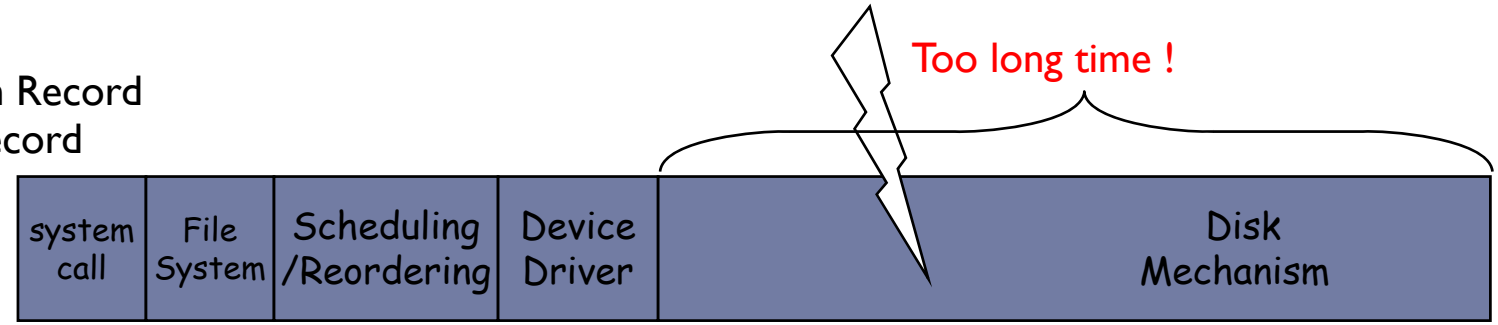
New(old) problems in HDDs: How to cope with them ?

Heon Y. Yeom

Seoul National University

Disk I/O Performance

Read 302-th Record
Write xxx to 10-th Record
Read 777-th Record



- Disk I/O performance sucks. Why?
  - Disks require "Mechanical Movement" to access stored data.
    - Head switch, Platter rotation.
- So, Reducing the mechanical delay is important.
  - This begins by an exact modeling of physical disk layout.
  - The Key challenge is that hard-disk itself is a "**black-box**".

What has changed in 25 years ?

- Smaller size
- Larger capacity
- About the same speed
  - seek time + rotational delay + transfer time
- More intelligence
  - TCQ, NCQ

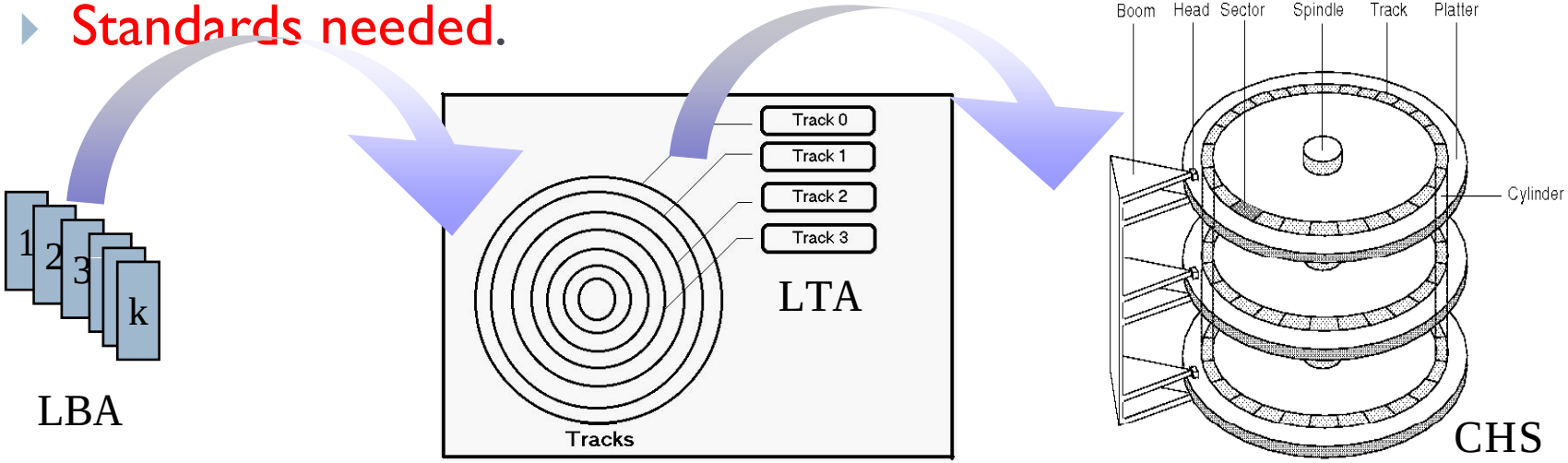
- Are they good ?

What does OS expect from HDD ?

