CS 153
Design of Operating Systems

Winter 2016

Final Review 2

True of False?

- For machines with 32-bit addresses (i.e. a 4GB address space), since 4GB physical memories are common and cheap, virtual memory is really no longer needed.

- Answer: False

True of False?

- A TLB miss could occur even though the requested page was in memory.

- Answer: True

True of False?

- A smaller page size leads to smaller page tables

- Answer: False

True of False?

- A smaller page size leads to more TLB misses
- Answer: True

True of False?

- A program allocating 100MB of memory cost only 100MB of memory

- Answer: False

True of False?

- The optimal page replacement algorithm is the best choice in practice
- Answer: False

True of False?

- Open() in NFS client translates into a RPC call into the server

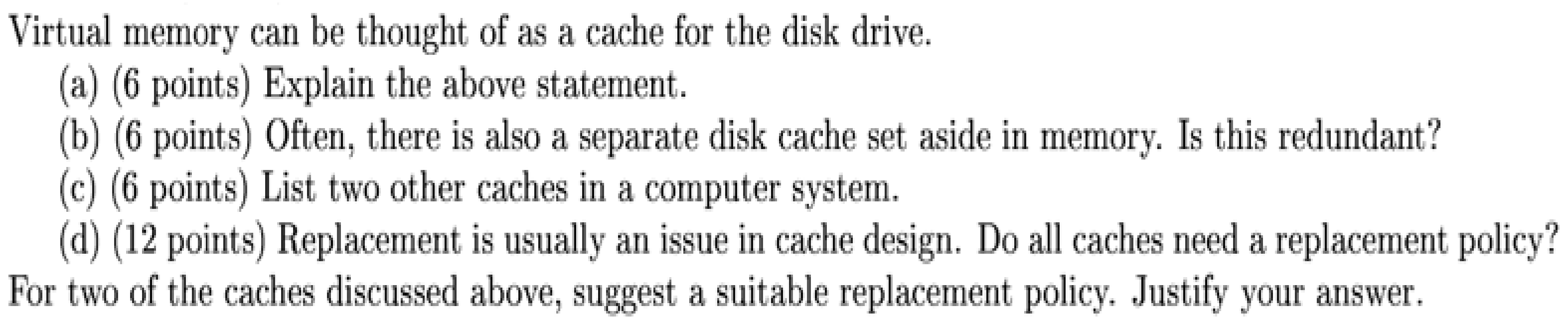
- Answer: False

True of False?

- Belady’s page replacement algorithm = LRU
- Answer: False

True of False?

- An Android app can read the memory of another app
- Answer: False


Virtual memory can be thought of as a cache for the disk drive.

(a) (6 points) Explain the above statement.

(b) (6 points) Often, there is also a separate disk cache set aside in memory. Is this redundant?

(c) (6 points) List two other caches in a computer system.

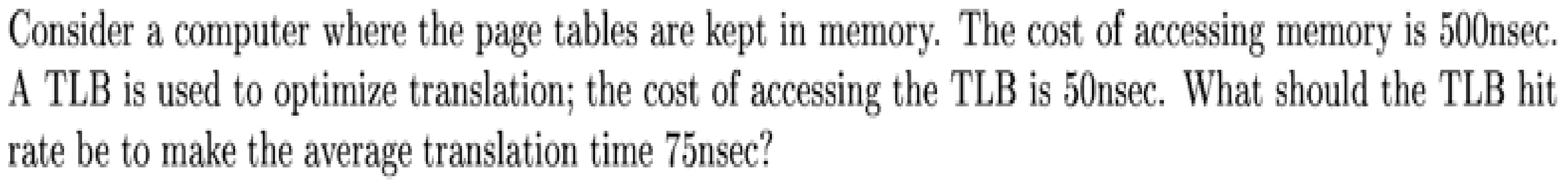
(d) (12 points) Replacement is usually an issue in cache design. Do all caches need a replacement policy? For two of the caches discussed above, suggest a suitable replacement policy. Justify your answer.

