HW5

姓名: 陈锐林, 学号:21307130148

2023 年 10 月 11 日

Chapter21

Question1

(1) 开始运行 mem 后，user time 在增加，空闲时间减小；并且能发现上下文切换和中断的时间也在增加。(2)user time 这个指标是有意义的，mem 1 就能让 user time 迅速达到 100。(3) 如果运行俩个 mem 1 实例，会发现 user time 翻倍，多次实验后发现应该是线性关系。

Question2

(1) 随着 mem 的退出，free 的数值是在变大的。(2) 但是 free 的增加数值并不完全如想象中的那么大，会偏小一些；部分内存没有释放，可能是因为 Swap 等操作得不到恢复。

Question3

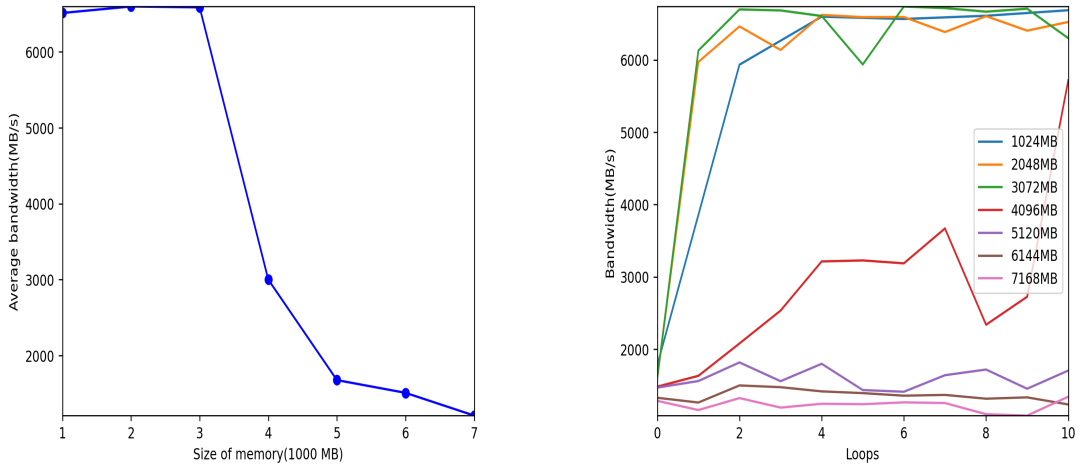
(1) 可以看到，第一个循环中，大量内存被交换出去，少量被交换进来；交换出去的部分非线性地减小到 0；它可能偶尔会再增加一些，但会到 0。(2) 不断更改命令，发现存在共性。第一个循环发生大量的交换之后，之后的循环只会发生少量的交换。

Question4

可以看到 CPU 利用率和 I/O block 都是上升的，表现在指标上就是 us 和 sy 有变大。

Question5

首先根据多组实验数据，能绘制出以下俩幅图。能看出，(1) 当所有事项都让内存舒适时，带宽是较大的；(2) 在 mem 较小时，首次循环和后几次的差距是最大的，mem 较大时差距会减小；而在 mem 适配内存时，循环间差距较小。



(a) (b)

图 1: Question5

Question6

当尝试 mem 更大内存时，会发现进程被迫结束。这里经过几次尝试，发现差不多到总内存的 20% 左右就会分配失败了。

Chapter22

Question1

经过计算，可以得到在不同的随机种子和策略下的命中率如下：

seed	policy	hit-rate	hit 的项
0	FIFO	10%	6
0	LRU	20%	6,9
0	OPT	40%	6,7,9,10
1	FIFO	20%	6,10
1	LRU	20%	6,10
1	OPT	30%	6,8,10
2	FIFO	40%	2,4,9,10
2	LRU	40%	2,4,9,10
2	OPT	40%	2,4,9,10

Question2

通过以下三条指令:"python3 ./paging-policy.py –addresses=0,1,2,3,4,5,0,1,2,3,4,5 –policy=FIFO –cachesize=5 -c", "python3 ./paging-policy.py –addresses=0,1,2,3,4,5,0,1,2,3,4,5 –policy=LRU –cachesize=5 -c", "python3 ./paging-policy.py –addresses=0,1,2,3,4,5,4,5,4,5,4,5 –policy=MRU –cachesize=5 -c". 就能得到最差的表现，此时的命中率都是 0。为了提高效率,我们只需提高 1 的 cache 大小;此后命中率有很大提升,都提到了 50%。

Question3

这里使用"python3 ./paging-policy.py -s 0 -n 10" 指令，随机生成。再考虑使用不同的策略；根据不同的策略进行计算，以及通过-c 标志验证，可以得到以下命中率：FIFO-10%; LRU-20%; OPT-40%; UNOPT-0%; RAND-0%; CLOCK-0%.

Question4

这里随机生成 32 个数 10,9,9,11,7,6,6,7,6,3,4,6,6,5,5,6,6,2,2,1,2,3,3,1,1,2,0 ,3,4,3,6,5,8；调用命令:"python3 ./paging-policy.py –addresses=10,9,9,11,7,6,6,7,6,3,4,6,6,5,5,6,6,2,2,1,2,3,3,1,1,2,0,3,4,3,6,5,8 –policy=?" 来计算。得到如下命中率:LRU-48.48%; RAND-42.42%; CLOCK(bit-1)-36.36%; CLOCK(bit-2)-39.39%。

Question5

首先这里我们要将生成的进行转化，主要用到了下面几行代码，根据掩码进行操作。之后就像前面一样，得结果，绘图分析。最终结果如下：

```
for line in traceFile:
    if (not line.startswith('=')):
        vpnFile.write(str((int("0x" + line[3:11], 16) & 0xfffff000) >>
                                                    12) + "\n")
```