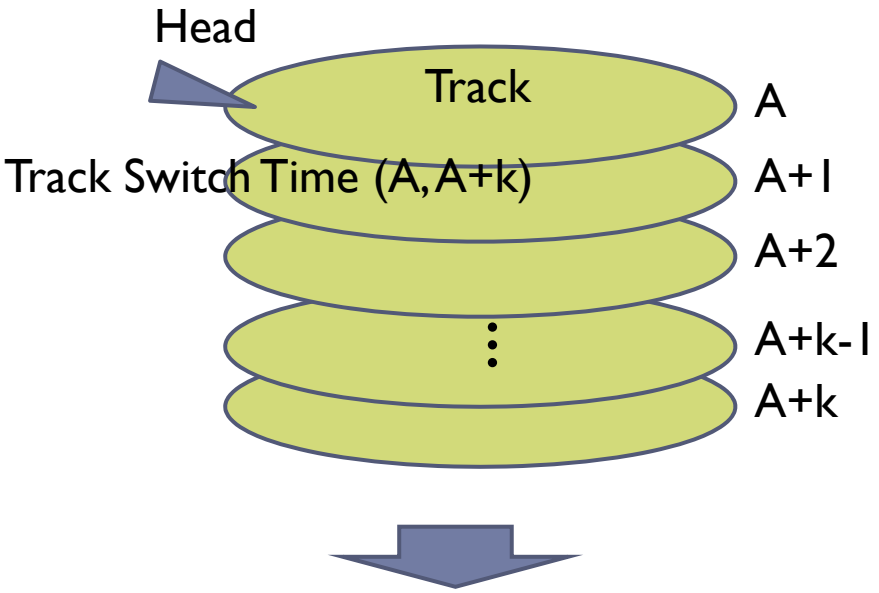
- Linear array of fixed size blocks.
  - not really
  - (Cylinder, Head, Sector).
- I want to read values, fast.
- I want to write values, fast, without loss.
  - => disk scheduling algorithms
- I want some performance guarantee !!

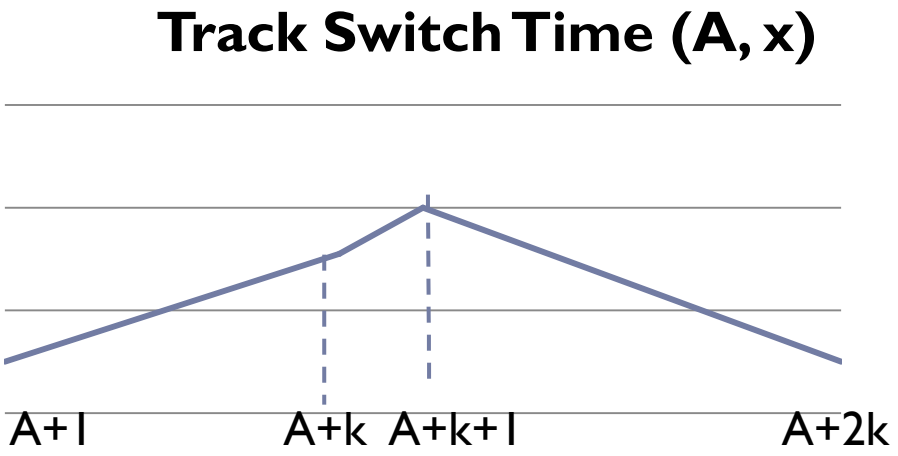
Disk layout information

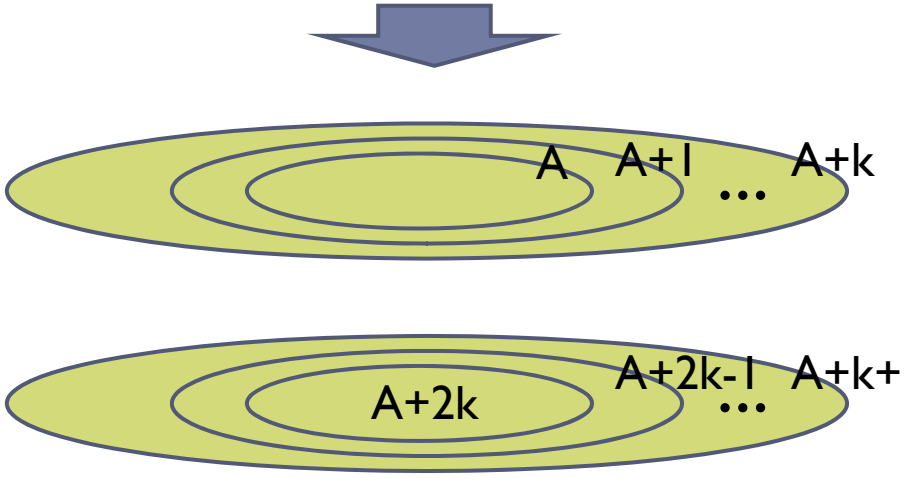
- Is hidden.
- Different from disk to disk.
- It is possible … but,
  - Takes 2-3 days for today's disk.
- If we have the information,
  - It is possible to predict exact timing.
- Standards needed.

