Storage and IO Systems
SGG: Chapters 12 & 13

I. Disk Scheduling [Worthington94]

A. We know disk performance is punitive

B. We've seen techniques at the OS level for mitigating
   1. Caching/Pre-fetching
   2. File System layout

C. We'll now look at some optimization algorithms performed by the device

D. Historically, these were algorithms implemented by the OS: back when the OS controlled the disk directly

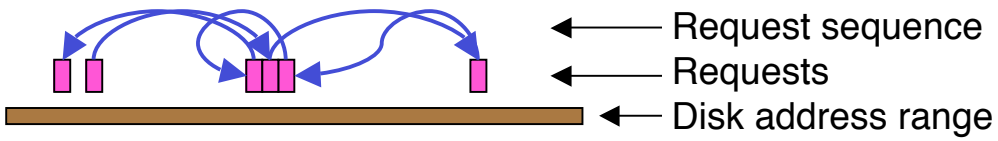
E. Nowadays, disks have their own controllers

F. Remember the costs associated with IO
   1. Command time, Seek time, Rotational delay, transfer time
   2. Seek time dominates

G. Other things to know: IOs are bursty and stochastic
   1. Bunches of requests will arrive with small inter-arrival times
      a) IOs request queue, which can grow quite large
   2. Followed by lengthy inter-arrival times (leads to idle time)
   3. Hard to predict what's going to be requested, and when

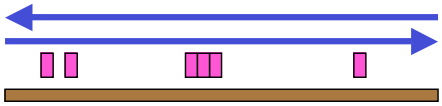
H. Q: How can we efficiently service this requests

I. Obvious first attempt: **first-come, first-served** (**FCFS**)

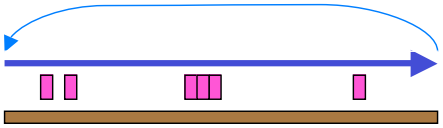


   1. Nothing done to optimize seek time, but fair
   2. almost always sub-optimal (i.e. a bad choice)

J. **SCAN** and **CSCAN**



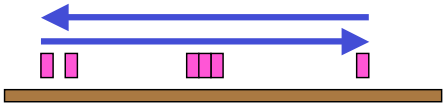
SCAN



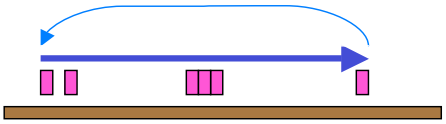
CSCAN

1. Early disk scheduling algorithm
2. SCAN: start at one end and sweep to the other end, then reverse direction
   a) sometimes called: the elevator algorithm
3. Cyclical (C) SCAN: like SCAN, but don't reverse
4. Q: Which is more fair?
   a) SCAN favors requests at the disk edges; non-uniform wait time for uniform request distributions

