Disks

Tore Larsen
Including material developed by
Pål Halvorsen, University of Oslo (primarily)
Kai Li, Princeton University

Overview

- Disks
  - Organization and properties
- Disk scheduling
  - traditional
  - real-time
  - stream oriented
- Data placement
- Multiple disks
- Prefetching
- Memory caching

Disks

Disks

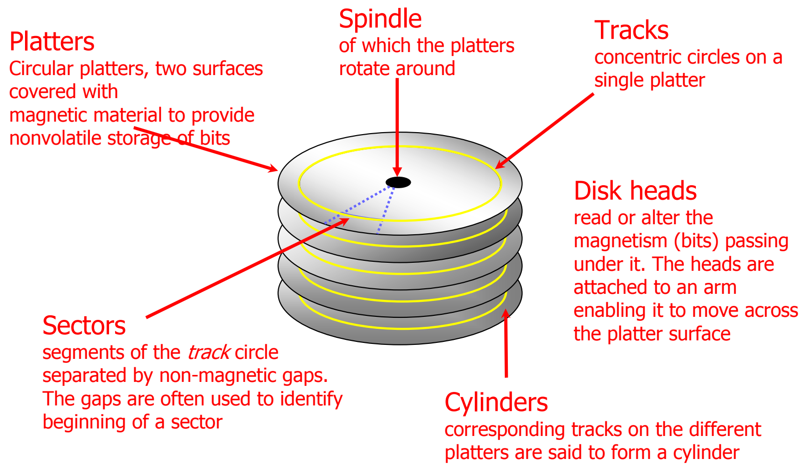
- Disks ...
  - Are I/O devices that can store data, including programs
  - While disk controller registers are directly accessible in SW *(through I/O ports or memory mapped I/O),* the data stored on disks are only accessible through *block-transfers* between disks and memory.
  - Offers **persistent storage,** in the sense that we expect data to survive a controlled cycling of power (power-down-power-up)
  - have *more capacity* than main memory
  - are *much cheaper* than main memory
  - are *orders of magnitude slower* than main memory

Disks

- Two resources of importance
  - storage space
  - I/O bandwidth
- Because...
  - ...there is a *large* speed mismatch (ms vs. ns) compared to main memory...disk I/O is *the* performance bottleneck for some applications
    - *May be the case also for computational tasks, i.e. oil reservoir modelling*
  - ...we need to minimize the number of accesses,
  - ...try to spread out the traffic in time and space
  - ...

  ...we must consider what disk technology to use, and how to use it!

Disk Organization



Disk Technology Trends

- Packing density is increasing
  - Linear density (bits/inch) is increasing exponentially
  - Track density (tracks/inch) is increasing exponentially
  - Areal density (the product of track and linear density) increases exponentially *(doubles per 18 months?)*
- Increasing transfer speed
  - Higher packing density
  - New interconnect technologies
  - Better buffering
  - Some increase in rotation speed
- Decreasing form factors
  - Less power/GB
  - New applications (ipods, cameras?)
  - Tighter packaging

Disk Market Trends

- Disks are getting cheaper
  - About a factor of two per year since 1991
- COTS Prevalence
  - Common-Off-The-Shelf technologies prevail in market
    - Technologies developed for mass market use continuously threatens technologies applied at higher price-ponts because development costs are amortized over more units sold. With lacking market shares, the more exclusive technology may loose first in performance/cost, and eventually also in performance
    - An aside: An important issue arises of when to hang on the true and tested, when to go with the winds of change? Too early and too late may be equally expensive.
  - COTS work when we have a synergy of technology push and market pull
    - We are able to develop the technologies further, and there are markets willing to pay for our development efforts and the products that arise

Is there a Future for Disks?

