CS353 Linux 内核 Final Project

实验要求

本实验包括内核态部分和用户态部分：在内核态中需要编写一个内核模块，可以获取程序的实际运行时间和内存读写量；在用户态中编写一个程序，定时通过内核模块跟踪程序的 CPU 利用率和内存读写量，两者比较分析程序是计算密集的还是内存密集的。

内核模块一个可能的实现方式为：创建一个 proc 文件，向 proc 文件写入进程 PID 之后，每次读取 proc 文件，可以得到上次读取文件之后进程调度到 CPU 上的运行时间以及读写的内存页个数。

用户态程序则需要启动 benchmark 进程，将进程的 PID 写入 proc 文件，然后定时读取 proc 文件，计算得到进程的 CPU 使用率以及内存读写频率，并写入一个日志文件。

具体的实现是开放的，但是你的实现应该通过内核模块编程从内核数据结构获得数据，而**不是通过内核已经实现的 proc 文件获得**。

实验报告

在实验报告中，需要描述实现的思路，以及实验的结果。实验结果的表现形式为：**选取不同的 benchmark 进程，将它们的 CPU 使用率和内存读写频率随时间的变化画图表示出来，并根据此分析进程的特点**。

另外，若是你在实现时遇到了什么困难并最终解决，或者是有什么心得体会，也可以在实验报告中写出。

实验报告最后请给出你关于**思考题**的回答。

提示

1. 进程实际运行时间的获取可以参考内核中对于 proc 文件 `/proc/[pid]/stat` 的实现。**该文件中有一个参数为 `utime`**，可以一定程度上表示该进程的实际运行时间。
2. **页目录项中有一个标志位 `young`**，CPU 在每次访问该页时会将该标志位置为 1。你可以通过先清空该标志位，一段时间后再读取的方式，判断进程在这段时间内是否读写该页。内核中有函数 `ptep_test_and_clear_young` 函数可以在获取 `young` 标志位的同时清空它，但是由于内核的限制，模块中无法使用该函数。你可以参考内核中该函数实现自己的版本。
3. 进程描述符中的 `task_stuct->mm->vma` 是进程向内核申请使用的内存区域（表示为一个 `struct vm_area_struct` 结构体）的链表，你可以只扫描用户内存空间的这部分区域，减少扫描的时间。你甚至可以根据这些 vma 的属性进一步跳过不必要扫描的页。
4. 用户态程序不限制编程语言的使用，你可以使用任意自己数量的语言进行编写。
5. 推荐可以使用的 benchmark 程序：`sysbench`，`7-zip benchmark`，`y-cruncher`。你也可以另外选择其他的 benchmark 程序。
6. 可以参考模版代码：https://github.com/chengjiagan/CS353-2022-Spring。

思考题

（注：以下问题可以在实验报告中回答，也可以实现在程序中，实现在程序中请注明）

1. 大多数 benchmark 是多线程或者是多进程的，你的程序中有考虑这种情况吗？若没有，应该怎么解决？
2. `utime` 表示程序程序的什么时间？`/proc/[pid]/stat` 还有一个参数为 `stime`，它表示什么？你觉得在这个实验里面适合使用它吗？

3. 你觉得当前实验中以页为单位统计进程的内存读写合适吗？如果合适，原因是什么？如果不合适，有没有更好的方法?

提交

提交渠道：Canvas

提交文件：`学号_projectfinal.zip`，源码文件夹 `学号_projectfinal_src`（所有源代码文件以及Makefile），实验报告 `学号_projectfinal_report.pdf`。