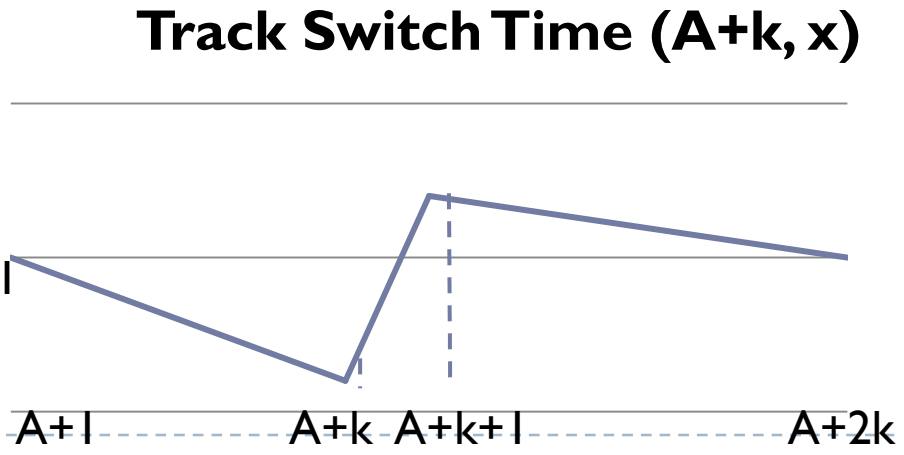


Track Layout Gathering Example



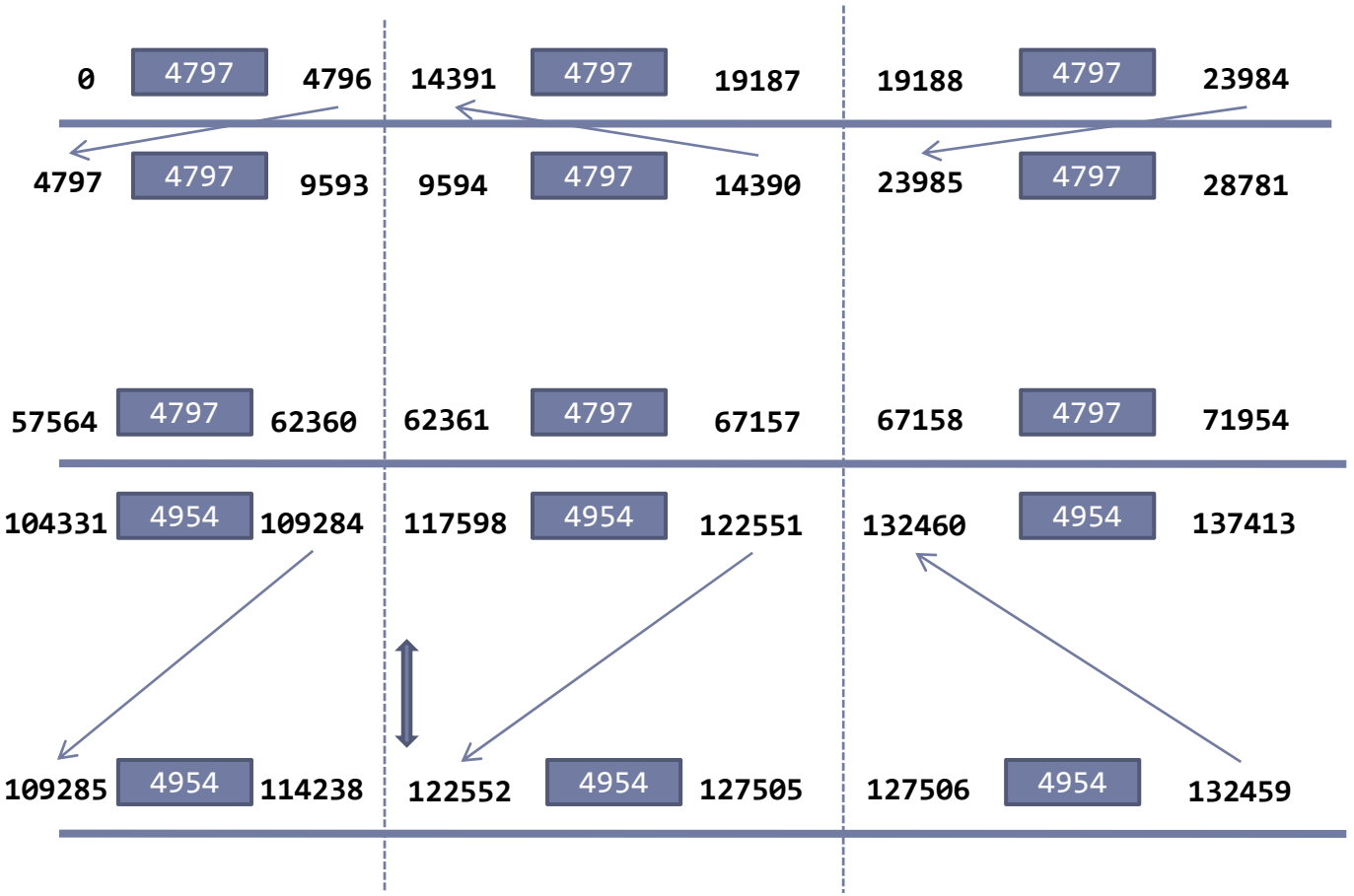






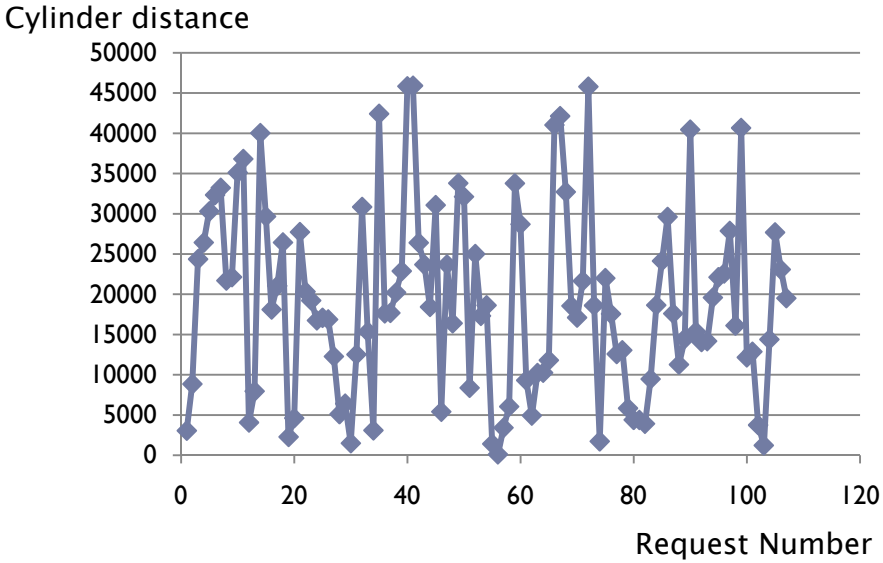
Possible Mapping : 2 platters

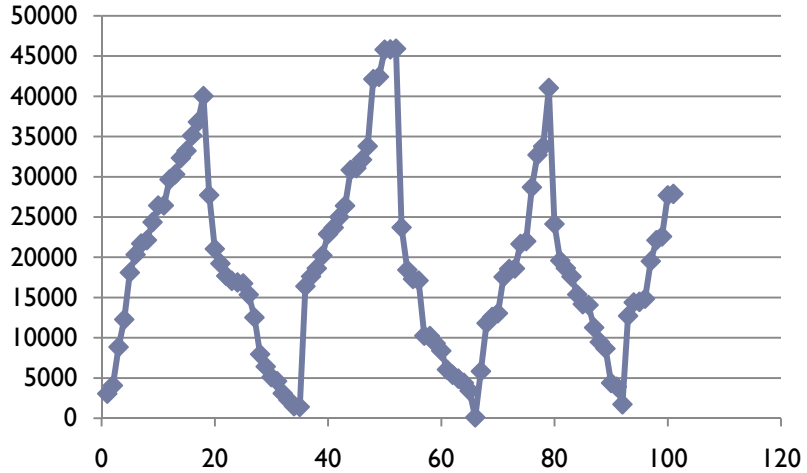
Samsung Disk **hd400ld** layout



Performance Evaluation (Samsung hd400ld)

Filebench Trace with 1,000 requests (Queue size : 10)





- LBA-based Scheduling

Total cyl distance	9,611,435
Average Service Time (usec)	10,975
Average Wait Time (usec)	111,437

