈结束 |
| 结论 | 围棋对弈成功 |

### 系统结构模型

类图主要是用来描述系统的类、接口以及它们之间的静态关系。在对系统类设计过程中，由于所涉及方法和属性数量较多，本文类图重点介绍各类之间的关系以便说明系统工作的原理，省略大部分类的方法和属性，只对一些重要类的方法和属性作出显示说明。系统总体类设计如图3-4所示。

图3-4 系统总体类图

系统核心类主要包括：User类、Role类、Student类、Teacher类、Class类、CourseInfo类、MessageRecord类、ChatInfo类、ConversationList类、Board类、Color类、Point类、SGFInfo类、ChessManual类、GTPInfo类、StrengthInfluence类、ParameValue类、EffectNum类、DataList类。

User类表示用户类，它主要包括用户ID、密码、类型、状态等属性，是一个抽象类，继承它的子类有Teacher类和Student类。User类中抽象了公共的属性，Teacher类自己额外的属性包括教师ID、班级等，Student类自己额外的属性包括学生ID、班级等。Class类表示班级的信息。CourseInfo类表示作业类，它主要包括作业ID、教师ID等属性。MessageRecord类表示聊天信息类，它主要包括消息ID、消息内容、消息类型、消息创建时间等属性，ChatInfo类用来记录用户之间发送的聊天消息，ConversationList类表示会话列表信息。Board类表示棋盘类，主要记录棋盘的一些信息，包括棋盘大小，属性等。Color类表示棋子颜色类，分为黑白两种颜色。Point类表示棋子坐标类。SGFInfo类表示SGF文件解析类，用来解析棋谱文件时的信息。ChessManual表示棋谱类。GTPInfo类表示GTP协议类，用来加载围棋对弈时，客户端和服务端传递的对弈信息。StrengthInfluence类表示影响模型类，用来保存围棋局面分析形势的类。ParameterValue类表示参数类。EffectNum表示有效子类。DataList用来保存一些数据信息类。

### 系统行为模型

本小节使用活动图对线上围棋教学系统运行的业务流程进行动态建模。此应用为基于线上围棋教学系统，下面将为系统注册登录、即时聊天、学校管理、围棋对弈、作业练习等主要业务流程使用UML活动图进行动态建模。

1）用户登录

根据需求，用户对系统进行操作时，首先要登录系统。

用户登录活动图如图3-5所示。



图3-5 用户登录活动图

用户登录具体流程如下：

（1）进入到系统登录主页面；

（2）用户选择普通用户或者第三方平台账号登录；

（3）如果是普通用户登录，则输入用户名密码；

（4）如果是第三方平台账号登录，则输入第三方平台用户账号；

（5）验证用户信息的合法性

（6）如果验证成功，则进入到主界面

（7）如果验证失败，则需要重新登录；

（8）结束；

2）即时聊天

聊天增强了用户对系统的粘性，关系到用户的回流率，提高了用户体验。聊天是用户与用户之间进行即时对话，解决了用户沟通，有利于用户了解围棋信息。用户首先需要进行登录成功并进入到系统，然后选择通讯录中的联系人，进入到聊天界面，用户可以输入文本、语音、图片等消息内容进行发送，发送成功后将消息推送给接收者。发送失败则会提示用户发送消息失败请重新发送。即时聊天活动图如图3-6所示。



图3-6 即时聊天活动图

即时聊天具体流程如下：

（1）用户A登录成功；

（2）进入通讯录，选择联系人B，进入到聊天界面；

（3）输入聊天内容，消息内容可以为文本、语音、图片等；

（4）点击发送按钮，判断是否发送成功；

（5）如果成功，则用户B会收到用户A发送的消息内容；

（6）如果失败，则需要重新发送消息内容，执行（3）；

（7）结束；

3）学校管理

用户在注册的时候，可以选择自己的身份，学生或者教师。不同的身份对应不同的操作流程。教师可以创建班级，并管理学生，比如批准学生进入班级，还可以在班级中发布作业信息等。学生可以查看班级信息，并且申请加入自己感兴趣的班级。

教师管理活动图如图3-7所示：



图3-7 教师管理活动图

教师管理活动具体流程如下：

（1）教师登录；

（2）进入到班级管理界面；

（3）教师可以创建新的班级，并维护该班级信息；

（4）教师查看自己管理的班级信息，并管理学生是否能加入该班级；

（5）对应的班级信息更新；

（6）教师可以在该班级中发布作业信息，该班级中的所有学生在作业界面都会看到新的作业信息；

（7）结束；

学生管理活动图如图3-8所示：



图3-8 学生管理活动图

学生管理活动具体流程如下：

（1）学生登录；

（2）进入到班级管理界面；

（3）学生可以查看班级信息，并申请加入自己感兴趣的班级，等待教师批准；

（4）如果教师没有批准，则不能查看班级成员，执行（3）；

（5）如果教师批准，则该学生可以查看新的班级里的成员；

（6）学生加入班级后，可以查看班级中发布的作业信息；

（7）结束；

4）围棋对弈

用户进入到围棋对弈模块，可以选择“人-人对弈”，然后邀请其他用户，如果对方同意，然后就可以和对方进行在线对弈。对弈界面有申请点目、申请和棋和认输三个按钮。申请点目即用户认为已经完成对弈，点击按钮之后，如果对方同意，系统会自动按照规则进行目数统计，分出胜负。申请和棋即用户认为双方旗鼓相当，点击按钮之后，如果对方同意，系统会自动按照和棋结束。认输即用户认为自己已经没有胜利的希望，点击按钮之后，如果对方同意，系统会自动按照认输结束。围棋对弈活动图如图3-9所示：



图3-9 围棋对弈活动图

围棋对弈活动的具体流程如下：

（1）用户A登录；

（2）进入对弈界面；

（3）用户A邀请用户B进行人-人对弈；

（4）如果用户B拒接邀请，则执行（2）。

（5）如果用户B同意邀请，则执行（6）；

（6）用户A和用户B进行对弈；

（7）开始对弈后，如果用户A认为对弈已经完成，可以申请点目，则会询问用户B；

（8）如果用户B同意申请点目，则系统会自动统计目数，计算胜负；

（9）如果用户B拒绝申请点目，则继续对弈，执行（6）；

（10）开始对弈后，如果用户A认为对弈已经双方旗鼓相当，可以申请和棋，则会询问用户B；

（11）如果用户B同意申请和棋，则系统会自动结束对弈，双方和棋；

（12）如果用户B拒绝申请和棋，则继续对弈，执行（6）；

（13）开始对弈后，如果用户A认为对弈已经无法取得胜利，可以申请认输，则会询问用户B；

（14）如果用户B同意认输，则系统会自动判定用户A失败，用户B胜利；

（15）如果用户B拒绝认输，则继续对弈，执行（6）；

（16）对弈结束，用户保存棋谱；

（17）结束

## 系统非功能需求分析

由于软件系统的非功能性需求也会影响到系统的功能性需求，考虑非功能性需求也是非常有必要的。结合实际项目的特点，客户端采用移动终端，服务端采用微服务化开发，所以本文系统的非功能性需求分为移动客户端的兼容性、服务端的可扩展性和服务端的可靠性需求三个方面。

### 移动客户端的兼容性需求

由于本系统的移动客户端能够运行在不同智能终端设备上，这些设备具有不同的分辨率和不同尺寸的屏幕，常见的包括1920\*1280、1280\*720、960\*640等，屏幕尺寸有4.7寸、5.0寸、5.5寸等。其次可能使用不同版本Android系统，现在主流使用的Android系统版本有Android5.0、Android6.0、Android7.0等。

在设计实现系统客户端时，需要充分考虑客户端在不同运行环境下的兼容性需求。对于不同分辨率的屏幕需要能够实现界面的相对布局，按照比例来分配控件所占屏幕大小。该系统需要能够在不同版本Android系统上运行。该系统需要能够适应不同硬件的性能差异，并且在性能比较低的机型上运行能够基本达到流畅和正常的响应速度要求。

### 服务端的可扩展性需求

该系统平台化服务框架将所有功能模块划分，允许开发者自定义框架的内容和开发扩展新的功能模块。服务接口的定义保持良好的对旧版本和新版本的兼容性。服务的部署和版本升级，应该支持平滑过渡。平台化服务API网关的反向代理检验流程接口化，松耦合，更好的应对API反向代理流程的变化。

### 服务端的稳定性需求

系统可靠性表示系统在规定的条件下和规定的时间内完成规定功能的能力。系统的稳定性对于一个系统来说也是非常重要的。每一个服务器的CPU和内存占用情况都不能太高，否则会影响功能的使用性。

## 本章小结

本章介绍了线上围棋教学系统的需求分析，完成了对系统功能需求的分析，通过UML分析模型中的功能模型和行为模型对系统进行了需求分析以及各个功能模块的分析。这些内容为第四章系统的实现做了准备。

# 基于微服务架构的线上围棋教学系统设计

本章在对线上围棋教学系统分析的基础上，从整体上描述系统设计方案，分别对移动客户端和API网关进行详尽描述和功能划分，服务端微服务化分为三个微服务，分别对学校管理模块、聊天管理模块和围棋引擎模块三个方面进行了详细设计。

## 系统概要设计

### 系统体系结构

从系统整体设计上来看，系统主要由移动客户端、API Gateway、学校管理服务、围棋引擎服务和聊天管理服务组成。系统的体系结构图如图4-1所示：



图 4-1 线上围棋教学系统体系结构图

1）移动客户端。用户通过移动客户端向服务器发送请求，管理学校和用户的信息，和其他用户进行实时聊天，进行围棋对弈等功能。

2）API Gateway。客户端通过API Gateway和服务器进行通信，它是进入后端服务器的唯一节点。所有来自客户端的请求都要先经过API Gateway，然后路由这些请求到对应的微服务。API Gateway封装内部系统的架构，并且提供REST API给客户端进行调用。API Gateway负责请求的转发、合成和协议转换，并且经常通过调用多个微服务来处理同一个请求以及聚合多个服务的结果。

3）学校管理服务。学校管理服务主要负责管理学校信息。用户注册登录的信息保存在该服务的数据库上。用户在登录的时候，检验用户信息的正确性。教师通过建立班级来管理学生信息。学生可以通过检索班级信息，并加入自己感兴趣的班级来学习围棋课程。教师还可以在班级中发布作业信息，该班级中的学生可以查看到该作业信息。

4）聊天管理服务。聊天管理服务主要提供单聊、查看会话列表和查看历史聊天记录等功能。用户之间的聊天信息也会保存到该服务的数据库，然后用户可以随时查看历史聊天记录。

5）围棋引擎服务。围棋引擎服务主要管理围棋对弈信息。用户在进行围棋对弈时，来保存棋谱信息或者加载新的棋局，包括查看棋谱、保存棋谱、申请点目和认输等功能。用户在进行围棋对弈时，还可以查看当前局面的形势分析，用户可以动态调整对弈策略来获得对弈胜利。

根据以上描述，将整个系统可以划分为三层，即展现层、服务管理层和数据服务层。系统的分层结构图如图4-2所示：



图 4-2 系统分层结构图

展现层主要负责与用户进行交互，通过移动客户端提供界面来交互信息，用户通过该模块可以管理各种信息，包括用户个人信息、对弈信息、学校班级信息、聊天信息等。

服务管理层主要完成对客户端请求的处理以及后端微服务的处理，包括API Gateway的管理，具体为对客户端请求的路由转发、路由过滤、服务的注册与发现等。

数据管理层主要与数据库系统进行交互，负责对数据信息在数据库方面的管理，包括数据的持久化、实现不同类型系统的之间数据之间的转换，连接池的处理等。

### 系统功能模块结构

针对系统的需求分析，将整个服务端划分为三大模块：学校管理模块、围棋引擎模块和聊天管理模块，而每个大模块又会细分为各个小模块。

系统功能模块结构图如图4-3所示：



图 4-3 系统功能模块结构图

学校模块设计包括普通账号的注册和登录、第三方平台账号的登录、班级管理和发布作业；围棋引擎模块包括查看或保存棋谱、围棋对弈、申请和棋、申请点目、认输、查看围棋局面分析；聊天管理模块包括即时聊天、查看即时聊天记录；

## 系统主要功能模块设计

针对系统的总体设计，本节分别对系统的主要功能做详细的设计，首先介绍了系统总体架构，然后介绍了移动客户端，API网关系统，最后对服务端，包括对学校管理模块、围棋引擎模块、聊天管理模块的设计。

### 系统总体架构

从系统整体设计上来看，系统主要由移动客户端、API Gateway和服务端组成。整个服务端采用微服务化的设计模式，根据业务划分为三个微服务模块：学校管理、聊天管理和围棋引擎。系统总体架构如图4-4所示：



图4-4 系统总体架构图

在系统总体架构中，所有微服务实例都在API网关上注册，服务与服务之间通过service-id区分。整个服务端提供API网关作为内外的门户，外部客户端访问采用REST风格的HTTP请求，API网关具有服务注册发现和服务路由等功能。每个微服务都具有自身独立的技术架构与数据库存储，通过对外暴露服务接口进行模块之间信息与数据的传递与交流。每个微服务部署在一个Docker容器上。

### 移动客户端

本系统的客户端主要在Android平台开发，并通过REST HTTP与服务端进行通信。该项目使用分层逻辑，客户端主要是UI层，在UI层中实现Android平台上的GUI绘制、响应用户操作等功能。用户通过客户端进行操作输入，传递命令给服务端，然后服务端返回相应的操作命令给客户端，反馈给用户并等待用户的响应。