K. **LOOK** and **CLOOK**



LOOK



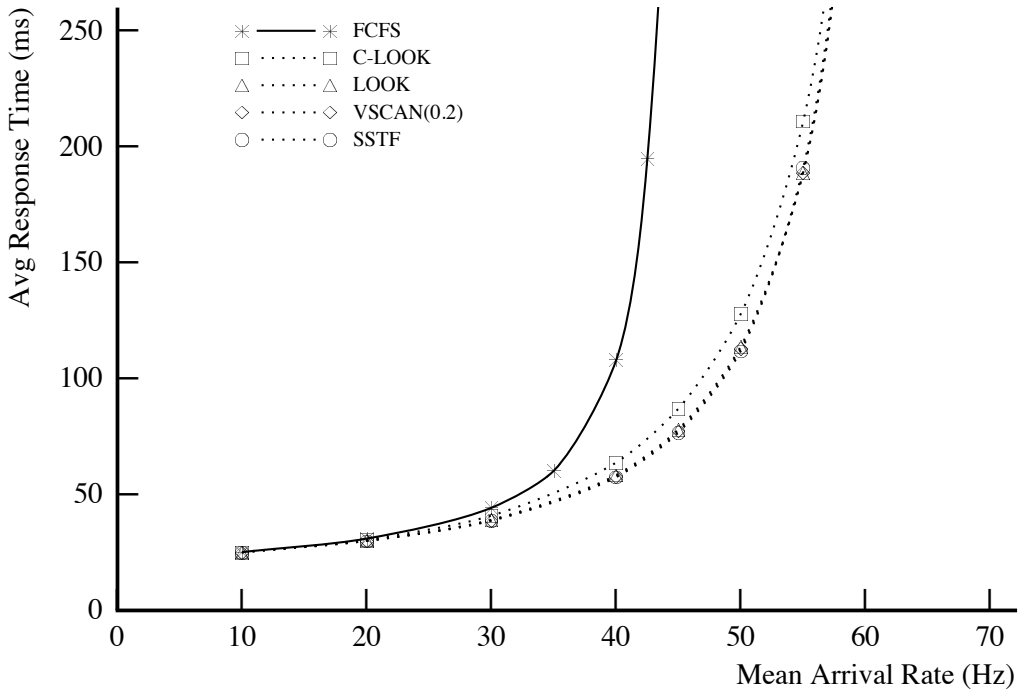
CLOOK

1. Like SCAN and SCAN, but don't blindly go to the end of the disk; stop and start only when there are actual requests
   a) True elevator scheduling; and usually what people mean when they say SCAN

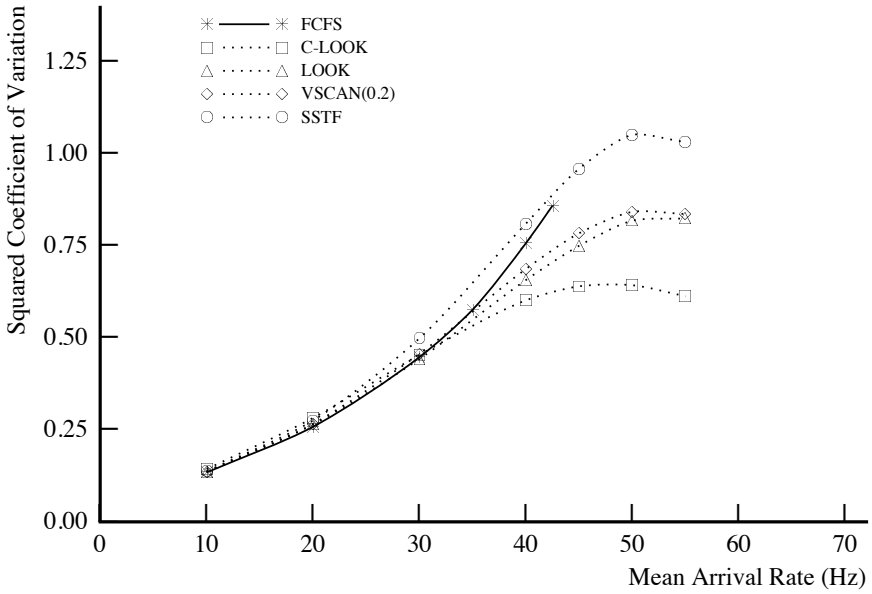
L. **Shortest Seek Time First (SSTF)**

1. service request with the shortest seek time from currently head location
   a) Assumes a time-distance metric
2. Our good friend: optimal (in terms of response time), but greedy and leads to starvation
   a) Requests far away from the current location may never get serviced

M. **Metrics** for algorithm comparison



1. Average response time (what is our average performance)



2. Squared coefficient of variation of response time
   a) variance divided by the squared mean
      (1) variance is squared std. deviation
   b) provides a indication of "starvation resistance"

N. Other variants

1. 2D-aware scheduling (remember the disk arm spiral?)
2. Cache-sensitive scheduling (cache aware)
3. Age-sensitive scheduling (to prevent starvation)
4. Priority-based scheduling (typically with priority queues)

O. Implementation

1. Request queueing takes place in both OS (at block device driver) and disk controller
2. Request scheduling can (and does) happen in both places
3. Can create conflicts
   a) BDD has limited knowledge of disk internals, so SSTF based on LBNs may be faulty
   b) Disk drive knows nothing about priorities, so may reorder for performance, resulting in deadline failures

II. Storage System Organizations

A. Single disks neither have the capacity, responsiveness, throughput or reliability to be supportive of many applications we desire

1. High-bandwidth applications: multi-media, scientific computing
2. High-capacity: “big data” applications
3. High-availability: any online service