- CHS-based Scheduling

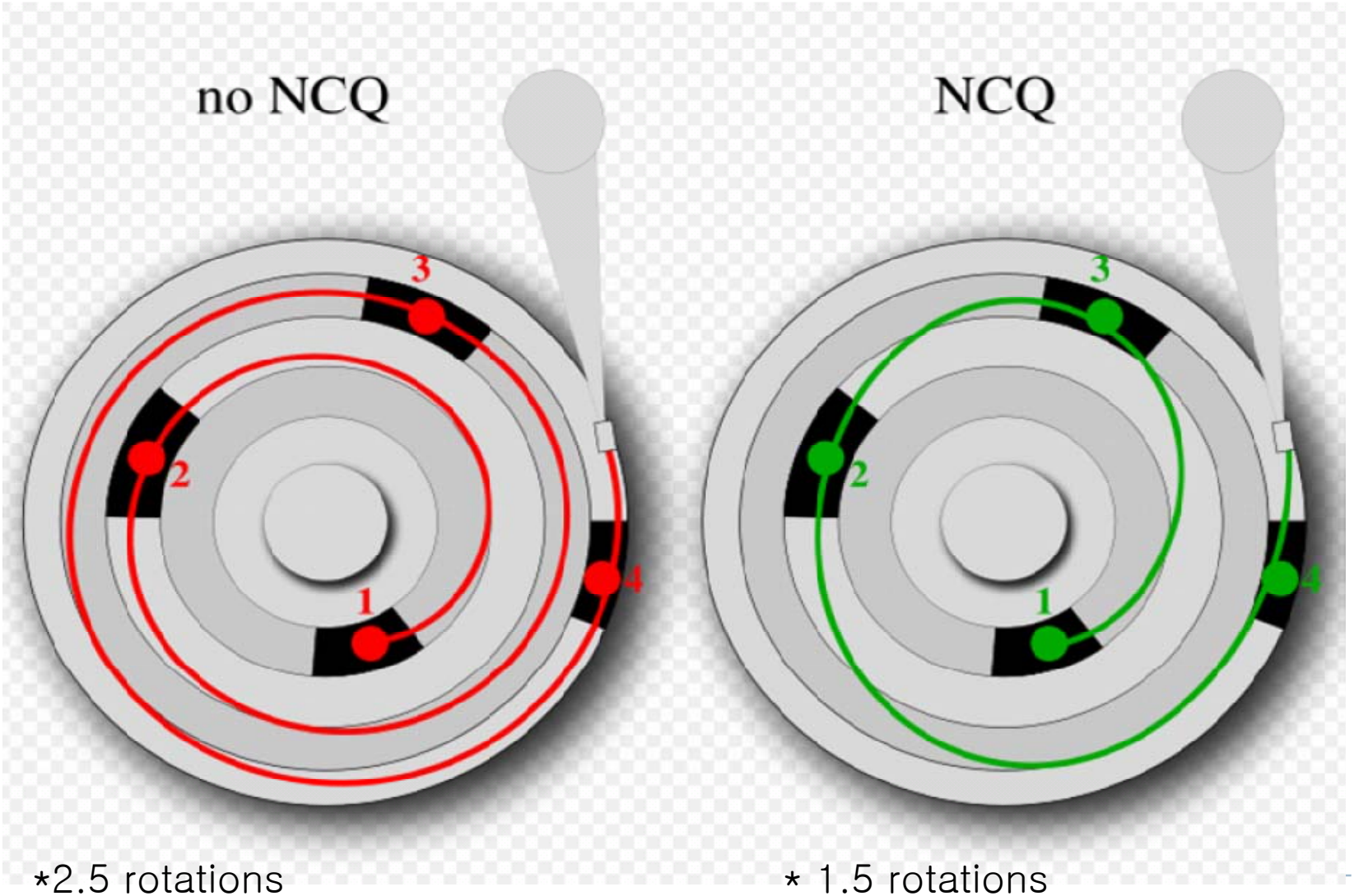
3,147,948	(32.7%)
9,532	(86.8 %)
97,133	(87.1 %)

Is it enough ? No !

Disk Model	Hd400ld	Wd2500jb	St330007lw	St314007lw
Interface	IDE	IDE	SCSI	SCSI
Capacity	400GB	250GB	300GB	146GB
Cylinder	6	6	8	4
LBA-based Average Response Time	10,975	10,209	4,048	5,113
CHS-based Average Response Time	9,532(86.8%)	9,567(93.7%)	3,809(94.1%)	5,000(97.8%)
Seek Distance (%)	32.7%	35.3%	33.7%	39.1%

We can probably do better inside the disk

Here it comes –
Native Command Queueing(NCQ)

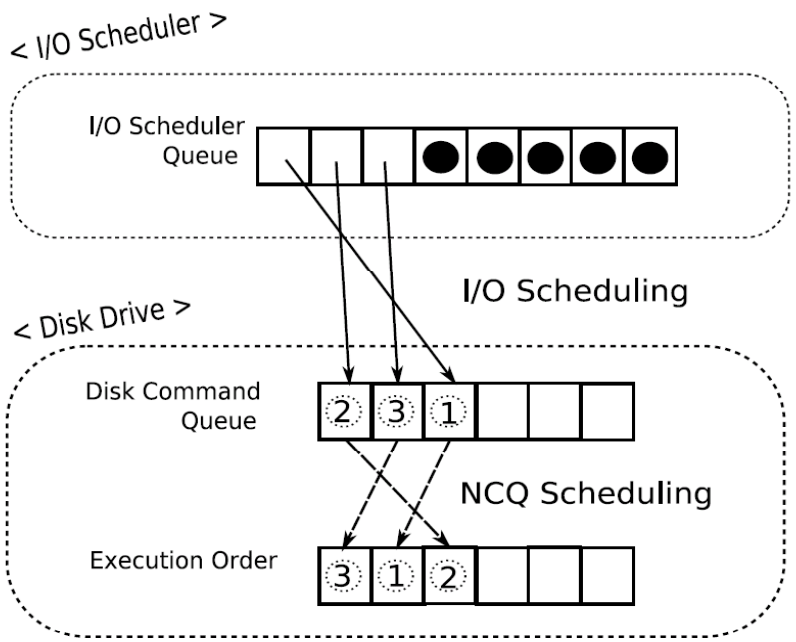


*2.5 rotations

* 1.5 rotations

From Wikipedia...

