摘　　要

ACM/ICPC 程序设计竞赛是世界上公认的规模最大、水平最高的国际大学生程序设计竞赛。可分布式部署的在线评测系统是一个辅助竞赛者的日常培训的应用系统，竞赛者可以通过它提交源程序并在互联网中实现在线自动评测。

可分布式部署的在线评测系统是一个运用 B/S 与 C/S 模式，并结合 Linux 内核编程的综合性可分布式部署应用系统。本系统不仅支持传统在线评测系统的动态 Web（B/S 模式）的访问，同时支持使用客户端（C/S 模式）访问。可分布式部署的在线评测系统分为 Web 服务模块、用户终端模块、中心服务模块、评测模块、数据接口模块等五个大的功能模块。其中，Web 服务模块和用户终端模块提供和用户的交互接口，中心服务模块负责整个系统的处理逻辑和数据导向，评测模块对用户提交的源代码进行评测，数据接口模块负责所有数据的存储与查询。

本文主要讨论可分布式部署在线评测系统的网络服务部分（包括 Web 服务模块，用户终端模块和中心服务模块的网络接口）的设计与实现。首先本文介绍了系统设计与开发中应用到的技术的背景知识，并从用户视图的角度阐述需求，然后阐述整个系统的架构，并详细讨论网络服务部分中关键模块的设计与实现，包括 Web 站点的结构，网络协议的定义，Web 安全策略的运用等，最后说明开发中用到的一些技术，并对开发做出总结。

关键字： ACM/ICPC；Web 服务；协议；Web 安全

Abstract

ACM/ICPC is recognized as the largest scale and highest level global college student programming contest. Distributed online judge system is used for supporting contestants' day-to-day training. Students can submit their source code and get results supplied by the system automatically on the Internet.

The distributed online judge system is an integrated application designed with B/S and C/S model, associated with Linux kernel programming. The system not only supports traditional dynamic web (B/S model) access, but also can be accessed through client (C/S model). The system can be divided into five major modules, they are web services module, user terminals module, center services module, judge module and data interface module. Web services module and user terminals module provides the interactive interface for users. The center services module is responsible for the whole system's logic processing and data directing. The judge module is to judge users' source codes. The data interface module takes care of all data's storage and enquiry.

This thesis mainly talks about the designed and implementation of the system's network services part (including the web services module, the user terminals module and the network interface of the center services module). At first, the paper gives a brief introduction to the background knowledge used in designing and developing, and then recounts the demands from users' view points. Then, the paper tells the structure of the whole system, and descriptions the design of some key modules of network services part, including structure of the web site, definition of the network protocols, using of the web security strategy, and so on. At the end, the paper tells some technology used in developing, and make a summary of the system's developing.

Keywords: ACM/ICPC; Web services; protocol; Web security.

目　录

第1章　　引言

1.1课题背景及意义

ACM/ICPC(ACM International Collegiate Programming Contest, 国际大学生程序设计竞赛)是由国际计算机界历史悠久、颇具权威性的组织 ACM(Association for Computing Machinery，国际计算机协会)主办的，世界上公认的规模最大、水平最高的国际大学生程序设计竞赛，其目的旨在使大学生运用计算机来充分展示自己分析问题和解决问题的能力。该项竞赛从 1970 年举办至今已历 32 届，一直受到国际各知名大学的重视，并受到全世界各著名计算机公司的高度关注，在过去十几年中，Apple、AT&T、Microsoft 和 IBM 等世界著名信息企业分别担任了竞赛的赞助商。可以说，ACM 国际大学生程序设计竞赛已成为世界各国大学生最具影响力的国际级计算机类的赛事， 是广大爱好计算机编程的大学生展示才华的舞台，是著名大学计算机教育成果的直接体现，是信息企业与世界顶尖计算机人才对话的最好机会。

在线评测系统(Online Judge System, OJ) 作为国内外诸多参加程序设计竞赛选手的训练交流平台，对每个由评测系统展示的题目接收用户提交的源代码，在编译执行后将用户程序的输出与系统的标准输出进行比较，并给出用户程序运行和比较的结果，从而判断用户提交的源代码是否正确。目前国内外已有类似网上评测系统，其中世界最著名的网上评测系统是 UVA Online Judge，国内著名的网上评测系统有 PKU Online Judge（北京大学）和 ZJU Online Judge（浙江大学）。使用者可以针对某个问题，写出程序解决它，并把你的程序代码上传至 Online Judge，由 Online Judge 反馈解题状态，这是使程序设计能力进步的一个很好的渠道。

国内计算机专业培养的学生理论功底一般会很好，但实践能力总是略显不足，这是因为该专业学生平常的实习课程受到的约束力量不够。大学计算机专业的实践性课程均设有教学辅导，且一般为该专业硕士研究生担任，但教学辅导员因需认真检查每个学生的实践成果而工作量过大，最后可能导致教学辅导对每个学生的检查力度无法达到预期的教学计划。

武汉大学在培养学生的理论知识的同时，也重视培养学生的动手能力。对于程序的爱好者而言，培养他们动手能力也就是多自己动手编程，再者就是多交流经验，互相取长补短。而武汉大学 ACM/ICPC 在线评测系统建设（Online Judge System）的目的正是为了培养学生的实际动手能力。其功能有在线测试（主要是用户通过网络直接检测自己编写的程序是否正确并提供在线比赛的平台）和论坛（提供用户在论坛上探讨问题，互相学习的机会）。通过此网站不但能让同学们对 ACM/ICPC 有所了解，而且还增添他们编程的兴趣。

随着参加 ACM/ICPC 的同学越来越多，原始的单服务器模式越来越不能承受多用户同时访问时的压力，表现在 Web 的响应速度无法跟上用户的大面积访问，以及判题服务的响应时间慢于用户提交新程序的时间，从而进入无法提供服务给新登录用户，且已登录用户的服务也会变得无法响应的恶性循环。近两年国内 ACM/ICPC 举行的亚洲区各赛点的网络预选赛充分暴露了这一问题。本课题将在武汉大学已有的在线评测系统上进行全面重构，将原始的单服务器模式转变成由一个底层数据库服务模块（本模块使用 MySQL），一个中心服务模块，多个 Web 服务模块，多个评测判题服务模块组成的服务集群，集群中的任何一个部件均可部署在不同的服务器上，且可以用普通 PC 临时替代，大大减轻服务器压力和提升服务性能。在提供服务集群的同时本课题还提供了评测系统的客户端软件，从而在 B/S 架构

的同时也实现了 C/S 架构，让用户在网络不稳定的情况下也能一样进行练习。

1.2 课题需解决的问题

1.2.1可分布式部署的实现

本课题所涉及的系统设计的原因之一就是解决单服务器模式在大量用户并发访问时的网络负载和 Judge 压力过重的问题。解决此问题一个可行的措施是设计一个可以分布式部署解决方案。多个 Web 服务入口以及额外的客户端访问入口，可以使在线评测服务的网络负载分散开了，多个 Judge 服务承担判题服务，从而减轻单个服务器的负担，使在大量用户并发访问时系统仍然能够正常相应用户的请求。

1.2.2 各模块之间通信协议

由于本课题所涉及的系统要求最后能将各模块部署在不同的机器上，这样就带来了各模块的通信问题，主要包括通信的安全及稳定。从尽可能降低各模块耦合的设计理念及提高性能和安全性的角度出发，我们选择使用一个中心服务模块来进行核心的通信交换和一些数据转换和处理，其他所有模块均只和该中心服务模块进行通信。中心服务器使用 MySQL 标准的 3306 端口和协议与 MySQL 数据库服务模块进行通信，使用自定义端口及协议与本系统中的 Web 服务模块及评测判题服务模块进行通信。自定义的端口均保证不与系统端口及常用端口冲突，且可以进行实时配置来选用合适的端口。自定义的协议均在保留可扩展性的前提下，对传输数据进行简洁高效的封装，可用最轻量级的协议来减少网络传输数据的开销。自定义的协议中也考虑了容错性，如果在数据传输过程中发生非预期的更改，也可以通过验证机制来判断正确与否，从而丢弃部分数据而请求重新发送。

1.2.3 各模块之间的连接——线程池的广泛应用

在传统的两层 C/S 连接结构中，客户端程序启动时打开连接，在退出时关闭连接。这样，在整个程序运行过程中，每个客户端均始终占用一个服务器的连接，实际上使用连接的数据传输只会占用整个运行时间的极小部分，这样就造成连接的使用效率低下，且会拖累服务端和客户端程序的性能。此问题将表现在中心服务模块对数据库服务模块的连接、中心服务模块对 Web 服务模块的支持、中心服务模块对客户端软件的支持和中心服务模块对评测判题服务模块的连接上。

在使用了线程池的三层 C/S 连接结构中，连接都通过线程池来进行管理。只有真正需要数据传输时，中间层才会从线程池中申请一个线程使用该线程的连接来进行数据传输，在数据传输结束后立即释放此线程到线程池中，以供其他连接使用。这样不仅大大提高了连接的使用效率，使得大量用户可以共享较少的连接，而且省去了建立连接的时间。

1.2.4 Web 安全对策

本系统作为一个在线提交与评测系统，允许用户提交代码，以多种方式方面查询结果。系统 Web 服务模块并不直接接触数据库，这从一定程度上保障了数据的安全。但 Web 本身存在大量的安全隐患，由于与用户的互动性很大，给非法用户攻击服务器或 Web 应用环境的程序提供了便利。