- Disks have repeatedly been doomed a dead-end technology by respected computer scientists, because of moving mechanical parts. That hasn't happened yet.
  - M. Flynn volunteers the information that he advised IBM to get out of the disk business. Fortunately, he says, they didn't follow his advise, and moved on to make lots of money on disks
- Look for new applications of disks …
  - Camera!?
  - Back-up!!
- …and new usage
  - "Hang to your life-time of data"

Disk Specifications

Note 1:
disk manufacturers usually denote GB as 10^9 whereas computer quantities often are powers of 2, i.e., GB is 2^{30}

- Disk technology develops "fast"
- Some existing (Seagate) disks today (2002):

	Barracuda 180	*Cheetah 36*	*Cheetah X15*	**X15.3**
Capacity (GB)	181.6	36.4	36.7	73.4
Spindle speed (RPM)	7200	10.000	15.000	
#cylinders (and tracks)	24.247	9.772	18.479	
average seek time (ms)	7.4	5.7	3.6	
min (track-to-track) seek (ms)	0.8	0.6	0.3	0.2
max (full stroke) seek (ms)	16	12	7	
average latency (ms)	4.17	3	2	
internal transfer rate (Mbps)	282 – 508	520 – 682	522 – 709	609 – 891
disk buffer cache	16 MB	4 MB	8 MB	

Note 2:
there is a difference between internal and formatted transfer rate. ***Internal*** is only between platter. ***Formatted*** is after the signals interfere with the electronics (cabling loss, interference, retransmissions, checksums, etc.)

Note 3:
At any given time, there is usually a trade off between speed and capacity

Disk Capacity

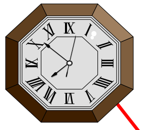
- The size (storage space) of the disk is dependent on
  - the number of platters
  - whether the platters use one or both sides
  - number of tracks per surface
  - (average) number of sectors per track
  - number of bytes per sector
- Example (Cheetah X15):
  - 4 platters using both sides: 8 surfaces
  - 18497 tracks per surface
  - 617 sectors per track (average)
  - 512 bytes per sector
  - Total capacity = 8 x 18497 x 617 x 512 $\approx$ 4.6 x 10^{10} = 42.8 GB
  - Formatted capacity = 36.7 GB

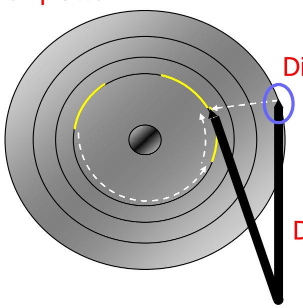
Note:
there is a difference between formatted and total capacity. Some of the capacity is used for storing checksums, spare tracks, gaps, etc.

Disk Access Time

- How do we retrieve data from disk?
  - position head over the cylinder (track) on which the block (consisting of one or more sectors) are located
  - read or write the data block as the sectors move under the head when the platters rotate
- The time between the moment issuing a disk request and the time the block is resident in memory is called *disk latency* or *disk access time*

Disk Access Time

I want block X →  → block x in memory



Disk access time =

Seek time

+ Rotational delay

+ Transfer time

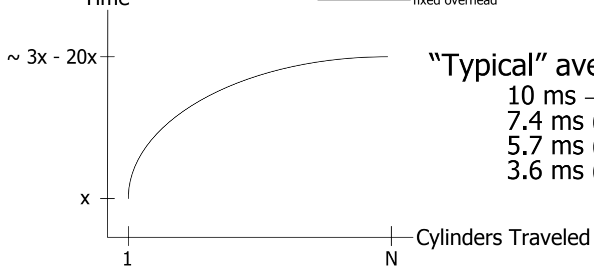
+ Other delays

Disk Access Time: Seek Time

- Seek time is the time to position the head
  - the heads require a minimum amount of time to start and stop moving the head
  - some time is used for actually moving the head – roughly proportional to the number of cylinders traveled
  - Time to move head:

$$\alpha + \beta\sqrt{n}$$

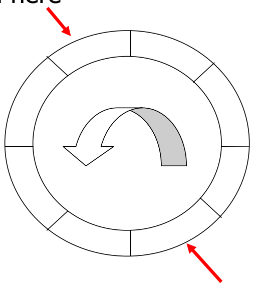
(n – number of tracks; β – seek time constant; α – fixed overhead)



