Homework5

Name Yu Yang   Student ID 892449550

9.8

	1	2	3	4	5	6	7
FIFO	20	18	16	14	10	10	7
LRU	20	18	15	10	8	7	7
OPT	20	15	11	8	7	7	7

9.16

a. Because it uses pure demand page, no page has been loaded as a process starts. At this time, page fault number is very high.
b. Once the working set is in memory, the page-fault rate falls because of locality. When the process moves to a new working set, the page-fault rate rises toward a peak once again, returning to a lower rate once the new working set is loaded into memory. The span of time between the start of one peak and the start of the next peak represents the transition from one working set to another.
c. Best way to handle it is to get more physical memory whenever possible.
The second way is to reclaim pages more aggressively.
The third way is to ignore it.

9.24

LRU requires that the page with smallest count (we keep a counter of the number of references to each page) to be replaced. But the problem is when a page is used heavily during the initial phase of a process but then is never used again. Since it was used heavily, it has a large count and remains in memory even though it is no longer needed. So when a initially heavily used page appears again after longtime, it will cause a page fault in LRU but not in LFU.

LRU is using past locality to predict future (optimal page replacement looking backward in time). If a certain program keep certain degree of locality over time, LRU should be better than LFU.

9.32 What is the cause of thrashing? How does the system detect thrashing? Once it detects thrashing, what can the system do to eliminate this problem?

Cause of thrashing: The CPU scheduler sees the decreasing CPU utilization and increases the degree of multiprogramming as a result. The new process tries to get started by taking frames from running processes, causing more page faults and a longer queue for the paging device. As a result, CPU utilization drops even further, and the CPU scheduler tries to increase the degree of multiprogramming even more. Thrashing has occurred, and system throughput plunges.
Detect: by monitoring the cpu utilization and multiprogramming level.
Eliminate: by decreasing the multiprogramming level.

10.2

Because on average, middle cylinders have the shortest distance to other tracks. So the seek time tends to be shortest.

10.11

FCFS: order 2150, 2,069, 1,212, 2,296, 2,800, 544, 1,618, 356, 1,523, 4,965, 3681
Total distance: 13011
SSTF: order 2150, 2069, 2296, 2800,3681,4965, 1618, 1523, 1212, 544, 356
Total distance: 7586
SCAN: order 2150, 2296, 2800, 3681, 4965, 4999(end), 2069, 1618, 1523, 1212, 544, 356
Total distance: 7492
LOOK: order 2150, 2296, 2800, 3681, 4965, 2069, 1618, 1523, 1212, 544, 356
Total distance: 7424
C-Scan: order 2150, 2296, 2800, 3681, 4965, 4999(end), 0, 356, 544, 1212, 1523, 1618, 2069
Total distance: 9917
C-Look: order 2150, 2296, 2800, 3681, 4965, 356, 544, 1212, 1523, 1618, 2069
Total distance: 9137

