Online Judge 系统的优化①

庄奇东[1,3]，王键闻[2]，张 楠[1,3]，张 爽[4]，任 娜[1]

[1](沈阳工程学院 信息工程系，沈阳 110136)
[2](沈阳工程学院 基础部，沈阳 110136)
[3](沈阳工程学院 电力系统信息安全重点实验室，沈阳 110136)
[4](昌兴建筑工程有限公司 ERP 软件事业部，东莞 523750)

摘 要：从 Web 页面和数据库缓存、服务器架构、多核评测处理规则、前端异步响应、数据表设计、跨平台支持、源代码抄袭检测、测试用例自动生成等方面优化了 Online Judge 系统，使得评测效率提高的同时减少了服务器数量，节约了运行成本。最后讨论了基于 Online Judge 系统实现智能优化算法的统一测试平台的方法。

关键词：Online Judge；缓存；多核；处理器亲和性；排队论；抄袭检测；测试用例自动生成

Optimization of Online Judge Systems

ZHUANG Qi-Dong[1,3], WANG Jian-Wen[2], ZHANG Nan[1,3], ZHANG Shuang[4], REN Na[1]

[1](Dept. of Information Engineering, Shenyang Institute of Engineering, Shenyang 110136, China)
[2](Dept. of Fundamental Sciences, Shenyang Institute of Engineering, Shenyang 110136, China)
[3](Key Lab of Information Security for Power System, Shenyang Institute of Engineering, Shenyang 110136, China)
[4](ERP Software Division, Changxing Construction Engineering Co., Ltd, Dongguan 523750, China)

Abstract: This paper describes the application and performance optimization of Online Judge Systems in terms of web page and database caching, server architecture, testing and processing rules in multi-core environment, front-end asynchronous response, data table design, cross-platform support, source code plagiarism detection, automatic generation of test cases, etc., which enhances the evaluation efficiency while reducing the number of servers, saving operating coses. And then, it discusses the general idea on the implementation of a unified test platform for intelligent optimization algorithms.

Key words: online Judge; cache; multi-core; processor affinity; queuing theory; plagiarism detection; test case auto generation

Online Judge 系统（简称 OJ）一般指在 ACM/ICPC（国际大学生程序设计竞赛）等各种形式的编程比赛中用来评测参赛选手的程序的正确性与时空效率的程序以及评测程序所依托的网络环境。一般包括以下几部分：Web 服务器、数据库服务器、编译程序以及评测程序。用户可以通过网络浏览服务器上的网页来选择题目，通过网页提交代码，由服务器端调用编译程序对用户提交的程序进行编译，然后由评测程序进行评判并返回相应的结果到网页上供用户查看。

1 OJ系统的历史和现状

最早用于程序设计竞赛的 Judge 系统是由西班牙 Valladolid 大学的 Ciriaco García de Celis 于 1995 年开发，用于为该校参加 ACM/ICPC 西南欧区域赛选拔队员[1]。十余年来经过数人的合作开发，UVa Online Judge 已被完善为一个具有比赛实时评判能力的功能强大、界面友好、统计数据齐全的分布式在线裁判系统。现在，基于它的 EduJudge 工程，亦即"Integrating Online Judge into effective e-learning"，已成为欧盟委员会基金支持下的，致力于推广终生学习和高效在线学习模式的 LifeLong Learning Programme 2007-2013 的一个核心部分[2]。

近年来，随着 ACM/ICPC 竞赛体制在国内的普及，众多定位各异的 OJ 系统也雨后春笋般地出现。2009

① 基金项目:辽宁省自然科学基金(20092045);沈阳市重点实验室建设资助项目(1091244-1-00);沈阳工程学院院内基金理工类(2009009)
收稿时间:2010-12-12;收到修改稿时间:2011-01-27

年课题组启动了 SIEOJ 项目，初始版本基于 Windows 和.net framework，后迁移到 Linux 平台上。

文献[3-6]或详细或简略地介绍了 OJ 系统的设计与实现，因此本文只着重介绍 SIEOJ 系统在多核系统中的实现难点、功能提升和性能优化方法，以及由 OJ 系统实现智能优化算法的统一测试平台的思路。