"Typical" average:
10 ms → 40 ms
7.4 ms (Barracuda 180)
5.7 ms (Cheetah 36)
3.6 ms (Cheetah X15)

Disk Access Time: Rotational Delay

- Time for the disk platters to rotate so the first of the required sectors are under the disk head



Average delay is 1/2 revolution

"Typical" average:

8.33 ms	(3.600 RPM)
5.56 ms	(5.400 RPM)
4.17 ms	(7.200 RPM)
3.00 ms	(10.000 RPM)
2.00 ms	(15.000 RPM)

Disk Access Time: Transfer Time

- Time for data to be read by the disk head, i.e., time it takes the sectors of the requested block to rotate under the head
- Transfer rate ≤ $\frac{\text{amount of data per track}}{\text{time per rotation}}$
- Transfer time = amount of data to read / transfer rate
- Example – *Barracuda 180:* 406 KB per track x 7.200 RPM ≈ 47.58 MB/s
- Example – *Cheetah X15:* 316 KB per track x 15.000 RPM ≈ 77.15 MB/s

Note: one might achieve these transfer rates reading continuously on disk, but time must be added for seeks, etc.

- Transfer time is dependent on data density and rotation speed
- If we have to change track, time must also be added for moving the head

Disk Access Time: Other Delays

- There are several other factors which might introduce additional delays:
  - CPU time to issue and process I/O
  - contention for controller
  - contention for bus
  - contention for memory
  - verifying block correctness with checksums (retransmissions)
  - **waiting in scheduling queue**
  - ...
- Typical values: "0"
  (maybe except from waiting in the queue)

Disk Throughput

- How much data can we retrieve per second?
- Throughput = $\frac{\text{data size}}{\text{transfer time (including all)}}$
- Example:
  for each operation we have
  - average seek - average rotational delay
  - transfer time - no gaps, etc.

  - Cheetah X15 (max 77.15 MB/s)
    4 KB blocks → 0.71 MB/s
    64 KB blocks → 11.42 MB/s
  - Barracuda 180 (max 47.58 MB/s)
    4 KB blocks → 0.35 MB/s
    64 KB blocks → 5.53 MB/s

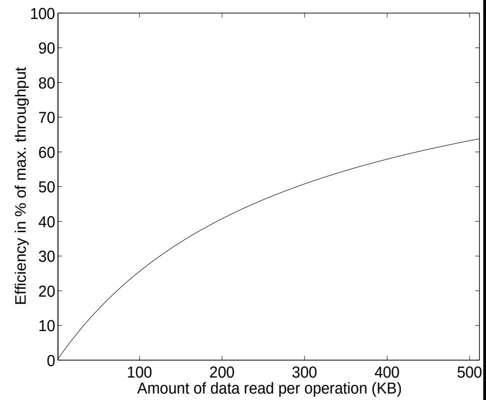
Block Size

- The block size may have large effects on performance
- Example:
  assume random block placement on disk and sequential file access
  - doubling block size will halve the number of disk accesses
    - each access take some more time to transfer the data, but the total transfer time is the same (i.e., more data per request)
    - halve the seek times
    - halve rotational delays are omitted
  - e.g., when increasing block size from 2 KB to 4 KB (no gaps,...)
    for *Cheetah X15* typically an average of:
    - ☺ 3.6 ms is *saved* for seek time
    - ☺ 2 ms is *saved* in rotational delays
    - ☹ 0.026 ms is *added* per transfer time

    } saving a total of 5.6 ms when reading 4 KB (49,8 %)
  - increasing from 2 KB to 64 KB saves ~96,4 % when reading 64 KB

Block Size

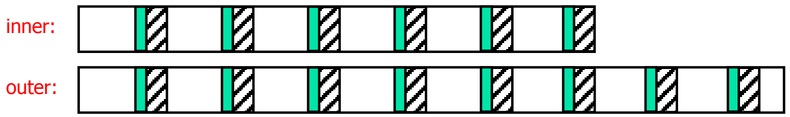
- Thus, increasing block size can increase performance by reducing seek times and rotational delays
- However, a large block size is not always best
  - blocks spanning several tracks still introduce latencies
  - small data elements may occupy only a fraction of the block



