CS-323 – Final Preparation

23 May 2019

Question 1: Disk optimization

Two of the primary disk optimizations are disk scheduling and clever disk allocation. One is known to be more effective under high load, and the other is known to be more effective under low load. Which one is which and why? Structure your answer as follows:

1. Disk scheduling is more effective under low load / high load (pick one).
2. Reason for answer to 1
3. Clever disk allocation is more effective under low load / high load (pick one).
4. Reason for answer to 3

Answer:

- Disk scheduling is more effective under **high load**.
- Why?
  - Many scheduling opportunities
    - Many requests in the queue
  - Allocation gets defeated
    - By interleaved requests for different files

Answer:

- Clever disk allocation is more effective under **low load**.
- Why?
  - Not much scheduling opportunities
    - Not many requests in the queue
  - Sequential user access → sequential disk access
  - Cache tends to reduce load

Question 2:

Suppose that a disk drive has 5000 cylinders, numbered from 0 to 4999. The drive is currently serving a request at cylinder 143, and the previous request was at cylinder 125. The queue of pending requests is:

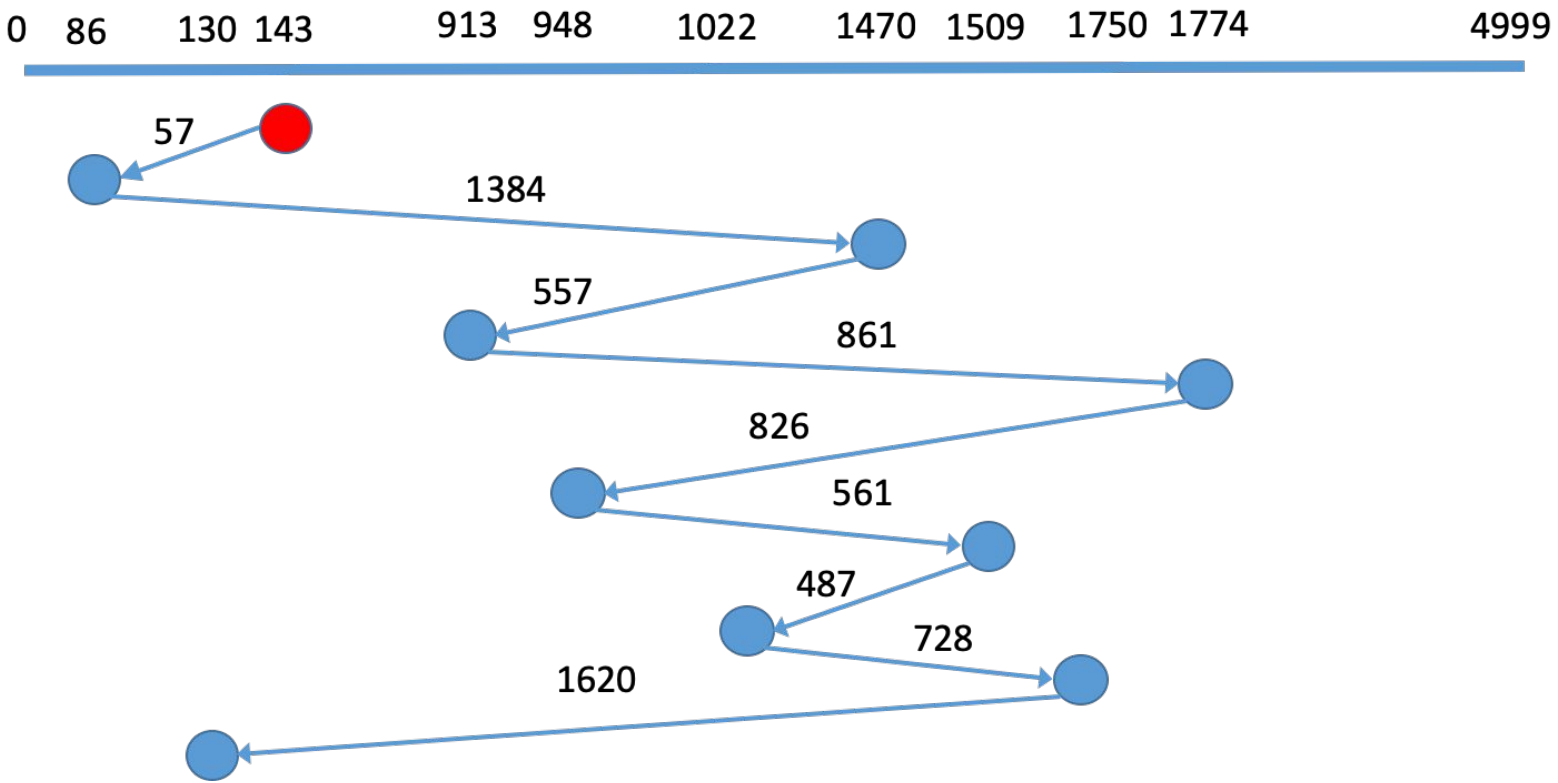
86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130

Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests for each of the following disk-scheduling algorithms?

1. FCFS
2. SSTF
3. SCAN
4. C- SCAN

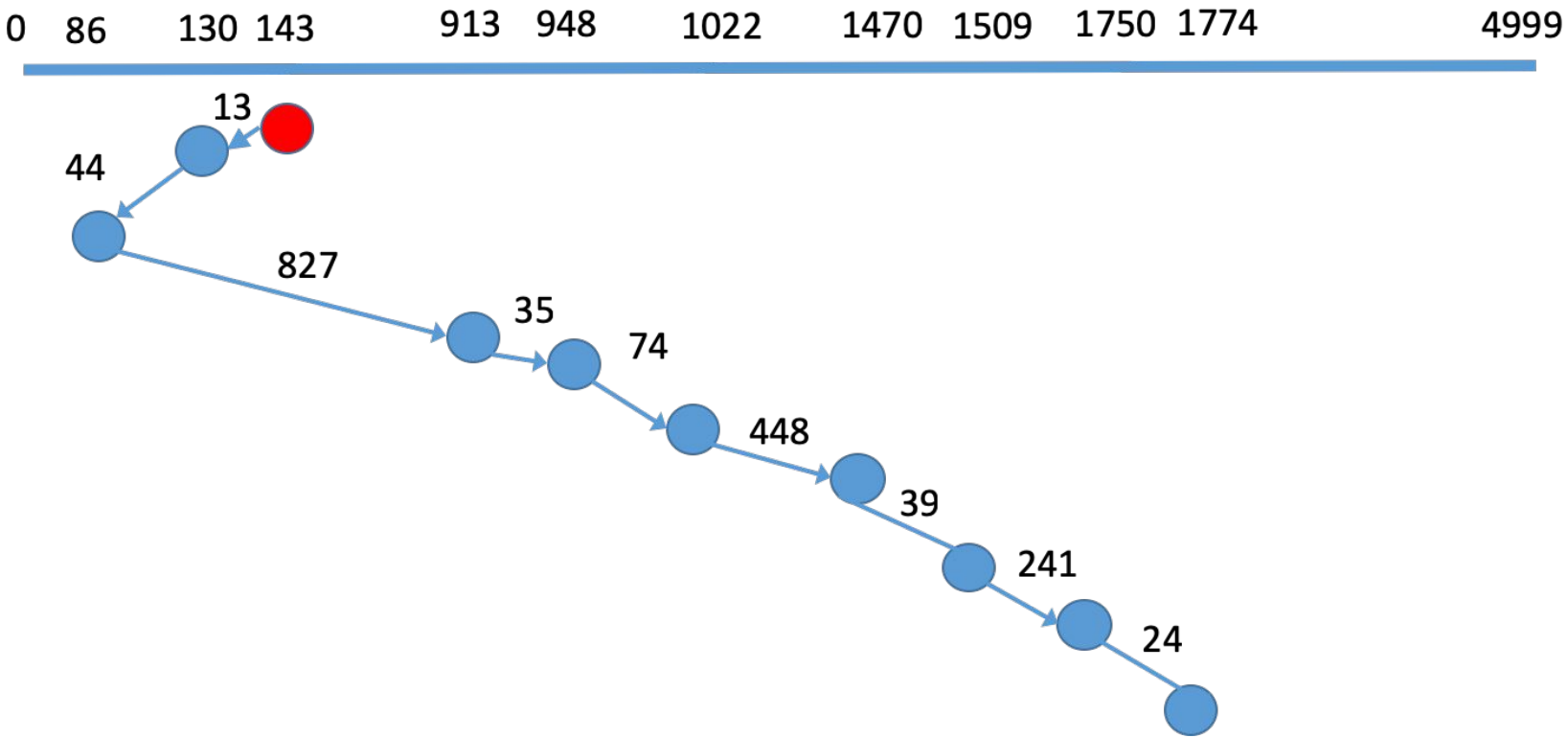
Req: 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130

