

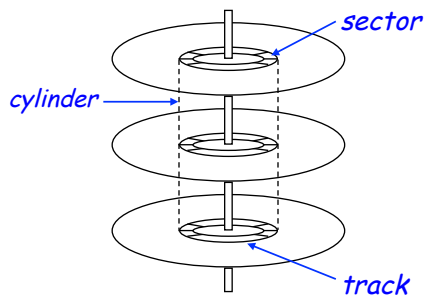
Disk Management

- Disk Structure
 - Disk Scheduling
 - RAID
 - Disk Block Management
 - Modern Secondary Storage Technologies
-

Disk Management

- Disk Structure
 - Disk Scheduling
 - RAID
 - Disk Block Management
 - Modern Secondary Storage Technologies
-

Disk Structure



Disk speed:

- **seek time** : head moves to correct track
- **rotational delay** : wait until sector is under head
- **transfer time** : transfer data between disk and memory

Disk Performance

Seek Time : T_S

n = number of tracks traversed

m = "track traversal time"

s = startup time

$$T_S = m \times n + s$$

Rotational Delay (Latency Time): T_R

r = # revolutions per time unit

$$T_R = \frac{1}{2r}$$

Transfer Time: T_T

b = # bytes to be transferred

N = number of bytes on track

$$T_T = \frac{b}{rN}$$

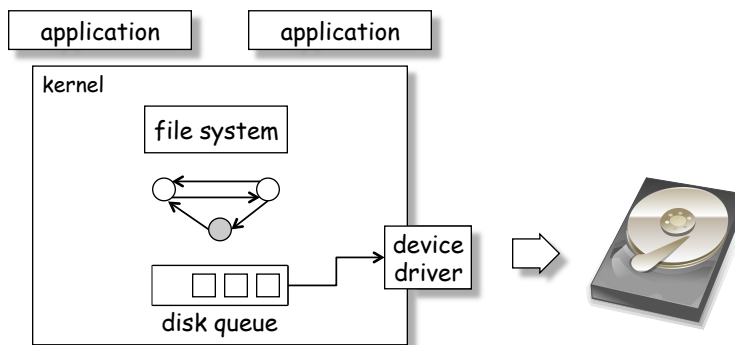
Disk Access Time:

$$T = T_S + T_R + T_T$$

Disk Management

- Disk Structure
- Disk Scheduling
- RAID
- Disk Block Management
- Modern Secondary Storage Technologies

Disk Scheduling

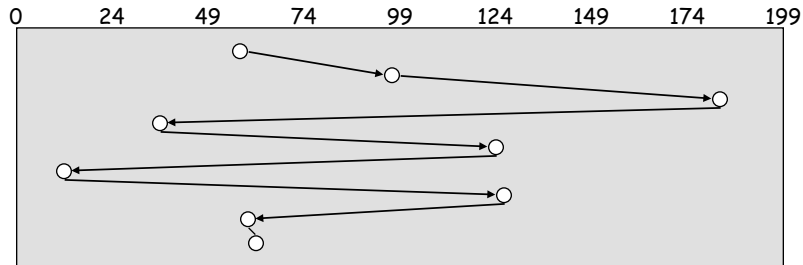


Q: Does it pay off to think about scheduling policy in disk queue?

Evaluation: Compare time for service for *given request sequence*.
Distinguish only *by cylinder*.

FCFS Scheduling

Request Sequence: 98, 183, 37, 122, 14, 124, 65, 67



total head movement: **640 tracks**

FCFS Pros:

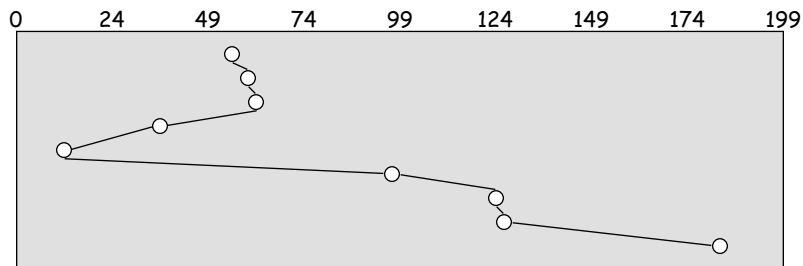
- simple
- fair

FCFS Cons:

- poor average service time

Shortest-Seek-Time-First (SSTF)

Request Sequence: 98, 183, 37, 122, 14, 124, 65, 67



total head movement: **236 tracks**

Always serve "closest" request.