2 性能优化

2.1 前端异步响应

平时，OJ 系统的访问量并不大，这时一台普通服务器应负起所有的 http 服务、数据库服务、判题服务都绰绰有余。但是当遇到承办大型竞赛尤其是网络赛时，正常同时访问者上千并且还有可遭遇 DDoS 等形式的黑客攻击的巨大可能，所以需要服务器具有强大的并发处理能力。这样一般要把 Web 前端、数据库和评测程序分离，即至少置三台服务器。

处理大量连接时，Web 服务器常用的 I/O 复用机制有 IOCP(Windows), epoll(Linux), kqueue(FreeBSD)等模型。虽然从理论上来说，IOCP 是纯异步的，阻塞程度最低，但由于其所依赖的 Windows 系统的文件系统性能的局限性，其实际表现一般并不如 epoll 和 kqueue。表 1 显示了基于 Select 模型的 Apache、基于 epoll 和 kqueue 模型的 Nginx 和基于 IOCP 模型的 IIS 的实测对比（在 Intel Xeon 5520 双 CPU、4G RAM 上测试）。

表 1　Apache、IIS 和 Nginx 的测试比较

	最大并发连接数	每秒处理简单静态页面请求数/ index.php 响应时间(s)	
		无 eAccelerator	有 eAccelerator
Apache2	约 3000	3478/0.034	10917/0.014
IIS7	约 5000	3930/0.029	3891/0.029
Nginx0.8	约 55000	3779/0.028	14557/0.009

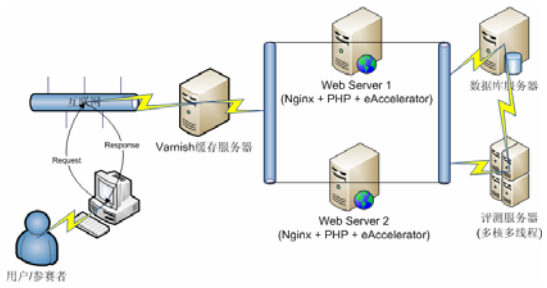

图 1　多核评测的系统架构

使用 eAccelerator 作为 PHP 脚本的预编译、加速和缓存器。其内存缓存的大小受限于 Linux 系统最大连续分配共享内存的大小，所以可以根据实际情况修改系统的/proc/sys/kernel/shmmax 来改变缓存容量。

Web Server 的前端置一台同样基于 epoll 模型的 Varnish 服务器，经测试，Varnish 缓存服务器还可减少约 90%的数据库查询。

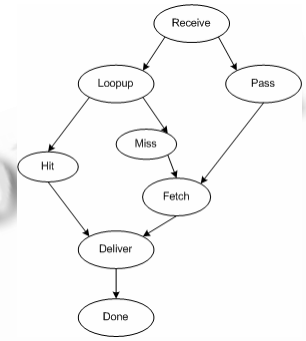

图 2　Varnish 的 http 请求状态图

2.2 前端异步响应

SIEOJ 采用 jQuery 实现 Ajax 技术，使得服务器和浏览器之间交换的数据减少到只有原来的 30%-40%，在得到更快的前端响应的同时减轻了服务器负荷。

Nginx 的配置也可告诉浏览器缓存静态文件。另外，由于使用了 Varnish 作为缓存服务器，jQuery 的 JavaScript 文件也被缓存到了内存中，所以当由新用户产生新的连接时，也不会造成服务器频繁的 I/O 操作。

2.3 多核评测

2.3.1 单核单线程环境下的时间测量

Linux 和 Windows 以及 Java 下都有系统函数调用可以获取当前进程/线程的运行时间。不过不同的系统实现都不用，而且用法也不同[4]。从测试程序段所在位置上讲，可分为在待测任务中插入测试代码和通过另外一个程序测试两种；从测试效果上讲，可分为间隔计数（各运行态累计得到 CPU 时间片之和）和周期计数（累计经过时钟周期数，即结束和起始时间之差）两种，从应用层次上讲，可分为用系统 API 测量和用低级语言直接访问 CPU 内的 8254 计数器测量两种。

