评测鸭：
稳定精确的评测系统如何改善编程教育

王逸松

清华大学计算机科学与技术系

wys19@mails.tsinghua.edu.cn

内容提要

稳定评测的重要性

评测鸭：稳定精确评测系统的实现

评测鸭如何影响编程教育

内容提要

稳定评测的重要性

评测鸭：稳定精确评测系统的实现

评测鸭如何影响编程教育

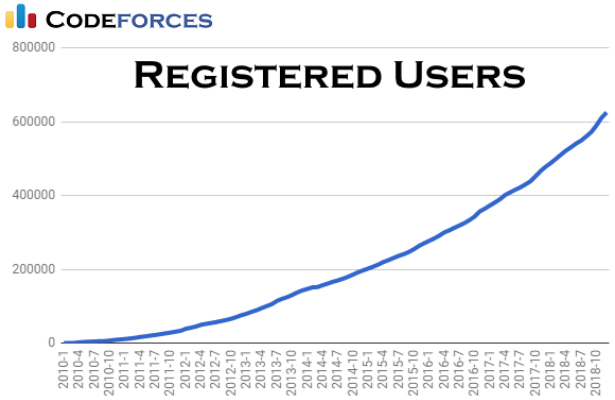
编程竞赛和训练

人工智能和互联网的快速发展

→ 编程教育得到高度关注和重视

→ 编程竞赛和训练的需求快速增长

* 编程/算法竞赛、课堂教学、技术面试



Tsinghua Online Judge

Department of Computer Science and Technology, Tsinghua University

Welcome!

Quick guide, video guide and PA handbook are available.

35,971 registered users, **14** active courses, and **94** closed courses.

878 test runs in the last 24 hours.
8,824 test runs during last week.
1,319,419 test runs since 09/01/2012.

力扣 探索 题库 圈子 竞赛 面试 职位 商店

2019 力扣杯全国秋季编程大赛 排名

排名	用户名	语言	得分	完成时间	题目 1 (2)
1526	maodouzhendou	Python3	2	2:10:07	2:05:07
1527	SinceQoc	Java	2	2:10:47	2:10:47
1528	xing-103	C	2	2:10:49	2:10:49