SSTF Pros:

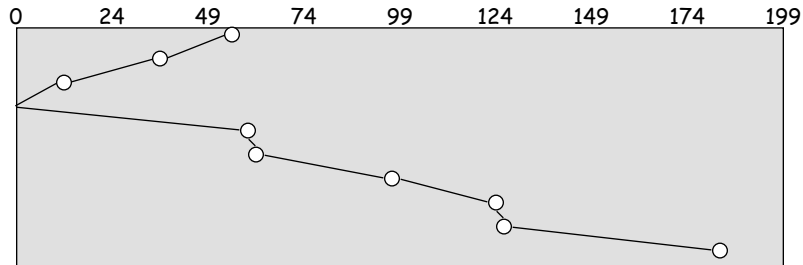
- short service times

SSTF Cons:

- Starvation!

Elevator Algorithm (SCAN)

Request Sequence: 98, 183, 37, 122, 14, 124, 65, 67



total head movement: 236 tracks

Continuously scan disk from one end to the other.

SCAN Pros:

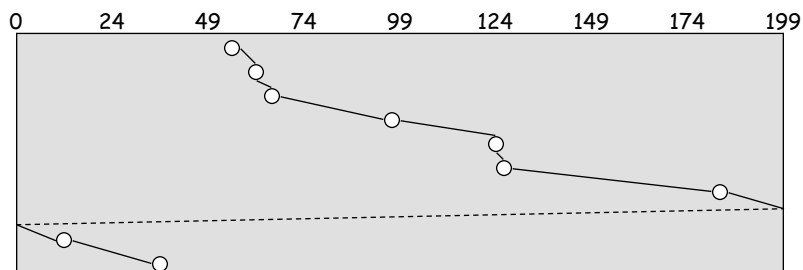
- short service times

SCAN Cons:

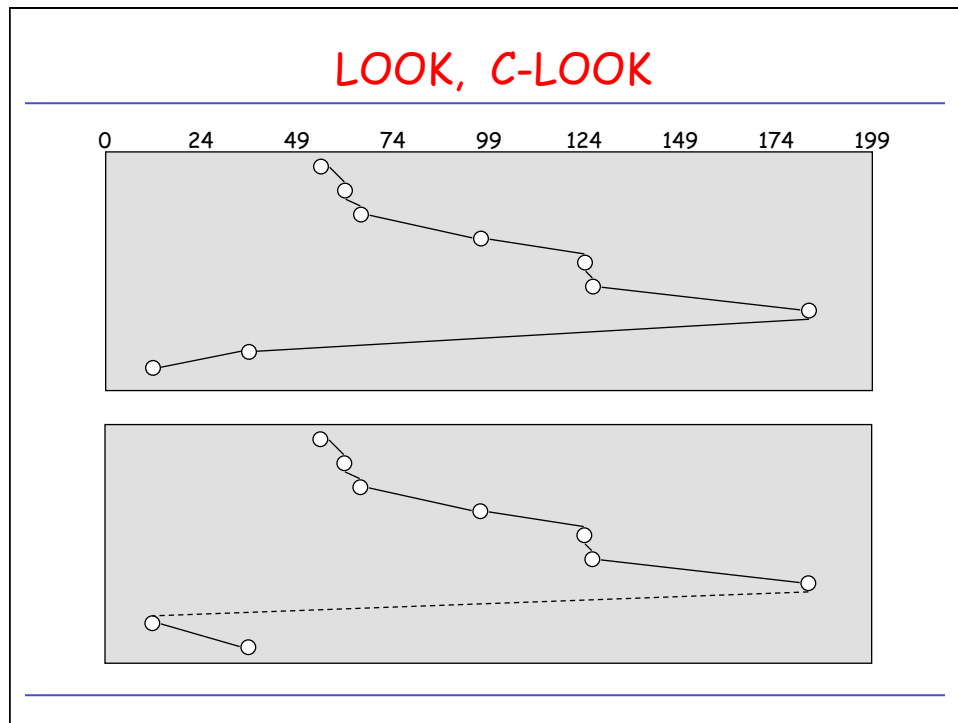
- When scanning, few requests after us, since just past through.
- Problem: When we change direction at end, requests there are *very new*.