1.3 课题取得的成果

本课题所实现的可分布式部署在线评测系统目前已在实验室内网络临时服务集群上稳定运行超过两个星期，为整个武汉大学 ACM/ICPC 集训队提供了练习、比赛的平台。在结合其他高校的经验和教训基础上，对在线评测系统提出了更符合队伍训练及教学练习发展方向和更好提升用户体验的需求，并对核心功能做了深入的探讨并实现了需求中提出的所有核心功能及提升用户体验的绝大部分功能。

1.4 本文的组织

因为在线评测系统本身是一个比较复杂的系统，涉及的背景知识也会相对复杂，实现过程中所使用的方法和实现的内容也相对繁琐，加上本系统要求可分布式部署，致使本文覆盖的内容和知识体系不可避免的会比较复杂。为简化阅读，现将整篇论文的结构介绍如下：

整篇论文共分五章及附录两节,各章节的主要内容安排如下：

1．引言。介绍论文选题的课题背景及意义、需要解决的问题、实际上取得的成果及论文的结构安排。

2．背景知识。简单讨论一下所用的技术的背景知识，包括 Web 开发技术，Web 安全与对策，Linux 与 Linux 下的网络编程等。

3．网络服务模块设计与实现。对网络服务模块的需求，分析，设计，以及用到的一些技术进行详细说明。给出系统总体架构，数据库的简要描述，站点构成图，Web 安全法则的运用，以及对关键子模块的描述。

4．结果分析。 对开发过程以及开发所取得的结果进行分析。

5．结束语。对开发做出总结，对有待改进的地方进行阐述。

第2章　背景知识

2.1 Web 开发技术

2.1.1 Apache Web 服务器简介

Apache 是世界使用排名第一的 Web 服务器。它可以运行在几乎所有广泛使用的计算机平台上。

Apache 源于 NCSAhttpd 服务器，经过多次修改，成为世界上最流行的 Web 服务器软件之一。Apache 取自“a patchy server”的读音。意思是充满补丁的服务器，因为它是自由软件，所以不断有人来为它开发新的功能，新的特性，修改原来的缺陷。Apache 的特点就是简单，速度快，性能稳定。

本来它只用于小型或试验 Internet 网络，后来逐步扩充到各种 Unix 系统中，尤其对 Linux 的支持相当完美。Apache 有多种产品，可以支持 SSL 技术，支持多个虚拟主机。到目前为止 Apache 仍然是世界上使用的最多的 Web 服务器，市场占有率达 60%左右，远远领先排名第二的 Microsoft IIS。世界上很多著名的网站如 Amazon.com、Yahoo!、W3Consorttium、Financial Times 等都是 Apache 的产物，它的优势主要在于它的源代码开放、有一支开放的开发队伍、支持跨平台的应用（可以运行在几乎所有的 Unix、Windows、Linux 系统平台上）以及它的可移植性等方面。

Apache 服务器拥有以下特性：

支持最新的 HTTP/1.1 通信协议

拥有简单而强有力的基于文件的配置过程

支持通用网关接口

支持基于 IP 和基于域名的虚拟主机

支持多种方式的 HTTP 认证

集成 Perl 处理模块

集成代理服务器模块

支持实时监视服务器状态和定制服务器日志

支持服务端包含指令（SSI）

支持安全套接字层（SSL）

提供用户会话过程的跟踪

支持 FastCGI

通过第三方模块可以支持 Java Servlets

支持多种平台，如 MS Win32/Unix/Linux/OS2 等

多语种，支持 UTF-8、GB2312、Shift JIS、BIG5 等多种字符集编码

多线程，Apche2 支持 Unix 运行 POSIX 线程

2.1.2 PHP5 Web 程序设计

PHP，全称为 PHP Hypertext Preprocessor，超文本预处理器之意。PHP 是一种在服务器

端执行的嵌入 HTML 文档的脚本语言，语言的风格类似 C 语言，现在被很多的网站编程人员广泛的运用。PHP 独特的语法混合了 C、Java、Perl 以及 PHP 自创的新的语法。它可以比 CGI 或者 Perl 更快捷的执行动态网页。用 PHP 做出的动态网页与其他的编程语言相比，PHP 是将程序嵌入到 HTML 文档去执行，执行效率比完全生成 HTML 标记的 CGI 要高很多。PHP 执行引擎还会将用户经常访问的 PHP 程序驻留内存中，其他用户再一次访问这个程序时就不需要重新编译程序了，只要直接执行内存中的代码就可以了。PHP 具有非常强大的功能，而且支持几乎所有流行的数据库以及操作系统。

使用 PHP 进行 Web 应用程序开发，具备以下的优点：

开发效率高，函数语句简洁明了

输出控制灵活，可在 HTML 中内嵌代码，也可以由 PHP 输出 HTML 运行，也可以在命令行下运行，将结果输出到其他设备。

可实现模板化，实现程序逻辑与用户界面分离

跨平台可运行在 Win32 或 Unix/Linux/FreeBSD/OS2 等平台

与多个 Web 服务器兼容如 Apache、MS IIS、Netscape Server 等

完全支持面向对象开发，并向下兼容，支持面向过程与面向对象两种风格开发

内置 Zend 加速引擎，性能稳定快速

内置函数丰富，几乎包含了 Web 开发的所有方面

组件化开发，提供 MySQL、Oracle、MS SQL 等多种数据库访问接口，支持 ODBC

支持正则表达式，内置 POSIX 与 Perl 兼容两类的正则表达式支持

开发成本低，开发工具多，且有众多使用 PHP 开发的源代码项目供我们参考和二次开发

2.1.3 Javascript 的应用

Javascript 是一种基于对象和事件驱动并具有安全性能的脚本语言，它运行于客户端浏览器。在 HTML 基础上，使用 Javascript 可以开发交互式的 Web 网页，使得网页和用户之间实现了一种实时的、动态的，交互性的关系，使网页包含更多活跃的元素和更加精彩的内容。Javascript 使有规律的重复的 HTML 文段简化，减少下载时间。Javascript 能及时响应用户的操作，对提交表单做即时的检查。Javascript 短小精悍，又是在客户机上执行的，大大提高了网页的浏览速度和交互能力。Javascript 的基本特点有：

是一种脚本语言，它采取小程序段的方式实现编程，无需编译，而是在程序运行过程中逐行解释，提供了一个简易的开发过程。

是一种基于对象的语言，能运用自己已经创建的对象。

是一种安全性语言，不允许访问本地硬盘，不允许对网络文档进行修改和删除，只能通过浏览器实现信息浏览和动态交互，从而有效防止数据的丢失。

它是动态的，可以直接对用户输入做出响应，无须经过 Web 服务程序。

它是跨平台的，Javascript 依赖于浏览器本身，与操作环境无关。

2.2 Web 安全与对策

Web 应用可能是静态的 HTML 文件或是复杂的、动态的、基于数据库的 Web 站点，无论哪种情况，安全性对于保持应用的完整性、用户的隐私、数据的保密、服务器的正常运行都是极为重要的。

Web 安全漏洞可以分为两类：一类包括平台自身的安全漏洞——许多 Web 应用程序共

享的部分，例如 Linux、Windows、Apache 和 Oracle 等；另一类则是指应用程序自身的安全漏洞，也就是说，Web 站点中的编程错误可能会暴露用户的隐私信息、允许恶意用户执行任意的数据库查询，甚至允许通过远程命令行访问服务器。

因此，任何 Web 应用都面临着多种多样的威胁，对于一些针对应用的攻击，例如 SQL 语句的注入、命令注入或是会话劫持，可以整理出一些最普遍的检查方法，例如基本的输入验证，密码加密传输，尽可能不使用隐藏域，尽可能不使用 URL 参数传递敏感信息，会话标志设置有效时限，并及时注销并清除会话等等。

2.3 Linux 与 Linux 下的网络编程

2.3.1 Linux 简介

Linux 是一个基于 Posix 和 Unix 的多用户多任务支持多线程的类 Unix 操作系统，这个系统是由全世界各地的成千上万的程序员设计和实现的，其目的是建立不受任何商品化软件的版权制约的、全世界都能自由使用的 Unix 兼容产品。

Linux 以它的高效性和灵活性著称。它能够在 PC 计算机上实现全部 Unix 特性，具有多任务、多用户的能力。Linux 是在 GNU 公共许可权限下免费获得的，是一个符合 POSIX 标准的操作系统。Linux 操作系统软件包不仅包括完整的 Linux 操作系统，而且还包括了文本编辑器、高级语言编译器等应用软件。

Linux 提供了丰富的网络功能。实际上，Linux 就是依靠互联网才迅速发展起来的，它不仅可以作为网络工作站使用，更可以胜任各类服务器，如 Web 服务器、文件服务器、邮件服务器等等。

Linux 之所以受到广大计算机爱好者的喜爱，主要原因有两个，一是它属于自由软件，用户不用支付任何费用就可以获得它和它的源代码，并且可以根据自己的需要对它进行必要的修改，无偿对它使用，无约束的继续传播。另一个原因是，它具有 Unix 的全部功能。