Android客户端开发采用MVC模式，视图层主要是指客户端UI，Android中与视图有关的布局文件都以XML的形式存储在res文件夹中。核心的UI控件有TextView、EditText、ListView等，还有一种视图表现方式是自定义View，本客户端的实现过程中自定义的View有CustomProgressDialog、HorizontalListView、XListView、PullToRefreshListView等UI控件。控制层由Activity来实现，所有Activity都由UI线程来保持，每个Activity用于处理视图层中的点击事件和更新。避免在Activity中实现复杂的业务逻辑和耗时操作，而这些操作是会单独开启新的线程调用模型层中的方法进行处理的，以这种分层结构实现低耦合，提高程序的健壮性和易维护性。由于Android中Layout种类繁多，导致很多数据都不能直接绑定上去，所以Android引入了适配器（Adapter）这个机制作为复杂数据的展示的转换体，各种Adapter就成了转换不同方式和能力的模型。客户端的整体框架设计核心类图如图4-5所示。



图4-5 移动客户端整体框架类图

BaseActivity类与Application类是系统的核心类。该移动客户端中作为一级菜单的三个Activity如“首页”界面的HomeActivity和“消息”界面的MessageRecordActivity等全部继承BaseActivity。BaseActivity中的customProgressDialog属性是自定义的一个View用来显示进度，并在该类中提供启动和关闭方法。

Application类是移动客户端应用程序最先加载的一个类，它采用单例模式进行设计，每个应用中只会存在一个唯一的Application类。Application属性用于存储、共享以及传递数据信息。Application和Activity、Service一样是Android框架的一个系统组件，它的生命周期在整个过程中最长，等于整个应用程序的生命周期。正因为它是全局的、单例的，所以在不同的Activity或者Service中获取的对象都会是同一个对象，这样就可以通过Application来实现程序中的数据传递、数据共享以及数据缓存等全局操作。本系统中的一个EasyApplication类继承自Application类，并且在Androidmanifest.xml中的application标签中进行注册，在该类中几乎定义了该应用中所有需要全局操作的方法和重要标识。为了减少网络的访问频率和流量消耗，在用户登录或者对弈时，会将一些重要的信息一次性加载到客户端，并存储到Application类中定义的相应数据类型中。Board类为加载棋盘信息，Color类加载棋子颜色信息等等。

### API网关系统

API Gateway（网关）是随着微服务概念而出现的架构模式，它主要用来解决微服务过于分散和统一进出口流量管理的问题。在微服务架构中，服务网关是整个系统架构唯一的入口，相对来说网关服务是一个是更基本的系统模块。API网关出现在系统边界，只提供访问的入口和串行集中式的强管控服务。它减少了客户端多次调用微服务，也可以在入口处进行负载或权限校验的处理。整个服务端提供API网关作为内部和外部的门户，并且客户端访问使用无状态REST风格的请求。API网关具有如下优点：

1）通过使用API网关模式，微服务变得更加轻量；

2）API网关可以提供非功能性的功能，比如安全和监控等；

3）在服务网关层级，可以进行轻量级的消息路由和消息转换；

4）在服务网关层级，可以为各个微服务提供一些抽象行为；

微服务架构中的服务网关具有重要的作用。API网关具有服务路由、请求过滤等功能，其包括的组件为Spring Cloud Zuul，微服务可以利用该组件来进行服务路由。Spring Cloud Zuul为微服务架构提供了入口，并且同时将权限控制这些比较重的非业务逻辑内容迁移到服务路由层面，使得服务集群主体能够具备更高的可复用性和可测试性。Spring Cloud Zuul网关实现了内部接口与外部的隔离，并提供统一入口和实现了路由功能来屏蔽诸多服务细节，另外网关还可以进行服务过滤。在服务网关中定义过滤器只需要继承ZuuIFilter抽象类实现其定义的四个抽象函数就可对请求进行拦截与过滤。在Zuul中定义了四种不同生命周期的过滤器类型，每个类型的描述如下所示：

1）pre类型在请求被路由之前调用。

2）routing类型在路由请求时候被调用。

3）post类型在routing和error过滤器之后被调用。

4）error类型处理请求时，如果发生错误则将会被调用。

Zuul的filterType生命周期具体如下图4-6所示。

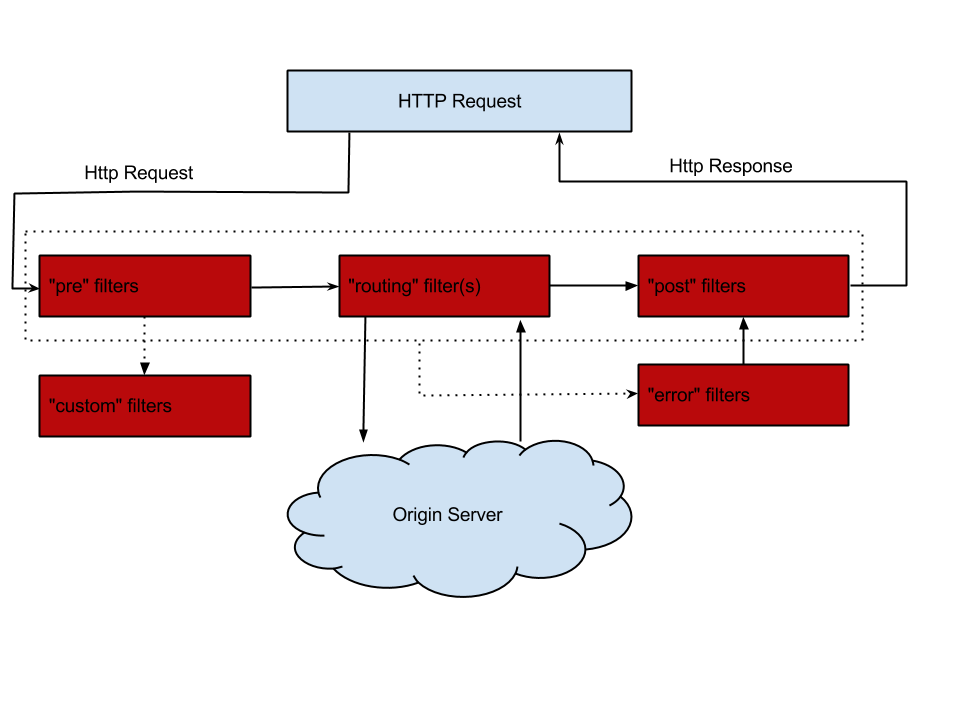


图4-6 Zuul的filterType生命周期

### 学校管理模块

学校管理模块的主要功能是用户的注册登录、班级管理。第三方平台账号的登录使用的是shareSDK提供的第三方jar包。第三方平台账号登录首先需要验证用户的第三方平台账号，验证通过后获取一个有时间限制的权限令牌，在这个有效时间内才能进行本系统的注册和登录功能。用户登录成功之后，就可以进行学校管理的操作。学校管理模块实现的整体类图如图4-7所示。



图4-7 学校管理模块结构类图

该模块的核心类为RegisterActivity、LoginActivity和ClassActivity，这三个类都继承自BaseActivity类，分别对应的功能为用户注册、登录和班级管理。RegisterActivity类中有普通账号注册和第三方平台账号注册，普通用户账号注册为用户直接输入账号和密码然后调用normalRegister()方法进行注册。第三方平台注册需要调用第三方平台jar包，然后输入账号密码调用threeRegister()方法进行注册。LoginActivity类中有普通账号登录和第三方平台账号登录，普通账号登录为用户直接输入账号密码点击登录，然后调用normalLogin()方法进行登录。第三方平台账号登录需要输入第三方平台的账号密码，然后调用threeeLogin()方法进行登录。ClassActivity类负责班级管理，主要用于用户信息管理和班级信息管理。

用户在进入到登陆界面后，可以选择普通账号登录或者选择第三方平台微博账号登录，只有通过第三方平台账号验证成功后才能进行登录。第三方平台账号第一次登录成功，并给验证后的账号一个有时间限制的权限令牌，在令牌有效期内不再需要平台账号验证，否则需要重新进行验证。用户在登录成功之后，通过不同的身份可以进行班级信息的维护。教师可以创建自己的班级，并允许学生的加入和管理。学生可以搜索班级信息，并选择自己感兴趣的班级加入。

学校管理的时序图如图4-8所示。



图4-8 学校管理时序图

学校管理具体步骤如下：

1）用户登录系统时，选择普通账号登录或者第三方平台账号登录；

2）普通账号登录，调用CheckUser()方法验证账号密码的正确性；

3）第三方平台账号登录，调用CheckUser()方法验证第三方平台账号的正确性；

4）用户登录成功以后，可以通过调用UserInfoImpl类的方法，得到或者修改用户的个人信息；

5）根据不同的身份，学生和教师分别调用ClassInfoImpl类的方法执行不同的班级管理操作；

### 聊天管理模块

聊天管理模块主要的功能是单聊、查看会话列表和查看历史聊天记录。单聊主要是用户之间可以进行实时聊天，并且需要保证消息收发的即时性。消息内容可以是文本、图片、语音等。语音和图片是通过先把发送的语音和图片文件存储到第三方服务器，该服务器会返回URL地址，然后本系统的服务端保存该地址。为了保证聊天的即时性，采用的是轮询机制。服务器会定时查看消息的收发内容等。查看会话列表是用列表的形式显示与所有用户历史聊天记录。查看历史聊天记录是用户点击某个会话列表，然后就进入到具体的聊天页面，下拉列表可以获取历史聊天记录，如果聊天记录过大，则会分页进行显示记录。聊天管理模块的整体类图如图4-9所示。

图4-9 聊天管理模块整体类图

聊天管理模块主要是单聊、查看会话列表和查看历史聊天记录。实体类包括、ChatData类、ChatInfoAction类、MessageRecordServiceImpl类、SingleChatServiceImpl类。SingleChatServiceImpl类的业务类包括SingleChatServiceImpl类、SingleChatInfoUtil类、MessageManager类、ChatDataServiceImpl。MessageRecordServiceImpl类的业务类包括MessageRecordDaoImpl类、MessageRecordAction类。接口类分别为SingleChatService、MessageRecordService和DataService。BaseDao接口是对业务类的抽象，主要是提取出业务类操作的共性，然后封装成接口，方便每个业务类实现具体的功能。MessageRecordDaoImpl类依赖于MessageRecordServiceImpl类和BaseDao接口，类似地，每个业务类依赖于实体类和BaseDao类，实现接口。

聊天管理时序图如图4-10所示：



图4-10 聊天管理时序图

聊天管理具体步骤如下：

1）用户登录系统；

2）用户登录成功后通过SingleChatService()方法和其他用户进行单聊，然后该系统会调用第三方平台的SDK进行发送聊天信息，发送成功后，返回具体信息；

3）该系统会通过SingleChatService()中的方法保存具体的聊天信息，比如url信息等；

4）用户也可以通过MessageRecordService()中的方法进行查看会话列表和历史聊天记录，在查看历史聊天记录时，直接查询本系统已经保存在数据库中的聊天信息；

5）系统获取聊天信息后，将聊天内容展示给用户；

### 围棋引擎模块

作为围棋教学中比较重要的部分，围棋引擎模块主要完成围棋对弈功能。围棋引擎用于实现围棋对弈的功能，并与客户端进行通信，具体的功能包括开始新的棋局、加载棋谱等。围棋引擎通过GTP（Go Text Protocol）协议与移动客户端进行通信，GTP是基于文本的消息传递协议，并具有简便可靠的特性。用户通过移动客户端进行围棋对弈功能，如开启新局、落子、认输等，客户端发送对应的GTP指令，服务端的围棋引擎收到指令并解析，然后进行相关的操作。用户如果保存当前棋谱，则围棋引擎模块会将当前的棋谱信息编码为SGF（Smart Game Format）文本字符格式，然后保存到当前服务器的数据库中[32][33]。

#### 1）GTP协议解析模块

目前计算机围棋程序间对弈大多数都采用GTP协议，所以本系统也同样采用GTP协议来实现围棋的通信，这样符合GTP协议的任何围棋引擎都可以与该系统进行链接，方便扩充对接其他开源的围棋引擎，比如以后可以对接人机对弈引擎来实现人机对弈功能。

围棋通信协议现在主要有两种：GMP（Go Modem Protocol）协议和GTP（Go Text Protocol）协议。早期计算机围棋就是使用基于GMP的信息交换协议。程序员只需要关注实际的功能，而不需要考虑绘制棋子，缩放棋盘和其他无关的事情。GTP协议是GNU GO 3.0引入的新接口，GTP协议相比于GMP协议的优点是非对称协议，更适合于双机通信，而且提供了更加简单有效的接口。GTP协议作为程序间通信方法，已经被越来越多的围棋程序采用。GTP协议既可以支持两个围棋程序的对弈，也可以支持两人利用计算机围棋软件对弈以及人和计算机之间的对弈[34][35]。

GTP协议采用的是应答模式，移动客户端和围棋引擎通过GTP协议进行通信，客户端向围棋引擎发送基于GTP协议格式的命名请求，然后围棋引擎根据该请求做出相应的反应，这样就实现了客户端与围棋引擎进行通信的功能。表4-1为GTP协议两个主要指令说明。

表4-1 GTP协议两个主要指令说明

|  |  |
| --- | --- |
| 命令请求 | 响应 |
| play b ab | =\n\n |
| genmove b | =ab\n\n |

genmove b表示请求获取当前围棋盘面下的黑子（b）的着子点的坐标，然后play b ab就是在ab坐标处着黑色（b）棋子。GTP协议对棋盘坐标的定义方式为：在19路棋盘中，棋盘坐标以字母和数字的组合来表示，横向从左向右分别是A-T（不包含I），纵向坐标从下至上分别是1-19。

#### 2）SGF棋谱解析模块

SGF（Smart Game Format）是现在最通用的棋谱文件格式，它是为了存储棋局对弈记录而设计的文件格式。SGF不包含二进制数据，是一种纯文本格式。它基于树形结构，从而可以方便的保存棋谱数据信息。本系统采用SGF格式来保存棋谱文件，当要打开棋谱文件，就可以按照SGF文件格式的规则进行过滤和解析。