- Which block size to use therefore depends on data size and data reference patterns
- The trend, however, is to use large block sizes as new technologies appear with increased performance – at least in high data rate systems

Disk Access Time: Some Complicating Issues

- There are several complicating factors:
  - the "other delays" described earlier like consumed CPU time, resource contention, etc.
  - unknown data placement on modern disks
  - zoned disks, i.e., outer tracks are longer and therefore usually have more sectors than inner - transfer rates are higher on outer tracks
  - gaps between each sector
  - checksums are also stored with each the sectors
    - read for each track and used to validate the track
    - usually calculated using Reed-Solomon interleaved with CRC
    - for older drives the checksum is 16 bytes
  - (SCSI disks sector sizes may be changed by user!!??)



Writing and Modifying Blocks

- A write operation is analogous to read operations
  - must add time for block allocation
  - a complication occurs if the write operation has to be *verified* – must wait another rotation and then read the block to see if it is the block contains what we wanted to write
  - Total write time ≈ read time + time for one rotation
- Cannot modify a block directly:
  - read block into main memory
  - modify the block
  - write new content back to disk
  - (verify the write operation)
  - Total modify time ≈ read time + time to modify + write time

Disk Controllers

- To manage the different parts of the disk, we use a *disk controller*, which is a small processor capable of:
  - controlling the actuator moving the head to the desired track
  - selecting which platter and surface to use
  - knowing when right sector is under the head
  - transferring data between main memory and disk
- New controllers acts like small computers themselves
  - both disk and controller now has an own buffer reducing disk access time
  - data on damaged disk blocks/sectors are just moved to spare room at the disk – the system above (OS) does not know this, i.e., a block may lie elsewhere than the OS thinks

Efficient Secondary Storage Usage

- Must take into account the use of secondary storage
  - there are large access time gaps, i.e., a disk access will probably dominate the total execution time
  - there may be huge performance improvements if we reduce the number of disk accesses
  - a "slow" algorithm with few disk accesses will probably outperform a "fast" algorithm with many disk accesses
- **Several ways to optimize .....**
  - block size
  - disk scheduling
  - multiple disks
  - prefetching
  - file management / data placement
  - memory caching / replacement algorithms
  - …

Disk Scheduling

Disk Scheduling

- **Seek time is a dominant factor of total disk I/O time**
- Let operating system or disk controller choose which request to serve next depending on the head's current position and requested block's position on disk (disk scheduling)
- Note that disk scheduling ≠ CPU scheduling
  - a mechanical device – hard to determine (accurate) access times
  - disk accesses cannot be preempted – runs until it finishes
  - disk I/O often the main performance bottleneck
- General goals
  - short response time
  - high overall throughput
  - fairness (equal probability for all blocks to be accessed in the same time)
- Tradeoff: seek and rotational delay vs. maximum response time

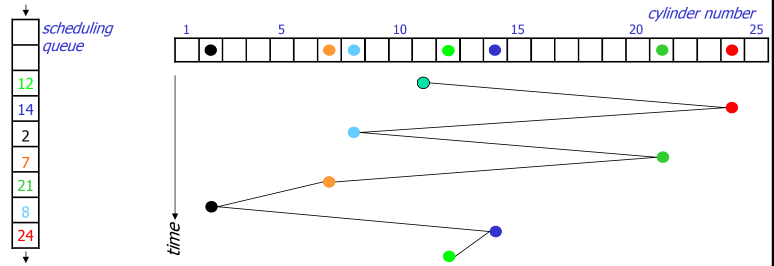
Disk Scheduling

- Several traditional algorithms
  - First-Come-First-Serve (FCFS)
  - Shortest Seek Time First (SSTF)
  - SCAN (and variations)
  - Look (and variations)
  - …

First–Come–First–Serve (FCFS)