codeforces.com, dsa.cs.tsinghua.edu.cn/oj, leetcode-cn.com, 2019-09-28

编程竞赛和训练

人工智能和互联网的快速发展

➔ 编程教育得到高度关注和重视

➔ 编程竞赛和训练的需求快速增长

* 编程/算法竞赛、课堂教学、技术面试

编程竞赛和训练的主要形式是编程任务：

* 编写一份程序，解决一个实际问题

编程任务的评价方式：

* 对程序进行**评测**

评测系统

通常情况下，评测由评测系统自动化地完成。

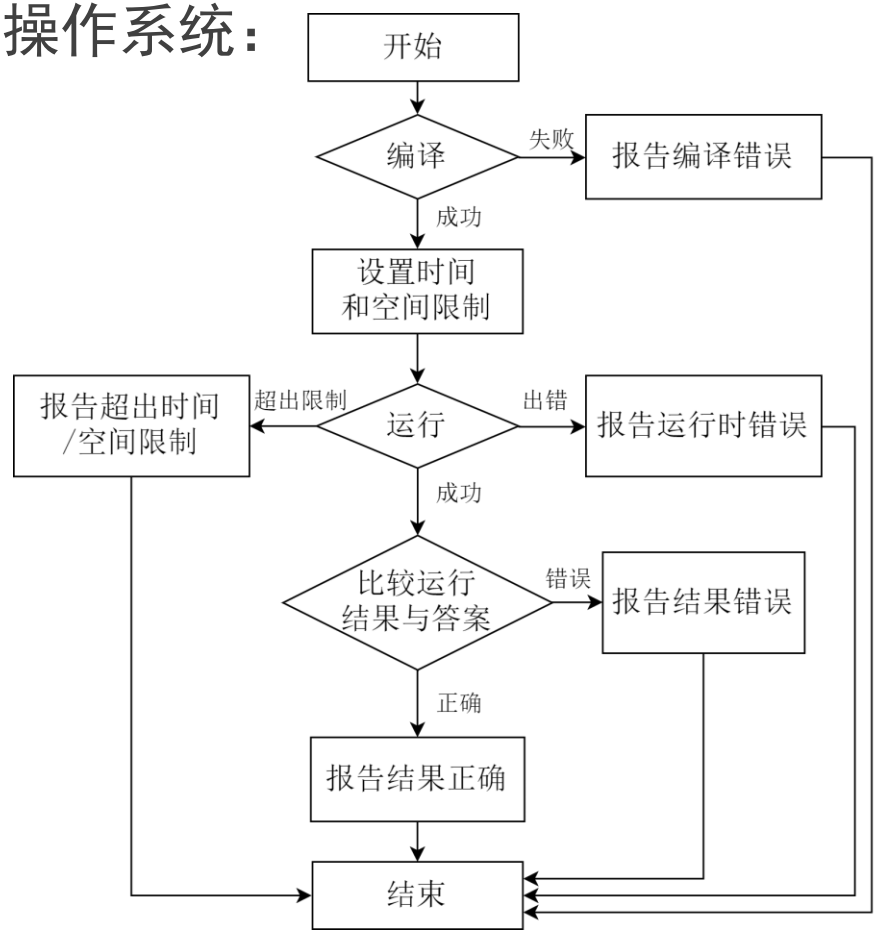
评测系统能接受一个源程序和一些数据作为输入，并测试：

1. 源程序能否成功编译
2. 对于每组数据，程序能否在一定的时间和空间内运行完成
3. 程序的运行结果是否正确

传统评测解决方案

传统评测系统通常使用 Linux 操作系统：

* Linux 提供了完整的工具链，包括编译、运行、时间/内存测量
* Linux 也是工业界最常用的操作系统之一



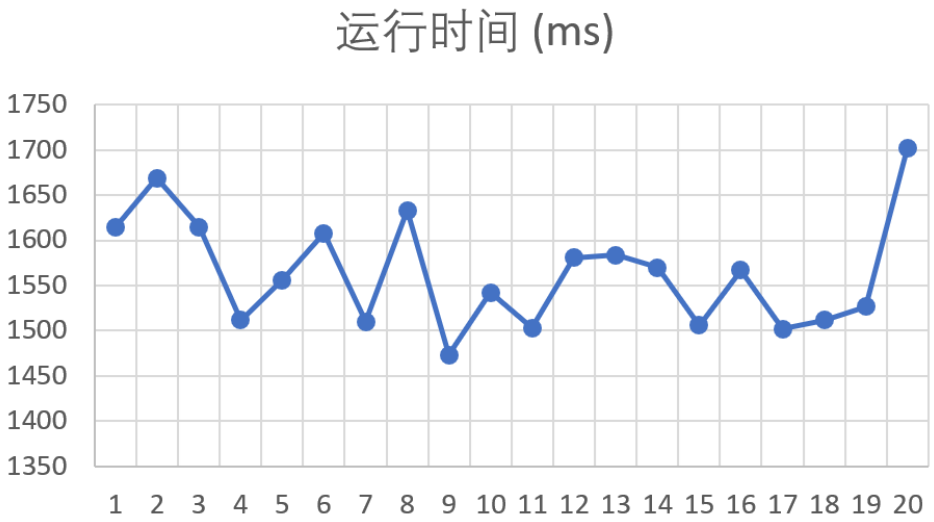
此方案已有近20年的实践，已成为评测解决方案的事实标准。

评测结果稳定性问题

在评测系统中，一个核心任务是测量选手程序的运行时间。

作为一种测量工具，误差或不确定因素必然存在。

如果误差较大，则足以影响评测系统对选手程序的评价。



数据	值
平均数 (ms)	1564.3
标准差 (ms)	61.6
极差 (ms)	229
标准差 (%)	3.94%
极差 (%)	14.64%

https://vijos.org/d/gggg/records/5d8dbbf9f41362044aac021f, 2019-09-27

评测结果稳定性问题

在评测系统中，一个核心任务是测量选手程序的运行时间。

作为一种测量工具，误差或不确定因素必然存在。

如果误差较大，则足以影响评测系统对选手程序的评价。

事实上，基于 Linux 的评测系统有时会产生10%以上的随机误差，在极端情况甚至会产生数倍于运行时间的误差。

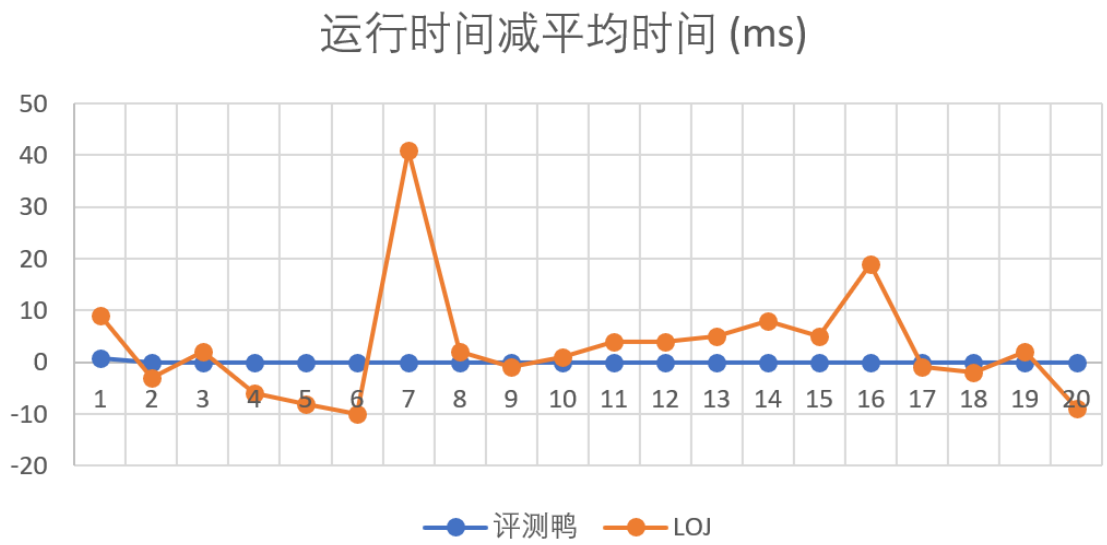
由于这可能会影响重要竞赛的成绩，因此，实现一个稳定的评测系统非常重要。

在过去国内外关于评测系统的研究[1-3]中，对评测结果稳定性问题的探讨不够充分。

评测鸭：稳定精确的评测系统

评测鸭实现了：

* 稳定：多次测量同一程序，误差不超过 1%+1微秒
* 精确：时间测量精度达1纳秒（10^{-9}秒）
* 兼容：对于 C/C++ 语言，支持源代码级别兼容



https://duck.ac/submission/7130, https://loj.ac/submission/618349, 2019-09-27

内容提要

稳定评测的重要性

评测鸭：稳定精确评测系统的实现

评测鸭如何影响编程教育

评测系统做什么

编程任务：实现一个完整的程序或程序的部分，来解决一个实际问题。其中，可以使用编程任务提供的函数或库。

* 例子1：实现一个完整的程序，从标准输入读入 n 个整数，将其相加并输出到标准输出。