图 3　单核系统下的物理时间

但是，在现代的计算机系统中根本不可能精确测

量某一个程序运行的确切时间。因为操作系统的进程和线程的分配和调度、CPU 的指令流水线处理等对于程序和用户来说都是透明的。在真实情况下计算机并不是只运行一个程序的，待测进程的优先级、调度策略、系统负载、同步或线程切换、各种中断、共享的多用户、网络流量、高速缓存的访问、转移预测等，都会对计时产生影响。

例如，现在的 Linux 和 Windows 等操作系统对线程进行抢占式调度，用户线程会因为时间片用完等原因而被中断。系统会从用户态转入核心态，即使在线程队列上没有其他线程，这种切换也会使用户线程的运行时间明显增加。

2.3.2 CPU 亲和性与任务优先级

在多核和多 CPU 系统中，情况更为复杂。又增加了伪共享的问题以及同一进程/线程在不同 CPU 核间切换的问题。

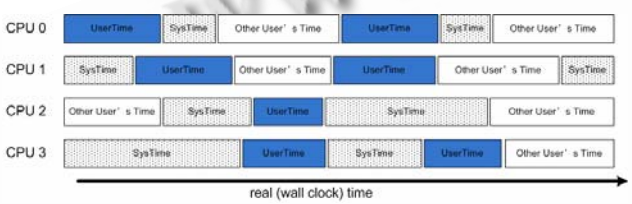

图 4 多 CPU/多核系统下的物理时间

CPU 读取 Cache 时是以行为单位读取的，如果两个硬件线程的两块不同内存位于同一 Cache 行里，那么当两个硬件线程同时在对各自的内存进行写操作时，将会造成两个硬件线程写同一 Cache 行的问题，它会引起竞争[7]，造成程序效率的成倍下降。

现代的许多支持多核处理器的操作系统，往往都有一种机制来维持各个 CPU 核心上的负载均衡。例如，在 2.6 版的 Linux 内核中便采用了一种简单方法—使用软中断来调整多核 CPU 上的压力，每秒一次。其大致基本原则是：从最繁忙的 CPU 核上挪一个任务到最空闲的 CPU 核上，如此下去直到负载均衡[8]。

对于伪共享问题，需要自己编写内存分配和管理的算法，十分复杂。对于核间切换的问题，虽然系统中其中还有一些优化算法尽量使切换时使任务尽量分配在同一个 Chip 上，但还是有较大可能使得切换后的任务丧失原来的指令和数据缓存、管线等“残留物”，甚至需要重新分配一些资源，造成额外的时间开销。

在多核 OJ 系统中，一个解决上述问题的最直接思路便是使评测服务器系统尽量不运行除评测任务外的其他任务，同时在维持各 CPU 核负载均衡的前提下尽量避免发生任务抢占和任务在不同核间的切换。

处理器亲和是在对称多处理器操作系统中本地中央队列调度算法的改进。每个任务（无论是进程还是线程）在队列中有一个标记，标明其首选/亲近处理器。在分配时，每个任务分配优先于其他处理器分配给其亲近处理器[9]。

在多核 OJ 系统的设计中，考虑利用处理器亲和性，以更大概率地保证被测的用户程序在单一的 CPU 和核心上执行，避免伪共享和任务在不同核间的切换，从而保证对用户程序执行时间测试的精确性。

表 2 不同操作系统下实现处理器亲和性的方法

	函数/API	命令行/UI
Windows	SetThreadAffinityMask()或 SetProcessAffinityMask()	TaskManager
Linux	sched_setaffinity()	taskset
FreeBSD	pthread_setaffinity_np()	cpuset
NetBSD	pthread_setaffinity_np()	psrset
MacOS	affinity API	----