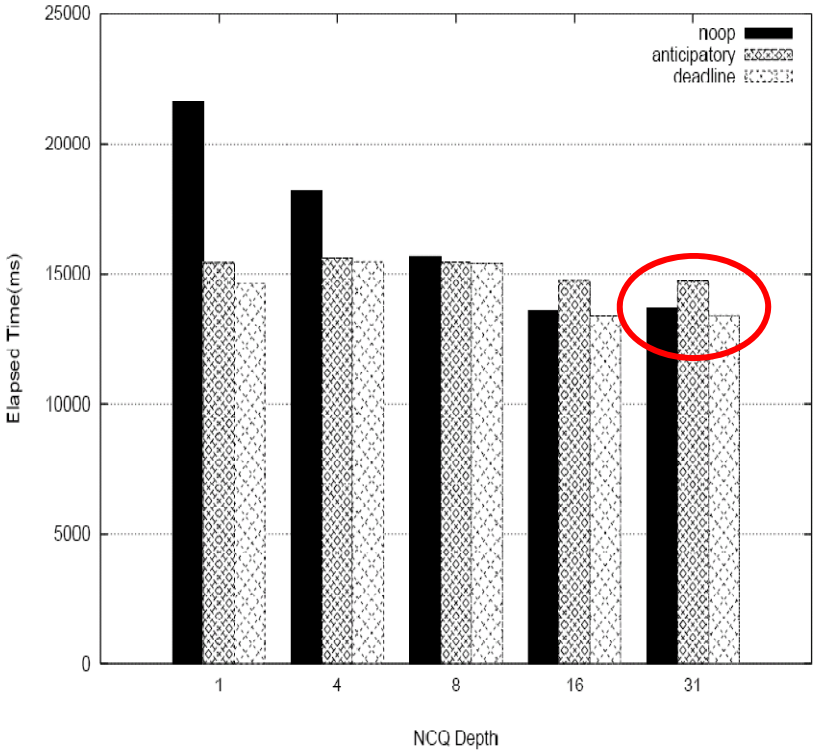
Problem 1. - Redundant Scheduling



- **Case I.**
  - No synergy effect.
  - I/O Scheduler + NCQ ≈ I/O Scheduler(or NCQ)
- **Case II.**
  - Sometimes, it's worse !
  - I/O Scheduler + NCQ < I/O Scheduler(or NCQ)

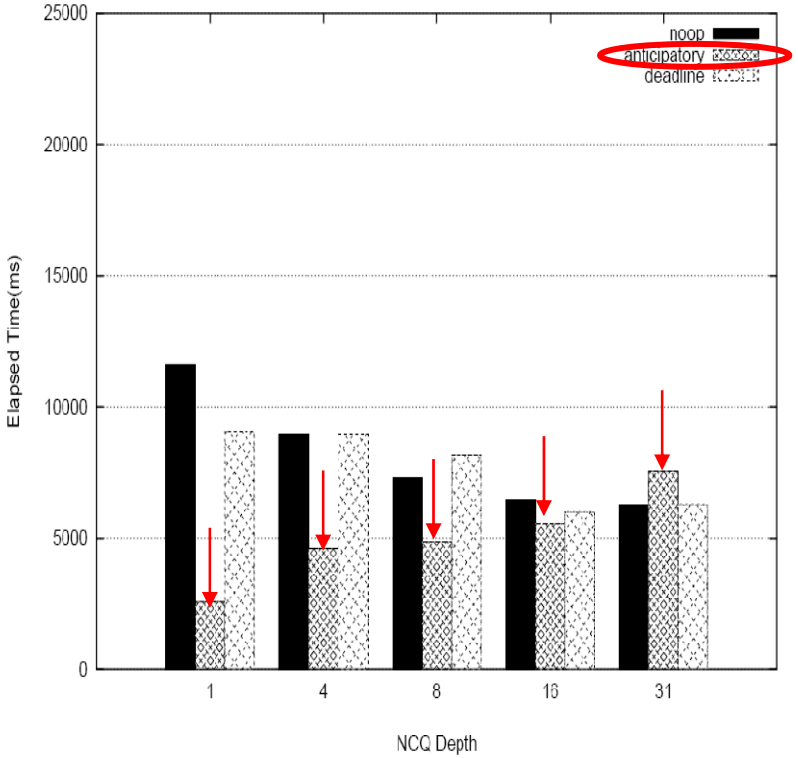
Problem 1. Redundant Scheduling.