Circular SCAN (C-SCAN)

Request Sequence: 98, 183, 37, 122, 14, 124, 65, 67



Reduce variance in service time by always starting at the beginning of the disk.



Disk Management

- Disk Structure
 - Disk Scheduling
 - **RAID**
 - Disk Block Management
 - Modern Secondary Storage Technologies
-

RAID

Observation: Traditional secondary storage devices are **slow!**

Improve their performance by using **multiple devices in parallel:**

RAID

- **Redundant Arrays of Independent Disks**
- **Redundant Arrays of Inexpensive Disks** (Berkeley)

Common characteristics:

- Array of physical disks that are **visible as single device** to OS.
- Data is **distributed** across physical drives of array.
- **Redundant disk capacity** is used for **error detection/correction**.

RAID (cont)

Approach: Replace **single large-capacity** disk with **array of smaller-capacity** disks.

Benefits:

- Improved I/O performance
- Enables incremental upgrade

Problems:

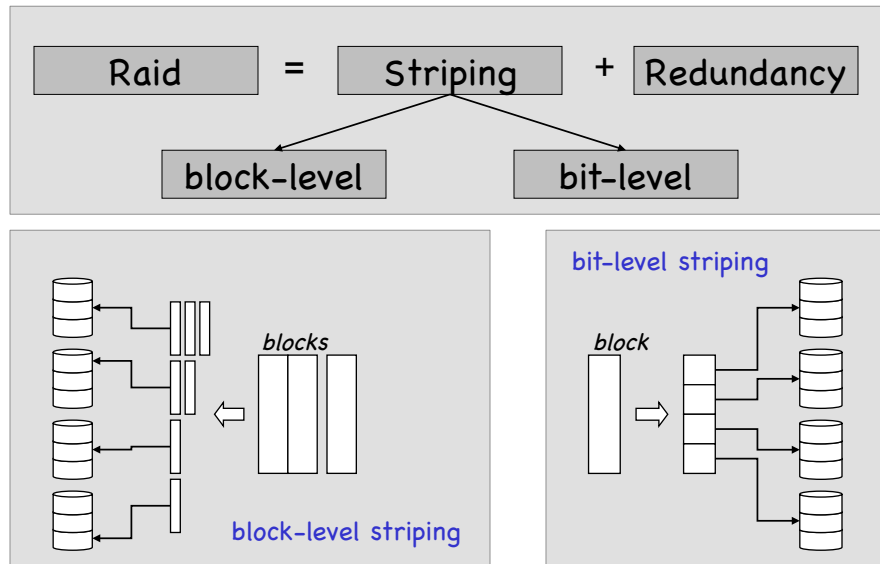
- **Reliability:** more devices **increase** the probability of failure.

$$R(t) = P[t_F > t] \quad \text{e.g.} \quad R(t) = e^{-\lambda t}$$

$$MTTF = E[t_F] = \int_0^{\infty} R(t) dt$$

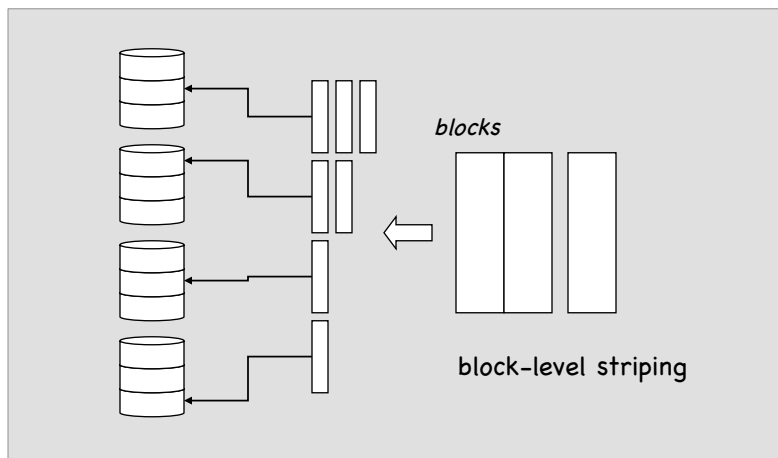
Solution: **redundancy**

Raid (cont 2)