FCFS: 7081



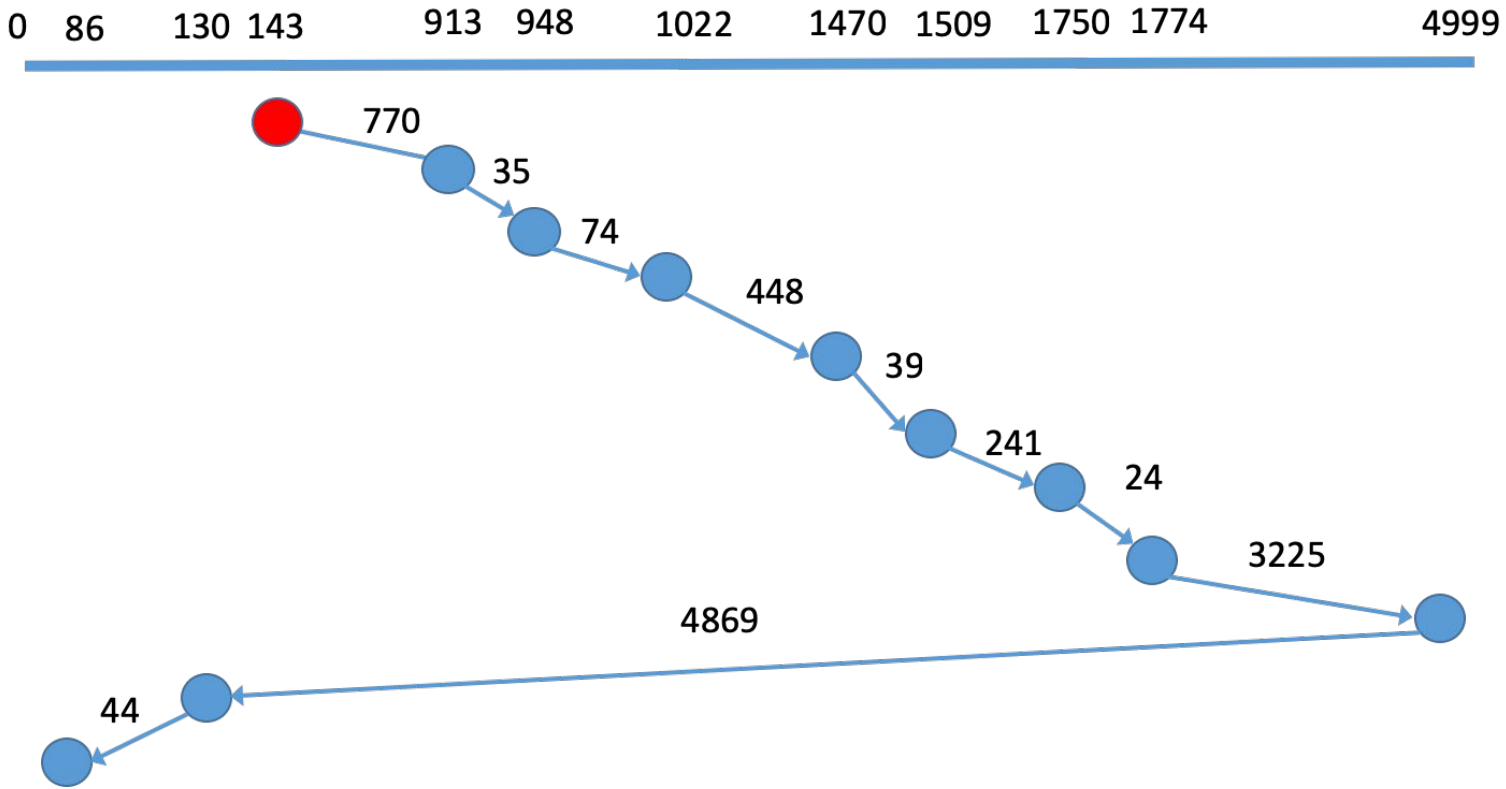
Req: 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130