另外，被测用户程序的优先级也会影响评测结果的准确性。如果太高，则调度的时间片少，因为操作系统的时间片轮转调度，一个运行时间大于调度时间片的线程会被中断；如果太低，则因为操作系统的优先级调度，会在大部分运行期间得不到 CPU 轮转的时间片。为确定采用哪种优先级会使对用户程序耗时的测量最准确，做了一组实验：随机选取 OJ 题库中的一道题目，将用户程序和多核系统的最后一个 CPU 核心建立处理器亲和，并将待测进程设置为不同的优先级，每次实验连续 10 次向 OJ 提交测试相同的正确代码，最后得到时间值。测试结果表明：采用最高的优先级运行程序得到的时间值最为稳定。

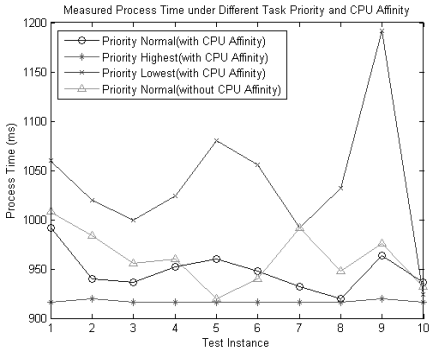

图 5 优先级、处理器亲和性与进程运行时间

猜想造成这个结果的原因是最后一个 CPU 核心较为空闲，即使被测用户进程时间片用完或被中断，也在很短时间内被重新调度而获得 CPU 时间片，缓存、指令管线等资源并未丢失且多次评测在相同条件下，从而得到了较强的一致性。

文献[10]说明了多 CPU 系统中，在未指定处理器亲和性的情况下，处理已有指派任务时，外部的中断往往被操作系统全部交与第一个 CPU 即 CPU0 来处理，这也在一个侧面支持了上述猜测。

文献[11]提出了 The K-Best Measurement Scheme 的测量方法，其中有一点是减少高速缓存不命中给我们程序执行时间带来的影响，即在测试之前先执行一遍程序，让指令高速缓存和数据高速缓存都得到“预热”。该文献也指出，任何一次单独的测量都无法精确获得进程运行时间，只有在多次运行后取最小的重复出现的若干个数据才能得到较为精确的值。然而这在高并发的 OJ 系统中是不可取的。但上述实验说明，在多核系统中，通过指定处理器亲和性，便可一次就能获得比较精确的测量结果。

2.3.3 多核评测模式

考虑使用一台多核评测机，指定操作系统的内核、judged 守护进程与第一个 CPU 核为处理器亲和，指定 Soulution_ID 为 i 的用户程序和 i mod (Total_CPU_core - 1)的 CPU 核亲和，并使 judge_client 进程和用户程序进程以最高的优先级运行。这样不仅尽可能保证了用户程序执行测试期间不会被其他进程抢占，也在一定程度上维持了系统的负载均衡。对于支持超线程的 CPU，可做适当调整，即把 Total_CPU_core－1 改为 2(Total_Logical_CPU_core-1)即可。其中 Soulution_ID 为用户提交的 ID 号，Total_CPU_core 为 CPU 的物理核心数，Total_Logical_CPU_core 为 CPU 的逻辑核心数。这些值可在 OJ 的配置文件中设定。

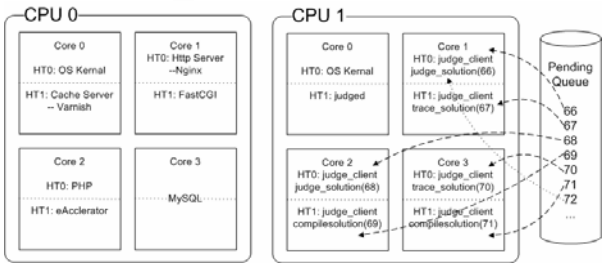

图 6　多 CPU 多核超线程系统上的可能瞬态

对于高性能的多 CPU 多核超线程的服务器且有双工模式的外存储设备如 SAS 硬盘，则 OJ 仅需要一台服务器：一个 CPU 上运行缓存服务器、http 服务器、数据库服务器，同样用处理器亲和性指定每个服务的亲和 CPU 核心；另一个 CPU 用来评测，规则如上。