基于 Linux 开放源代码的特性，越来越多的大中型企业及政府投入更多的资源来开发 Linux。

2.3.2 Linux 环境下的网络编程

套接口提供的是 OSI(计算机通信开放系统互连)模型中上三层(会话层、表示层、应用层)与传输层的接口。Socket(套接字)，用于描述 IP 地址和端口，是一个通信链的句柄，应用程序通常通过 Socket 向网络发出请求或应答网络请求。Socket 接口是 TCP/IP 网络的 API，Socket 接口定义了许多函数或例程，程序员可以用它们来开发 TCP/IP 网络上的应用程序。常用的 Socket 类型有两种：流式 Socket(SOCK_STREAM)和数据报式 Socket(SOCK_DGRAM)。流式 Socket 是一种面向连接的 Socket，针对于面向连接的 TCP 服务应用；数据报式 Socket 是一种无连接的 Socket，对应于无连接的 UDP 服务应用。

C／S 模式，即 Client／Server（客户机／服务器）模式是常见的程序间网络通信模式，下图（图 2-1）是一对典型的 TCP 客户与服务器进程之间通信过程：

服务器首先启动，稍后某个时刻客户端启动，它试图连接到服务器。假设客服端给服务器发送一个请求，服务器处理该请求，并且给客户端发回一个响应。这个过程一直持续下去，知道客户端关闭连接，从而给服务器发送一个文件结束通知为止。服务器接着也关闭本地连接，然后或者结束运行，或者等待新的客户连接。

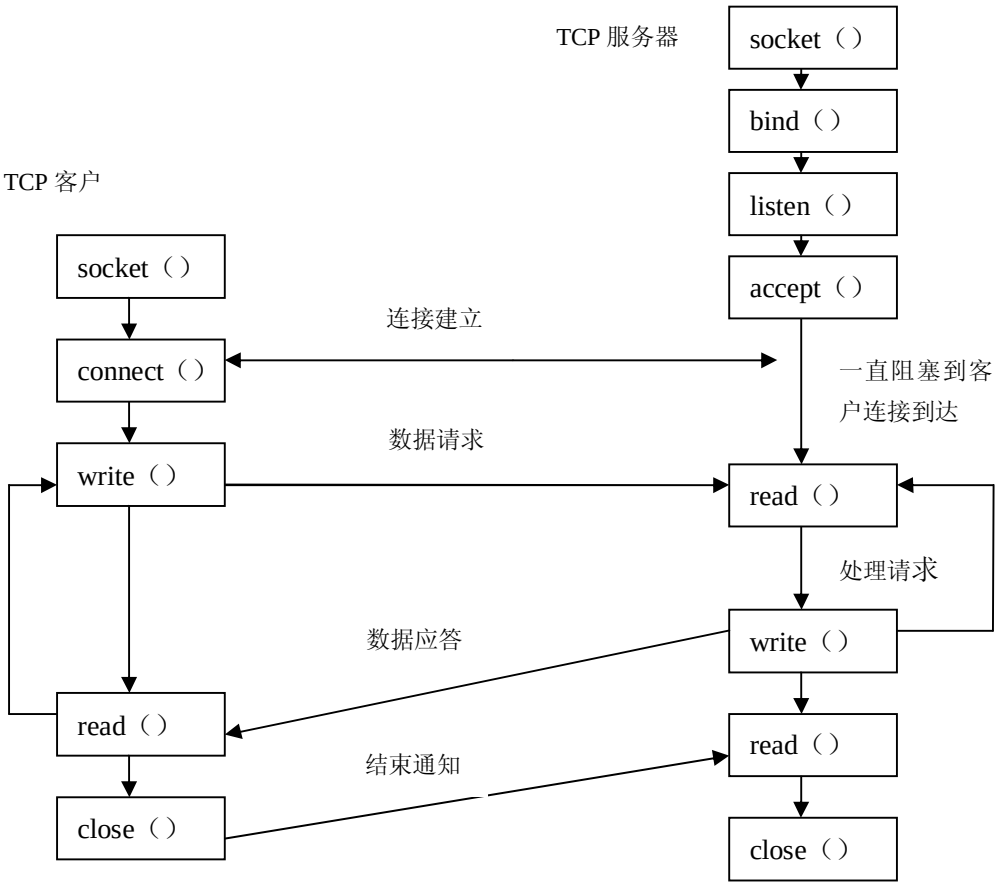

图 2-1 基本 TCP 客户端/服务器程序的套接口函数

第3章　网络服务模块设计与实现

3.1 需求分析

在线评测系统作为程序设计竞赛选手的训练和交流平台，其核心功能是对用户提交的源代码进行评测并反馈相应结果的功能，并提供用户管理，题目列表，在线比赛，讨论交流，信件管理等基本功能。在线评测系统的用户包括一般一般用户和管理员，下面从用户界面的角度分别讨论这两方面的需求：

3.1.1 一般用户

一般用户获取的信息和功能需求包括：首页（Homepage），题目（Problems），比赛（Contests），实时状态（Status），实时提交（Submit），用户排名（Ranklist），讨论区（Discuss），用户控制台（User），帮助（FAQ）。

1．首页（Homepage）

首页包括欢迎信息，最近的通知通，下场比赛提醒，刻苦用户排行榜，友情链接等信息。

2．题目（Problems）

题目包含题目列表，题目详情，题目统计信息等内容。

题目列表：题目列表每页显示 100 个题，每行一个题目的信息，信息包括用户是否尝试/通过此题、题目编号、该题通过率、该题提交次数和通过次数等信息。题目列表下方搜索框提供根据题号、标题、来源、隶属比赛等条件的搜索。

题目详情：题目详情包含此题目的标题、描述、输入说明、输出说明、样例输入、样例输出，时间限制、内存限制、提交次数、通过次数等必要部分，描述中支持图片、特殊符号等内容的显示。每个题目还可以包含提示，来源，相关比赛三个非必要部分。

题目统计信息：每个题目统计信息应包含本题的提交信息统计表和通过的提交表，本题提交信息统计表包括尝试的用户数，通过的用户数，提交总次数，通过次数，格式错次数，超内存次数，超时次数，答案错次数，超输出次数，运行错次数，编译错次数。本题通过的提交表格式同实时状态信息，这里只选结果为 AC 的，同一用户多个 AC 选成绩最好的，所有提交信息按成绩排序，排序规则为耗时越少越好，耗时一样时耗内存越少越好，时间内存均一样按提交时间排序。

3．比赛（Contests）

比赛包含比赛列表、比赛详情、比赛排名、比赛统计信息、虚拟比赛列表、预定虚拟比赛、管理虚拟比赛等内容。

比赛列表：每行显示一场比赛，比赛信息包括比赛名称、开始时间、比赛时长、比赛状态、比赛类型等。

比赛详情：主题为比赛题目列表，显示当前用户是否/尝试通过此题，题目编号，题目标题等。其他信息包括比赛开始时间、结束时间、当前时间、比赛状态等。

比赛排名：比赛排名包含一个表格，每行分别是一个用户的信息，包含排名（Rank），昵称（Nickname），解决题数（Solved），罚时（Penalty），及各题的状态。

比赛统计信息：比赛统计信息包含对用户和题目的统计信息。对用户的统计信息包括参

加的用户数（即有提交的用户），用户出题数的分布。对题目的信息包括每个题的各种结果（AC, CE, WA, TLE, RE, MLE, OLE）数，各种语言数，总提交数的统计。

虚拟比赛列表：同比赛列表，类型为 Virtual。

预定虚拟比赛：一个提交表单，内容包括一场虚拟比赛需要的所有参数，即申请用户，申请用户密码，比赛标题，比赛描述，比赛开始及结束时间，比赛包含的题目列表等必要信息。

管理虚拟比赛：对虚拟比赛的各个选项进行修改。

4．实时状态（Status）

实时状态是一个独立的页面，同时提供服务给特定题目或特定比赛，提供服务给特定题目可以直接使用实时状态的搜索功能实现，对特定比赛需要另外提供限制服务，同时查看代码页面，查看错误输出信息页面也由此部分提供。实时状态表的每行应包含提交编号，提交者，提交题号，结果，内存使用，耗时，编程语，代码长度，提交时间等。

5．实时提交（Submit）

一个提交表单，包括用户名，密码，提交题号，编程语言，是否共享代码，源代码内容等项目。

6．用户排名（Ranklist）

每行包含一个用户的排名，用户 ID，用户昵称，用户解决题目数，用户提交次数，和成功率等信息。

7．讨论区（Discuss）

次部分主体为一树状结构森林，即讨论主题森林，每个讨论主题下的信息按照回复关系组成树状结构，每个讨论主题可以选择在树根部进行折叠或展开。讨论区还包括发表，回复，按条件搜索等功能。

8．用户控制台（User）

用户控制台包含用户注册，用户登录，用户信息修改，用户邮箱，用户共享代码控制台，个人信息页面。

用户注册：表单，包括用户 ID、密码、昵称、学校、邮箱等注册必需个人信息。

用户登陆：用户 ID 和密码，登陆后记录会话信息，退出后清除。

用户信息修改：修改个人信息。

用户邮箱：包括写邮件，收件箱，发件箱等。

9．帮助（FAQ）

常见问题解答。

3.1.2 管理员

管理员控制台的功能需求包括快速首页（QuickPath），题目控制台（Problems），比赛控制台（Contests），结果判断控制台（Judge），用户管理控制台（Users），讨论区控制台（Discuss），下部分别说明。