SSTF: 1745



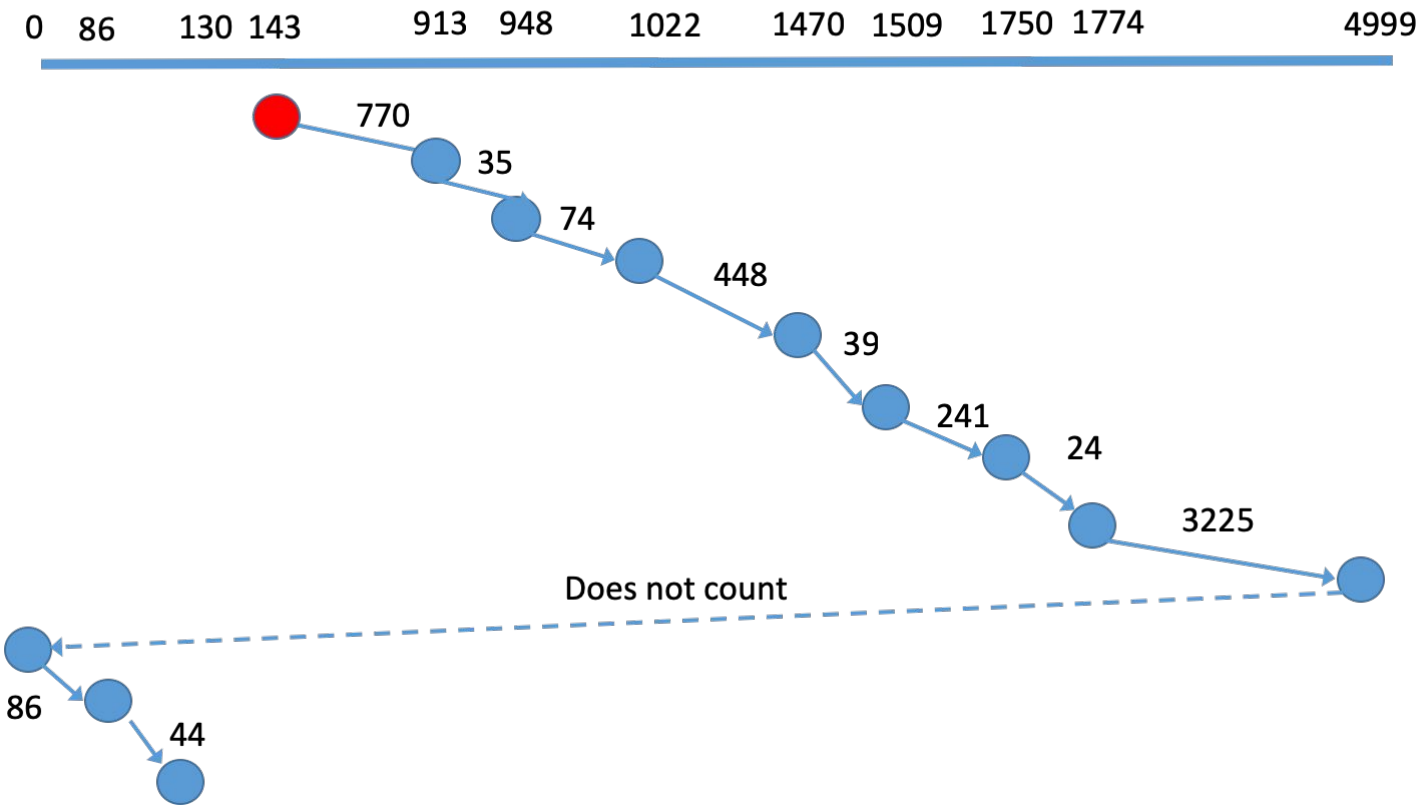
Req: 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130