Case I (anticipatory, deadline ≈ noop)



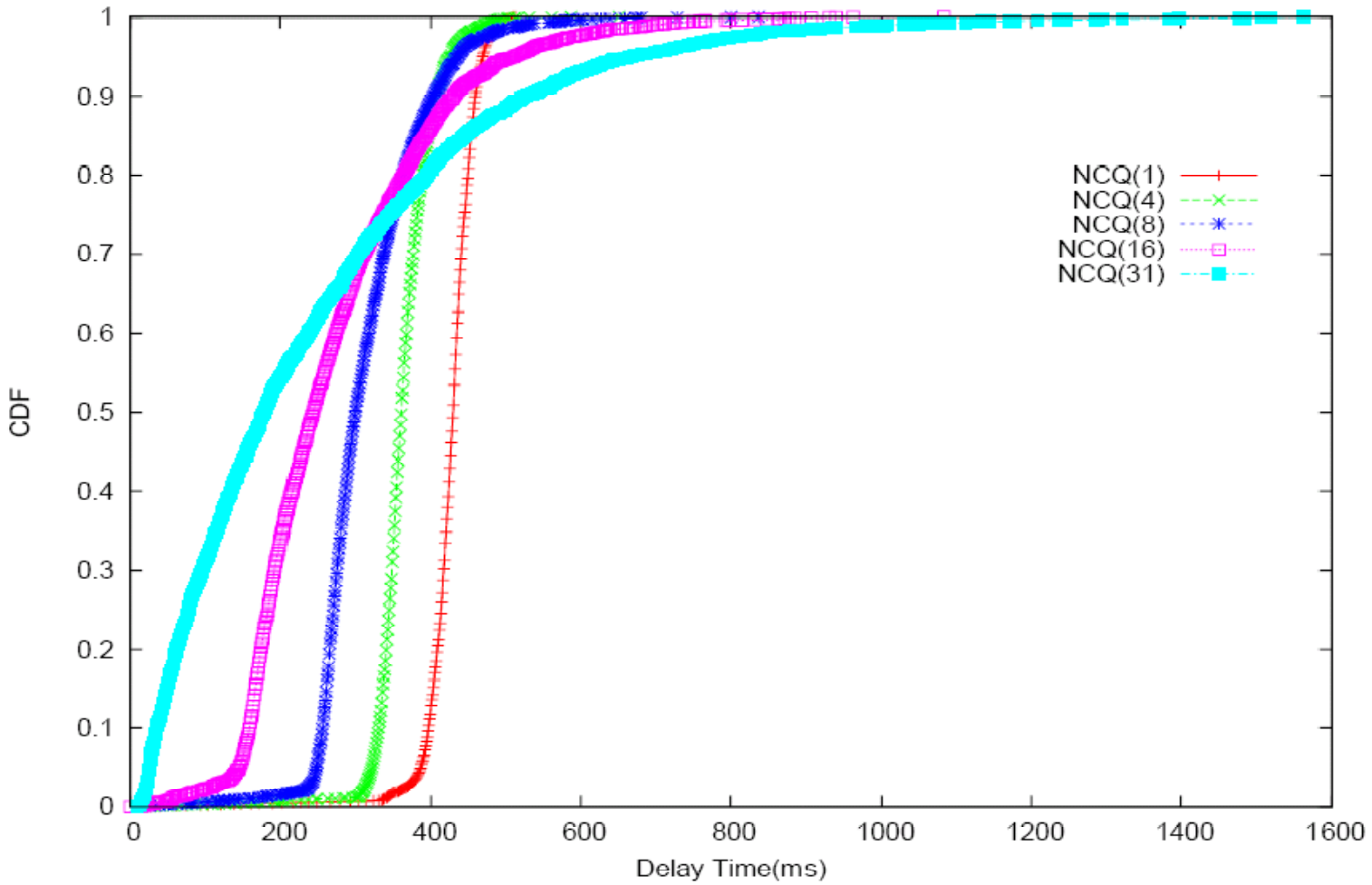
(a) $p = 16, s = 4000, r = 100$(Random Workload)

Case II (anticipatory case)



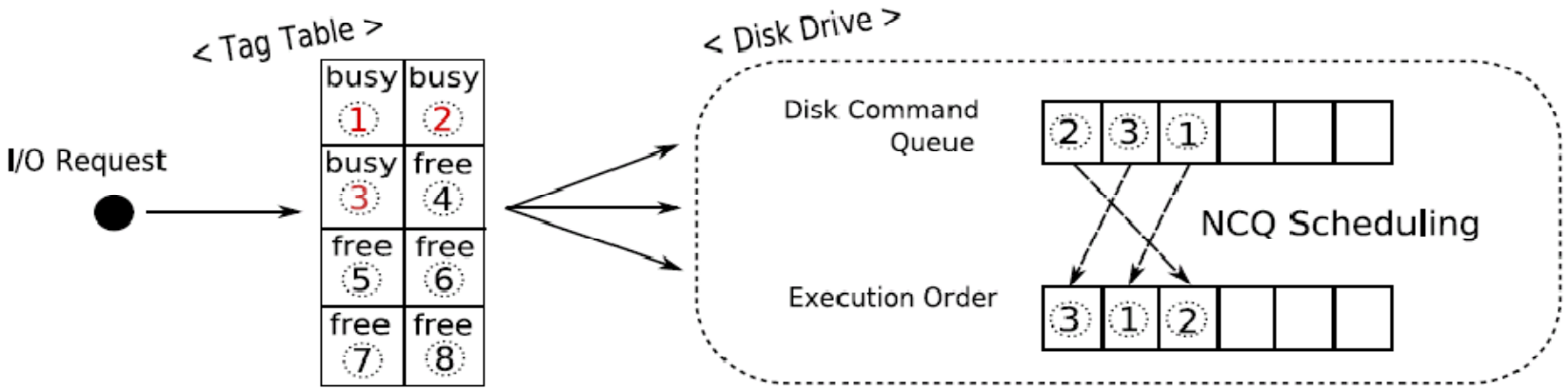
(b) $p = 16, s = 10000000, r = 1$(Sequential Workload)

Problem 2. When requests are reordered,
- some requests are starved !