1．快速首页（QuickPath）

快速首页包括其他一切功能的入口，没有在导航上提供入口的功能需要在此提供入口，如用户 Web 界面飘动信息，首页通知通告，内部提交，内部状态等。

2．题目控制台（Problems）

题目控制台包括题目列表，添加新题，修改已存在的题，查看题目详情。

题目列表：同一般用户的题目列表，每行多加修改及是否屏蔽的选项。

添加新题：输入信息包括题号，标题，总时间限制，单个测试文件时间限制，总内存限制，题目描述，输入描述，输出描述，样例输入，样例输出，提示，来源，相关比赛，相关文件，是否 Special Judge 等。

修改题目：修改题目的各个属性，如标题，描述等。

查看题目详情：显示题目的各个属性。

3．比赛控制台（Contests）

比赛控制台包括比赛列表，添加新比赛，修改已存在的比赛，查看比赛详情，管理虚拟比赛列表等。

比赛列表：同一般用户比赛列表，多加修改按钮。

添加比赛：输 入信息包括比赛号，标题，类型，描述，开始时间，结束时间，关联题目，相关文件等。

修改比赛：修改比赛的各个属性，如标题，描述等。

查看比赛详情：显示比赛的各个属性。

管理虚拟比赛列表：比赛的类型为虚拟（Virtual）。

4．结果判断控制台（Judge）

结果判断控制台包括一个带有搜索框和 Rejudge 功能的状态信息页。

5．用户管理控制台（Users）

用户管理控制台包括一个用户列表页及一个批量添加用户信息页。

6．讨论区控制台（Discuss）

讨论列表页有一个包含所有讨论主题的表，每行显示一个讨论主题的第一帖的主题，作者，关联的比赛和题目，发表时间,并有一个是否屏蔽此讨论主题的开关按钮。

3.2 系统总体架构

3.2.1 可分布式部署

本系统不仅要满足竞赛者的平时训练的要求，还要保证一定规模比赛的顺利进行。比赛进行时，会有大量的并发用户在线，这些并发连接给网络模块和 Judge 模块带来很大的压力，本系统实现网络模块和 Judge 模块的分布式部署，即在多台计算机上部署多个网络模块和 Judge 模块，提高服务质量，关于部署的具体方式在 3.2.2 节给出。

同时，可分布式部署显示了极大的适应性和灵活性，而且易于对系统进行扩充和缩小。系统中的功能部件充分隔离，中心处理模块集中处理数据流向。

3.2.2 模块划分

要实现在线评测系统的可分布式部署，同时维持数据的唯一性，必须进行模块的功能划分，模块划分的要求是：一，要实现功能逻辑上的模块化，二，各模块还要实现物理部署上的独立性，三，数据要保持一致。

基于以上考虑，下面对整个系统划分层次和模块。

以数据为对象进行层次划分，系统可分为三个层次，即网络交换层、数据控制层和内核层。网络交换层主要完成对用户数据的收集以及向外显示数据。数据控制层主要功能是进行数据分发，控制数据流向。内核层的主要工作是进行数据处理，反馈处理结果。

从功能方面出发，系统可划分为五个功能模块，即 Web 服务模块，用户终端模块，中

心服务模块，评测模块，数据库模块。

Web 服务模块和用户终端模块提供与用户的交互。其中 Web 服务模块即 Web 服务器，提供对外的 B/S 服务，用户可以在这里完成所有的功能，包括注册，登陆，查看用户、题目、比赛信息，提交源代码，以及发表讨论，收发邮件等等。用户终端模块是一个轻量级的客户端程序，用于在网络状况不好时提供一些基本服务，如登陆，提交源代码等。

中心服务模块是整个系统的中心，它提供与各个模块的接口，将各个模块有机的结合在一起，推动数据在各个模块之间的流动，并保持各个模块的性能平衡。例如，用户提交的源代码经由 Web 服务模块传递给中心服务模块，中心服务模块把相应信息写入数据库，然后从空闲队列里找到一个评测模块，将数据传递给它处理。得到处理结果后，在将信息经由 Web 服务模块显示给用户。

评测模块主要是对用户提交的代码进行编译，运行，输出比对，并反馈评测结果。

数据库模块管理着一个数据库和一些输入输出文件。数据信息包括用户信息，题目信息，比赛信息，邮件信息，讨论信息，结果信息等等。

下图(图 3-1)显示系统的总体架构：

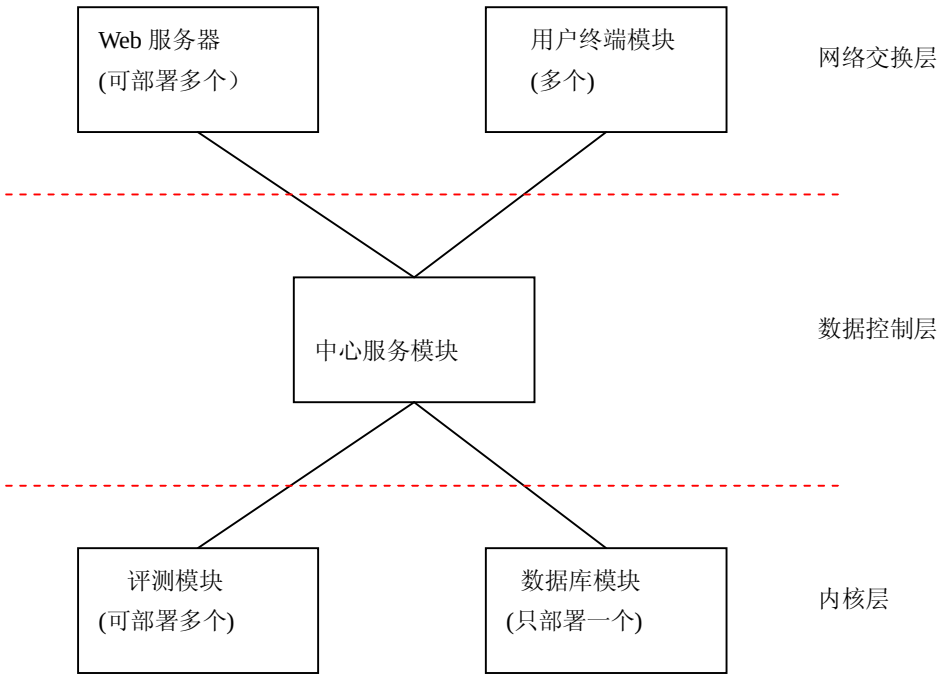

图 3-1 系统总体架构图

3.3 数据库设计

下面给出主要的数据库表的简要描述（带*号的为主键）：

1．用户表(Users)：*user_id（用户名），password（用户密码）， email（用户邮箱），show_email（是否显示用户邮箱信息）， submits（用户提交数），solves（用户通过的题目数），available（用户是否有效）， last_login_ip（用户上次登陆 ip）， last_login_time（用户上次登陆时间）， volume（用户上次题目卷号）， language（用户默认提交语言）， reg_time（注册时间）， nickname（用户昵称）， school（学校名称）， permission（权限字段）， share_code（是否共享代码）。

2．题目表(Problems)：*problem_id（题目编号），title（标题）， description（描述）， input（输入描述）， output（输出描述）， sample_input（输入样例）， sample_output（输出样例），hint（题目提示）， source（题目来源）， addin_time（加题时间）， time_limit（题目总时间限制），case_time_limit（单个测试用例时间限制），memory_limit（题目内存限制），available（题目是否有效）， accepted（题目被通过次数）， submit（题目被提交次数）， solved_users（通过该题的用户数目）， submit_users（提交该题的用户数目）， standard_time_limit（标准程序运行时间）， standard_memory_limit（标准程序运行占用内存）， version（题目版本号）， spj（题目是否 special judge）。

3．比赛表(Contests)：*contest_id（比赛编号），public_id（公开显示的比赛编号），title（比赛标题），start_time（比赛开始时间）， end_time（比赛结束时间），contest_type（比赛类型）， description（描述），version（版本号），available（比赛是否有效）。

4．邮件表(Mails)：*mail_id（邮件编号），topic_id（邮件主题编号），title（邮件标题），content（邮件内容），unread（邮件是否已读），time，to_user（接收用户），from_user（发送用户），reader_del（邮件是否被收件人删除），writer_del（邮件是否被发件人删除）。

5．讨论表(Discusses)：*message_id（讨论编号），reply_id（回复对象的编号），topic_id（讨论主题编号），user_id（用户名），problem_id（题目号），contest_id（比赛号），title（标题），content（内容），time（发表时间），available（是否有效）。

6．文件表(Files)：*file_id（文件编号），path（文件存放的路径），type（文件类型），disable（是否失效）。

7．错误信息表(Errors) ：*error_id（错误信息编号），content（错误信息内容）。

8．代码表(Codes)：*code_id（代码编号），share（代码是否共享），code_content（代码内容）。

9．新闻表(News)：*news_id（新闻编号），publishtime（新闻发布时间），title（标题），content（内容）。

10．提交表(Statuses)：*status_id（状态编号），user_id（用户名），problem_id（题目编号），contest_id（比赛编号），time（运行所用时间），memory（运行所占内存），submit_time（提交时间），result（运行结果），language（代码所用的编程语言），code_id（代码编号），code_length（代码长度），submit_ip（提交的 ip），error_id（错误信息编码），type（类型）。