SCAN:9769



Req: 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130

C-SCAN: 4986



Question 3: In-memory and on disk data structures

	Active File Table	Open File Table
Scope		
Each entry corresponds to		
Each entry contains		

Question 3: In-memory and on disk data structures

A. Fill in the following table, comparing the Active File Table and the Open File Table.

B. List the steps that take place when a process invokes *tid = Open(uid).* In particular, show any accesses to disk and any updates to the above data structures. Assume that the file with identifier *uid* exists, but that all in-memory data structures are empty when the operation is invoked.

Answer:

	Active File Table	Open File Table
Scope	• System-Wide • One array for the entire system	• Per-Process • One array per process
Each entry corresponds to	• An *open file*	• A *file open* by the process
Each entry contains	• Device directory entry of file • Reference count of number of file opens • Additional info	• Pointer to entry in active file table • File pointer *fp* • Additional info

Answer:

Steps:

1. Check in the Active File table to see if the file is used by another process. In this case it won't be.
2. Find free entry in Active File Table
3. Read inode and copy it in an Active File Table entry
4. Refcount = 1
5. Allocate entry in Open File Table
6. Point to entry in Active File Table
7. Set *fp* = 0

Question 4: Log-structured Key-Value Stores

Similar principles that are used by the log-structured file system can be applied to different contexts. One such example is *log-structured merge key-value stores (LSMs)*. LSMs are data stores holding key-value pairs. They have a *sorted* in-memory component and an on-disk component. All writes are performed (only) on the in-memory component. Once the in-memory component gets full, it is sequentially written to the disk component. The disk component hence contains multiple files corresponding to the former memory components, which were persisted to disk.

Question 4: Log-structured Key-Value Stores

A. Describe what a read operation look likes in this LSM. In other words, given a particular key, how do you find the corresponding value.

B. A problem of the described LSM is that the files stored on disk could contain many duplicates of the same key, wasting space. Describe a “cleanup” operation that would solve this problem.

C. For what kinds of workloads or applications would the LSM described above be the most suitable for? Motivate your answer.

Answer:

A.

First, check if the key is in the in-memory component.

If the key is not found, search for it in all files on disk in reverse chronological order.

To keep read performance acceptable, LSM KV stores usually sort the files on disk.

Answer:

B.

This cleanup operation is called "compaction".

Compaction is performed by merging multiple files on disk into a single file, removing any duplicate and deleted keys. As the files are sorted on disk, a simple parallel merge sort can be used to implement compaction.

Answer:

C.

High write workloads such as transactions log file, persistent message queues, social network updates, file backups, etc.

LSM KV stores are better than B+ Trees for such workloads as they remove the need to perform dispersed write/update operations.