SGF文件由节点组成并构造成对局树，SGF文件中节点有且只有一个父节点，每个节点可以有多个子节点。SGF文件可以存储实战变化图和围棋对局记录。节点是文件中最小的可见单元，用户可以沿着对局树一步步的查看节点。节点由属性组成，属性包含一定信息，例如属性W[ ]描述白子着子，B[ ]描述黑子着子，属性C[ ]包含评论文本。编辑棋谱有两种层次分别为增加或者删除节点、增加或者删除属性。一个着子是一个节点的一部分，而非节点是着子的一部分，着子通过属性来描述，而且节点可以包含多个属性信息。表4-2为SGF文件主要规则说明。

表4-2 SGF文件主要规则说明

|  |  |
| --- | --- |
| 规则名称 | 规则说明 |
| FF（） | 文件格式，SGF规范的版本管理该文件 |
| GM | 游戏类型（如属性值1表示19路围棋） |
| PB | 黑方的名字 |
| WR | 白方的等级 |
| HA | 障碍，预先放置在棋盘上的黑子，放置的位置使用AB属性进行标识 |
| KM | 目数 |
| RE | 结果使用格式“B+R”（黑中盘胜）或者“B+3.5”（黑胜3.5目） |
| RU | 规则集（如中国规则、日本规则） |
| SZ | 棋盘尺寸 |

### 围棋局面分析的影响模型

围棋通常按照棋盘上占领的地域多少作为判断胜负结果的标准，现在大多数围棋程序根据空目数的多少作为最佳棋步的评判标准。但是很多时候并不能直观的观察到棋子的领先数目。在对围棋当前局面进行分析时，一般采用影响模型来评估，并且利用影响模型将棋子向棋盘其他部分辐射的影响量化[22][23]。

棋子的作用不仅体现在其几个相邻点上，同样也不仅作用于接下来的一两步棋中，它的作用可以蔓延较远地域并持续很长时间，特别是在开局阶段和棋盘上比较广阔的地方。通常子力通过棋盘上每个点的辐射影响值的和对周边的点施加辐射影响，其中黑白子辐射为正负相反的值。子力辐射按距离的增加而单调递减。因此计算机围棋研究者们认为，应该为棋子所产生的影响建立可计算模型，从而可以将这种影响量化并进行评估的目的[24][25][26]。

棋子的影响模型通常是围棋战略决策的基础，局面的静态评估、棋子的安全性判断与分块、领地的计算与划分等一般都基于影响模型的计算结果。通常在影响模型中，多个棋子的作用被看作是单个棋子作用的叠加，而单个棋子的作用根据相对位置不同进行一定的修正，其中黑白子会产生正负相反的影响。

实地就是在棋盘上预期终局时，成为一方或者另一方实际地域的部分。和实地相对应的是厚势，厚势是一种围棋概念，它表明了每一方棋子对空点的控制潜力。一般职业棋手在围棋对局的过程中，需要计算厚势的价值，并且以此作为判断全局形势的基础。所以计算厚势价值也是计算机围棋发展中比较重要的部分，特别是在进行静态评估，然后量化获胜概率的时候，计算的厚势价值的准确性，将直接影响获胜概率的最终量化结果。人类棋手一般在计算厚势价值时依靠子效分析，或者有时源自于对围棋的感觉，或者对局经验的反馈，这种计算缺少规律性，所以计算厚势价值一般难度比较高。本系统提出了一种新的基于厚势价值的影响模型，可以用于分析围棋的局面形势。

#### 1）Zobrist的影响函数

Zobrist首次引入了影响函数[14][15]，并根据该方法将棋盘划分为黑方地域和白方地域。Zobrist影响函数通过计算围棋棋盘上每一个交叉点的数值，其中黑子取值+50、白子取值-50和空白点取值0。然后将正数效果的点给该点的相邻点进行加一，负数效果的点给该点的相邻点进行减一，该算法然后进行递归执行四次，将棋盘上的点最终数值化。Zobrist影响函数进而可以得出一个黑子形成，并对其周围辐射的影响域，该影响函数的影响模型具体如图4-11所示。

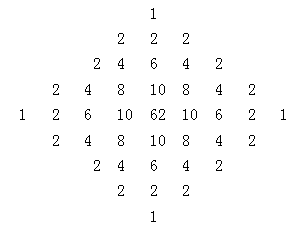


图4-11 Zobrist影响函数的影响域

#### 2）Ryder的影响函数

Ryder影响函数是在Zobrist影响函数的基础上，Ryder同样也是对棋盘上棋子产生的影响进行赋值[16]。Ryder影响函数也是将黑子取正值和白子取负值，然后围棋中某点影响的取值由其相邻点的影响传播进行线性累加，Ryder影响函数的传播系数固定和传播范围也没有太大。Ryder影响函数的影响模型具体如图4-12所示。

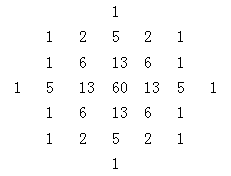


图4-12 Ryder影响函数的影响模型

Ryder通过结合影响函数和联络的概念来进行判断围棋局势，棋子联络的概念为：如果两个棋子相邻，那么这两个棋子就构成了联络；如果一个空点与某方中的至少一个棋子相邻，并且它不与另一方的任何棋子相邻，那么这个空点与这一方就构成联络，而与另一方就不构成联络。Ryder规定了一个阈值，如果一组有联络的空点所受的影响值大于或等于该阈值，那么它们就形成了棋势。

#### 3）陈克训的影响函数

陈克训对电脑围棋有着较深研究，他开发的围棋程序Go Intellect战斗力较强，在各项电脑围棋赛上屡获佳绩[17]。陈克训的影响模型中，每个棋子都要对盘面发出影响，这影响在棋子的紧邻（距离为1）为最大值m，并且随着距离增加而按比例衰减，衰减因子为f。围棋程序“探索者”选用、。 衰减因子为0.5就是距离每增加1时影响值减半。所以一个棋子对邻位的影响值为64，尖位和关位为32,小飞和大关位（拆二位）为16等。m值取为2的幂，而f取为0.5，就使各级影响值均为整数，避免了小数运算。围棋程序因为要计算很多问题，所以应该尽量节省计算量，因此通常都避免小数运算而只用整数计算。陈克训影响函数的影响模型如图4-13所示。

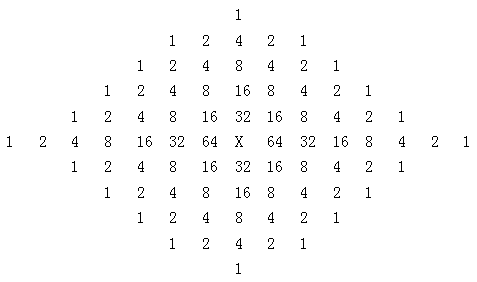


图4-13 陈克训影响函数的影响模型

陈克训主要应用影响函数对棋群进行分块，分块方式为：

（1）如果棋子紧密相连，并且颜色相同，则属于同一块。

（2）设定界限值a=16，影响值大于或等于16的空点，则看作被黑方所控制，影响值小于或等于-16的空点，则看作被白方所控制。如果某方的若干棋子和己方所控制的空位相连，那么这些棋子属于同一块。

（3）有时相同颜色的两串棋子间并没有足够大影响值的空位，但是无法分断的。 陈克训影响函数通过将相连的己方子与相连的己方控制点合在一起来形成棋块。但是如果只用影响值来确定棋块是一般是不全面的。存在这样一种情况，相同颜色的两串实际上不可分割，却并非通过具有足够大的影响值的空位相连。所以为了使分块方案更为完善，陈克训还补充了链的概念。一般链是不可分割的相同颜色串的整体。然而如何确定不可分割性却不简单。陈克训提出了实际上采用的直观推断准则。通过影响和链来做出分块，与人类棋手的直觉是颇为一致的。一旦分了块，其安全性就可用它的影响值和自由度来估计。

块的自由度意为它周围的敞开程度，它和逃出及利用周围敞开处做眼的难易程度有关。块中的影响值和自由度足够大则安全。如果不够大的可检查它能否做两眼，从两眼及逃出的难易程度来对安全程度作估计，并加以定量化而用数字0至64来表示：64为安全，较少的则不尽安全。安全性为0的即完全无望，也就是死块，其中的棋子和邻近的空位均可判为对方的地域。若我方某块不安全而非无望，即安全性小于64但大于某界限，就会有适当的防守着点使其加强。若某敌块不安全而非无望，就会有攻击点以图攻杀或欺凌该块而取利。即分块能用来产生攻防着点。

#### 4）GNU GO的影响函数

GNU GO是国外的一款开源软件，它的算法和源代码都是公开的，任何人都可以免费地用作研究和扩展。这种开源性为GNU GO带来了更有利的竞争优势，许多围棋研究者都已它为基础进行电脑围棋的开发[18]。

围棋棋盘中的黑子和白子对空点的影响总是不可避免，并且相互阻碍和干扰。过去的想法认为棋子的影响力是可以穿透的，也就是一个棋子对空点产生的影响可以穿透途中的其他棋子从而进行辐射，但这并不符合围棋的逻辑。

GNU GO影响函数的基本思想是把棋盘上所有活棋看成是影响源，然后影响会向各个方向进行辐射，并且影响值会随着距离的增大而减小，而且只有当前棋盘上没有棋子阻挡时才能无限扩张，不论是我方棋子还是对方棋子，所有活棋都视为障碍物，如果遇到障碍物，影响就不再扩张。GNU GO影响函数的基本过程就是为通过队列对邻近点和距离更远的点采用宽度优先搜索，同时结合模式匹配，就可以找到辐射时遇到的棋子障碍。

GNU GO影响函数的计算步骤如下:

设为影响源坐标，为辐射时遇到的空点，棋盘上任意一点都会对距离其最近的八个点和对角线方向上的点产生影响：

对于八个最近点的方向：

（1）首先计算向量的积：。

（2）如果积的计算结果为负数或零，则表明方向是在外部，从而继续另一个方向。

（3）如果在棋盘外部，或者已经被使用过，则继续寻找下一个方向。

（4）设S为空点的影响强度，那么从该点辐射至的影响强度计算公式为。

其中P、A和D分别为三种阻尼系数：P表示渗透度，它是棋盘上点的属性，通常为1，即无限制辐射。当需要停止辐射的时候，就需要在模式匹配时将该点的P值置为0；A表示衰减度，影响源在不同方向上衰减度的值也不同，默认为3，对角线方向上的衰减度默认为6，当改变影响源大小时，可以改变A的值；D表示方向阻尼，它是和的夹角余弦的平方，作用是阻止棋子产生的影响越过障碍物，并在该方向上继续辐射。在访问影响源的时候，令D值，即方向阻尼为1。

在计算棋子或者棋块对空点产生的影响时，每个棋子都被看作是一个独立的影响源，然后再分别计算1次，将每个值进行累加，最终所得到的和就是棋子或者棋块的影响值。在GNU GO的影响函数中，它主要用来计算实地，并确定棋块的稳定性和评估棋块的强弱等。

根据以上算法计算可以得到，强度等于100和衰减度的影响源形成的影响模型如图4-14所示，在实际的计算过程中，结果为浮点型，为了方便表示，所以进行了取整。

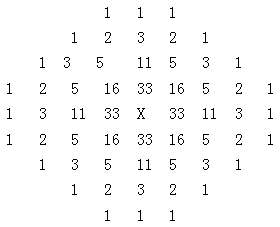


图4-14 GNU GO影响函数的影响模型

#### 5）基于厚势价值的影响函数

上面几种棋子的影响函数主要是用于数值化棋盘.、分块棋群和评估优劣等方面，并没有用于量化棋子的影响函数。其中Zobrist、Ryder和陈克训的影响模型简单，避免了小数运算，计算速度快，但是它们棋盘上各点的影响值是根据邻点的影响值叠加所得，并且距中心点更远的空点会对距中心点更近的空点造成影响，这些都不符合围棋逻辑。GNU GO考虑情况充分，计算精度较高，对于评估棋子或棋块的强弱和确定棋子或棋块的稳定性能够起到一定的作用，但是它认定“遇到阻挡活棋，影响辐射就不会再扩张”的说法不够全面，且各影响值均为浮点型，在计算P即渗透度的值时需要通过查找模式库，因此GNU GO的计算速度较慢。

由于上述四种影响函数都不是为计算厚势价值而设计，所以在计算围棋影响模型时存在缺陷，就像陈克训的影响函数，棋子的最大影响距离为7，这对围棋厚势来说相对较长。而GNU GO和Zobrist的影响函数中，棋子的最大影响距离为4。Ryder影响函数中，棋子的最大影响距离为3，相对较短。为了使厚势价值量化模型的精度更高，运算速度更快，因此需要设计符合厚势价值量化问题特殊需求的棋子影响函数。