* 例子2：实现一个函数 sort(a[], n)，对 n 个元素进行排序，其中可以使用 cmp(a, b) 函数进行元素比较。

评测系统将测试：

* 编译是否成功
* 运行时间和空间是否不超过题目限制
* 运行结果是否正确

现有评测系统及稳定性问题

现有评测系统使用 Linux 操作系统：

* 编译：g++ a.cpp lib.cpp -o a
* 运行：./a < in > out
* 时间/内存测量：getrusage
* 答案正确性：diff out ans （或自定义比较）
* 资源限制：ptrace，namespace/cgroup 等

现有评测系统存在的稳定性问题：

* 对相同程序相同数据进行多次测试，测得时间可能相差10%以上。
* 是什么因素导致了不稳定？这些因素可能消除吗？

为什么不稳定

被测程序在 Linux 操作系统上运行：

* 中断的影响：进程切换、系统调用、缺页异常带来不确定影响。

  约1us/中断（或异常），若所有中断耗时随机且独立，则每秒影响数ms。

* 资源的共享：其他正在运行的程序、设备占用内存和总线带宽等资源，使被测程序运行时间不稳定。

  在Linux中同时运行多个程序，再运行被测程序，可看到数倍于运行时间的影响。

* 缓存的性质：CPU最后一级缓存（LLC）基于物理地址，而进程使用虚拟地址，因此操作系统对虚拟地址到物理地址映射的不确定性导致了缓存作用的不确定性。

  在极端情况（如步长为缓存大小的访存）下，一个程序的缓存命中率每次运行时都可能有较大变化。这会影响运行时间数倍。

如何提高稳定性

被测程序在 Linux 操作系统上运行：

* 避免中断的影响：运行被测程序时关闭中断，且尽可能避免缺页异常。

  需要改变操作系统的中断机制

* 防止资源的共享：只运行被测程序，关闭其他程序和设备。

  关闭其他无关进程，关闭DMA设备，使被测程序独占CPU资源

* 降低缓存的不确定性：每次运行被测程序时，固定虚拟地址到物理地址的映射。

  需要改变操作系统的内存管理机制

如何实现？更改已有操作系统，还是，重写操作系统？

方案设计

重写操作系统：

* 消除 Linux 的各种不稳定性较为困难。
* Linux 系统本身过于复杂，而实现一个“评测专用操作系统”较为简单。
* 若自己实现操作系统，需考虑与现有评测系统的兼容性：
  * 评测鸭：源代码级别兼容

基于 MIT JOS 项目：

* 这是一个支持多进程、多核、文件系统、网络的微内核教学操作系统。
* 关闭多核，增加“评测”相关的系统调用，并修改内存分配和进程调度机制。

** MIT JOS: https://pdos.csail.mit.edu/6.828/2016

模块设计

操作系统内核（内核态）

* 提供操作系统的基本功能：内存分配、进程调度等
* 提供评测相关的系统调用
* 保障评测时的资源管理

评测服务进程（用户态）

* 随操作系统启动，持续接收并执行评测任务
* 使用评测相关的系统调用

被测进程（用户态）

* 每次评测时由评测服务进程启动
* 读取必要的数据，调用被测函数，最后输出答案正确性等结果

其他进程（用户态）

* 实现网络等功能

系统调用设计

sys_enter_judge(void *eip, struct JudgeParams *prm)