Question 5: RAID

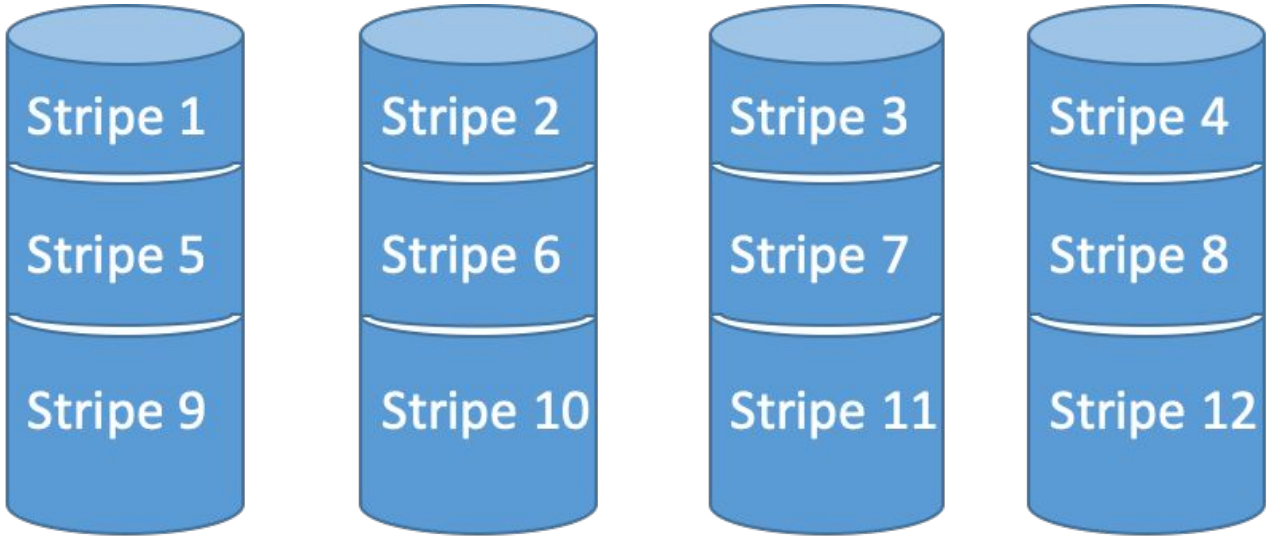
Consider a 4-disks, 256GB-per-disks RAID array. What is the available data storage capacity if:

A. Disks are organized as RAID 0?

B. Disks are organized as RAID 5?

Answer - RAID 0:

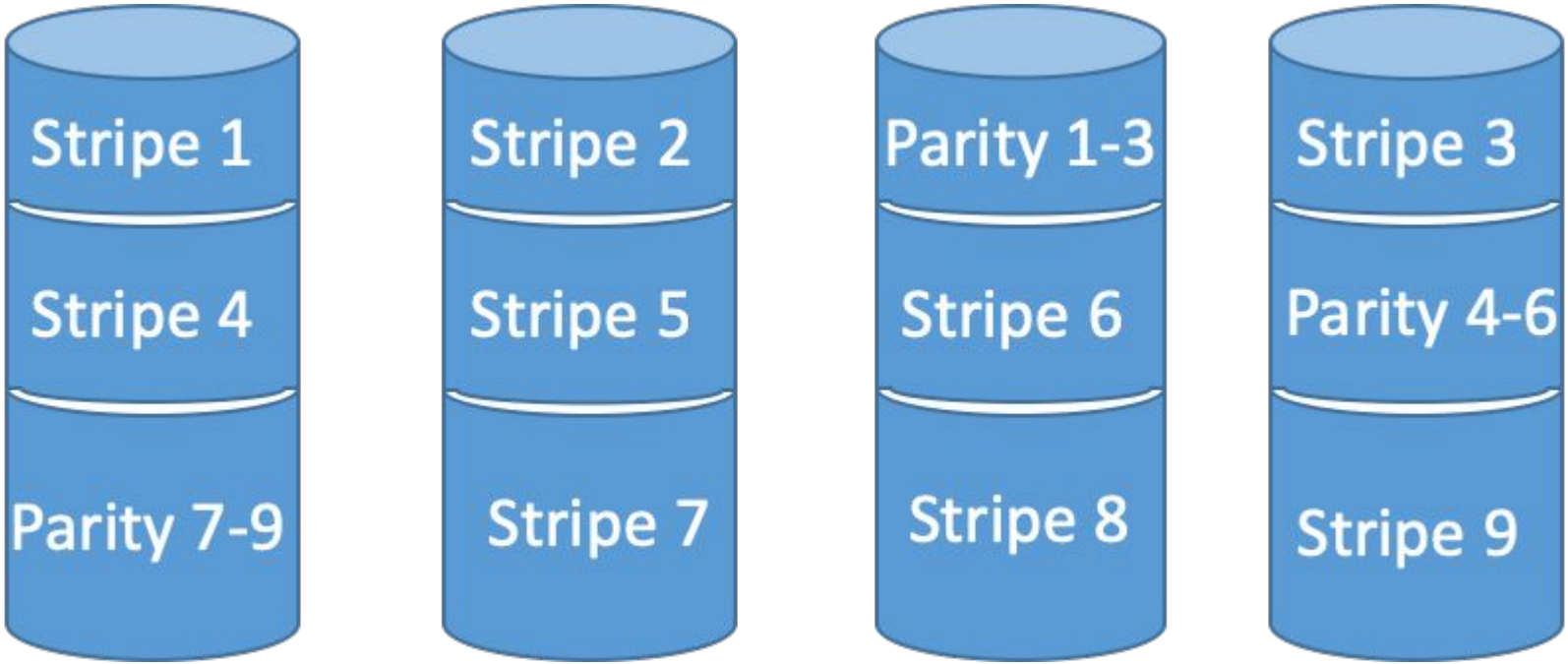
- No redundancy
- Data striped across **all** available disks