根据围棋厚势的特性，总结出了围棋中棋子影响函数需要满足以下要求：

（1）越靠近围棋中心的空点，所受影响值应该越大。

（2）在远离围棋中心时，衰减度应该逐渐增大。

（3）经验表明，围棋中棋子最大影响距离为5时，函数影响模型的效果较好。

（4）为了提升运行速度，进行取整，删除没有用处的增加运算时间的步骤。

根据上述需求设计了基于围棋厚势价值的影响函数，该函数的具体规则如下：

（1）设是影响源坐标，并且影响强度为。则在第一象限中，位于X轴和Y轴各点的影响强度为：

（2）位于45度方向各点的影响强度为：

（3）棋盘上其余各点的影响强度为：

（4）以Y轴为对称轴，得到第二象限各点的影响强度：

（5）以X轴为对称轴，得到第三、四象限各点的影响强度为：

根据以上步骤可以得到：设影响强度，则基于围棋厚势的基础上，独立影响源形成的影响域如图4-15所示。图中各数值为了方便计算和表示，进行取整。

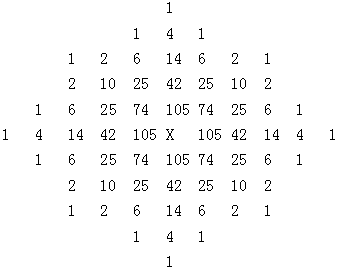


图4-15 基于厚势的影响函数的影响模型

基于围棋厚势的影响函数的基础上，建立围棋厚势价值的量化模型。该模型更加符合职业棋手的估值，构建步骤如下：

（1）计算厚势的有效子数为n，将气数大于或等于3气的棋子或棋串定义为厚势的有效子，并且计算它们的数量。

（2）计算阈值上限和阈值下限。

（3）然后计算归一化参数：。

（4）再计算围棋厚势影响域范围内每个空点受到的影响值。黑子产生正值影响，白子产生负值影响，每个空点所受到的影响值为所有棋子对该点影响的线性叠加。当计算黑棋厚势价值时，如果某点，令此点。计算白棋厚势价值时，如果某点，令此点。

（5）重复计算步骤4，将影响值进行线性叠加，得到围棋所有空点所受的影响值。

（6）由于围棋棋盘中空点所处的位置不同时，计算效率也不同。所以要进行边角调整：由于边和角更容易接受棋子的影响，所以要对一线和二线的空位的影响值进行修正，具体的修正算法为：。

（7）当某空点所受的影响值大于阈值上限的时候，即时，表示它所受棋子的影响很大，但影响值值再大，最多也只是成为某方的一目地，因此令。当某空点所受的影响值小于阈值下限的时候，即时，表示它所受棋子的影响很小，为了提高计算效率，令。当某空点所受的影响值介于阈值上限和下限之间的时候，即时，就令。

（8）计算围棋中所有空点所受棋子的影响值的总和。

其中N为空点总数。如果影响总值的值为正数，就表示黑方对棋子的影响总值更大，当前盘面黑子领先于白子。如果影响总值的值为负数，就表示白方对棋子的影响总值更大，当前盘面白子领先于黑子。

在本系统中基于厚势的影响函数更加符合职业棋手下棋的经验。基于厚势的影响函数类图如图4-16所示：



图4-16 基于厚势价值的影响模型类图

GroupValueInfo类是厚势价值信息类，ParameterValue类是参数类，该类记录了一些阈值参数，并通过该类方法进行获取或者设定阈值大小。EffectNumInfo是计算有效子数的信息类。GroupValueInflu类是厚势价值影响算法类，PointCoords类作为参数传递给GroupValueInflu类，通过调用GroupValueInflu类的getPointValue()方法得到空点的影响强度大小，然后调用ModifyCornerService边角类的getCornerInfo()方法进行修正，最后调用countAllValue()方法计算得到棋盘上所有空点影响强度的总值。

基于厚势价值的影响模型算法的时序图如图4-17所示。

图4-17 基于厚势价值的影响模型算法时序图

#### 6）算法的性能分析

本实验采用Intel(R) Core(TM) i7-4790处理器，模拟实验上述五种影响函数的性能。首先将职业棋手的判断作为空点所受影响值的理论值，即，然后通过基于厚势价值量化模型，将经过参数优化后的结果作为影响值的实验值，即。将各参数分别带入相对误差（公式4-1）、平均相对误差（公式4-2）和标准误差（公式4-3）的公式中进行计算这5组模型的各项参数，程序运行情况的结果如表4-3所示：

相对误差计算公式如下所示：

 （4-1）

平均相对误差计算公式如下所示：

 （4-2）

标准误差计算公式如下所示：

 （4-3）

表 4-3 五种模型的实验结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 模型 | 题目数量 | 平均相对误差 | 标准误差 | 平均解答时间/s |
| Zobrist影响模型 | 40 | 41.26% | 6.2861 | 0.0083 |
| Ryder影响模型 | 40 | 29.43% | 4.3672 | 0.0098 |
| 陈克训影响模型 | 40 | 21.39% | 2.7521 | 0.0099 |
| GNU GO影响模型 | 40 | 18.32% | 2.5937 | 0.0135 |
| 基于厚势价值影响模型 | 40 | 9.85% | 1.2095 | 0.0149 |

图4-18 五种模型的实验结果比较图

根据上面图表可知，基于厚势价值的影响函数模型的平均相对误差值和标准误差都大大降低，说明该影响模型的量化结果更为精确和可靠，从而说明了棋子基于厚势价值的影响函数量化问题中发挥了巨大作用。它模拟人类棋手的思维方式，将厚势的价值与棋力相关联，使其精度得到进一步的提升，更好的满足了围棋实战要求。

## 基于REST的通信接口设计

本系统中客户端与服务器有大量通信交互，这些通信借助HTTP协议，使用RESTful风格的通信接口，主要用于客户端和服务器交互类的软件。本文设计的基于HTTP协议的REST风格通信格式，用请求行标识服务资源的地址，基于JSON格式的数据的请求体和响应体来标识服务交互的主体，具有较高的处理速度和较强的通信能力。本文将系统中与主要功能相关的通信接口具体说明如表4-4所示。

表4-4 相关通信接口设计

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ServiceID | 名称 | 接口名 | 类型 | 参数类型 | 参数ID | 参数数据类型 |
| schoolservice | 登录 | UserLogin | POST | in | ID | int |
| in | password | String |
| out | message | String |
| 注册 | UserRegister | POST | in | type | String |
| in | password | String |
| in | name | String |
| in | gender | String |
| in | image | url |
| in | age | int |
| 创建班级 | TeahcerClass | POST | in | ID | int |
| in | classname | String |
| 申请加入班级 | StudentClass | POST | in | ID | String |
| in | name | String |
| in | message | String |
| 批准加入 | ClassOpera | PUT | in | studentID | String |
| in | classID | String |
| in | status | String |
| 发布作业 | TeacherWork | POST | in | teacherID | String |
| in | Work | String |
| 查询作业 | StudentWork | GET | in | teahcerID | String |
| in | classID | String |
| out | workInfo | String |
| chatservice | 单聊 | SingleChat | POST | in | fromuserID | String |
| in | touserID | String |
| in | chatInfo | String |
| 查看会话列表 | ConversationList | GET | in | userid | String |
| in | chatList | String |
| 查看历史聊天记录 | RecordList | GET | in | fromuserID | String |
| in | touserID | String |
| out | recordInfo | List |
| goengineservice | 查看棋谱 | ChessManual | GET | in | ID | String |
| out | ChessList | List |
| 保存棋谱 | SaveChess | POST | in | userID | String |
| in | SGFInfo | List |
| in | time | Datetime |
| 对弈 | PlayChess | POST | in | fromusetID | String |
| in | touserID | String |
| in | GTPInfo | List |
| 认输 | GiveUp | POST | in | userID | String |
| in | type | String |
| 申请点目 | ApplyCount | POST | in | userID | String |
|  |  |  | in | type | String |
| 查看局面分析 | SituationAnalysis | GET | in | userID | String |
|  |  |  | Out | GTPInfo | List |

## 系统的数据库设计

软件系统设计过程中比较重要的一节就是数据库设计，良好的数据库设计是系统实现的基础。数据模型设计包含了数据结构、数据操作和约束三个方面，其中数据结构为数据操作和约束提供基础，数据操作主要描述相应数据结构上的操作类型和方式，约束主要描述数据结构之内以及数据结构之间的依赖关系和数据的变化规则。数据模型设计一般先进行概念数据模型的设计，然后在概念数据模型的基础上考虑存储介质的组织结构，进一步完成物理数据模型的设计。本文将从概念数据模型和物理数据模型两个方面来阐述系统的数据库设计。

### 系统的概念数据模型

概念数据模型（CDM, Conceptual Data Model）是大多数开发者在进行数据库设计时首先创建的模型，它代表了整个数据库设计的最高层抽象。数据库设计离不开ER模型，而概念数据模型就是建立在传统的ER模型理论之上的。E-R图中有三大元素：实体型、属性和联系。其中实体型对应到CDM中的Entity，属性对应到CDM中每个Entity的Attribute，在概念上基本上是一一对应的。本文将利用概念数据模型来阐述本系统的数据库设计，如图4-19所示为系统概念数据模型设计。



图4-19 系统数据库E-R图

### 系统的物理数据模型

物理数据模型设计的目标是明确用什么样的数据库模式来实现概念数据模型，并进行数据的存储。物理数据模型的内容包含了确定所有的表和列和定义主键和外键等内容。下文将分别从字段名称、字段类型、字段约束等几个方面阐述本系统中主要的数据库表的设计。

系统需要的主要数据库表包括：user表，class表，board表，color表，messageRecord表，GTPInfo表，SGFInfo表，groupValueInfo等。

User表用户表，用于存放系统中用户的信息。字段包括姓名，密码，邮箱，角色ID，用户类型。

表4-5 User表清单

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 名称 | 数据类型 | 主键 | 外键 |
| id | 主键 | bigint | TRUE | FALSE |
| name | 姓名 | varchar | FALSE | FALSE |
| password | 密码 | varchar | FALSE | FALSE |
| email | 邮箱 | varchar | FALSE | FALSE |
| roleId | 角色ID | bigint | FALSE | TRUE |
| userType | 用户类型 | varchar | FALSE | FALSE |

class表是班级表，用于存储学校班级的信息。字段包括班级名称，班级类型，创建时间，描述，班级状态，用户ID。

表4-6 class表清单

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 名称 | 数据类型 | 主键 | 外键 |
| id | 主键 | bigint | TRUE | FALSE |
| name | 班级名称 | varchar | FALSE | FALSE |
| type | 班级类型 | int | FALSE | FALSE |
| createtime | 创建时间 | datetime | FALSE | FALSE |
| description | 描述 | varchar | FALSE | FALSE |
| status | 班级状态 | varchar | FALSE | FALSE |
| userId | 用户ID | bigint | FALSE | TRUE |

board表是棋盘表，用记录对弈时的棋盘信息。字段包括棋盘名称，对弈开始时间，对弈结束时间，对弈描述，棋盘大小，对弈A用户，对弈B用户。

表4-7 board表清单

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 名称 | 数据类型 | 主键 | 外键 |
| id | 主键 | bigint | TRUE | FALSE |
| name | 棋盘名称 | varchar | FALSE | FALSE |
| starttime | 对弈开始时间 | datetime | FALSE | FALSE |

表4-7 （续）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 名称 | 数据类型 | 主键 | 外键 |
| endtime | 对弈结束时间 | datetime | FALSE | FALSE |
| description | 对弈描述 | varchar | FALSE | FALSE |
| size | 棋盘大小 | int | FALSE | FALSE |
| userA | A用户 | bigint | FALSE | TRUE |
| userB | B用户 | bigint | FALSE | TRUE |

color表是棋子颜色表，用于记录棋盘的棋子信息。字段包括白子，黑子，X轴，Y轴，用户ID。

表4-8 color表清单

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 名称 | 数据类型 | 主键 | 外键 |
| id | 主键 | bigint | TRUE | FALSE |
| white | 白子 | int | FALSE | FALSE |
| black | 黑子 | int | FALSE | FALSE |
| pointX | X轴 | int | FALSE | FALSE |
| pointY | Y轴 | int | FALSE | FALSE |
| WuserId | 白子用户ID | bigint | FALSE | TRUE |
| BuserId | 黑子用户ID | bigint | FALSE | TRUE |
| description | 描述 | varchar | FALSE | FALSE |

messageRecord表是聊天记录表，用于存储用户的聊天记录信息，是聊天模块最重要的数据表。字段包括聊天内容，消息时间，消息类型，发送用户ID，接收用户ID，消息地址。

表4-9 messageRecord表清单

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 名称 | 数据类型 | 主键 | 外键 |
| id | 主键 | bigint | TRUE | FALSE |
| chatContent | 聊天内容 | varchar | FALSE | FALSE |
| chatTime | 消息时间 | datetime | FALSE | FALSE |
| chatType | 消息类型 | varchar | FALSE | FALSE |
| sendUserId | 发送用户ID | bigint | FALSE | TRUE |
| receiUserId | 接收用户ID | bigint | FALSE | TRUE |
| chatURL | 消息地址 | varchar | FALSE | TRUE |
| receiTime | 接收时间 | datetime | FALSE | FALSE |
| description | 描述 | varchar | FALSE | FALSE |

GTPInfo表是GTP协议信息表，用于存储对弈过程中的一些对弈信息。字段包括控制器，围棋引擎，X轴，Y轴，命令ID，命令返回，发送时间，描述。

表4-10 GTPInfo表清单

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 名称 | 数据类型 | 主键 | 外键 |
| id | 主键 | bigint | TRUE | FALSE |
| controlURL | 控制器 | varchar | FALSE | TRUE |
| engineURL | 围棋引擎 | varchar | FALSE | TRUE |
| pointX | X轴 | datetime | FALSE | FALSE |
| pointY | Y轴 | bigint | FALSE | FALSE |
| orderId | 命令ID | varchar | FALSE | FALSE |
| orderBack | 命令返回 | varchar | FALSE | FALSE |
| sendTime | 发送时间 | datatime | FALSE | FALSE |
| description | 描述 | varchar | FALSE | FALSE |

SGFInfo表是SGF解析信息表，用于存放保存或者解析SGF文件中的一些棋谱信息。字段包括对局ID，节点，子节点，属性，对弈时间，规则名称，规则描述，白方用户，黑方用户。