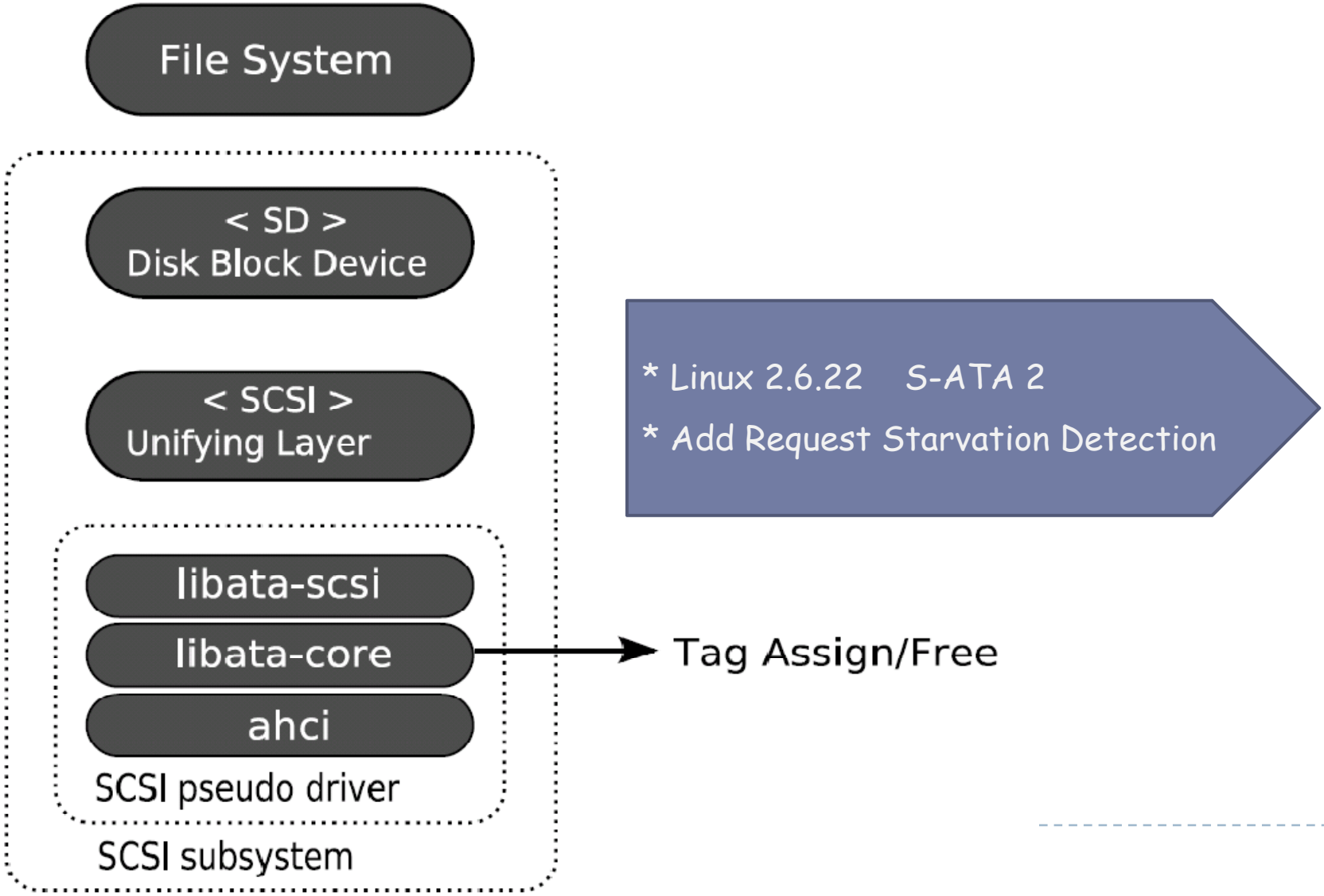
(c) Delay Time CDF(Varying NCQ Depth)

Solution for starvation – Control the tagging



- Count the number of reordering events.
- If exceeds threshold, stop the NCQ mechanism.
- Modification to the device driver.

Request Starvation Detection Architecture



Conclusions

- Revisit the old HDD performance problem.
  - Utilizing the internal disk layout information.
  - It is cumbersome
  - Can the disk manufacturers help ?
- Identified a new problem.
  - NCQ is great, but
  - It's not panacea…
- Some cooperation would be great…..