对于大内存的 64 位机，可以用内存文件映射提高 I/O 速度，还可将比赛题目测试用例全部读入内存（或内存虚拟硬盘的方法），这样可消除绝大部分 I/O。

大量测试表明，采用以上策略，在多核系统中，对于相同的正确代码，Linux 和 Windows 下的时间测量值的差异一般不超过 4ms 和 1ms。

2.3.4 理论分析和进一步优化

图 6 看似只有一个用户提交队列，但其实是模 6 同余的 ID 的提交组成的三个队列，这可看成是一个多队多服务台的排队模型。大量的实际数据表明，在竞赛中，用户的提交事件近似为泊松流。按照排队论，若输入过程服从泊松分布，采用单队多服务台的排队规则比采用多队多服务台的工作效率高。

改进：设法变成多评测机单队列模型。增加一布尔数组 available[USING_JUDGE_CORE]，初始全为 1，当从待测队列中取出一条记录时，顺序搜索此数组，若无非零值则等待一小段时间再搜索，若有为 1 的值则记住该下标 i，将 available[i]改为 0，顺次启动编译、跟踪、评判程序，并将这些进程同 i 对应的 CPU 逻辑核心（下标不一定为 i，但一般与 i 具有线性关系）建立处理器亲和性，评测完成后，将数据库中写入记录并将 available[i]改回 1。另外，由于对单字节变量的赋值只对应一条汇编指令，具有原子性，所以这里无需考虑共享变量的互斥问题。

一般大型竞赛中一场约有 8000 次提交，评测机不够或策略不当往往会导致大量的待测队列堆积，严重影响选手水平的发挥。如 2007 年亚洲北京站点赛就产生了最高 20 分钟的待测序列。

选取 2008 年 ACM/ICPC 亚洲哈尔滨站点赛的统计数据，由于不能获得具体的提交和评测情况，按最坏的假设估计各种类型提交所需要的评判时间，算出两个分布的参数：平均到达率 λ =0.433/s，平均服务率 μ =0.128/s，根据排队论中的 M/M/C 模型不难算出此系统的其他性能指标，如表 3。

表 3 假设下各种不同的模型的性能指标

	4 评测机 4 队	4 评测机 单队	6 评测机 6 队	6 评测机 单队
服务强度 ρ	0.846	0.846	0.564	0.564
平均队长 Ls	21.937	7.119	6.776	3.586
平均等待队列长 Lq	18.554	3.736	4.373	0.203
平均逗留时间 Ws	50.697	16.453	17.927	8.287
平均等待时间 Wq	42.879	8.634	10.108	0.468
系统空闲概率 P0	0.154	0.019	0.436	0.033
等待时间小于 5s 的概率 P(tq<5)	0.234	0.541	0.573	0.971
逗留时间小于 10s 的概率 P(ts<10)	0.179	0.409	0.428	0.698

通过生成服从特定分布的序列作离散事件模拟得出的结果也与表 3 相似。从中可以看出，采用多机单队评测正是通过提高系统的利用率来减少了等待队列长度和等待时间。

此外，如果 Web 等其他服务器也是多核的且有计算资源剩余，则也可用上述方法利用其剩余资源处理一些判题工作。比之文献[5]所述的集群式 OJ，上述设计也极大地节省了硬件成本。

2.4 数据表设计与查询优化

在数据库系统中，采用外键能够保证数据库的数据完整性。然而，虽然在大型比赛时读写操作频繁，但 OJ 系统一般没有复杂的强约束性事务，所以采用外键一方面会造成系统并发处理性能下降，另一方面会使得系统遇到数据灾难时恢复困难。所以在 SIEOJ 数据库设计中，完全摒弃了外键约束，关键的几条完整性约束由外部的 C 和 PHP 语言程序来控制。