* 声明该进程要进入评测状态，由被测程序调用。
* 调用后系统将被测进程挂起。
* 参数：被测函数的指针，时间、内存限制，允许使用的系统调用列表。

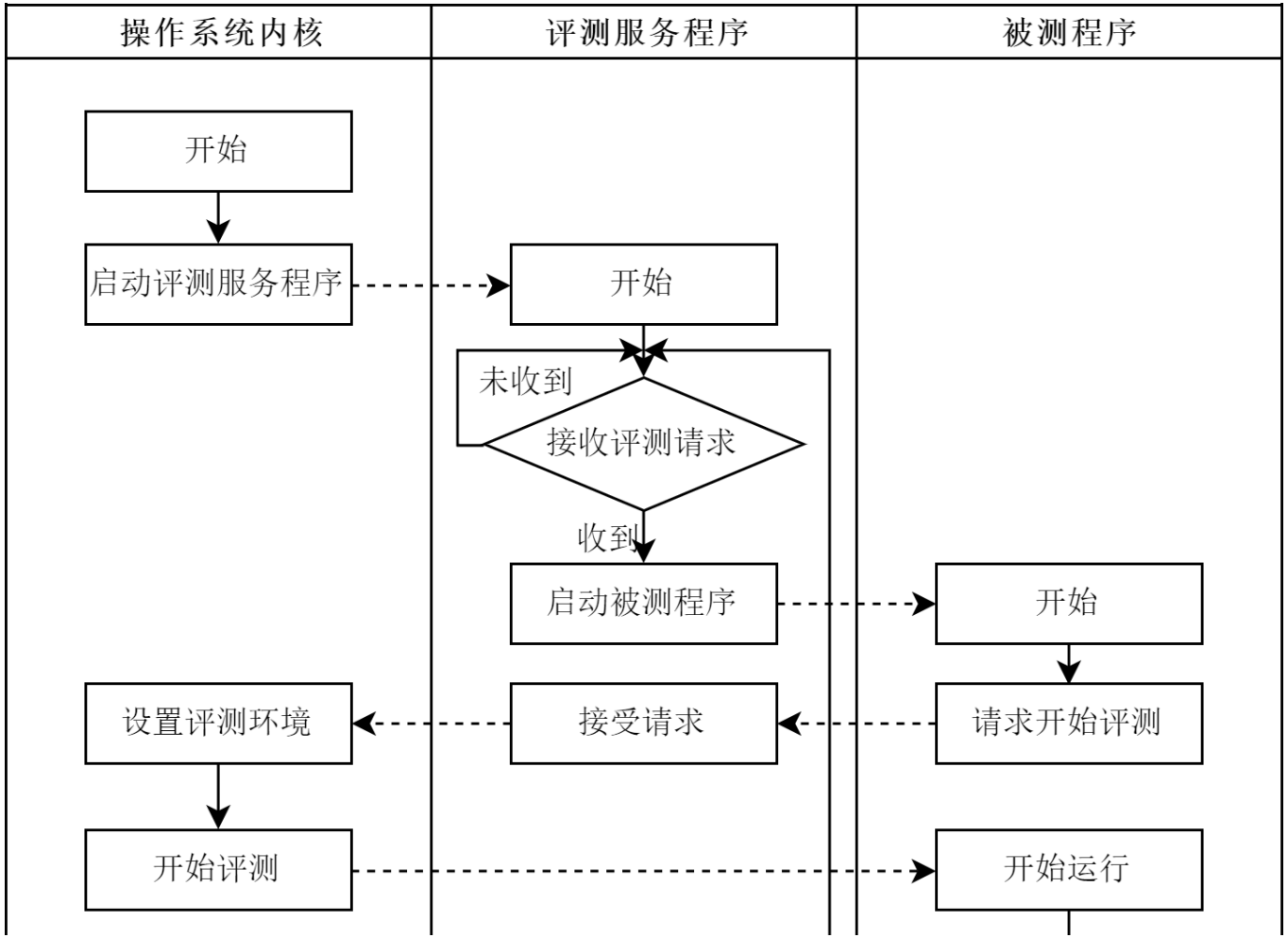
sys_accept_enter_judge(envid_t envid, struct JudgeResult *res)

* 接受一个进入评测状态的请求，由评测服务程序调用。
* 调用后系统启动被测进程，并在测试完成后将评测结果返回给评测服务程序。
* 返回值：运行的结果（是否运行时错误），时间、内存的使用量。