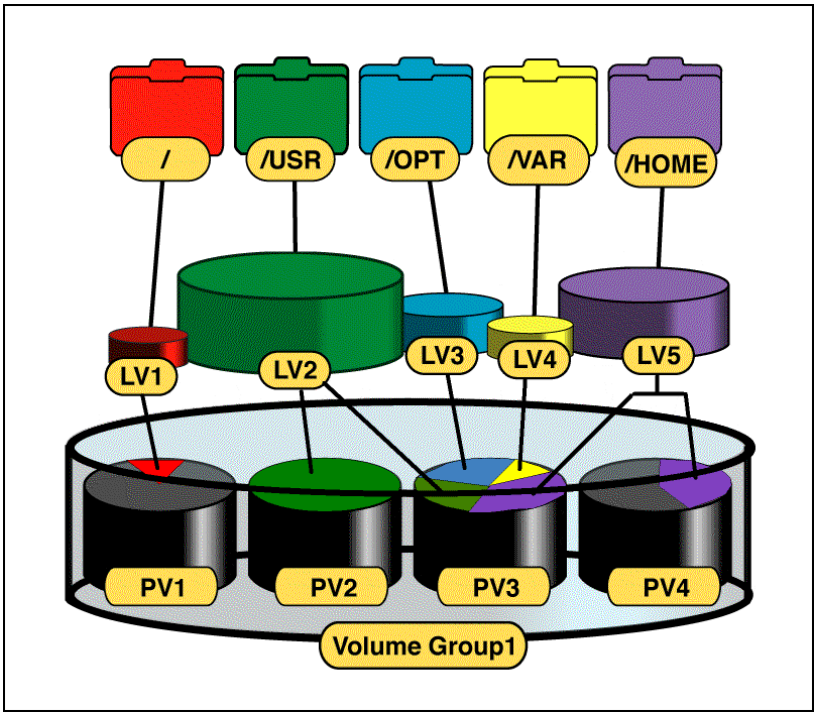
B. We’re limited by physics to increase capacity: bit-densities are approaching theoretical limits

C. Form factor and energy limit the physical size

D. Q: How do we organize multiple drives into a single logical unit of storage?

E. **Logical Volume Manager**: a way of organizing/abstracting multiple block devices into a single logical unit of storage

1. A software translation layer
2. Can provide additional features, like: encryption, snapshots, redundancy, hot-swapping, resizing



F. **Redundant Array of Inexpensive Disks (RAID)**: another type of volume manager

1. Often a physical organization of disks, and a hardware controller, providing a single logical view of all disks
   a) (Although software controllers exist)
2. Can provide: high throughput, parallel IO, reliability and transparent management
3. Consider Mean-time-til-failure (MTTF)
   a) e.g. 750,000 hours for a single disk
   b) A data center has 1,000,000 disks
   c) 1 disk failure every .75 hours (45 minutes)
   d) RAID gives us a way to tolerate and recover from these failures gracefully

G. RAID Terminology:

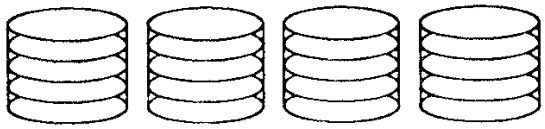
1. **Striping**: Spreading data across multiple drives, allowing parallel access to the drives
2. **Strip**: The allocation size on a particular disk
3. **Stripe**: The size of all strips; typically the unit of writing to a RAID device

H. Consider the stripe size:

1. Smaller means more flexibility in size of data stored (reduced internal fragmentation); decreased maximum throughput
2. Larger means high throughput (more disks writing/reading in parallel); higher internal fragmentation

III. RAID Levels

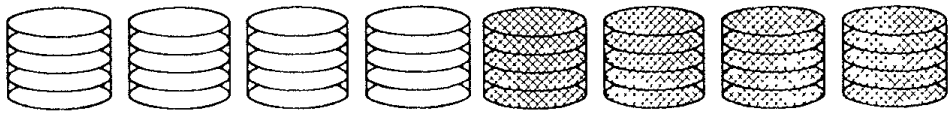
A. RAID 0



Non-Redundant (RAID Level 0)

1. Data are striped for increased capacity and throughput
2. A single disk failure results in data loss

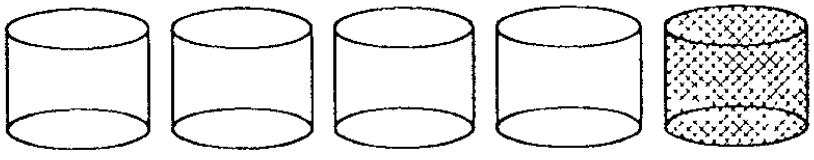
B. RAID 1: Mirroring



Mirrored (RAID Level 1)