表4-11 SGFInfo表清单

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 名称 | 数据类型 | 主键 | 外键 |
| id | 主键 | bigint | TRUE | FALSE |
| gameId | 对局ID | bigint | FALSE | TRUE |
| node | 节点 | bigint | FALSE | FALSE |
| childNode | 子节点 | bigint | FALSE | FALSE |
| attribute | 属性 | varchar | FALSE | FALSE |
| gameTime | 对弈时间 | datatime | FALSE | FALSE |
| ruleName | 规则名称 | varchar | FALSE | FALSE |
| ruleDes | 规则描述 | varchar | FALSE | FALSE |
| whiteUser | 白方用户 | bigint | FALSE | TRUE |
| blackUser | 黑方用户 | bigint | FALSE | TRUE |

groupValueInfo表是基于厚势价值信息表，用于存放基于厚势价值信息影响算法中的一些信息。字段包括有效子数，影响强度，空点X轴，空点Y轴，中心点X轴，中心点Y轴，阈值上限，阈值下限，距离，描述。

表4-12 groupValueInfo表清单

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 名称 | 数据类型 | 主键 | 外键 |
| id | 主键 | bigint | TRUE | FALSE |
| effectNum | 有效子数 | bigint | FALSE | FALSE |
| strengthInflu | 影响强度 | double | FALSE | FALSE |
| blankX | 空点X轴 | int | FALSE | FALSE |

表4-12 （续）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 名称 | 数据类型 | 主键 | 外键 |
| blankY | 空点Y轴 | int | FALSE | FALSE |
| centerX | 中心点X轴 | int | FALSE | FALSE |
| centerY | 中心点Y轴 | int | FALSE | FALSE |
| thresholdUpper | 阈值上限 | double | FALSE | FALSE |
| thresholdLower | 阈值下限 | double | FALSE | FALSE |
| distance | 距离 | double | FALSE | FALSE |
| description | 描述 | varchar | FALSE | FALSE |

在数据库设计时，理论上需要遵循第三范式的数据库设计原则，但是由于项目的实际需要，往往会降低范式级别，增加冗余字段以提高査询效率和达到缩小响应时间的目的。

## 本章小结

本章对线上围棋教学系统的整体架构进行了设计，通过类图、活动图、时序图对各个功能模块，即移动客户端模块、学校管理模块、聊天管理模块、围棋引擎模块和围棋的影响模型模块进行了设计，对每个模块设计中的关键点进行了详细的描述，为系统的实现提供了依据，对系统合理的设计是系统实现的理论基础和前提条件。

# 基于微服务架构的线上围棋教学系统实现与测试

根据上一章对线上围棋教学系统的详细设计，本章介绍系统主要模块的实现，包括服务端的微服务部署、API网关的实现、学校管理模块的实现、聊天管理模块的实现、围棋引擎模块的实现，并对相应模块进行了功能测试，非功能测试，并且分析了测试的结果。

## 系统开发环境

表5‑1 系统开发环境

|  |  |
| --- | --- |
| 项目 | 内容 |
| 操作系统 | Ubuntu GNOME 14.04 |
| 服务端开发工具 | IntelliJ IDEA |
| 客户端开发工具 | Android Studio+SDK |
| 开发平台 | J2EE |
| 开发语言 | JAVA |
| 服务端数据库 | Mysql5.6 |
| 客户端数据库 | SQLite |
| 应用服务器 | Tomcat7 |
| Web框架 | spring+springMVC+mybatis |

## 系统的实现

### 服务端微服务部署

单体式应用在漫长的发展过程中，其局限性也越来越突出，一个简单的单体式应用随着时间的推移会逐渐变大，由于应用的更新和新的需求，软件需要加入新的代码和数据。从而简单的小应用就会变得复杂，从而导致开发者维护越来越困难。因此本文提出在服务端采用微服务架构来解决上述问题。首先对微服务的设计模式进行选择，然后提出结合容器技术实现微服务的部署与编排。

对微服务进行划分实现之后，需要对微服务进行部署。通常一个应用由多个微服务构成，每个服务都是一个单一的应用，可以进行独立部署，并且拥有独立的资源等，所以应用中微服务的部署必须要快速、可靠和性价比高。微服务的部署策略包括：单主机多服务实例模式和单主机单服务实例模式，使用单主机多服务实例模式可以快速部署微服务，并且资源利用的有效性较高，但是使用该模式最大的问题就是各服务之间的隔离性不好，而当使用单主机单服务实例模式时，服务实例之间都是相互独立的，所以本系统选择单主机单服务实力模式来部署微服务。

单主机单服务实例模式包括两种：单虚拟机单实例模式和单容器单实例模式。使用单虚拟机单实例模式通常是将服务打包成虚拟机映像，每个服务实例是一个使用此镜像启动的虚拟机。这种部署模式具有如下好处:

1）这种部署模式使得每个服务都完全独立，且都拥有各自独立的CPU和内存，不会被其他服务占用。

2）用户可以使用成熟云架构，例如阿里云提供的云服务，该服务提供负载均衡和扩展性等功能。

3）服务实施技术被各自包含，如果服务被打包成虚拟机映像，则就成为了一个黑盒子，虚拟机映像的管理API就变成了部署服务的API，部署成为一个非常简单和可靠的事情。

但是单虚拟机单实例架构模式也存在一些缺点，具体如下所示:

1）资源利用效率不高。每个服务实例占有整个虚拟机的资源，而且在一个典型的公有IaaS环境中，虚拟机资源都是标准化的，有可能未被充分利用。

2）部署服务新版本比较慢。虚拟机镜像因为大小原因创建起来比较慢，同样虚拟机初始化速度也比较慢。

3）需要强大的运维团队。对虚拟机的创建和管理都是很繁琐的，因此需要投入很多时间和人力从事与核心业务不太相关的工作。

综上所述我们在实际微服务架构中使用另外一种部署模式，就是结合Docker容器技术实现单容器单实例模式的微服务部署模式。在单容器单实例模式中，每一个微服务都运行在各自的容器中。从进程的角度看，每个容器都有各自的根文件系统和命名空间，可以限制容器的CPU和内存资源，所以在同样配置的虚拟机上可以运行多个容器实例，从而使得资源利用率得到最优化。

为了实现微服务的集群化部署和管理，使用docker compose可以快速的实现微服务模块的编排和部署，从而使得一个微服务模块可以进行快速的弹性伸缩，本文使用docker compose实现微服务水平扩展的实现方式，docker compose是用来定义和运行多容器应用的工具，使用docker-compose.yml来描述多容器的定义，通过使用下面命令运行整个应用，主要代码如下：

//首先下载docker compose

curl -L https://github.com/docker/compose/releases/download/1.1.0/docker-compose-

‘uname -’-’uname -m’>/usr/local/bin/docker-compose

//增加执行权限

chmod +x /usr/local/bin/docker-compose

sudo mkdir compose&&cd compose

sudo touch docker-compose.yml&&sudo vim docker-compose.yml

//编辑docker-compose.yml

discovery:

image: "discovery:1.0.0-SNAPSHOT"\

hostname:discovery

ports:"8761:8761"

volumes:

一/www: /usr/share/nginx/html: ro

schoolservice:

image: "schoolservice:1.0.0-SNAPSHOT"#使用学校管理微服务镜像

hostname: schoolservice

links:

-discovery

-config

environment:

EUREKAes HOST: discovery

EUREKA PORT: 8761

SPRING PROFILES ACTIVE: docker

ports:

-"8081:8081"

chatservice:

image: "chatservice:1.0.0-SNAPSHOT"#使用聊天管理微服务镜像

hostname: chatservice

links:

-discovery

-config

environment:

EUREKAes HOST: discovery

EUREKA PORT: 8761

SPRING PROFILES ACTIVE: docker

ports:

-"8082:8082"

goengineservice:

image: "goengineservice:1.0.0-SNAPSHOT"#使用围棋引擎微服务镜像

hostname: goengineservice

links:

-discovery

-config

environment:

EUREKAes HOST: discovery

EUREKA PORT: 8761

SPRING PROFILES ACTIVE: docker

ports:

-"8083:8083"

//environment就是给容器使用的变量，在容器中使用${}来调用

//links表示当前容器依赖的容器，可直接使用依赖容器的已有端口

//ports表示将要暴露的端口映射出来，不需要暴露的端口则不做映射

//将docker-compose.yml上传到Linux服务器，在文件当前目录下执行下面命令

//启动docker compose

docker-compose up -d

完成部署微服务模块，可以通过docker ps命令查看各个微服务模块的当前运行状态。如图5-2所示，当前所有的微服务：学校管理(schoolservice)、聊天管理(chatservice)和围棋引擎(goengineservice)均处于up状态，说明服务端的微服务运行正常。



图5-2 服务端微服务正常运行效果图

### API网关的实现

API网关的主要功能就是进行服务路由、服务过滤等，并提供整个系统的唯一入口，对外可以屏蔽内部的具体实现，它封装了各个应用的内部结构。移动客户端只需要同API网关进行交互，而不必去调用特定的服务。通过服务网关中的过滤功能，在各生命周期中去校验请求的内容。为了保证微服务的无状态性，将在对外服务层进行的校验前移。与此同时降低了微服务的测试难度，让各个微服务更关注自身业务逻辑的处理。创建API Gateway的module，需要引入spring-cloud-starter-zuul。API网关具体的核心代码如下：

apply plugin: 'org.springframework.boot'

dependencyManagement {

imports {

mavenBom "org.springframework.cloud:spring-cloud-dependencies:"+ springCloudVersion

mavenBom "org.springframework.boot:spring-boot-dependencies:"+ springBootVersion

mavenBom "org.springframework.boot:spring-boot-starter:"+ springBootVersion

}

}

dependencies {

compile('org.springframework.cloud:spring-cloud-starter-config')

compile('org.springframework.cloud:spring-cloud-starter-eureka')

compile('org.springframework.cloud:spring-cloud-starter-zuul')

compile('org.springframework.cloud:spring-cloud-stream')

compile('org.xerial.snappy:snappy-java:' + snappyVersion)

compile ('org.springframework.boot:spring-boot-starter-web')

testCompile('org.springframework.boot:spring-boot-starter-test')

testCompile group: 'junit', name: 'junit', version: '4.12'

}

configurations {

all\*.exclude module: 'spring-boot-starter-logging'

all\*.exclude module: 'logback-classic'

all\*.exclude module: 'log4j-over-slf4j'

all\*.exclude module: 'snappy-java'

}

sourceSets {

main {

resources.srcDirs = ['src/main/resources', 'src/main/java']

resources.includes = ['\*\*/\*.xml', '\*\*/\*.yml']

}

}

jar {

baseName = 'apigateway-boot'

}

1）API 网关的服务路由

在服务网关的路由上，Spring Cloud Zuul具有很强大的功能。通过服务路由的功能，系统在对外提供服务的时候只需要公开Spring Cloud Zuul配置的调用地址，然后就可以让调用方采用统一的方式来访问系统服务，从而可以屏蔽具体提供服务的主机实例信息。URL直接映射的YML配置代码如下：

//学校管理服务

zuul:

routes:

auth:

path:/school/\*\*

serviceId:schoolservice

sensitiveHeaders:Cookie,Set-Cookie

//聊天管理服务

zuul:

routes:

auth:

path:/chat/\*\*

serviceId:chatservice

sensitiveHeaders:Cookie,Set-Cookie

//围棋引擎服务

zuul:

routes:

auth:

path:/go/\*\*

serviceId:goenginelservice

sensitiveHeaders:Cookie,Set-Cookie

该配置定义了所有到Spring Cloud Zuul的路由规则为：/school/\*\*的访问都映射到serviceId为schoolservice学校管理的微服务实例上，也就是说当客户端访问/school/\*\*的时候，Zuul会将该请求路由到schoolservice学校管理服务的实例上，即由学校管理模块来处理用户的学校管理方面的请求。同理可以得到chatservice聊天管理模块和goengineservice围棋引擎模块的路由规则。sensitiveHeaders用来配置REST API无状态特性，防止将authorization的头部带到目标服务。

2）API网关的服务过滤

除了服务路由之外，系统对外开放服务还需要一些保护措施来确保客户端只能访问特定的资源。所以需要利用Spring Cloud Zuul的过滤器来实现对外服务的安全控制。在服务网关中定义过滤器只需要继承ZuuIFilter抽象类，然后实现其定义的四个抽象函数就可以对请求进行拦截和过滤。ZuuIFilter抽象类中的四个抽象函数如下：

（1）filterType()函数表示返回一个字符串代表过滤器的类型，也就是上述提到的pre，routing， post，error四种类型。

（2）filterOrder()函数表示通过整型值来定义过滤器的执行顺序。

（3）shouldFilter()函数表示返回一个boolean类型来判断该过滤器是否要执行，所以通过此函数可实现过滤器的开关。

（4）run()函数表示过滤器的具体逻辑，一些自定义的业务逻辑在此方法内实现。

本项目中自定义过滤器主要核心代码如下所示:

@Override

public Object run(){

RequestContext ctx=RequestContext.getCurrentContext();

HttpServIetRequest request=ctx.getRequest();

//定义规则，访问URL中需要有accessToken参数

Object accessToken=request.getParameter("accessToken");

if(accessToken == null){

ctx.setSendZuuIResponse(false);

ctx.setResponseStatusCode(401);

return null;

)

return null;

上面的代码定义了一个Zuul过滤器，重写了run()函数，实现了在请求被路由之前检查请求中是否有accessToken参数，如果有就进行路由，若没有就拒绝访问，返回401 Unauthorized错误提示。本系统是通过ctx.setSendZuuIResponse(false)令网关拒绝该请求，并通过ctx.setResponseStatusCode(401)设置了其返回错误码，同样可以通过ctx.setResponseBody()对返回body内容进行编辑，然后进一步优化返回的提示信息等。