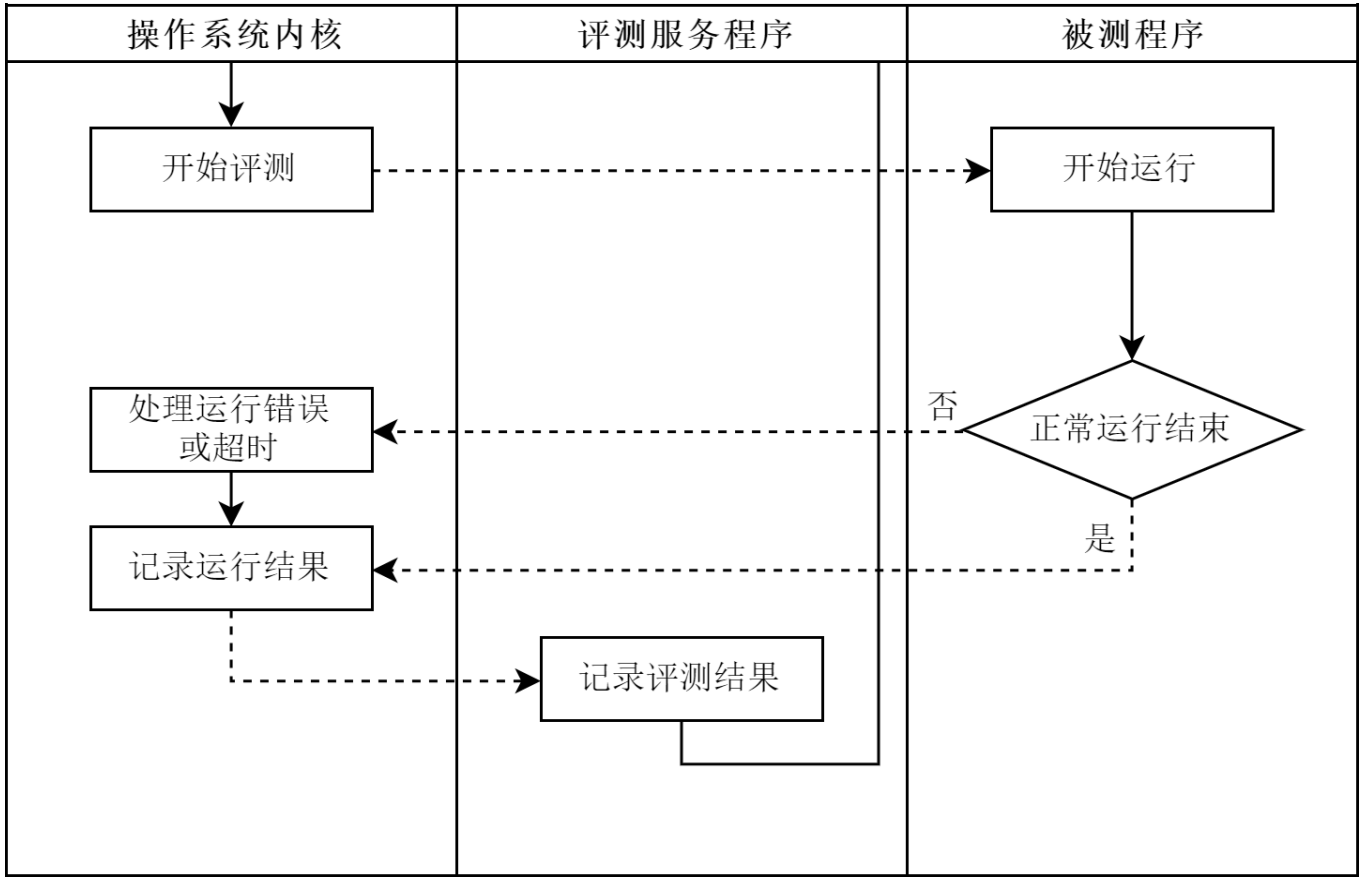
sys_quit_judge()

* 退出评测，由被测进程调用。
* 调用后立刻结束评测状态，并使评测服务程序得到评测结果。

评测流程


操作系统内核
评测服务程序
被测程序
开始
启动评测服务程序
开始
未收到
接收评测请求
收到
启动被测程序
开始
设置评测环境
接受请求
请求开始评测
开始评测
开始运行

评测流程（续）


操作系统内核
评测服务程序
被测程序
开始评测
开始运行
正常运行结束
否
处理运行错误
或超时
是
记录运行结果
记录评测结果

评测环境

关闭外部设备（除时钟外）的中断，仅开启单次时钟中断，用于实现时间限制。

预先分配内存，包括堆空间和栈空间（略大于内存限制）。

保证被测程序虚拟内存到物理内存的映射在多次评测中一致。

清除CPU的各级缓存。 （使用 `wbinvd` 指令）

被测程序不得进行除 `sys_quit_judge` 之外的系统调用，因为这会带来较大的不稳定性。

实现和测试

基于 JOS Lab 4（多进程/多核/文件系统/lwIP网络协议栈）

* 实现了评测相关的系统调用。
* 修改了内存管理机制，使得被测程序能获得连续物理内存。
* 实现了一个用户库（排序库）和一些被测程序（排序算法）。
* 实现了 e1000e 的网卡驱动，以实现通过网络发送评测请求。