在大型比赛中，用户的提交队列数据表的访问时最关键且最频繁的，但一般而言这个表通常都是临时表，数据量不大且每条记录处理结束后都会被删除而处理结果会被写入其他表中去。因此，用 memory 存储引擎把此表建在了内存上。

OJ 中还有一些数据表的数据量较大而读操作比较频繁，因此在关键的列上建立了 Hash 或 B 树索引。

2.5 安全性保障

由于需要执行用户提交的代码，所以在 OJ 系统运行中的安全性是一个必需考虑的因素。目前，在已有的 OJ 系统中，常用到的安全策略有：1) 敏感字符串过滤[6]，如对于 while(1) fork();、system(“shutdown”);等会对系统造成危害的恶意代码，如果在对用户提交的代码扫描后检测出来，则不编译用户提交的程序，数据库中亦记录下有关信息。常用的检测手段为正则表达式匹配；2) 管道技术，在 Linux 系统下为了让用户提交上来的程序不破坏服务器的文件系统，在执行用户程序之前，先把输入流定向到标准输入文件，然后使用 chroot 改变用户程序的执行目录，让其只能在一个临时目录下面做操作；3) 沙箱技术[12]，使用户程序在创建的“Sandbox”环境中运行，与系统中的其它部分隔离开来，从而使用户程序对操纵系统、编译器与其它服务器组件等做的更改在程序结束后完全无效。但是对于方法 1)，用户还有可能用宏定义，或字符串分拆后拼接等特殊手段突破字符串过滤技术，工作量大且对程序员要求高。对于方法 2)和 3)，容易被突破，并不能有效保证系统安全。有人提出了底层的 API Hook 方法，但难度过大。

在 SIEOJ 的实现中，兼用了字符串过滤和管道技术，并增加了跟踪用户程序的方法，即先运行并测试一遍用户程序（ptrace()），如果遇到白名单中没有的调用，则强行终止用户程序，状态置为 Runtime Error，反之则再测试一遍用户程序，得到准确的时间。一般程序设计竞赛中能够用到的系统调用非常有限，所以白名单的创建也较为简单。这样做还可以“预热”高速缓存从而得到更精确的时间，第二次测试时也可省去许多处理，比如格式错的判断。

3 功能增强

3.1 跨平台支持

前台，使用标准的 HTML，使得在不同浏览器上呈现出一致的效果。为了使使用智能手持式终端设备的用户也能够访问，专门设计了不含 JavaScript 脚本且节省流量的 wap 版本，同时提供了 gcc 编译器的移植版本及其 IDE[13]，使得用户在智能手机上也能够编写代码、调试程序并向 OJ 提交。

后台，为了能够充分利用老版本基于 Windows 和.net 的 OJ 中的题目，设计了用 XML 文件自动导入和导出题目的方法。OJ 内核评测模块也很好地处理了如何跨平台的问题，比如不同系统下时间测量单位不同的问题和测试用例文件格式的问题。因为 Windows 和各种类 Unix 系统对于线程的处理机制不同，线程切换所需的代价的差异较大[14]，所以 OJ 的评测模块采用了多进程模型而不是多线程模型。