### 学校管理模块的实现

学校管理模块主要用户的注册登录和班级管理，用户的个人信息和班级信息都保存在学校管理模块的数据库中。

1）用户的注册和登录

首先用户账号分为普通用户和第三方平台的账号。普通用户登录和注册使用界面LoginActivity来完成，第三方平台的注册与登录使用第三方平台ShareSDK继承的SDK中的界面ShareSDKUIShell进行第三方平台账号验证，然后返回一个权限令牌，就可以完成注册和登录功能，获取第三方平台账号验证授权核心代码如下：

public static void authorize(Platform plat){

if(plat == null){

return;

}

if(plat.isValid()){

String userId = plat.getDB().getUserId();

Log.e(“userId------>”, plat.getDB().getUserIcon());

if(userId != null && !userId.equals(“”)){

handler.sendEmptyMessage(MSG\_USERID\_FOUND);

login(plat.getName(), userId, plat.getDB().getUserIcon(), null);

return;

}

plat.setPlatformActionListener(this);

plat.SSOSetting(true);

plat.showUser(null);

}

}

获取第三方平台账号登录授权后，调用第三方平台注册接口将用户的账号信息注册到本服务器上，注册成功后就可以自动登录到本系统中。用户在获取第三方平台给的限时权限令牌后，才能进行注册和登录。所有的操作都是在权限令牌有效的情况下进行的。第三方平台登录核心代码如下：

Handler handler = new Handler(){

public void handlerMessage(Message msg){

if(msg.content == MSG\_AUTH\_COMPLETE){

login(platformsh.getName(), platformsh.getDb().getUserId(), reses);

}

}

}

public static void login(String userId, String plat, HashMap<String, Object> userInfo){

if(!plat.equals(“ThirdParty”)){

return;

}

AppContext.uid = userId;

AppContext type = plat;

if(userInfo != null){

//调用第三方平台注册接口

thirdPartyRegister(userId, (String)userInfo.get(“password”), type);

}else{

//调用第三方平台登录接口

thirdPartyLogin(userId, type);

}

}

用户的注册和登录界面的运行效果图，如图5-3所示：

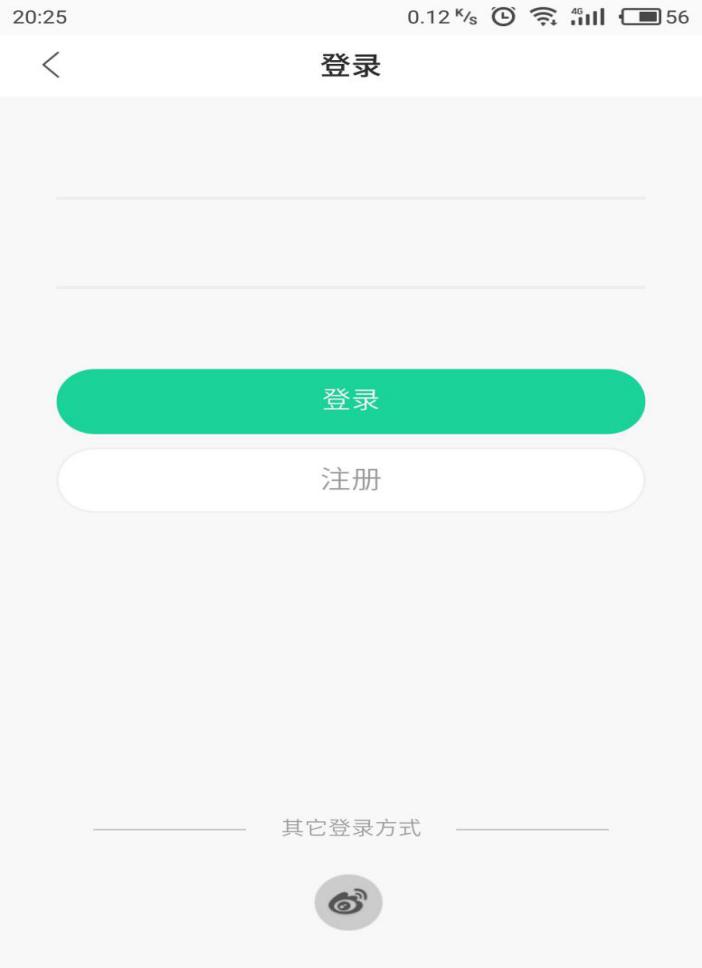


图5-3 用户的注册与登录界面

2）班级管理

班级管理模块中，用户在登录成功以后，可以管理自己的个人信息，以及学生与教师管理班级信息，比如教师在登录成功后，可以创建班级，然后学生批准加入班级，教师批准后，再进行管理班级的学生信息。以及教师可以在班级中发布作业信息，方便学生查看。班级管理的部分代码如下：

//修改用户个人信息

public static void setUserInfo(String userId, UserInfo userInfo){

UserInfo dbUser = getUserDAO(userId);

If(!userInfo.getUserName.equals(“”)){

dbUser.setUserName(userInfo.getUserName);

}

If(!userInfo.getUserPassword.equals(“”)){

dbUser.setUserPassword(userInfo.getUserPassword);

}

If(!userInfo.getUserImage.equals(“”)){

dbUser.setUserImage(userInfo.getUserImage);

}

If(!userInfo.getUserEmail.equals(“”)){

dbUser.setUserEmail(userInfo.getUserEmail);

}

modifyUserInfo(dbUser);

}

//教师管理班级信息

public static void teacherManageClass(Student stu, ClassInfo class){

//如果当前班级ID不存在，则创建该班级

if(getClassDAO(class.getClassID) == null){

createNewClass(class);

}

if(verify(stu, class.getClassID) == true){

return;

}else{

approveJoin(stu, class);

}

}

//学生申请加入班级

public static String studentManageClass(Student stu, ClassInfo class){

if(getClassDAO(class.getClassID) == null){

return “false”;

}

if(applyToJoin(stu, class.getClassID) == true){

return “success”;

}else{

return “false”;

}

}

//教师在班级中发布作业信息

public static String assignHomeWork(Teachert teacher, HomeWork work){

if(getTeacherDAO(teacher.getUserID) == null){

return “false”;

}

//检查该教师新发布的作业标题是否已经存在

if(checkWorkDAO(work.getWorkTitle) == true && getTeacherDAO(teacher.getUserID) == true){

return “false”:

}

if(assignHomeWork(teahcer.getUserID, work) == true){

return “success”;

}else{

return “false”;

}

}

学校管理界面的运行效果图，如图5-4所示：



图5-4 学校管理界面

### 聊天管理模块的实现

聊天管理模块主要是提供用户和用户之间进行即时聊天、查看会话列表以及查看历史聊天记录功能。聊天的内容可以是文字、表情、图片、语音形式。用户登录成功后，有两种方式可以进行单聊，第一种是在通讯录界面，点击某个用户，然后可以发起聊天。第二种是用户消息界面，点击一个历史会话列表，然后可以进行聊天。聊天模块接收消息采用轮询机制，方便及时接收用户的聊天信息。服务端在接收到第三方服务推送来的聊天记录时，会同时保存到本服务器的数据库中，方便下次直接查询。聊天管理模块，客户端发送聊天信息的核心代码如下：

public static void postMsg(String iUid, String iTuid, String sContent, String iType, String width, String heigth, String seconds){

String url = MyProperUtil.getProperties(context, “appConfigHost.properties”).getProperty(“Host”)

+MyProperUtil.getProperties(context, “appConfig.properties”).getProperty(“createmsginfo”)+”&data”

+makeSingleParam(iUid, iTuid, sContent, iType, seconds, “0”,”0”, width, heigth);

DhNet net = new DhNet(url);

//添加系数

/net.addParam(“data”, makeParameters(uid, type, page));

Net.doGet(new NetTask(context){

@Override

Public void doInUI(Responce responce, Integer transfer){}

});

}

聊天管理模块中用户收到消息后会进行提醒用户，用户A和B在聊天时，如果用户B不在当前系统，系统会提醒用户有未读的消息。如果用户在系统，但是不在消息界面，“消息”按钮会进行红点提醒，然后用户就会知道有新的未读消息。并且会同时以通知栏进行消息推送，如果用户在聊天界面，会直接将信息展示到聊天界面。消息接收通知栏提醒并展示的方法showCustomNotification()需要参数Context和Intent数据意向。聊天管理后台服务端创建消息接口实现的核心代码如下：

public static function createMsgInfo(){

da = newArray();

if(isset(this.data[‘msg\_to’]) && isset(this.data[‘msg\_from’])){

//获取请求参数

getArrayFromServer(“MSG\_SERVER\_URL”);

}else{

return array(“参数不正确”);

}

msginfoService = this.serviceLocator.getService(“msginfo”);

result = msginfoService.createMsgInfo(da);

if(result){

ret[‘code’] = cls\_config\_ErrorCode.code\_num[‘EN\_SUCCESS’];

ret[‘message’] = “success”;

ret[‘item’] = result;

}else{

ret[‘code’] = cls\_config\_ErrorCode.code\_enum[‘EN\_FAIL’];

ret[‘message’] = “fail”;

}

return result;

}

聊天管理模块的主要运行效果图，如图5-5所示：



图5-5 聊天管理界面

### 围棋引擎模块的实现

围棋引擎模块主要完成围棋对弈功能，是围棋教学中比较重要的部分。围棋引擎用于控制所有围棋对弈的功能，包括围棋吃子、落子等[20][21]。这些围棋基本功能在GNU GO开源软件中都已经有了实现，本系统采用了GNU GO的算法。本系统中的围棋引擎通过GTP协议与移动客户端进行通信，它是基于文本的消息传递协议，具有简便可靠的特性。用户通过客户端进行对弈，如开启新局、落子、认输等，客户端发送相关GTP指令，服务端的围棋引擎收到指令并解析，然后进行相关的操作。用户如果保存当前棋谱，则围棋引擎模块会将当前的棋谱信息编码为SGF文本字符格式，然后保存到当前服务器的数据库中。

1）GTP协议解析模块

本系统采用GTP协议来实现围棋的通信，这样符合GTP协议的任何围棋引擎都可以与该系统进行对接，方便扩充连接其他开源的围棋引擎[19]。GTP协议解析模块主要为围棋对弈时，移动客户端和服务端进行通信，发送基于GTP协议的通信数据，然后服务端进行解析。GTP协议解析模块的核心代码如下：

public class GTP{

private int boardsize;

private double komi;

private static final int PLAYOUTS\_NUM = 5000;

private final String[] commands = {"boardsize",

"list\_commands",

"name",

"play",

"clear\_board",

"quit",

"genmove"};

private Montecarlo mc;

public GTP(){

boardsize = 19;

board = null;

}

public void start(){

String input;

BufferedReader stdIn;

Point p;

int stoneType = Board.ENEMY;

long startTime;

Montecarlo.Move aiMove = null;

String humanMove;

boolean aiPasses = false;

try{

stdIn = new BufferedReader(new InputStreamReader(System.in));

while (true){

input = stdIn.readLine();

if (input.equals(" name")){

continue;

}

if (input.equals("quit")){

break;

}

if (input.startsWith("clear\_board")){

board = new Board(boardsize);

continue;

}

if (input.startsWith("genmove")){

if (input.split(" ")[1].equals("b")){

mc = new Montecarlo(board, Board.ENEMY);

stoneType = Board.ENEMY;

}

if (input.split(" ")[1].equals("w")){

mc = new Montecarlo(board, Board.FRIENDLY);

stoneType = Board.FRIENDLY;

}

for (int i = 0; i < PLAYOUTS\_NUM; i++){

mc.playOneSequence();

}

aiMove = mc.getWinner();

if (aiMove == null || aiPasses == true){

aiPasses = false;

continue;

}

board.makeMove(aiMove,stoneType);

if (aiMove.getMoveScore() < 0.25){

continue;

}

continue;

}

if (input.startsWith("play")){

humanMove = input.split(" ")[2];

if (humanMove.equals("pass")){

if (aiMove != null && aiMove.getMoveScore() > 0.7){

aiPasses = true;

}

continue;

}

p = strToPoint(humanMove,board.getSize());

if (input.split(" ")[1].equals("b")){

board.makeMove(p, Board.ENEMY);

}

if (input.split(" ")[1].equals("w")){

board.makeMove(p, Board.FRIENDLY);

}

continue;

}

}

}

}

}

2）SGF棋谱解析模块

SGF是为了存储棋局对弈记录而设计的一种文件格式，它是纯文本格式并且基于树形结构，所以可以方便的保存棋谱数据。本系统就是采用SGF格式来保存棋谱，当获取到棋谱的所有内容时，即可以按照SGF文件格式，按照一定的规则进行过滤和解析。另外在本系统中，从网上找到一些经典棋谱，已经以SGF文件的格式保存到数据库中，用户可以直接进行查看。也可以在对弈时，保存自己的棋谱。SGF核心代码如下：

//解析sgf文件的根节点

public SGFTreeNode parse(String sgf) throws Exception {

SGFTreeNode root = null;

int pos = 0;

while (pos < sgf.length()) {

char c = sgf.charAt(pos);

if (c == '(') {

root = new SGFTreeNode();

root.type = SGFTreeNode.TYPE\_PARENT;

parseInternal(root, sgf, pos);

break;

}

pos++;

}

return root;

}

//从根节点解析sgf文件

public int parseInternal(SGFTreeNode node, String sgf, int startPos){

String currentPropertySet = "";

boolean foundNodeClose = false;

int pos = startPos + 1;

boolean insideValue = false;

boolean previousWasEscape = false;

while (!foundNodeClose && pos < sgf.length()) {

//调用parsePropertySet()函数遍历解析sgf字符串

}

return pos - 1;

}

//对弈时，保存新的SGF文件，新的棋谱存入到数据中

public void saveSGF(String sgfName, String sgf){

SGFInfo sgfInfo = new SGFInfo();

SGFTreeNode node = parse(sgf);

sgfInfo.SGFTreeNode = node;

sgfInfo.SGFName = sgfName;

insertIntoFile(sgfInfo);

