Linux Kernel大作业

0. 作业简介

- 数据内存页的定位，划分，热度收集及加速
- 将相关的统计信息 传输至 用户空间

1. 程序数据页，热度信息的收集

1.1 基础步骤

1. 每个进程 通过PID 获取 process descriptor，真正放置数据的内存页结构， 也就是一个进程类中的VMA结构，该结构中的数据页链表保存着需要的数据
2. 根据VMA的特征，判断该页面是否保存着真正的数据
3. 找到包含数据的VMA后，取出该数据页链表，通过四次或者五次页表转换，获得真正的数据页面
4. 找到数据页后，根据pte_young()函数判断，该页面在上一个间隔中，是否被访问。 然后通过pte_mkold()重置所有标志位。
5. 循环步骤三和步骤四，获取所有页面的访问热度信息。

1.2 提示

0. 假设，程序中有一个二维数组。
1. VMA中的page list的起始虚拟地址，和数组中每个一维数组的起始虚拟地址，不相同且不对应。
2. VMA之间的地址，可能不连续，且VMA之间的大小可能不相同。
3. 对于每个page的实际寻址过程，非常缓慢。 1000 000个page，也就是4GB，全部扫描一遍，需要约3s钟。 而一个访存密集型程序，运算一遍可能只有300ms。 首先考虑，对VMA进行逻辑划分，其次，考虑使用多线程。 尽可能收集更多更准确的热度信息。

2. 程序数据页，热度信息的传递

2.1 基础步骤

1. 同时启动，用户态程序和内核态程序
2. 用户态程序将PID写入共享内存，内核态程序开始收集页面热度信息
3. 用户态完成五次迭代，睡眠固定时间，等待内核态程序写入热度收集信息
4. 内核态程序定期，强制将所有页面热度信息写入共享内存
5. 用户态程序拿到热度信息后，打印出所有程序内存页的使用热度情况

2.2 提示

0. "/proc" virtual file system
1. 简单来说，可以用户态等待时间较长一些即可
2. 若想做的好一点，可以估算自己扫描页面的时间，然后根据扫描页面的时间，决定用户态睡眠时间

3. 检查

3.0 Benchmark

0. 64线程
1. 访存均匀程序 -- heat.cpp
2. 访存不均匀程序 -- heat_rand.cpp

3.1 第一部分实验检查

1. 打印出，筛选出的，VMA结构及其起始地址及结束地址，并输出程序中，数组的起始地址及结束地址。 展示出，数组的起始地址及结束地址 包含在 筛选后的VMA的起始地址及结束地址中。
2. 打印出，整个程序的page数，筛选出的VMA结构的page数，程序中内存申请的page数 筛选出的page数应该不超过申请的page数的40%
3. 打印出，程序每次迭代的时间，页面收集所需的时间。并在内核态打印出所有页面的热度信息

3.2 第二部分实验检查

1. 演示两个程序的启动，交互
2. 并在用户态打印出所有页面的热度信息