11．参加比赛权限表(ContestPermissions)：*user_id（用户名），*contest_id（比赛编号）。

12．比赛题目隶属表(ProblemsToContests)：*problem_id（题目编号），*contest_id（比赛编号），in_contset_id（题目在对应比赛中的编号）。

13．题目相关文件(FilesToProblems)：*file_id（文件编号），*problem_id（隶属题目编号），version（题目版本号）。

14．比赛相关文件(FilesToContests)：*file_id（文件编号），*contest_id（隶属比赛编号），version（比赛版本号）。

3.4 Web 站点结构

Web 服务模块是用户与系统交互的接口，Web 服务模块的主要部分是运行在 Apache 上的 Web 站点，这里我们用 PHP5 开发。根据需求，Web 服务对象有两个，即一般用户和管理员，根据两者的功能需求，分别设计 Web 视图和功能接口。Web 包括两个入口，即一般用户入口和管理员入口，如图 3-2 所示。

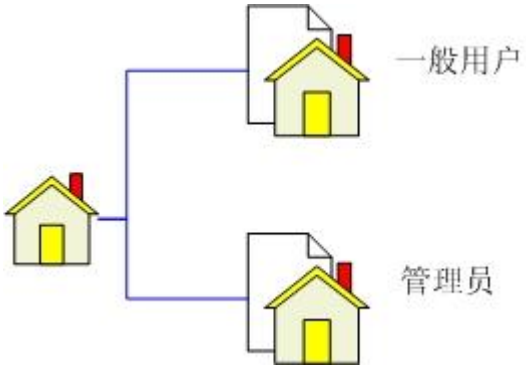

图 3-2 站点主结构

普通用户能查看的内容以及所享有的权限如图 3-3 所示。普通用户能查看的内有：主页新闻，题目列表，题目具体信息，比赛列表，比赛具体信息，比赛排名，题目提交状态，用户排名，用户信息，讨论区信息等。普通用户能使用的功能有注册，登陆，修改个人信息，提交代码，参加比赛，发表评论，收发信件，举办虚拟比赛等。

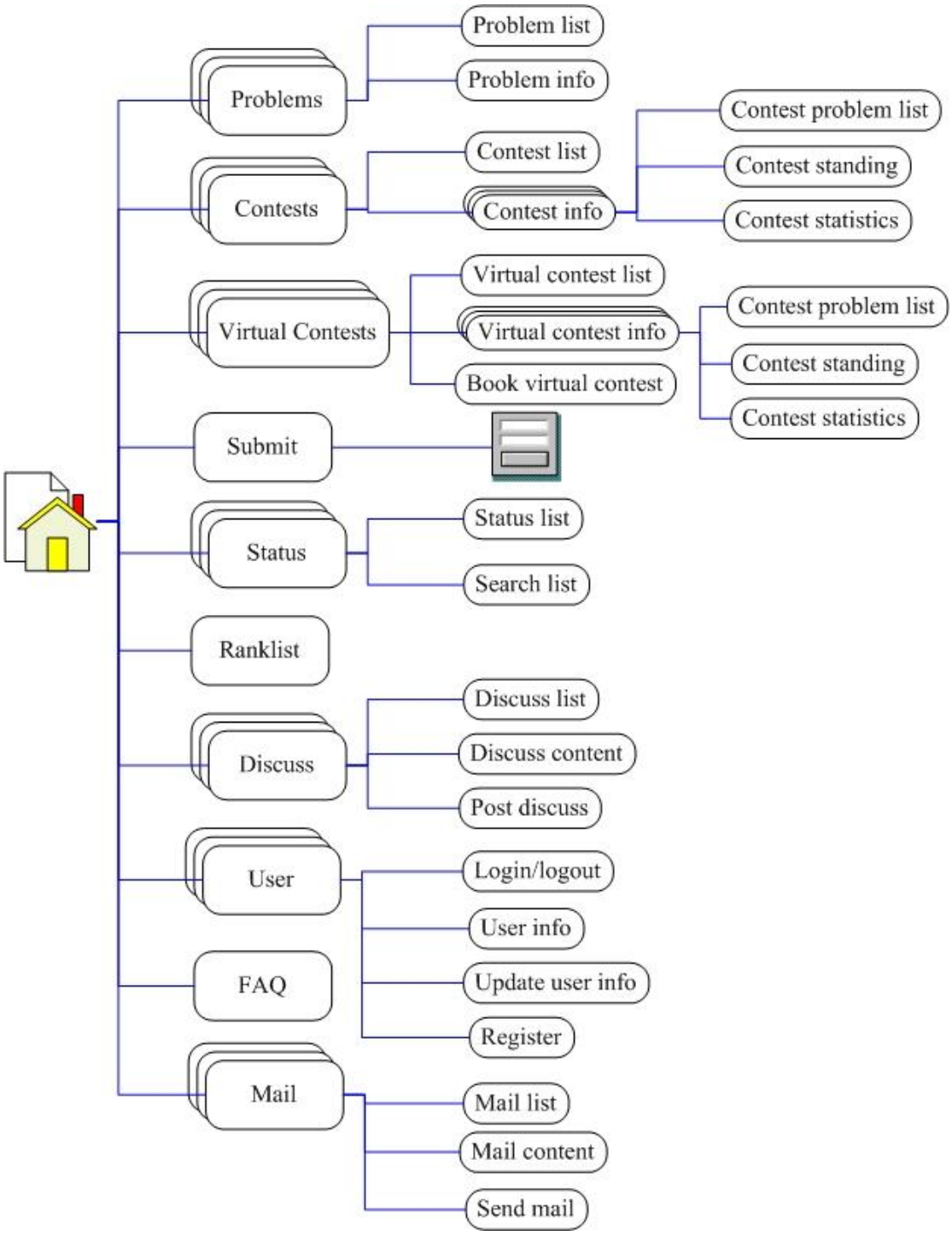

图 3-3 Web 站点结构(一般用户)

管理员能查看的内容以及能行使的功能如图 3-4 所示。管理员必须先登陆才能查看信息和使用相关功能。管理员能够查看的内容有题目列表，题目相关信息，比赛列表，比赛相关信息，讨论区信息，用户信息，实时提交状态等。管理员能行使的功能有设置飘动信息，添加新闻，添加/删除题目，添加/删除比赛，对题目和比赛进行编辑，重判提交，删除讨论信息，批量添加用户等等。

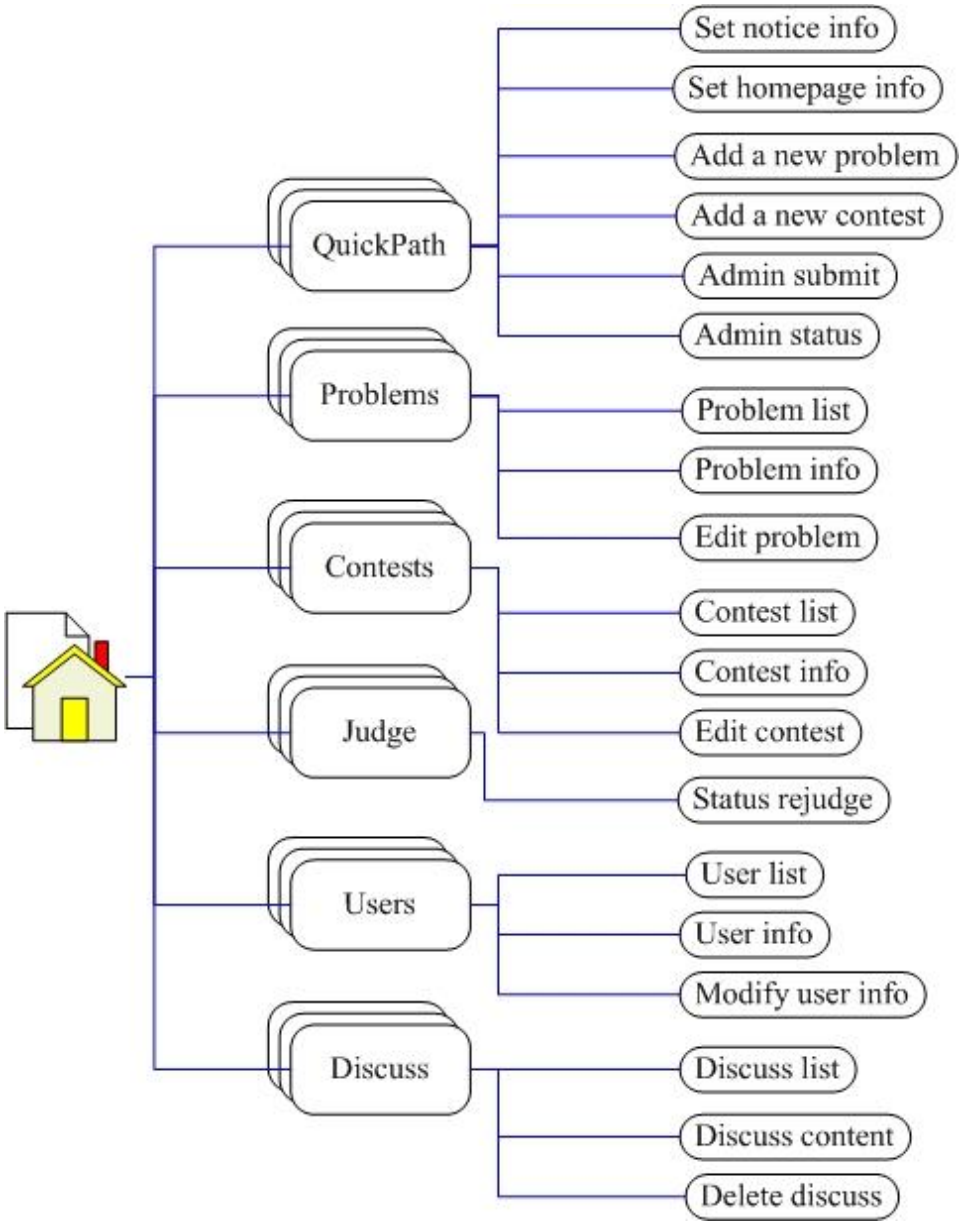

图 3-4 Web 站点结构(管理员)