模块	C 源程序	C 头文件	汇编程序	其他文件	总计
boot	125	0	85	55	**265**
kern	23234	1378	351	323	**25286**
user	969	33	0	115	**1117**
fs	1596	49	0	178	**1823**
lib	3003	0	139	48	**3190**
inc	0	1983	0	154	**2137**
总计	**28927**	**3443**	**575**	**873**	**33818**

代码行数：https://github.com/JudgeDuck/JudgeDuck-OS, 2019-09-28

实现和测试

和其他评测环境相比，各种程序均更稳定。

	评测鸭	Linux 静态编译	Linux 动态编译	BZOJ	UOJ	LOJ	Vijos
std::sort	**1**	13.6	12.2	6.1	15	17.5	177.4
基数排序	**1**	609	721.4	115.1	925.6	148.7	运行失败
空循环	**1**	3207.6	3744.9	2198.3	2841	6803.3	59541.3

* 表格中的数值：归一化的相对不确定度
根据 95%置信区间 / 运行时间平均值 计算，并除以一行中的最小值

上述实现的不足

缺少 C/C++ 的标准库

* 程序设计竞赛通常需要完整的语言支持。

网络协议栈性能不足，以致发送数据文件耗时过大

* 基于微内核的 lwIP 网络协议栈仅能实现约12Mbps（1.5MB/s）的吞吐量。需要改进，才能支持较大（数十MB）的数据文件。

启动和结束被测进程时存在两次上下文切换

* 当前所有被测程序均有 1~2 us 的时间，为这两次上下文切换。
* 消除这两次切换，可提高时间测量的精确度。

尝试改进

C/C++ 的标准库

* 移植了 musl-libc 和 GNU libstdc++-v3。
* 将文件 IO 操作映射至内存；不支持其他系统调用。

实现了用户态网络协议栈

* 重写了用户态网卡驱动和用户态 UDP/IP 网络协议栈。
* 能够支持千兆线速收发包；约 1500 行代码。

消除启动和结束时的上下文切换

* 开启多核，使用共享内存与被测进程同步运行状态。
* 两次上下文切换（约 2us） → 两次内存同步（约 200ns）

相关工作讨论

编程竞赛中的其他评测系统：

* 大多数评测系统不考虑“稳定性”和“精确性”，而更多地关注权限管理和易用性。
* 基于 Linux 的评测系统可使用 ptrace，cgroup 等系统调用来实现简单的权限管理。
* 一些评测系统支持更多种类的语言，如 Java，Python 等。
  * 这些解释性语言本身就含不确定因素，支持它们没有必要。

在 Linux 上的稳态和高性能：

* 高性能通常指存储、网络等设备和应用的高吞吐量，稳态通常指网络等设备的稳定低延迟。
* 实现高性能：实现用户态驱动，应用程序直接访问设备，使用无锁数据结构。
* 实现稳态：关闭中断，避免进程调度，减少缺页异常（如使用大页）。

** Intel DPDK, C10M 问题

总结和展望

问题分析

* 提出了编程竞赛中的稳定评测问题
* 分析了现有评测系统中的不稳定因素及其解决方案

实现和测试

* 实现了稳定的评测系统，与现有评测系统进行了比较

尝试改进

* 改进评测系统，以支持编程竞赛的更多要求

讨论和展望

* 技术方面：是否需要支持多种编程语言？是否需要支持64位？
* 非技术方面：有了稳定评测系统，可以做什么？

内容提要

稳定评测的重要性

评测鸭：稳定精确评测系统的实现

评测鸭如何影响编程教育

最初的尝试：Online Judge

为了让大家体验这种新型的评测技术，我们开发了一个在线评测系统——“评测鸭在线”。

提供了多个算法题目，对提交的程序进行排名。

题目列表

全部题目 | 传统题 | 模板题 | 比赛题

题号	题目
1000	测测你的 A+B
1000f	测测你的实数 A+B
1000i	【传统题】 A+B Problem
1001	测测你的排序

测测你的二分查找 - 排行榜

排名	提交编号	用户	用时
1	5099	foreverpiano	32.82 us
2	2066	negiizhao	32.94 us
3	2240	Simpson561	35.08 us
4	2120	Decyx_asmend	35.41 us
5	2008	lastans7	36.88 us