}

SGF棋谱界面的运行效果图，如图5-6所示：



图5-6 SGF棋谱模块界面

### 围棋局面分析的实现

围棋通常按照棋盘上占领的地域多少作为判断胜负结果的标准，通常围棋程序根据空目数的多少作为最佳棋步的评判标准，但是很多时候棋子的领先数目并不能直观的观察到。棋盘上预期终局时，成为一方或者另一方实际地域的部分成为实地。与实地相对应的是厚势，厚势是一种围棋概念，它表明了每一方棋子对空点的控制潜力。职业棋手在对局的过程中，需要计算厚势的价值，并且以此作为判断全局形势的基础。所以计算厚势价值同样是计算机围棋发展中至关重要的一节，尤其是在进行静态评估，并量化获胜概率的时候，厚势价值计算的是否准确，将直接影响获胜概率的最终量化结果。本系统提出了一种新的基于厚势价值的影响模型，可以用于分析围棋的局面形势。棋子的影响模型往往是围棋战略决策的基础，局面的静态评估，棋子的安全性判断与分块、领地的计算与划分等通常都基于影响模型的计算结果。通常在围棋影响模型中，多个棋子的作用被看作是单个棋子作用的叠加，而单个棋子的作用根据相对位置不同作一定的修正，黑白棋子会产生正负相反的影响。

本系统中提出的基于厚势价值的影响模型，更加符合职业棋手的估值。首先，基于职业棋手的判断和大量的下棋经验，棋子最大影响距离为5时，函数影响模型的效果最好。在基于厚势的影响函数的基础上，建立了围棋厚势价值量化模型。然后计算每个空点所受的影响值，由于空点的位置不同，所以要边角调整，然后进行修正。最后计算总的影响值之和。基于厚势价值的影响模型算法的核心代码如下：

public Map getInfluence(Blank blank, Center center){

//棋子最大影响距离

int maxDistance = 5;

int decay = 0;

if(blank.size()<=0 || center.size()<=0){

return null;

}

Map influInfo = new HashMap<Point, Integer>();

Point point = new Point<>();

point.put(blank.getPoint);

//分别计算X轴和Y轴的影响强度

int StrengthX = getStrengthInflu(blank.getPointX, center);

int StrengthY = getStrengthInflu(blank.getPointY, center);

//计算其他点的影响强度

int OtherStren = getStrengthInflu(blank.getPoint, center, StrengthX, StrengthY);

int quadrant = getStrengthInflu(OtherStren);

map.put(point,quadrant);

return map

}

public Model getQuantizedModel(PointList plist, int effectNum, Center center) {

Model model = new Model();

int allInfluValue = 0;

//分别计算得到阈值上下限

int threUpper = getThreUpper();

int threLower = getThreLower();

//计算归一化参数

int parameter = getNormalPara();

for(Point point : plist){

Map<> map = getInfluence(point, center);

int value = map.get(point);

if(value > threUpper){

//如果空点影响值大于阈值上限，修改该值

modifyStren(value,point,threUpper);

}

if(value < threLower){

//如果空点影响值小于阈值下限，修改该值

modifyStren(value,point,threLower);

}

allInfluValue += value;

}

model.setValue(allInfluValue);

model.setResult(true);

return model;

}

围棋的局面分析界面的运行效果图，如图5-7所示：

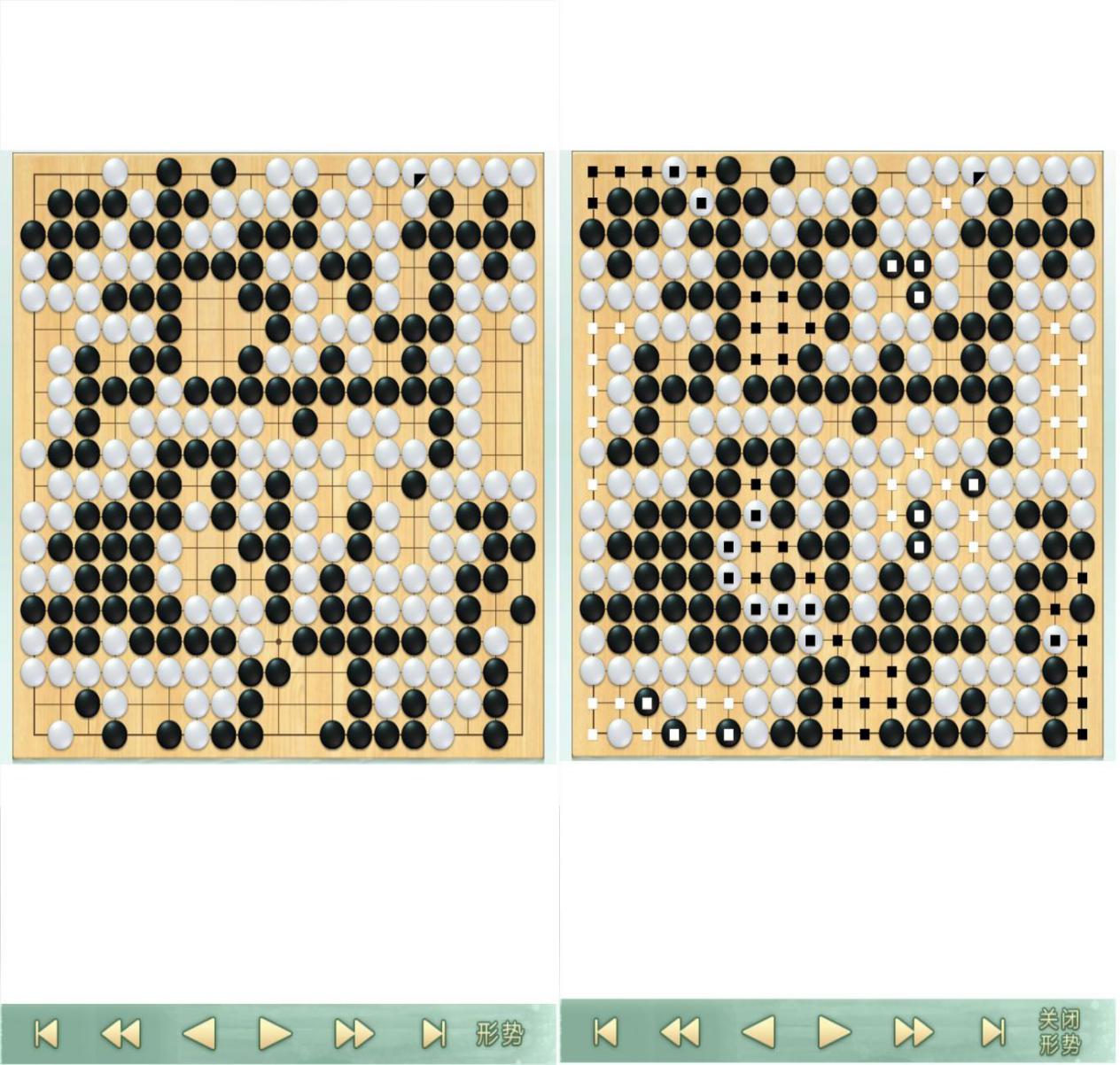


图5-7 围棋的局面分析模块界面

## 系统的测试

本章节主要介绍线上围棋教学系统的测试，系统测试是保障系统能够正常运行的关键。测试的主要目的是为了保证代码的正确性和功能的完善性，测试该系统能否达到设计的预期。所以本节首先介绍了系统测试的环境，然后在测试环境下，对线上围棋教学系统的各个功能以及非功能进行了测试，最后给出测试结果的具体分析并得出结论。

### 系统测试环境

本文服务端的测试环境搭载在阿里云上，一共使用了2台服务器，一台作为API网关的服务器，另外一台服务器构建Docker容器来部署微服务。系统的测试环境必须包含能够正常运行系统所具备的硬件、软件和网络环境。系机器的详细配置和功能角色如下：

1）移动客户端测试环境

表5-2 移动客户端测试环境

|  |  |
| --- | --- |
| 手机型号 | 小米手机6 |
| 运行内存 | 4GB |
| CPU型号 | 高通骁龙835 |
| CPU频率 | 2.45GHz |
| 机身存储 | 64GB |
| 操作系统 | Android系统7.1.1 |
| 通讯类型 | 移动4G |
| 连接外网 | 支持 |

2）服务端测试环境

服务端主要包括API网关和三个微服务Docker容器环境，包括学校管理、聊天管理和围棋引擎。

表5-3 API网关服务器信息

|  |  |
| --- | --- |
| 主机IP | 192.168.105.30 |
| CPU | Intel(R) Core(TM) i7-4790 |
| 内存 | 2G |
| 操作系统 | Ubuntu 14.04 |
| 服务器功能 | API网关，进行请求的路由和转换 |

表5-4学校管理服务器信息

|  |  |
| --- | --- |
| 主机IP | 192.168.106.41 |
| CPU | Intel(R) Core(TM) i7-4790 |
| 内存 | 1G |
| 操作系统 | Ubuntu 14.04 |
| 服务器功能 | 处理学校管理的请求 |

表5-5 聊天管理服务器信息

|  |  |
| --- | --- |
| 主机IP | 192.168.106.42 |
| CPU | Intel(R) Core(TM) i7-4790 |
| 内存 | 1G |
| 操作系统 | Ubuntu 14.04 |
| 服务器功能 | 处理聊天管理的请求 |

表5-6 围棋引擎服务器信息

|  |  |
| --- | --- |
| 主机IP | 192.168.106.43 |
| CPU | Intel(R) Core(TM) i7-4790 |
| 内存 | 1G |
| 操作系统 | Ubuntu 14.04 |
| 服务器功能 | 处理围棋引擎的请求 |

### 功能测试

功能测试是将任何程序或者系统看成是从输入定义域取值，然后映射到输入值域的函数。即只关注对于特定的输入能否得到指定的输出结果，将整个程序或处理过程看作是黑盒，所以功能测试又叫做“黑盒测试”。通过对系统主要功能模块的不断测试，逐步提高系统功能需求的符合度，最终到达需求中各个功能的要求，并通过了功能测试。具体功能测试如表5-7所示。

表5-7 系统功能测试表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 功能模块 | 主要功能点 | 期望结果 | 实际结果 | 结论 |
| 学校管理 | 注册 | 注册包括普通账号和第三方平台账号注册，输入用户信息，点击注册，注册成功 | 注册成功 | 通过 |
| 登陆 | 登陆包括普通账号和第三方平台账号登录，输入账号密码，点击登录，登录成功 | 登陆成功 | 通过 |
| 修改用户信息 | 用户登录成功以后，可以修改自己的个人信息 | 修改成功，并且页面显示修改后资源的用户信息 | 通过 |
| 创建班级 | 教师登录成功以后，可以创建新的班级 | 创建班级成功 | 通过 |
| 加入班级 | 学生登录成功以后，可以选择自己感兴趣的班级，并选择加入 | 申请加入班级 | 通过 |
| 批准加入 | 教师收到学生加入的信息，并且批准加入 | 批准加入，学生成功加入该班级 | 通过 |
| 发布作业 | 教师可以在该班级中，发布作业 | 该班级的作业界面显示教师发布的作业 | 通过 |
| 聊天管理 | 单聊 | 用户A进入到通讯录界面，选择一个好友B，点击“聊天”，进入单聊界面，输入聊天内容 | 用户B收到用户A发送的聊天内容，并显示在聊天界面 | 通过 |
| 查看会话列表 | 用户发送聊天内容后，会话列表显示新的会话内容 | 用户可以查看新的会话内容 | 通过 |

表5-7 （续）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 功能模块 | 主要功能点 | 期望结果 | 实际结果 | 结论 |
| 聊天管理 | 查看历史聊天记录 | 用户进入到消息界面，然后点击与某用户的会话，进入到单聊界面，然后屏幕下拉 | 单聊界面成功显示历史的聊天记录 | 通过 |
| 围棋引擎 | 选择用户进行对弈 | 可以选择其他用户，然后点击“对弈”，对方同意后，进行对弈 | 加载新的棋谱，双方进入对弈界面 | 通过 |
| 落子 | 用户在棋盘上对弈时，可以点击棋盘上某个坐标，然后用户点击“确认落子” | 点击某个坐标，点击“确认落子”，落子成功 | 通过 |
| 交替落子 | 用户A落子之后，要轮到用户B来落子 | 用户A落子成功，用户B可以落子，并落子成功 | 通过 |
| 吃子 | 用于A落子之后，如果用户B的棋子没气了，要自动提子 | 用户A落子，用户B的棋子没气，提子成功 | 通过 |
| 认输 | 用户如果认为胜利无望，可以点击“认输”，对弈结束 | 用户点击“认输”，对方用户同意，对弈结束 | 通过 |
| 查看棋谱 | 本系统中预存了一些经典对弈棋谱，用户可以选择查看 | 界面显示预存的棋谱列表，并在棋盘上加载选择的棋谱对弈信息 | 通过 |
| 保存棋谱 | 用户可以在对弈时，保存当前棋谱 | 用户点击“保存棋谱”，保存成功 | 通过 |
| 查看围棋局面分析 | 用户可以在对弈过程中，点击“形势”，可以查看对当前局面的形势判断 | 用户点击“形势”，棋盘显示形势判断情况 | 通过 |

从测试用例的运行结果来看，线上围棋教学系统实现了学校管理、聊天管理、围棋引擎功能，且各模块均达到了需求分析和系统设计的功能要求，取得了预期的效果。

### 非功能测试

在软件系统汇总，非功能测试一般是执行测试，以确定系统在特定工作负载下的响应性和稳定性方面的表现。本文将分别从系统的兼容性、丢包率、稳定性进行测试。测试内容如下：

1）兼容性测试

系统采用360真机自动化云测试，采用50个测试终端，测试得到的测试结果如表5-8所示。

表5-8 兼容性测试表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 通过终端数 | 安装失败终端数 | 运行失败终端数 | 卸载失败终端数 |
| 个数 | 50 | 0 | 0 | 0 |
| 占比 | 100% | 0% | 0% | 0% |