3.5 数据通信接口协议设计

3.5.1 协议概述

作为可分布式部署的在线评测系统，需要确定各个模块之间的数据往来格式和传送方式。

模块之间的通信采用 C/S(Client / Server)的方式。从系统架构来说，Web 服务器和独立客户端可以看成一个整体，它们与中心 Server 通信，并且相对中心 Server 来说，它们是客户端(Client)，而中心 Server 是服务器(Server)。

从本系统的需求出发，为 C-S 之间的通信制定协议需要考虑四个方面：一，中心 Server

的唯一性，独立客户端与 Web Server 各自直接与中心 Server 通信，而中心 Server 以相同的方式处理来自这两部分的请求，即中心 Server 不会去识别请求来自哪里，其对外的数据接口是无差别的。二，编程方面的要求，为了区分内部请求种类，校验请求的合法性，以及保证传输数据的有序性和完整性。三，功能扩展的需求，制定有效的协议，有助于系统功能的扩展与升级。四，协议简单有效，复杂的协议只会加重网络负担和编程的难度，同时协议要保证实用有效。

3.5.2 协议实现

本节主要讨论 Web 服务器和独立客户端与中心 Server 之间的网络数据传输协议。

Web 服务器与中心服务器之间利用基于 TCP 的网络套接字（Socket）传送数据。进行数据传输通信时，发送方先将待传输的数据项打包，各个数据项之间以 ASCII 码（001）隔开，在发送数据包之前，先发送一个固定长度为 10 字节的协议头，协议头的内容包括两项，2 个字节标示请求类型，8 个字节标示待传输的数据包的长度。接收方先接收 10 字节的协议头，判断出请求类型和待接收数据包的长度，再接收数据包，并向发送方发出确认信息，然后根据协议类型对数据包做出相应处理。

下面给出部分数据传输请求的协议实现：

为了描述的简便和直观，先对一些名词做一些符号化的简化：

HEADER：表示一个协议头，意义如上所述。

C：表示 Web 服务器和独立客户端，相对中心 Server 来说，它们是客户端。

S：表示中心 Server，相对 Web 服务器和独立客户端来说，它是服务器。

PACK（）：表示一个数据包，括号内是数据包的具体内容，意义见数据库表字段意义。

ACK：确认信息，一个字节，Y 或 N。

->：表示数据的流向。

1．邮件列表（请求类型为 ml）

C->S ： HEADER

C->S ： PACK（user_id，pagenum）

S->C ： HEADER

S->C ： PACK（mail_id，to_user，from_user，title，in_date，read）

2．提交状态（st）

C->S ： HEADER

C->S ：PACK（pagenum，problem_id，user_id，result，language，contest_id，share_code，type，cur_user_id）

S->C ： HEADER

S->C ： PACK（solution_id，user_id，problem_id，result，memory，time，language，code_length，in_date，code_id，error_id，permission）

3．注册信息（rg）

C->S ： HEADER

C->S ：PACK（user_id，password，nickname，share_code，school，email，share_email，language ）

S->C ： ACK

4．题目列表（pl）

C->S ： HEADER

C->S ： PACK（pagenum，problem_id，title，source，contest_id，user_id，）

S->C ： HEADER

S->C ： PACK（pagenum，(problem_id，title，ac，total，result)）

5．更新用户信息（uu）

C->S ： HEADER

C->S ：PACK（user_id，old_password，new_password，email，show_email，nickname，school，share_code，language）

S->C ： ACK

6．提交代码（sm）

C->S ： HEADER

C->S ： PACK（user_id，password，problem_id，contest_id，language，code_length，share_code，ip，type）

C->S ： PACK（source code）

S->C ： ACK

7．上传输入输出文件（io）

C->S ： HEADER [：这里 herder 后 8 位为输入输出文件对数]

C->S ： HEADER [：这里 header 为 problem_id]

下面为一循环过程，逐一上传文件

C->S ： HEADER

C->S ： input file

C->S ： HEADER

C->S ： output file

S->C ： ACK

8．批量添加用户（su）

C->S ： HEADER

C->S ： PACK（number，(user_id，password)）

S->C ： HEADER

S->C ： PACK（y/n，...）

3.6 服务端守护程序实现

本系统中的中间层——中心服务模块从功能上可进一步划分为网络接口，评测控制和数据接口三个子模块，而其中的网络接口子模块则是本文要讨论的内容。

在这里，网络接口子模块实际上是一个服务器守护进程，它监听网络端口，接收来自网络（包括 Web 服务器和用户独立终端）的数据，对于来自 Web 服务器的请求，它从数据接口获得数据，并通过 socket 发送出去，对于来自 Web 的提交，它将接收的数据传递给处理模块。

网络接口和各个模块之间使用基于 TCP 的 Socket 通信。对于中心服务器来说，它接收的 TCP 连接量是非常大的，差不多每一次 Web 页面浏览就会带给中心服务器一次 TCP 连接，所有有必要采用多进程或者多线程技术来提高程序的并发性，而 Linux 正好是支持 POSIX 的 pthread 多线程编程的，于是本系统采用 TCP 多线程并发服务器的技术，提高网络服务质量。

由于创建一个线程要访问内核，对内核系统调用较多，所以这里采取预先派生一个线程池以取代为每个连接现场创建一个线程的做法，来进行性能加速。本模块的基本设计是预先创建一个线程池，并让每个线程各自调用 accept，用互斥锁以保证任何时刻只有一个线程在

调用 accept。

为了实现上述过程，需要一些全局变量，例如监听口套接字和一个由所有线程共享的互斥锁变量。下面先给出一个网络线程类的定义：

```
pthread_mutex_t socket_lock; //线程互斥锁变量
int listen_fd; //监听套接口描述字

class ProcessThread : public Thread {
public:
  ProcessThread(){}  //构造函数
  ProcessThread(int     listen_fd,     pthread_mutex_t*     lock)     :
  m_socket(listen_fd), m_lock(lock){
      flag = false;
  }
  virtual ~ProcessThread() {}  //析构函数

  void quit();
  void running();   //线程运行函数
private:
  int m_socket;   //监听套接口描述字
  ProcessImp* m_process_imp;  //数据处理类
  pthread_mutex_t* m_lock;  //线程互斥锁指针
  bool flag;
};
```

线程运行函数 running()的的工作过程是这样的，先调用 pthread_mutex_lock()对 accept()函数加以保护，在与客户的连接完成，accept()函数返回后，pthread_mutex_unlock()对 accept()解锁，接着完成数据的收发和处理工作。

主线程的工作过程是：创建监听套接字描述字，并对其绑定和监听，然后建立线程池，让各个线程各自阻塞于 accept()。主线程等待各子线程的返回。

由于篇幅原因，线程运行函数和主线程的具体实现将在附录中给出。

3.7 Web 安全策略的运用

在线评测系统的 Web 应用部分提供了很强的与用户的交互能力，同时也给 Web 应用以致整个系统的安全与稳定带来了各种各样的威胁。其实评测系统的架构本身就能防范一定威胁，所有的数据都保存在数据库和中心服务端，而 Web 服务端本身不接触数据库，它读取的数据都是经过自定义的网络接口协议从中心服务器获得，这样就把恶意用户和数据库给隔开了。

下面本文将从更多的方面，如输入的验证，防 SQL 注入，会话保护，加强主机安全性角度论述 Web 安全策略的运用。

1. 输入数据的验证

输入数据来自表单，URL 参数，cookies，应用程序期望使用某些特定数据作为参数，恶意用户有可能使用非法的输入，使应用程序出错或执行非法操作。如 URL：http://website/list.php?page=1 中的 page 字段表示页码，恶意用户可能修改该参数，使用-1 或者其他非数字符号 ，可能造成程序不可预期的错误，因此必须对这类输入数据进行验证，使用 PHP5 中的正则表达式函数可以很简单的实现一些验证，下表是一些验证方法：

表 3-1　Web 输入数据验证方法

需校验的项	正则表达式或 php 函数	备注
电子邮件地址	^\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*$	限制不正确邮件格式
用户名	^[a-zA-Z0-9]{6, 20}$	由 6 到 20 个字母数字组成