https://duck.ac/problems, https://duck.ac/problem/1003/board, 2019-09-25

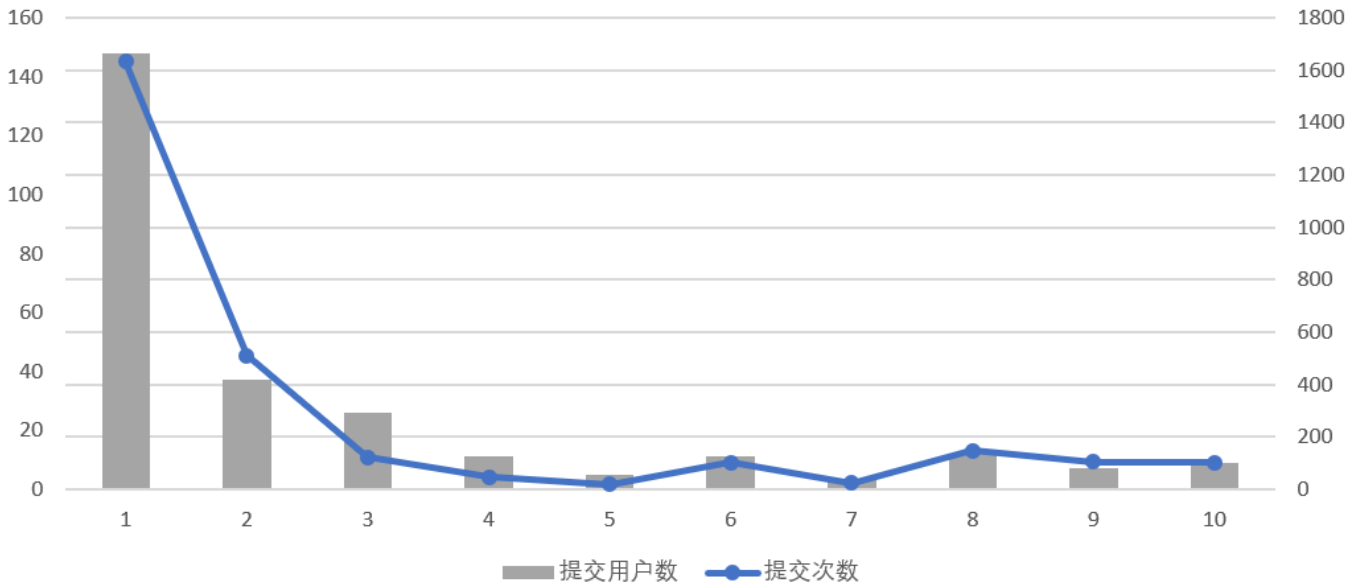
最初的尝试：Online Judge

第一天：1632个提交，来自148个不同的用户

第二天：509个提交，来自37个不同的用户

第三天：121个提交，来自26个不同的用户

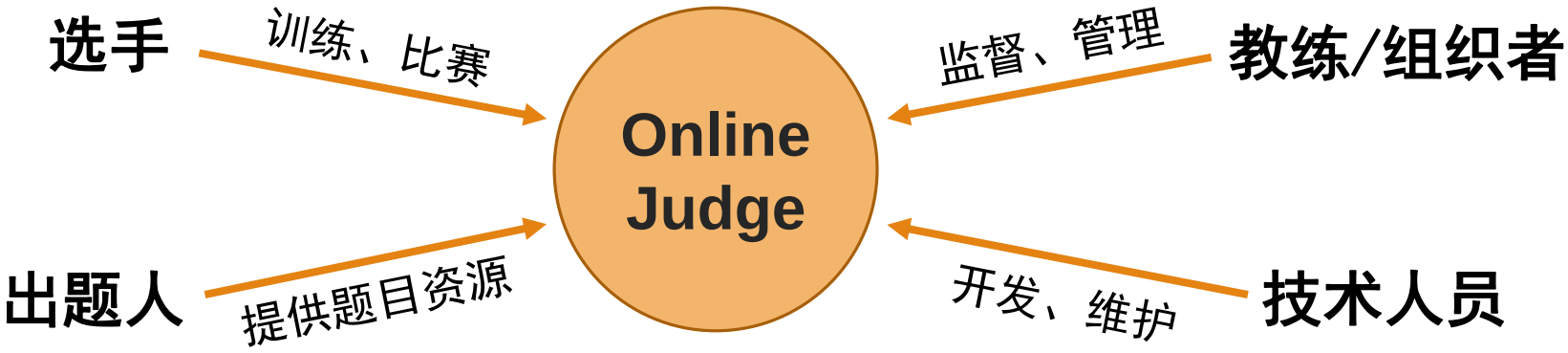
……



Online Judge 与编程教育

Online Judge：以评测为核心，提供多种多样在线服务

* 训练、比赛、讨论……



Online Judge 与编程教育

Online Judge：以评测为核心，提供多种多样在线服务

* 训练、比赛、讨论……

现今多数编程教育相关活动使用了 Online Judge 技术：

* 编程竞赛/考试：IOI、ICPC、CSP 等
* 训练/交流平台：UOJ、LOJ、BZOJ、Vijos、洛谷 等
* 编程相关课程：如清华大学《程序设计基础》《数据结构》等课程