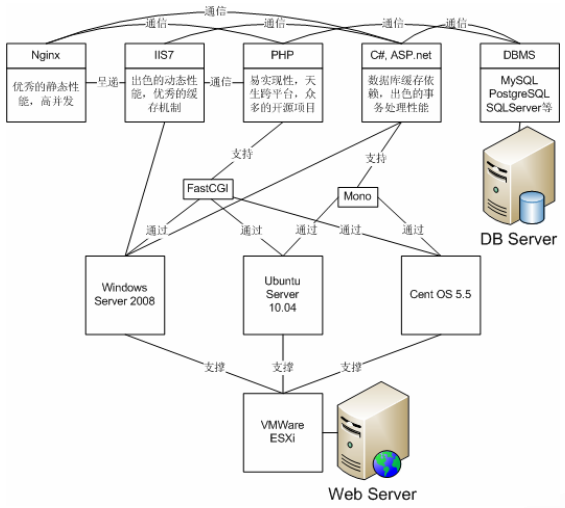

图 7　Web 和数据库服务器

为在 Linux 下也能够支持 C#和 ASP.net，使用了开源项目 mono。为了提高 Windows 下页面处理和数据库查询的效率，实现类似于 Varnish 的效果，ASP.net 采用了数据库缓存依赖以提高缓存命中率。.net 下利用了微软的 C#沙箱开源项目为系统的安全性提供了更多一重保障。同时运用了虚拟化和云计算的一些技术，比如使用 VMware 的免费软件 ESXi，在一台服务器上同时支持多个异构的操作系统来提供服务。

3.2 测试用例自动生成

比赛中的题目往往需要大量的测试用例，而且往往需要某些特殊的用例，能够将可解区域覆盖均匀且全面、有效涵盖特殊边界或能很好地测试出用户考虑不足的地方。然而，即使知道一道题目的正确解法和正确代码，人工编写成千上万组测试用例也是一件十分繁重的工作，而且很难进行验证。

SIEOJ 基于俄罗斯开源项目 testlib 实现了一个测试用例的自动生成器，用类似于正则表达式的语言描述数据特征和规则，即可让出题者方便地生成大量所需的测试用例。

3.3 代码抄袭检测

OJ 系统不仅用于竞赛，还常常用来作为众多计算机课程的教学和考试平台，因此在这些应用背景下，需要系统具有代码抄袭自动检测功能。

SIEOJ 在实现代码抄袭功能时选用了开源项目 sim2.26[15]，先通过 Unix 的 lex 命令对用户程序做词法分析，将程序源代码表示为另一种紧缩的格式，建立向前引用表，再采用一种检测 DNA 序列相似性的算法，将紧缩串分成若干份，每份到另一个紧缩串中进行匹配，匹配出的子串被删去，直到不能再发现匹配子串。系统只将当前用户提交代码与对应题目的以往其他用户的正确代码进行匹配，相似度若大于某个阈值则在数据库中留下相关记录。该功能为可选，只在需要时启用，大型比赛时关闭，以防影响速度。

4　由OJ实现智能优化算法的统一测试平台

近年来，智能优化算法一直是计算机科学界的一个研究热点，每年都有许多学者发表许多有关的论文，大部分都涉及对一些算法解决某类问题的改进，论文中常常出现一些测试比较的数据。然而这些数据并不一定很客观，因为不同的结果有可能不是在相同或相似条件下测得，由前文可知他们测定的方法也不一定合理（如通常只取两个时间值相减，这样往往会得到不正确的程序运行时间），论文作者也有可能对测试结果做一些人为的筛选以突出自己算法的高效性。因此，建立一个智能优化算法的统一测试平台很有必要。

不同于通常 OJ 中的题目的正确程序，智能优化算法的程序往往需要运行数十秒、数分钟甚至数小时，而且现在的智能优化算法的实现往往是并行的程序，因此需要有针对性地改进 OJ 系统。

首先，为 OJ 系统增加更多编译器的支持，如支持 OpenMP、TBB 等并行模型的 Intel C++和 Fortran 编译器（两者的 Linux 版本都是免费的），其次，限制 OJ 同一时间只能评测一个用户程序，其他的提交放到等待队列上，再次，每个程序只运行一次，无需“预热”。

因为世界各地从事智能优化算法研究的高校和科研机构有众多的测试用例提供下载，所以目前首先初步实现了对于 TSP 问题的统一在线测试。

5　结语

数本文所述的方法不仅适用于 OJ 系统，还可用于实时性要求较高的其他 Web 服务和编译应用，所用第三方组件均为免费或开源解决方案，而且实现方法简单、易行、有效。

参考文献

1 Revilla M, Manzoor S, Liu RJ. Competitive learning in informatics: the UVa online judge experience. Olympiads in Informatics, 2008,2:131−148.