1. The next most obvious organization
2. Data are striped across two identical RAID sets
3. Individual drives within a RAID set can fail, so long as their logical counterpart survives
   a) New disks can be hot-swapped and data rebuilt
4. Reads can be serviced by either set (e.g. whichever responds first)
5. High overhead (2X disk; 1/2 the capacity); writes are slower; two drive failures can cause data loss

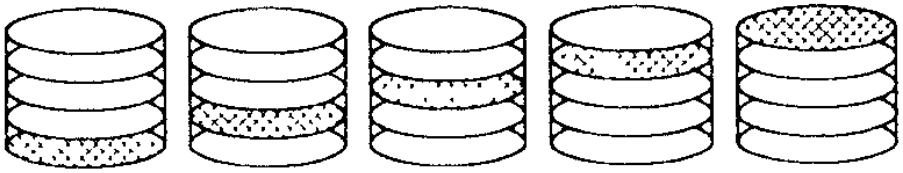
C. RAID 2/3: Parity disks



Bit-Interleaved Parity (RAID Level 3)

1. Designate a disk as a parity disk; parity being encoded data for redundancy
2. Many different encoding schemes; here's a simple one:
   a) $A \otimes B \otimes C = P$
   b) A dies; $P \otimes B \otimes C = A$
3. RAID-2 is bit-level striping with a dedicated parity disk (using a Hamming ECC); RAID-3 is byte-level striping with a dedicated parity disk (Ignore the label on the fig above); RAID-4 is block-level striping with dedicated parity disk
4. Reads and writes must always be in stripes
   a) Parity doesn't participate in reads, so reduces total array throughput
5. RAID levels 2-4 requires quite a bit of drive symmetry
   a) Even the synchronization of rotations
6. Further, they have a write bottleneck; only as fast as the parity drive

D. RAID 5: Block-interleaved, distributed parity



Block-Interleaved Distributed-Parity (RAID Level 5)

1. Round-robin assignment of parity; only supports a single drive failure, but
2. All drives perform in reads and writes
3. Best small read, large read, large write performance
   a) Terrible small write performance
   b) Small write problem: small-writes incur a lot of IO; consider the small update to a large stripe; this may require the entire stripe to read in, modified, new parity computed, and written back out again

E. RAID 6: Block-interleaved with double-distributed parity

1. Uses multiple parity blocks to guard against multiple disk

failures

F. Et al.: RAID levels have become a marketing tool, so RAID levels beyond 6 (10, 0+1, 100, 30, 50, 60) are typically some combination (often nested) of the above
G. **Hot-spares**: a drive in a RAID group that is blank and inactive, can be used to immediately rebuild a RAID should a single drive fail (without human administration); mitigates concerns about a double-drive failures while waiting for humans
H. Choosing a RAID level
    1. Applications typically dictate levels
    2. RAID 0 is high performance, and ok to tolerate data loss
    3. Other levels: How fast do you want to be able to recover
        a) RAID 1 instant recovery (plus background rebuilding)
        b) Parity levels requires time to rebuild, or at least compute lost data
I. Throughput

Table 3. Throughput per Dollar Relative to RAID Level 0.

	Small Read	Small Write	Large Read	Large Write	Storage Efficiency
RAID level 0	1	1	1	1	1
RAID level 1	1	1/2	1	1/2	1/2
RAID level 3	1/G	1/G	(G-1)/G	(G-1)/G	(G-1)/G
RAID level 5	1	max(1/G,1/4)	1	(G-1)/G	(G-1)/G
RAID level 6	1	max(1/G,1/6)	1	(G-2)/G	(G-2)/G

IV. Software-RAID

A. LVM can support RAID through software;
B. requires RAID computations (layout decisions, parity calculations, data recovery, etc) be done by the CPU;
C. doesn't have the insight to support all RAID levels (e.g. RAID 6);
D. RAID is now tied to the OS, making the hardware less portable/less transparent; not shared between OSes

E. Storage must be attached to the host; no support for remote storage
F. Additional bus congestion due to parity data