Cache

Virtual Memory

Physical Memory

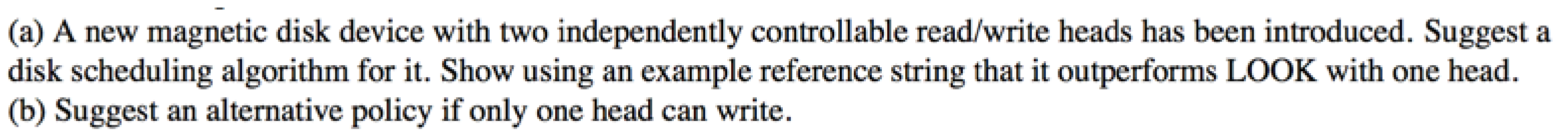
Disk



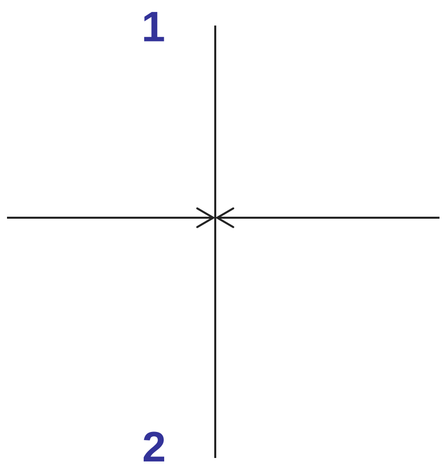
Consider a computer where the page tables are kept in memory. The cost of accessing memory is 500nsec. A TLB is used to optimize translation; the cost of accessing the TLB is 50nsec. What should the TLB hit rate be to make the average translation time 75nsec?

$X * 50 + (1-X) * 500 = 75$
→ $450X = 425$
→ $X = 94.4\%$

LOOK ~= SCAN except that it “looks” ahead to see if any requests are pending in the forward direction. If not, it may serve requests in the reverse direction. (Exactly the same algorithm as elevators)



(a) A new magnetic disk device with two independently controllable read/write heads has been introduced. Suggest a disk scheduling algorithm for it. Show using an example reference string that it outperforms LOOK with one head.
(b) Suggest an alternative policy if only one head can write.



Problem 8: (25 pts) Suppose that you have a UNIX file system where the disk block size is 512 bytes, and an inode takes a full block. Disk addresses take 4 bytes, and the inode contains space for 64 bytes of data (a recent optimization), 8 direct addresses, one indirect, one double-indirect and one triple- indirect. An index block is the same size as a disk block.

(1) (12 pts) How much space (including overhead) do files that are: a) 1 byte long, b) 750 bytes long, c) 80000 bytes long, and d) 1048576 (1MB) bytes long require? Hint: it may help if you draw a picture of how inodes are used to locate the blocks making up a file.

(2) (4 pts) How does having the 64 bytes of data in the I-node help?

(3) (9 pts) A proposed optimization for high-performance file systems is to write updated blocks to a nearby free block on the disk (instead of saving it in the same location). Discuss advantages and disadvantages of this approach.

- (18 points) In class we described three file descriptor structures:

  (a) Indexed files.

  (b) Linked files.

  (c) Contiguous (extent-based) allocation.

Each of the structures has its advantages and disadvantages depending on the goals for the file system and the expected file access pattern. For each of the following situations, rank the three structures in order of preference. Be sure to include the justification for your rankings.

(a) You have a file system where the most important criteria is the performance of sequential access to very large files.

(b) You have a file system where the most important criteria is the performance of random access to very large files

(c) You have a file system where the most important criteria is the utilization of the disk capacity (i.e. getting the most file bytes on the disk).

Revisiting questions

- What causes your code to “dump a core file” when you access a NULL pointer?

- Why can multi-threaded code be slower than single-threaded code?

OS Principles

- Tradeoffs
  - Performance (e.g., speed)
  - Cost (e.g., memory)
  - Complexity (e.g., difficulty to reason, lines of code)
  - …
- Optimize for common case and make the corner case work
  - Cache (replacement algorithm), inode structure, scheduling, etc.
  - Security, however, is often scrutinizing the corner cases
- You can always solve a problem with an extra layer of indirection
  - Virtual memory (indirection of virtual to physical address)

The End

- Congratulations on surviving CS 153!
  - It’s a challenging course, but I hope you found it worthwhile

- Good luck for the final!