FCFS serves the first arriving request first:
- Long seeks
- "Short" average response time

incoming requests (in order of arrival):
12 14 2 7 21 8 24



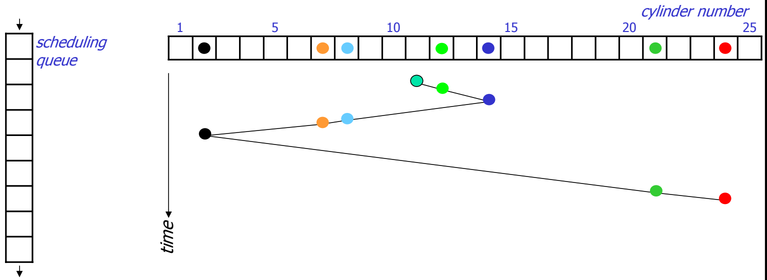
Shortest Seek Time First (SSTF)

SSTF serves closest request first:

- short seek times
- longer maximum response times – may even lead to starvation

incoming requests (in order of arrival):

12 14 2 7 21 8 24



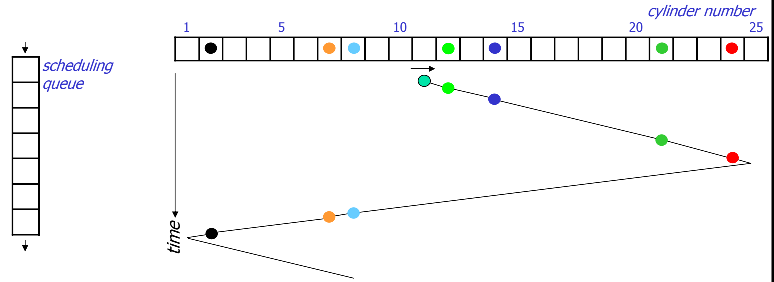
SCAN

SCAN (elevator) moves head edge to edge and serves requests on the way:

- bi-directional
- compromise between response time and seek time optimizations

incoming requests (in order of arrival):

12 14 2 7 21 8 24



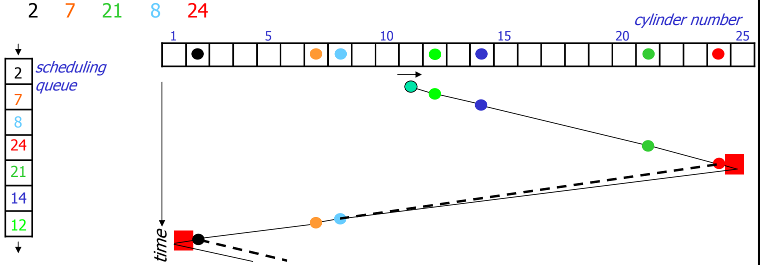
LOOK

LOOK is a variation of SCAN:

- same schedule as SCAN
- does not run to the edges
- stops and returns at outer- and innermost request
- increased efficiency
- SCAN vs. LOOK example:

incoming requests (in order of arrival):

12 14 2 7 21 8 24

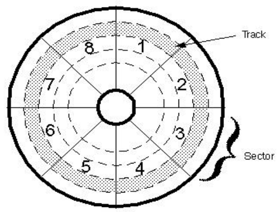


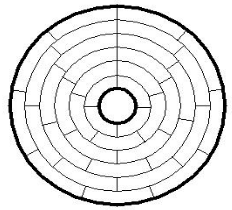
Data Placement on Disk

Data Placement on Disk

- Disk blocks can be assigned to files many ways, and several schemes are designed for
  - ❑ optimized latency
  - ❑ increased throughput
  - ➪ access pattern dependent

Disk Layout





- Constant angular velocity (CAV) disks
  - ❑ equal amount of data in each track (and thus constant transfer time)
  - ❑ constant rotation speed
- Zoned CAV disks
  - ❑ zones are ranges of tracks
  - ❑ typical few zones
  - ❑ the different zones have
    - different amount of data
    - different bandwidth
    - i.e., more better on outer tracks

Disk Layout

- Cheetah X15.3 is a zoned CAV disk:

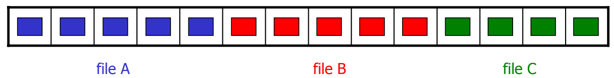
Zone	Cylinders per Zone	Sectors per Track	Spare Cylinders	Zone Transfer Rate Mb/s	Sectors per Zone	Efficiency	Formatted Capacity (Mbytes)
0	**3544**	**672**	7	**890,98**	19014912	77,2%	**9735,635**
1	3382	652	7	878,43	17604000	76,0%	9013,248
3	3079	624	6	835,76	15340416	76,5%	7854,293
4	2939	595	6	801,88	13961080	76,0%	7148,073
5	2805	576	6	755,29	12897792	78,1%	6603,669
6	2676	537	5	728,47	11474616	75,5%	5875,003
7	2554	512	5	687,05	10440704	76,3%	5345,641
8	2437	480	5	649,41	9338880	75,7%	4781,506
9	2325	466	5	632,47	8648960	75,5%	4428,268
10	**2342**	**438**	5	**596,07**	8188848	75,3%	**4192,690**

✓ Always place often used data on outermost tracks (zone 0) …!?

➪ **NO**, arm movement is often more important than transfer time

Data Placement on Disk

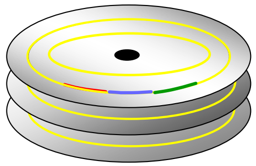
- Contiguous placement stores disk blocks contiguously on disk



- ❑ minimal disk arm movement reading the whole file (no intra-file seeks)
- ❑ possible advantage
  - head must not move between read operations - no seeks or rotational delays
  - can approach theoretical transfer rate
  - often WRONG: read other files as well
- ❑ real advantage
  - do not have to pre-determine block (read operation) size (whatever amount to read, at most track-to-track seeks are performed)
- ❑ no inter-operation gain if we have unpredictable disk accesses

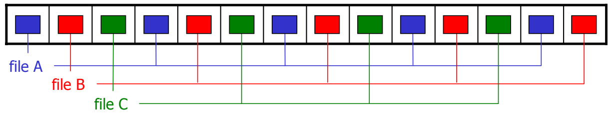
Data Placement on Disk

- To avoid seek time (and possibly rotational delay), we can *store data likely to be accessed together* on
  - adjacent sectors
    (similar to using larger blocks)
  - if the track is full, use another track
    on the same cylinder
    (only use another head)
  - if the cylinder is full, use
    next (adjacent) cylinder
    (track-to-track seek)



Data Placement on Disk

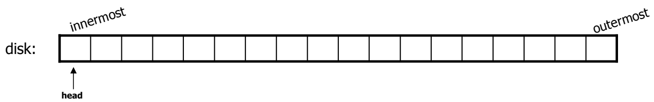
- Interleaved placement tries to store blocks from a file with a fixed number of other blocks in-between each block



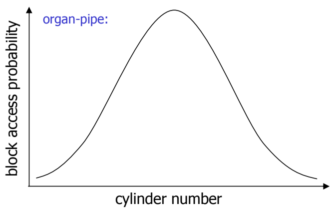
  - minimal disk arm movement reading the files A, B and C
    (starting at the same time)
  - fine for predictable workloads reading multiple files
  - no gain if we have unpredictable disk accesses
- Non-interleaved (or even random) placement can be used for highly unpredictable workloads

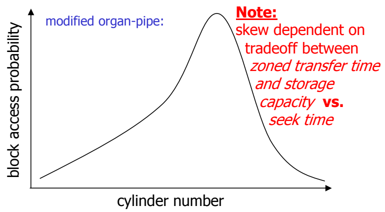
Data Placement on Disk

- Organ-pipe placement consider the usual disk head position
  - place most popular data where head is most often



  - center of the disk is closest to the head using CAV disks
  - but, a bit outward for *zoned* CAV disks (modified organ-pipe)





Prefetching and Buffering

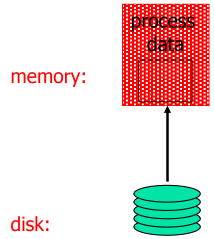
Prefetching

- If we can predict the access pattern, one might speed up performance using prefetching
  - a video playout is often linear → easy to predict access pattern
  - eases disk scheduling
  - read larger amounts of data per request
  - data in memory when requested – reducing page faults
- One simple (and efficient) way of doing prefetching is read-ahead:
  - read more than the requested block into memory
  - serve next read requests from buffer cache
- Another way of doing prefetching is double (multiple) buffering:
  - read data into first buffer
  - process data in *first* buffer and at the same time read data into *second* buffer
  - process data in *second* buffer and at the same time read data into *first* buffer
  - etc.