密码	/^\w{6,16}$/i	由 6 到 16 个字母数字组成，大小写敏感
HTML 格式标签	/<(\ /)?(b> | i> | u> | br>)/i	允许使用修改字符表现形式的 HTML 标签
	Ctype_alnum()	是否为英文字符或数字
	Ctype_alpha()	是否为英文字符
	Ctype_digit()	是否为数字
	Strlen()	检查字符串的长度

2. 防 SQL 注入

SQL 注入是一种特殊形式的输入验证，它通过发出原始 SQL 语句，试图操纵应用程序数据库，如 URL：http://website/login.php?user_id=magiii&password=asdfgh，会执行 SQL 查询：SELECT * FROM Users WHERE uid=magiii AND pwd=asdfgh，若恶意用户修改 URL，变成 URL：http://website/login.php?user_id=magiii&password=foo OR 1=1；查询语句就会变成：SELECT * FROM Users WHERE uid=magiii AND pwd=foo OR 1=1，这会返回一个 true，导致用户非法登陆。

防止这类攻击可以使用下面一些方法：

（1） 查找有漏洞的参数。测试像%00(NUL)、%27(‘)、%3b(;)这样的基本 SQL 注入字符。检查错误，以识别 SQL 注入。

（2） 防止使用 OR true=true 来绕过身份验证。

（3） 使用强类型的变量和数据库列定义

（4） 限制数据的长度，例如用户名限制在 20 字符以内。

（5） 使用 addslashes()进行字符串转义

3. 密码保护

本文的做法通过在客户端使用 Javascript 对用户密码进行 MD5 加密，可以防止密码在传输过程中或在 cookie 中被盗取。

4. 加强主机安全性

通过对 Apache 进行配置加强主机安全性。如 Apache 配置文件 httpd.conf 文件的<Directory>、<Location>及<Files>等指示符控制对文件系统的访问。下面的实例允许对目录的基本访问，包括列出其内容的能力：

```
<Directory “/user/local/apache/htdocs”>
   Options Indexes FollowSymLinks MultiViews
   AllowOverride None
   Order allow, deny
   Allow from all
</Directory>
```

默认时，应将 Options 设置为 None，以阻止列举目录、SSI 及脚本执行。

5. PHP 安全编码

在用 PHP 开发 Web 应用程序时，遵循下列编码规范：

（1） 禁用 php.ini 中的 allow_url_open，从而防止在 URL 中包含指示符。

（2） 使用 strip_tags()来防止 PHP 命令注入攻击。

（3） 使用 htmlspecialchars()来防止 SQL 注入攻击

（4） 使用 addslashes()来防止 SQL 注入攻击

（5） 留意用户提供的尝试运行 pssthru()或 exec()的数据

（6） PHP 包含文件必须有.php 后缀，而不是.inc 后缀。

（7） 当用户从应用程序注销时，使用 session_destroy()显示的结束会话。

第4章　　结果分析

4.1 利用 PHP 生成静态页面，来提高 Web 访问的效率

PHP 脚本是一种服务端脚本程序，它的基本工作方式是：客户端请求某一页面 à Web 服务器载入相应的脚本 à 由服务器指定的 PHP 解析器把脚本解析成 HTML 语言形式 à 将解析后的 HTML 语句以包的方式传回给浏览器。这种页面处理方式称为“动态页面”。静态页面是指在服务端确实存在的 HTML 以及 JS，CSS 等客户端运行的页面，它的处理方式是：客户端请求某一页面 à Web 服务器载入页面并以包的形式传递给浏览器。对比这两个过程，可以看出，动态页面需由 Web 服务器的 PHP 解析器进行解析，而且还需要连接数据库，进行数据库存取操作，然后形成 HTML 包，而静态页面无须解析，无须连接数据库，直接发送，可大大减轻服务器的压力，提高服务器负载能力，大幅提高页面打开速度。

对比动态页面和静态页面的优劣后，对于一些内容很少改动的页面，如题目描述页面，本系统采用 PHP 生成静态页面的方法，提高 Web 访问效率。

静态页面的生成一般有以下几个思路：

1．程序编写过程中，不使用直接输出的语句，而是将所有的输出连接至输出字符串，输出完成后，再直接将字符串内容写入文件

2．编写中按照正常的方式编写，通过 ob 函数组捕获输出，然后将输出写入文件。

3．使用模板类，用 str_replace()函数替换内容，并写入文件。

本文中使用的是第二种方法，该方法的好处是，使用方便，不影响正常的编程。

具体实现以上方法时，又有两种途径：

1．管理后台添加记录时，直接生成目标 HTML 页面，并且前台调用连接直接指向生成的 html 页面。这种方法程优点是程序效率最高。服务器负荷轻，不过由于生成的是纯静态页面，一旦页面样式上有所改动就必须重新生成所有的内容页。所以实际使用中应用一般不是太多。更多的是使用 JS，SSI，XML/XSL 等客户端手段，生成的静态文件中仅保存数据，不涉及样式，这样能达到速度和维护性的平衡，不过相对前后台程序要复杂些。

2．前台访问链接指向 PHP 程序，PHP 程序首先检查是否存在相应的静态文件，如果静态文件不存在，则生成并重定向至此文件，否则直接重定向。这种方法在效率上略有损失，不过程序结构简单，便于调整。

在本系统中，由于 Web 服务器是可分布部署的，即 Web 服务器有可能不止一台，所以方法一在添加记录时生成静态页面不现实，故采用上述的第二种方法。其实现过程如图 4-1 所示。

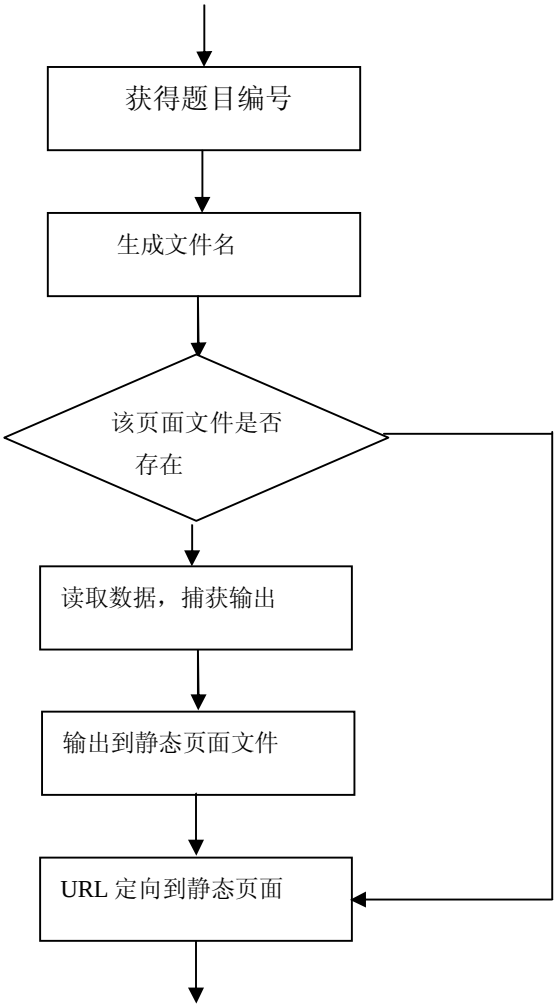

图 4-1 静态页面生成过程

4.2 Javascript 的运用

Javascript 是运行于用户客户端浏览器的脚本语言，在本系统的 Web 应用中，很多地方都用到了 Javascript。

4.2.1使用 Javascript 增加网页动态性

在讨论区的信息显示中，使用了 Javascript + CSS 的技术，使用户的讨论信息能够按树形结构显示。每个讨论主题下的信息按照回复关系组成树状结构,若某个信息有回复，则可以选择在此信息处进行折叠或展开，很大程度上增加了阅读的层次性和条理性。

4.2.2使用 Javascript 增强网页交互性

很多地方使用了 Javascript 来增强和用户的交互性。如添加题目描述页面是一个可即时预览的 HTML 编辑框，编辑框可编辑文字，图片，超链接等，它有两个模式，分别是编辑

模式和预览模式，用户可随意的切换，给编辑特定格式的题目描述带来了很大的方便。

4.2.3使用 Javascript 减轻 Web 服务器计算负担。

Web 应用中多处运用 Javascript 技术，在用户客户端完成一些复杂的计算，以此减轻 Web 服务器的负担。如使用 Javascript 对题目列表进行多关键字排序，对用户密码进行 MD5 加密等。

第5章 结束语

本课题涉及的在线评测系统目前已运行在武汉大学 ACM/ICPC 集训队内网服务器上。至论文完稿之日，本评测系统对于需求提出的绝大部分功能都已经完成且稳定运行在实际环境中，未完成的部分均为不影响正常使用的小功能改进和用户体验改善。

网络服务模块业已实现了预期的需求。对于普通用户来说，现在已经能够提供用户登陆/退出，用户信息管理，查看题目信息，查看比赛信息，提交题目，预订虚拟比赛，参加比赛，收发邮件，查看/发表讨论功能。对管理员来说，也实现全部的需求，包括添加、修改和管理题目和比赛，管理讨论、提交、用户信息等等。而且，对于 Web 服务器的安全也做了很多的工作，比如输入数据验证，防止 SQL 注入攻击，保护用户密码等等。在改善用户体验方面，也做了不少的工作，比如 Web 界面的设计，运用 Javascript 增加和用户的交互等等。