标准化的考试、社区化的平台、可复用的技术……

评测鸭与 Online Judge

评测鸭：评测技术的创新和改进

OJ需要：用户社区、题目资源、活动组织

现有OJ的挑战：评测结果的不稳定问题

评测鸭如何发展？

* 帮助改进各OJ的评测技术。

OJ数量众多，功能各异，如何改进其评测技术？

* 提供一套标准化的**评测服务**。

稳定精确的评测服务

稳定和精确：

* 精确测量程序的运行时间，最高精度可达纳秒级。
* 多次测量同一程序，时间误差一般不超过 1%+1微秒。

兼容：

* 兼容 C/C++ 等语言及其标准库的主要特性，实现低迁移成本。

易用：

* 提供标准化的接口，任何 OJ 或其他软件都可方便地接入。

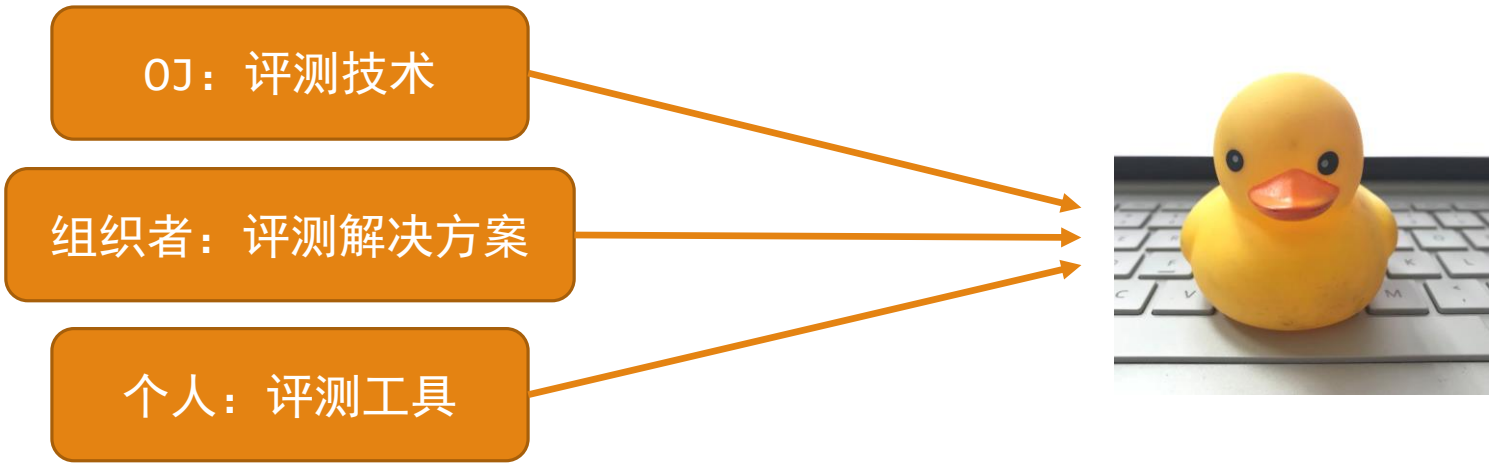
稳定的评测系统 → 确定的竞赛成绩 → 减少选手的顾虑

稳定的评测系统 → 更好的代码性能比较 → 更好的教学环节

评测服务：分享技术

当技术被作为一种服务来提供时，任何人都可以受益：

* 向各种OJ提供基于评测鸭的评测服务
* 向没有能力开发OJ的组织提供封装好评测服务的OJ
* 向个人用户提供评测服务客户端



评测服务：事实标准

评测鸭使用一种相同的机器配置向所有人提供评测服务：

* 想要比较几个算法，却没有一个好的测试平台？
* 想要出题/办比赛，却不知使用什么评测环境？
* 想要与其他评测系统得到同样的运行结果？

同一份源程序、相同数据在不同OJ上的运行时间

OJ	评测鸭在线	LOJ	UOJ	洛谷
最大运行时间	604 ms	685 ms	726 ms	781 ms
总运行时间	4332 ms	5157 ms	5264 ms	5100 ms

https://duck.ac/submission/10722, https://loj.ac/submission/619712,
http://uoj.ac/submission/365268, https://www.luogu.org/record/24442315, 2019-09-27

期待你的到来！

访问 https://duck.ac/，即可体验稳定精确的评测系统。

评测鸭是开源项目，你可在 Github 上搜索 JudgeDuck，找到其源代码。

你也可以扫描这个二维码，获取更多有关评测鸭的信息：

谢谢！

本次报告的PPT下载：

扫描此二维码，或访问

https://wys.life/2019-09-judge-duck

参考文献

[1] Stefano MAGGIOLO, Giovanni MASCELLANI. Introducing CMS: A Contest Management System[J]. Olympiads in Informatics, 2012: 86–99.

[2] Ágnes Erdősné Németh, László ZSAKÓ. Grading Systems for Algorithmic Contests[J]. Olympiads in Informatics, 2018: 159–166.

[3] 李扬. 基于 Python 的在线评测系统的设计与开发[D]. 青岛: 青岛大学, 2016.