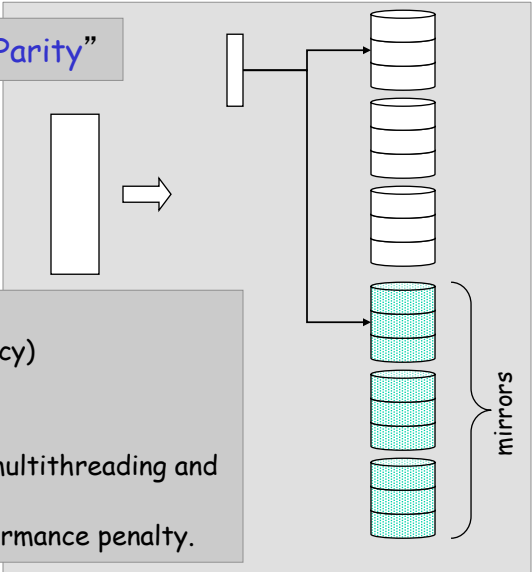
RAID Level 0

“Block-level Striped Set without Parity”



RAID Level 1

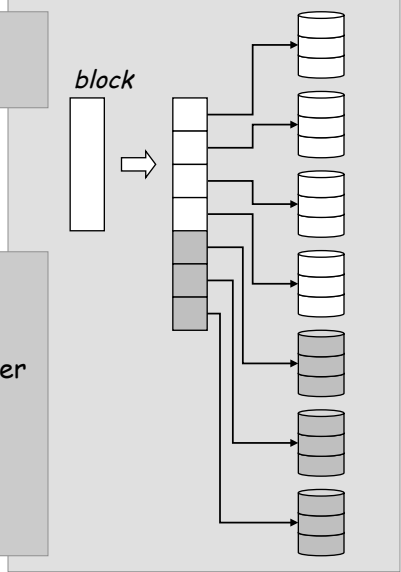
“Mirrored Set without Parity”



- Problem:
 - cost (100% redundancy)
- Performance
 - READs : good (with multithreading and “split reads”)
 - WRITEs: small performance penalty.

RAID Level 2

“Memory-Style Error-Correcting Parity”



- Head and spindles **synchronized**
- Small strips
- **Error correction code** calculated over bits of data disks. (Hamming Code)
- Appropriate for systems with **many failures**.
- **Typically not implemented.**

RAID Level 3

“Bit-Interleaved Parity”

- Heads and spindles synchronized.
- Small strips.
- Simple parity bits instead of ECC.

e.g. $P(S) = S_4 = S_3 \oplus S_2 \oplus S_1 \oplus S_0$

Disk 1 fails:

$$S_1 = S_4 \oplus S_3 \oplus S_2 \oplus S_0$$

RAID Level 4

“Block level Parity”

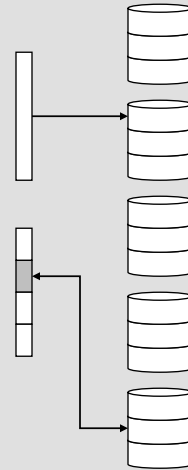
- Same as RAID 3, but with block-level striping.
- No synchronization across disks.
- Large strips.
- Each strip on parity disk contains parity information for all corresponding strips.
- Parity computation upon READ:

$$\begin{aligned}
 X4(i) &= X3(i) \oplus X2(i) \oplus X1(i) \oplus X0(i) \\
 X4'(i) &= X3(i) \oplus X2(i) \oplus X1'(i) \oplus X0(i) \\
 &= X3(i) \oplus X2(i) \oplus X1'(i) \oplus X0(i) \oplus X1(i) \oplus X1(i) \\
 &= X4(i) \oplus X1(i) \oplus X1'(i)
 \end{aligned}$$

RAID Level 5

“Striped Set with Interleaved Parity”

block

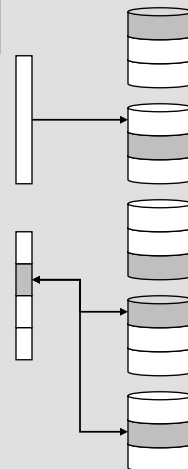


- Same as RAID 4, but parity **spread** across all disks.
- No **synchronization** across disks.
- **Large strips**.