10.16

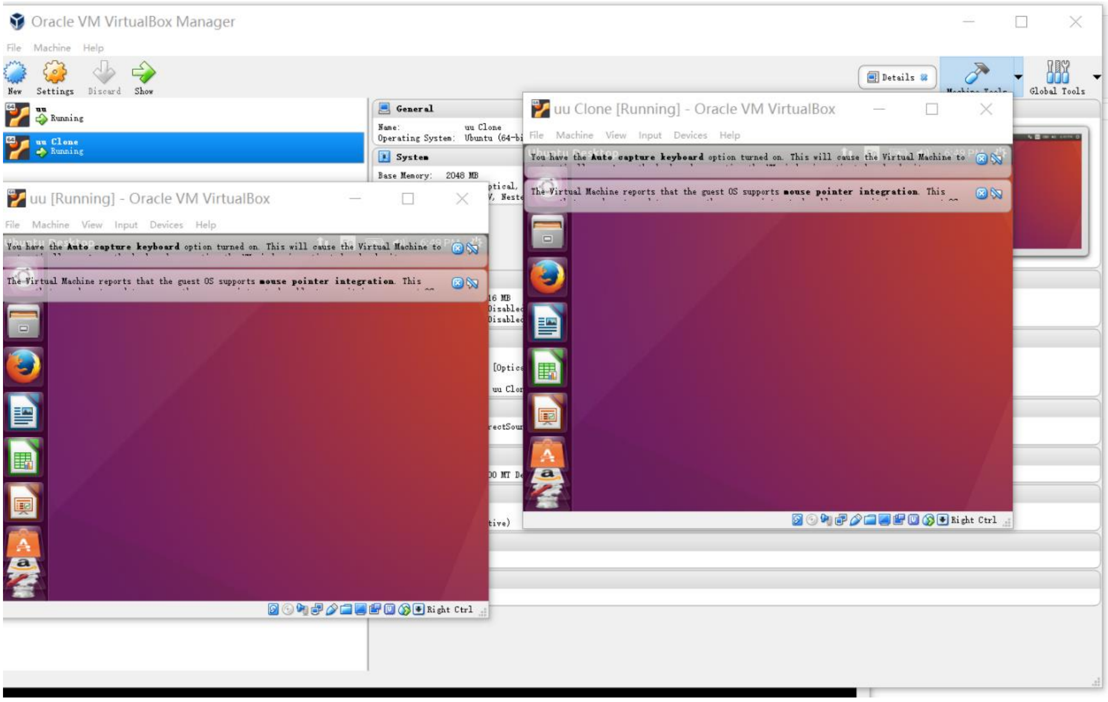
a. SSTF is particularly good for this case. Using SSTF would take advantage of the high-demand cylinders, because using SSTF would keep the arm around those positions for the most time. Neither scan(or look) or fcfs has more movements far away from those high-demand area.
b. Using SSTF would cause arm moving around hot spot for most time, at the risk of starvation. Modifying SSTF so that only a maximum number of requests by hot spot could be served consecutively,

10.18

a) The benefits of RAID5 are mainly on multiple read access can proceed in parallel. But in this case, read op only happens on single blocks, RAID5 could not give the parallel proceeding. Conversely, RAID 1 has one copy for each block, which could give more flexibility to choose which disk to read.
b) As discussed in question a, in multiple contiguous blocks reading, RAID5 could give the parallel proceeding.

Virtual Machine

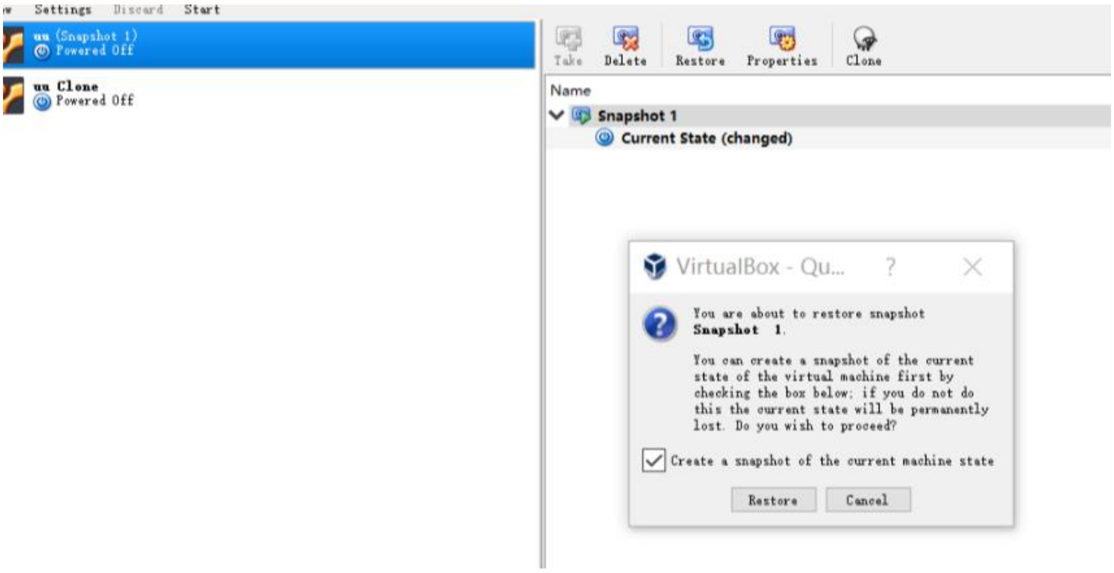
Cloning



Snapshot

After running sudo rm -rf /*

Restore



Recovered