Multiple Buffering

- Example:
  have a file with block sequence B1, B2, ...
  our program processes data sequentially, i.e., B1, B2, ...
  - single buffer solution:
    - read B1 → buffer
    - process data in buffer
    - read B2 → buffer
    - process data in Buffer
    - ...
    - if P = time to process a block
      R = time to read in 1 block
      n = # blocks

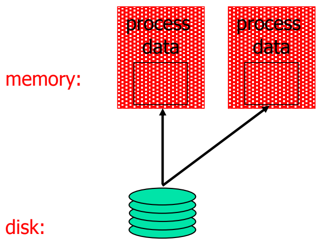
    single buffer time = n (P+R)



Multiple Buffering

- double buffer solution:
  - read B1 → buffer1
  - process data in buffer1, read B2 → buffer2
  - process data in buffer2, read B3 → buffer1
  - process data in buffer1, read B4 → buffer2
  - ...
  - if P = time to process a block
    R = time to read in 1 block
    n = # blocks

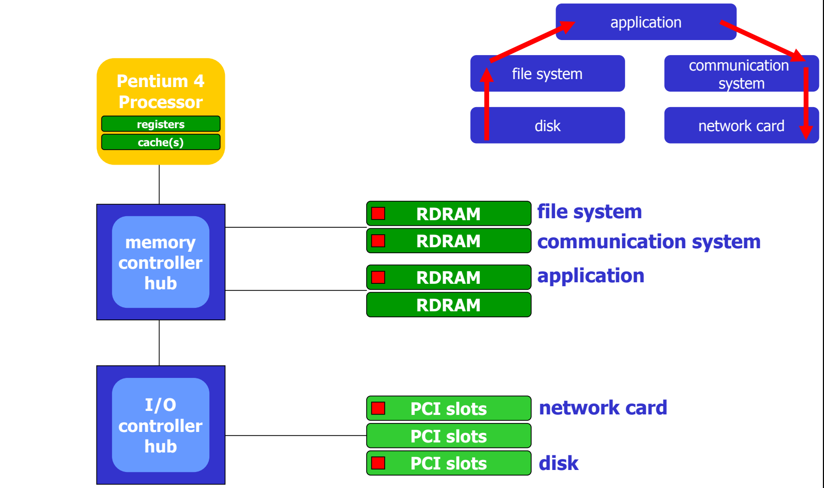
  if P ≥ R
  double buffer time = R + nP



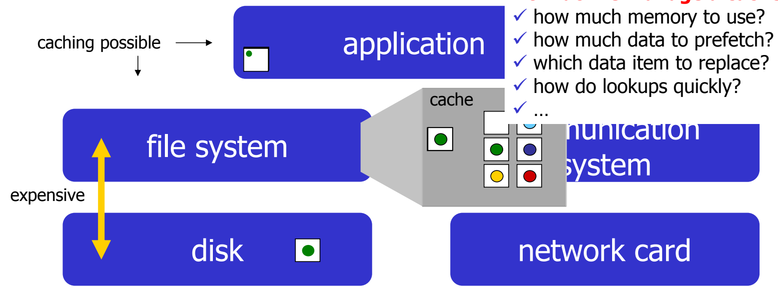
- if P < R, we can try to add buffers (*n* - buffering)

Memory Caching

Data Path (Intel Hub Architecture)



Memory Caching



How do we manage a cache?

- ✓ how much memory to use?
- ✓ how much data to prefetch?
- ✓ which data item to replace?
- ✓ how do lookups quickly?
- ✓ ...

Memory Caching