4x 256GB = 1 TB

Answer - RAID 5:

- For each disk with data - 1 parity
- Interleave parity with data



1 disk reserved for parity: 3 x 256GB = 768GB for data

Question 6: SSDs and LFS

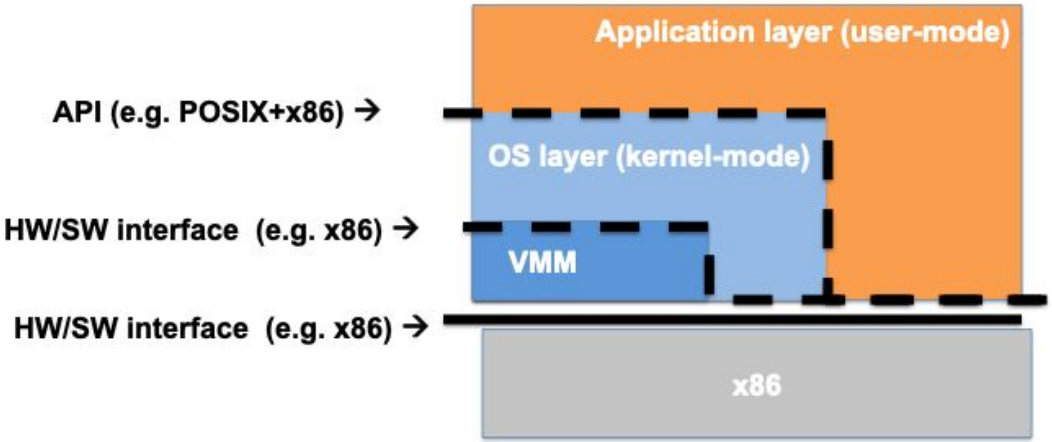
What makes a log-structured file system suitable as a file system for SSDs?

Answer:

- LFS is basically working the same way as SSDs
  - Append changes at the end of the log
  - As a result, you never update in place!
  - Writes are copy-on-write
    - Mark the old version as free space (~ erase)
    - Write the new version at the end of the log

Question 7: Virtualization

Assume an architecture that meets the Popek/Goldberg criteria for virtualization and follows the layering of this figure.



Question 7: Virtualization

Provide one example of a transition across each of the 5 boundaries that are pictorially represented in the figure, i.e.

A. Application layer – x86
B. Application – OS
C. OS – x86
D. OS – VMM
E. VMM – x86

Answer:

A - Application layer -x86

Unprivileged instructions (ex. Addition of two registers), regular CPU instructions in user mode

B - Application - OS

System calls: fork/wait, raise/signal, open/read/write/close file

C - OS - x86

Unprivileged instructions of the guest kernel

Answer:

D - OS - VMM

Privileged instructions of the guest kernel (generate a trap by the VMM). Hardware emulated by the virtual machine used by the virtualized OS.

E - VMM - x86

Any monitor instruction. Management of the real hardware emulated/multiplexed by the VMM, for use by the virtualized OS.