2 Luisa MR, Elena V, Juan PC, María ÁP, et al. A proposal of user interface for a distributed asynchronous remote evaluation system: An evolution of the QUESTOURnament tool. Proc. 9th IEEE Int'l Conf. on Advanced Learning Technologies. Riga, 2009.75–77.

3 康海燕,樊孝忠,汤世平.基于 J2EE 的在线测评系统的研究与设计.计算机工程,2004,13:169–171.

4 何静雯.ACM/ICPC 评测系统综述.计算技术与自动化,2005,4:405–409.

5 王辉,胡新华,张广泉.集群式程序设计竞赛评测系统设计与开发.计算机应用与软件,2009,9:119–122.

6 李哲.在线程序竞赛评判系统的设计与实现[硕士学位论文].大连:大连理工大学,2008.

7 周伟明.多核计算与程序设计.武汉:华中科技大学出版社,2009.

8 董昊.Linux 在多核处理器上的负载均衡原理.淘宝核心系统团队博客. [2010-11-11]. http://rdc.taobao.com/blog/cs/?p=379.

9 Wikipedia. Processor Affinity. [2010-11-11]. http://en. Wikipedia.org/wiki/Processor_affinity.

10 Foong A, Fung J, Newell D. An in-depth analysis of the impact of processor affinity on network performance. IEEE Transactions on Networks, 2004,1:244–250.

11 Hollinger D.Time Measurement. [2010-11-11]. http://www.cs.rpi.edu/~hollingd/comporg/notes/timing/timing.pdf.

12 Ahmed SA, Muhammad AR, Shusmita AS, et al. Secured programming contest system with online and real-time judgment capability. Proc. of the 8th Int'l Conf. on Computer and Information Technology, 2005.

13 张胜,洪明.基于 Pocket PC 的 IDE 设计与实现.计算机系统应用,2008,17(11):14–19.

14 Reinders J. Intel Threading Building Blocks: Outfitting C++ for Multi-core Processor Parallelism. O'Reily, 2007.

15 Grune D. The software and text similarity tester SIM. [2010-11-11]. http://www.few.vu.nl/~dick/sim.html.

(上接第 75 页)

证明，基于概念的向量空间模型比基于词语的向量空间模型的聚类分析性能更好。本文中还有很多值得进一步研究的地方。对于情感词概念的选取，本文直接选用了第一义项，没有具体的概念排歧工作。在使用 k-means 算法进行聚类分析时，每次随机选取的簇中心不一样，使得聚类的结果不稳定。今后需要在情感词概念提取以及算法优化方面加强研究，使聚类结果性能更好。

参考文献

1 杨勇涛.Web 舆情观点挖掘关键技术研究.成都:电子科技大学,2009.

2 董振东,董强.HowNet's HomePage.Http://www.keenage.com, 2010.

3 夏天,樊孝忠,刘林.利用 JNI 实现 ICTCLAS 系统的 Java 调用.计算机应用,2004,24(12):177.

4 胡静,蒋外文,朱华.Web 文本挖掘中数据预处理技术研究.现代计算机(专业版),2009,(3):48-50.

5 陈龙,范瑞霞,高琪.基于概念的文本表示模型.计算机工程与应用,2008,44(20):162-164.

6 刘金岭.基于语义的高质量中文短信文本聚类算法.计算机工程,2009,35(10):201-202.

7 廖浩,李志蜀,王秋野等.基于词语关联的文本特征词提取方法.计算机应用,2007,27(12):3010.

8 Liu B. Web Data Mining. Chicago: Springer Press, 2006. 120-121.

9 游春晖.基于语义情感倾向的文本相似度计算.西安:电子科技大学,2008.

10 Pang J, Xu DP, Feng S, et al. A Novel Clustering Approach Based on Graph Similarity for Chinese Blogs on Authors' Sentiment. The 7th International Conference on Fuzzy Systems and Knowledge Discovery. Yantai: Yantai University Press, 2010. 2344-2348.