从表中的数据可以看出，本系统在真机系统中测试通过。

2）丢包率测试

系统运行时，数据传输过程中的丢包率的测试如表5-9所示。

表5-9 丢包率测试表

|  |  |
| --- | --- |
| 时间（单位：秒） | 丢包率 |
| 0-10 | 0.102 |
| 30-40 | 0.268 |
| 70-80 | 0.212 |

从表中的数据可以看出，服务端接收到的数据会有一定的丢失，但是丢包率控制在0.1%~0.3%之间，根据实际的通信情况，会存在一定的误差，丢包率在正常容许范围之内，不会影响系统数据的准确性。

3）稳定性测试

系统运行时，服务器资源占用情况测试如表5-10所示。

表5-10 CPU与内存测试表

|  |  |  |
| --- | --- | --- |
| 服务器 | CPU利用率 | 内存占用率 |
| API网关 | 31% | 39% |
| 学校管理 | 19% | 30% |
| 聊天管理 | 13% | 29% |
| 围棋引擎 | 22% | 34% |

通过表中数据可以看出，服务器的CPU平均利用率和内存占用率符合通用的性能衡量标准。

## 本章小结

本章在线上围棋教学系统分析和设计的基础上，详细的阐述了需求分析中各个模块的实现过程，然后对系统进行了测试，并且对测试结果进行了分析，系统满足设计目标，完成了对集群系统的资源监控，同时，系统满足数据的完成性、低负载和准确性。

# 结论与展望

## 结论

线上围棋教学系统集围棋对弈与教学功能于一体，采用移动客户端和服务端微服务化的设计方式开发了一个方便教师和学生的围棋教学系统。本文还提出基于厚势价值的影响模型，在围棋的形势判断上，更加符合职业棋手的估值。本系统为围棋教学提供了一个方便快捷的平台，并且对围棋的推广进行了新的探索和尝试，也因此开拓了一个新的市场，具有重要的理论意义和实用价值。

根据本论文中对性能需求的要求，搭建的移动客户端框架能够满足其要求，并且服务端的微服务化开发符合低耦合的特性，方便后续的开发与维护，并且提高了开发效率和系统性能。

本文主要设计和实现了基线上围棋教学系统，根据需求，给出线上围棋教学系统的需求分析，按照功能需求将系统划分为：学校管理模块、聊天管理模块、围棋引擎模块。结论如下：

1. 通过对Android常用的UI框架进行研究，结合了自定义View的特点，并解决了界面交互和组件通信等问题，完成了客户端的整体框架构建。通过对系统的运用，证明了构建的移动客户端框架能够保证系统的正常运行、系统的性能要求以及后期版本迭代的要求。
2. 通过采用服务端微服务化的开发方式，更好的提升了各个模块的可复用性和可组装性。通过微服务架构更好的实现了原单体应用内部各个组件或者模块的彻底解耦，降低了原单体应用内部的复杂度。

3）通过采用了第三方平台的ShareSDK提供的jar包，将新浪微博接入到本系统中，然后调用第三方平台提供的接口，完成了第三方平台账号的验证，并且结合本系统中的业务要求，完成了第三方平台的注册和登录功能。

4）通过对班级信息的管理，教师与学生可以更加专注自己的角色功能，教师可以管理班级中的学生，然后在班级中发布新的作业。并且通过提过聊天管理功能，用户之间可以进行即时聊天，进而咨询他人来解决自己遇到的问题等等。

5）用户可以进行人-人对弈，并且通过对围棋局面的分析，采用了基于厚势价值的新的影响函数模型，更加符合职业棋手的估值。在围棋对弈时，用户可以实时的看到对当前的局面分析，进而动态的调整围棋策略，来争取对弈的胜利。

## 展望

本文完成了线上围棋教学系统的设计与实现，但尚存在需要改进的地方，特别是某些细节上有待进一步的优化和改善，可以在以下几个方面进行拓展：

1）由于对UI设计的经验不足和技术缺乏，导致移动客户端的系统界面不是很美观，以及交互上面也并没有做到专业水准的那么流畅，所以导致影响了系统用户界面的友好化，所以这些问题都是需要进一步的优化和完善。

2）由于个人的精力比较有限，因此本系统中只是开发是安卓的平台客户端，这不利于用户的普及和推广，会造成一部分用户的使用障碍。因此在后续的时间里，需要将IOS客户端开发出来，并在应用商店上线，这样才能使本系统真正的服务于大众，是围棋教学变得更加普及。

3）由于客户端和服务端之间都通过API Gateway进行通信，所以导致该组件可能成为项目的瓶颈。后期要进行该组件的重点开发和维护。

# 致 谢

时光如白驹过隙，转眼间已临近毕业季。随着硕士论文的完成，意味着我的研究生生涯即将结束。虽然这三年的学习生活在我们的人生旅途中是那么的短暂，但是回首往事，一切都历历在目。在这里帮助和鼓励过我的同学、朋友还有教诲过我的老师不仅仅使我收获了知识，也让我得到了极大的成长。

首先我想衷心的感谢我的导师赵银亮老师。在这三年里，我受到了赵老师极大的帮助。在学习和研究过程中，赵老师一直对我悉心指导，孜孜不倦的教诲我。除此之外，赵老师身上那认真严谨的钻研态度以及高尚的品德也对我影响非常深远。在生活中，赵老师就像家中长者一样，总是非常耐心且尽心尽力的指引着我，从而帮助我找到解决问题的办法。在此谨向赵老师致以最诚挚的谢意！

感谢学校给我提供了非常优越的科研环境和良好的学习氛围。通过西安交通大学网络图书馆，我可以阅读且学习很多也很优秀全面的中外相关文献以及期刊，丰富了我的知识，并且拓展了我的视野。

感谢我的同学以及朋友们。谢谢你们陪伴着我一起学习进步一起成长，一起分享生活中的点点滴滴喜怒哀乐，是你们的存在，让我的青春更加的丰富多彩。还有感谢我的家人，感谢他们对我的养育之恩，对我无私的奉献和关爱，让我知道我在未来前进的道路上，不管多么艰辛，都会有坚强的后盾。

最后，非常的感谢百忙之中抽出时间来评审本文的专家以及答辩委员会的专家。谢谢你们对我提出的宝贵意见。

# 参考文献

1. Sam Newuman. Building Microservices. 1 edition. O'ReillyMedia, February 20, 2015: 1-280.
2. 张玉志. 计算机围棋博弈系统. 北京: 中国科学院计算机研究所, 2017.
3. Kierulf, Anders. Smart Game Board: A Work bench for Game-Playing Programs, with Go and Othello as Case Studies: [Ph.D. Thesis No. 9135]. Switzerland: Swiss Federal Institude of Technology (ETH) Z”urich 2016.
4. H. Remus. Simulation of a learning machine for playing Go. In: Proceedings of the International Federation of Information Processing Congress (IFIP), North-Holland, Amsterdam 2015.
5. 石田芳夫. 围棋定式大全. 蜀蓉棋艺出版社. 2014.
6. 刘知青,李文峰. 现代计算机围棋基础. 北京: 北京邮电大学出版社, 2015:182-183．
7. Chang-Shing Lee, Mei-Hui Wang, Chaslot, G. The computational intelligence of Mongo revealed in Taiwan’s computer Go tournaments[C]. IEEE Transactions on Computational Intelligence and AI in games, 2016, 1(1):73-89.
8. 谷蓉. 计算机围棋博弈系统的若干问题研究. 北京: 清华大学, 2015．
9. 马越,黄刚.基于Docker的应用软件虚拟化研究[J].软件,2015, 36(3): 10-14.
10. 刘思尧，李强，李斌.基于Docker技术的容器隔离性研究[J].软件,2015, (4): 110-113.
11. 郭洁, 围棋人工智能中的几个上限值的研究. 长沙: 中南大学, 2016.
12. McQuade B. Machine learning and the Game of GO[D]. USA: Middlebury, Vermont: Department of Computer Science, Middlebury College, 2015 .
13. Atsushi Yoshikawa, Takuya Kojima, Yasuki Saito. Relations between skill and the use of terms-an analysis of protocol of the game of GO. In:H.J.van den Herik, H. Iida, eds. Computers and Games, 2015, No.1558 in LNCS:282-299．
14. Albert L. Zobrist. A model of visual organization for the game of GO. In: Proceedings of the Spring Joint Computer Conference. Spring Joint Conference.Boston: AFIPS Press, Montvale NJ, 2015.103-112.
15. Albert L. Zobrist. Feature extractions and representation for pattern recognition and the game of GO[D]. Madison: Graduate School of the University of Wisconsin, 2016.
16. Jon Ryder. Heuristic analysis of large trees as generated in the game of GO. Department of Computer Science, Stanford University, 2015．
17. Walter Reitman, Bruce Wilcox, R. Nado. Goals and plans in a program for playing Go. In:Proc. 29th ACM conference. New York: ACM, 2016:123-127．
18. Sander, P.T.A strategic approach to the Game of Go [M]. London:Department of Computer Science, Faculty of Graduate Studies, The University of Western Ontario, London, Ontario, 2015．
19. Kerwin, J. and Reitman, W. Video game no.3:a Go protocol with comments. Mental Health Research Institute, Communication No.300, University of Michigan, Ann Ardor, Michigan, 2015．
20. 岳鹏. 计算机围棋中的算法研究. 博士, 西南大学, 2015.
21. 余磊. 基于认知科学的计算机围棋博弈问题的研究. 华东师范大学, 2016.
22. Yao Meng, Dai Chun-ni,Pei Min, et al. The Species Compete-die Out(SCD) Algorithms Model for Evolutionary Computation [C]. Proc. Of Genetic and Evolutionary Computation Conference. Washington:GECCO-2015, 2015:158-160.
23. Dai Chun-ni, Yao Meng, Xie Zhu-jie. Parameter Optimization for Growth Model of Greenhouse Crop Using Genetic Algorithms [J]. Applied Soft Computing, 2015, 9(1):12-19.
24. 叶品星.一种博弈树静态估值算法[J]. 计算机工程与设计, 2014, 25(7).
25. B. Bouzy, B. Helmstetter. Monte-Carlo Go Developments. H. J. van den Herik, H. Iida and E.A. Heinz (eds.). Proceedings of the 10th Advances in Computer Games Conference (ACG-10)(Kluwer Academic), 2015, 159-174.
26. 陈磊.计算机围棋领域概念网的设计与实现.硕士, 北京邮电大学, 2015.
27. 仇臣.Docker容器的性能监控和日志服务的设计与实现. 浙江大学,2016.
28. Lei Yu,Jingao Liu,Improved Winning Probability Model in Go Based on Strong Group Quantization and Multi-level Species Compete-Die Out Algorithms. Proc.of 2013 the 2nd International Conference on Computer and Communication (ICFCC 2015).
29. Zhengwei. Zhou, Shijing..Development Situation of Computer Go Communications of the Institute of Information and Computing Machinery,Vol.l,No.2, April. 2014,pp: 23-30
30. L. Kishimoto, Correct and efficient algorithms in the presence of repetitions. Albertson College, 2015.
31. Martin Muller, Ralph Gasser. Experiments in Computer Go Endgames. In: R.Nowakowski, eds. Games of No Chance. Cambridge University Fress, 2016, 29:273-284
32. Muller, M. Computer Go as a sum of local games: an application of combinatorial game theory: [PhD thesis]. Zurich: Swiss Federal Institute of Technology, 2015
33. Jay Burmeister, Janet Wiles. An introduction to the computer Go field and associated Internet resources.Technical Report CS-TR-339, Department of Computer Science, University of Queensland, 2015.
34. Remi Coulom Computing Elo ratings of move patterns in the game of Go. Draft, submitted to ICGA Computer Games Workshop 2014, 2014.
35. 李文峰，余平，刘知青.计算机围棋博弈中模式的表示和提取.2014年中国机器博弈学术研讨会，重庆，2014.

学位论文独创性声明（1）

本人声明：所呈交的学位论文系在导师指导下本人独立完成的研究成果。文中依法引用他人的成果，均已做出明确标注或得到许可。论文内容未包含法律意义上已属于他人的任何形式的研究成果，也不包含本人已用于其他学位申请的论文或成果。

本人如违反上述声明，愿意承担以下责任和后果：

1．交回学校授予的学位证书；

2．学校可在相关媒体上对作者本人的行为进行通报；

3．本人按照学校规定的方式，对因不当取得学位给学校造成的名誉损害，进行公开道歉。

4．本人负责因论文成果不实产生的法律纠纷。

论文作者（签名）： 日期： 年 月 日

学位论文独创性声明（2）

本人声明：研究生 所提交的本篇学位论文已经本人审阅，确系在本人指导下由该生独立完成的研究成果。

本人如违反上述声明，愿意承担以下责任和后果：

1．学校可在相关媒体上对本人的失察行为进行通报；

2．本人按照学校规定的方式，对因失察给学校造成的名誉损害，进行公开道歉。

3．本人接受学校按照有关规定做出的任何处理。

指导教师（签名）： 日期： 年 月 日

学位论文知识产权权属声明

我们声明，我们提交的学位论文及相关的职务作品，知识产权归属学校。学校享有以任何方式发表、复制、公开阅览、借阅以及申请专利等权利。学位论文作者离校后，或学位论文导师因故离校后，发表或使用学位论文或与该论文直接相关的学术论文或成果时，署名单位仍然为西安交通大学。

论文作者（签名）： 日期： 年 月 日

指导教师（签名）： 日期： 年 月 日

(本声明的版权归西安交通大学所有，未经许可，任何单位及任何个人不得擅自使用)