用户独立终端也已开发完成，现在，Web 服务器和独立用户终端与中心服务器之间连接也保持良好，数据传输畅通，快速。

当然，网络服务模块也有很多可以改进的地方，如 Web 服务器性能的进一步优化，可以考虑把更多的数据变化改动比较小的页面做成静态页面。而 Web 安全性也可以进一步提高，如可以做一个对 Apache 日志进行分析的工具，从 Apache 访问日志中我们可以发掘出很多安全方面的信息。用户界面的友好性和易操作性等方面也有进一步改善的地方。

可分布部署在线评测系统是一个复杂的系统，本系统的开发历时近两个月，在开发过程中综合运用了软件工程，编程语言，数据结构与算法，操作系统，网站美工等方面的知识。期间开发人员查看了大量的文档，并进行了很好的合作交流。

参考书目

[1](美)W.Richard Stevens, Stephen A. Rago 著, 尤晋元、张亚英、戚正伟译，UNIX 环境高级编程（第二版），机械工业出版社, 2006

[2](美) Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides 著，李英军、马晓星、蔡敏、刘建中等译，设计模式：可复用面向对象软件的基础，机械工业出版社，2000

[3](美)W.Richard Stevens, Bill Fenner, Andrew M.Rudoff 著，杨继张译，UNIX 网络编程（第一卷：套接口 API）（第三版），清华大学出版社，2005

[4]杜江编著，PHP5 与 MySQL5 Web 开发技术详解，电子工业出版社，2007

[5]Cbuck Musciano, Bill Kennedy 著，技桥译，HTML 与 XHTML 权威指南(第五版)，清华大学出版社，2003

[6]Mike Andrews, James A. Whittaker 著， 汪青青译， Web 入侵安全测试与对策， 清华大学出版社， 2006

[7]曹力 著，JavaScript 高级程序设计， 人民邮电出版社，2007

[8]李文新，郭炜，“北京大学在线评测系统及其应用”，吉林大学学报（信息科学版），2005，2

[9]吴海燕，苗春雨，刘启新，孙方成，“Web 应用系统评测研究”，计算机安全，2008，04

[10]苑文会，“基于黑箱测试的源代码在线评测”，北京化工大学，2006

附　　录

附录一 服务端网络守护进程

1．处理线程类的定义

```
class ProcessThread : public Thread {
public:
   ProcessThread(){}
   ProcessThread(int listen_fd,    pthread_mutex_t* lock) : m_socket(listen_fd), m_lock(lock){
         flag = false;
   }
   virtual ~ProcessThread() {}

   void quit();

      void running();
   bool check(const string& ip);
private:
      int m_socket;
      ProcessImp* m_process_imp;
   pthread_mutex_t* m_lock;
   bool flag;
};
```

2．处理线程类的实现

```
bool ProcessThread::check(const string& ip) {
   return Configure::getInstance().getNetWorkIpTabs().count(ip) > 0;
}

void ProcessThread::running(){
   //cout<<"Socket is "<<m_socket<<endl;
   int connect_fd;
   struct sockaddr_in childaddr;
   socklen_t len = sizeof(struct sockaddr);
   while(!flag){
      pthread_mutex_lock(m_lock);
      connect_fd = accept(m_socket, (struct sockaddr *)&childaddr, &len);
      pthread_mutex_unlock(m_lock);
      LOG(DEBUG) << "Connection happen";
      string ip = getIp(ntohl(childaddr.sin_addr.s_addr));
      if (!check(ip)) {
         LOG(WARNING) << "Unknown connection from:" << ip;
```

```
    close(connect_fd);
    continue;
}
LOG(INFO) << "Connection from :" << ip;
char buf[20];
if (socket_read(connect_fd, buf, 10) != 10) {
    LOG(ERROR) << "header reader error";
    close(connect_fd);
    continue;
}
LOG(DEBUG) << buf;
int type = (buf[0] - 'a') * 26 + buf[1] - 'a';
bool unknown = false;
switch (type) {
    case 448:    //rg
        m_process_imp = new RegisterProcessImp();
        break;
    case 294:
        m_process_imp = new LoginProcessImp();
        break;
    case 112:
        m_process_imp = new ExistUserProcessImp();
        break;
    case 401:
        m_process_imp = new ProblemListProcessImp();
        break;
    case 197:
        m_process_imp = new HomePageProcessImp();
        break;
    case 470:
        m_process_imp = new CodeProcessImp();
        break;
    case 90:
        m_process_imp = new DisableMailProcessImp();
        break;
    case 12:
        m_process_imp = new AddMailProcessImp();
        break;
    case 323:
        m_process_imp = new MailListProcessImp();
        break;
    case 314:
        m_process_imp = new MailContentProcessImp();
        break;
```

```
            case 391:
                m_process_imp = new ProblemProcessImp();
                break;
            case 487:
                m_process_imp = new StatusProcessImp();
                break;

            case 80:    //dc
                m_process_imp = new DiscussContentProcessImp();
                break;
            case 3:     //ad
                m_process_imp = new AddDiscussProcessImp();
                break;
            case 81:    //dd
                m_process_imp = new DisableDiscussProcessImp();
                break;
            case 540: //uu
                m_process_imp = new UpdateUserProcessImp();
                break;

            case 476:    //si
                m_process_imp = new SetUserInfoProcessImp();
                break;
            default:
                LOG(ERROR) << "Unknown type data.";
                close(connect_fd);
                unknown = true;
                break;
        }
        if (unknown)
            continue;
        //sendReply(connect_fd, 'a');
        int length = atoi(buf+2);
        m_process_imp->process(connect_fd, ip, length);
        delete m_process_imp;
        m_process_imp = NULL;
            close(connect_fd);
    }
}

void ProcessThread::quit(){
    flag = true;
}
```

3．主线程的定义

```
class Server{
public:
    Server();
    ~Server();
    void initServer();
    void start();
    void join();

private:
    pthread_mutex_t socket_lock;
    ProcessThread threads[MAXCLIENT];
    int max_client_;
    int port_;
    int listen_fd_;
};
```

4，主线程的实现

```
Server::Server() {
    pthread_mutex_init(&socket_lock, NULL);
}

Server::~Server() {
    pthread_mutex_destroy(&socket_lock);
}

void Server::initServer() {
    Configure configure = Configure::getInstance();
    max_client_ = configure.getNetWorkMaxClient();
    if (max_client_ > MAXCLIENT)
        max_client_ = MAXCLIENT;
    port_ = configure.getNetWorkPort();
}

void Server::start() {
    struct sockaddr_in server_addr;
    if ((listen_fd_ = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
        LOG(SYS_ERROR) << "Cannot create socket";
        exit(-1);
    }
    bzero(&server_addr, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    server_addr.sin_port = htons(port_);
```

```
    LOG(DEBUG) << stringPrintf("port:%d", port_);
    if (bind(listen_fd_, (struct sockaddr*) &server_addr, sizeof(server_addr)) == -1){
        LOG(SYS_ERROR) << "Cannot bind socket.";
        close(listen_fd_);
        exit(-1);
    }
    if (listen(listen_fd_, max_client_) == -1) {
        LOG(SYS_ERROR) << "Cannot listen the socket.";
        close(listen_fd_);
        exit(-1);
    }
    LOG(INFO) << "Begin listen socket.....";
    for (int i = 0; i < max_client_; i++){
        threads[i] = ProcessThread(listen_fd_, &socket_lock);
        threads[i].start();
    }
    LOG(INFO) << "Process Thread Pool create complete.";
}

void Server::join() {
    for (int i = 0; i < max_client_; i++){
        threads[i].join();
    }
    close(listen_fd_);
}
```

致　　谢

在我们完成毕业设计的过程中，有非常多师长，同学给予了我们帮助。

感谢我们毕业设计的指导老师董文永老师，给予了我们大量专业知识和求学处事的指导。并且身为武汉大学 ACM/ICPC 集训队的总教练，在我们参加 ACM/ICPC 竞赛的过程中也给予了大量的帮助和支持，对我们的学习生涯有重要的指导意义，并让我们在完成毕业设计的过程中发扬武汉大学 ACM/ICPC 集训队快速高效能吃苦的精神，在短期内完成了这一复杂的系统的设计实现和测试。

感谢武汉大学在线评测系统（noah 及 oak）的开发人员：杨传辉、吴永坚、刘高杰、杨宝奎、董超、高双星，他们的工作才奠定了本课题最原始的理论和实践基础，他们也在本毕业设计实现的过程中给出了大量建设性的意见和帮助。

感谢武汉大学 ACM/ICPC 集训队，给我们提供了开发的环境和平台，让我们能完成毕业设计并能将其部署以提供服务，感谢集训队的同学们给我们大量有重要价值的用户反馈和改进建议，让我们将毕业设计做的更好。

感谢共同开发的队员叶文和刘潜，我们的团队是最棒的。

感谢在我大学四年中遇到的各位老师，是你们辛勤的培养和教诲，才能有我们今天的知识储备和专业修养，在面对未来的学习和工作时能够有的放矢、得心应手。

马陈吉尔
二〇〇八年五月
于武汉大学 ACM/ICPC 集训队训练基地