RAID Level 6

“Striped Set with **Dual** Interleaved Parity”

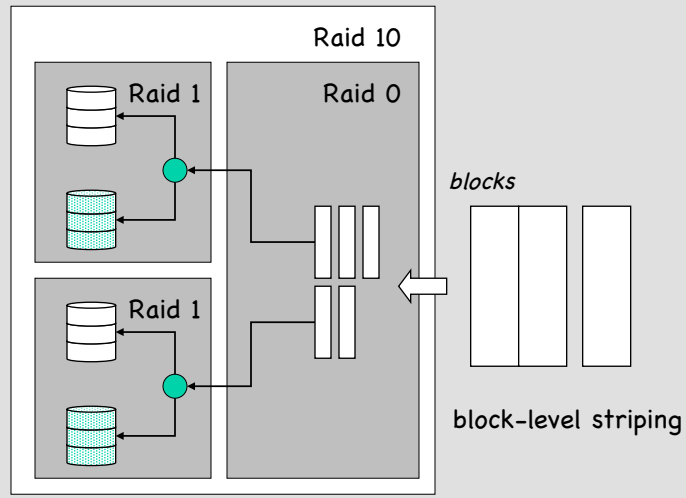
block



- Same as RAID 5, but uses **2 bits** to store “parity”.
- No **synchronization** across disks.
- **Large strips**.
- Uses **ECC** instead of parity.
- Tolerates **two failures**.
- In practice, a second drive can fail during recovery from first drive failure.

Nested Levels: RAID Level 1+0 = RAID 10


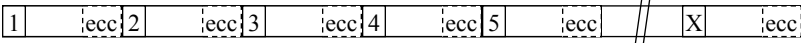

Raid 10 = “Mirrored Set in a Striped Set”



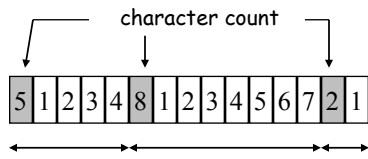
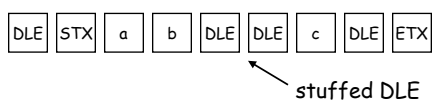
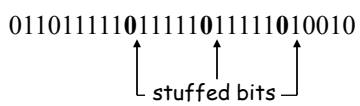
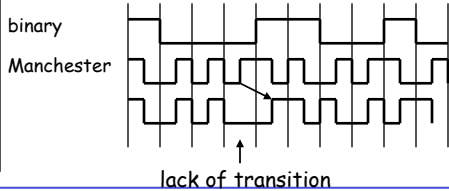
Disk Management

- Disk Structure
- Disk Scheduling
- RAID
- **Disk Block Management**
- Modern Secondary Storage Technologies

Disk Formatting

- Bare disk:**

- Physical formatting:**
 - “cut” into sectors
 - identify sectors
 - add space for error detection/correction
- Logical formatting:**
 - add blank directory, FAT, free space list, ...

Framing

<ul style="list-style-type: none"> Character count 	<ul style="list-style-type: none"> Starting and ending chars, with character stuffing 
<ul style="list-style-type: none"> Starting and ending flags, with bit stuffing <p>framing pattern: 01111110</p> <p>011011111011111011111010010</p> 	<ul style="list-style-type: none"> Physical layer coding violations 

Bad Block Management

- One or more blocks become unreadable/unwritable: **bad blocks**
 - **Off-line management** of bad blocks:
 - Run bad-block detection program and put bad blocks on **bad-block list**. (Either remove them from free list or mark entry in FAT.)
 - May have to run file recovery utility.
 - **On-line management**:
 - Have the device driver map the bad block onto a good block
 - Block **X** goes bad. Whenever OS requests block **X**, the disk transparently accesses a replacement block **Y**.
 - Problem: interferes with scheduling!
-

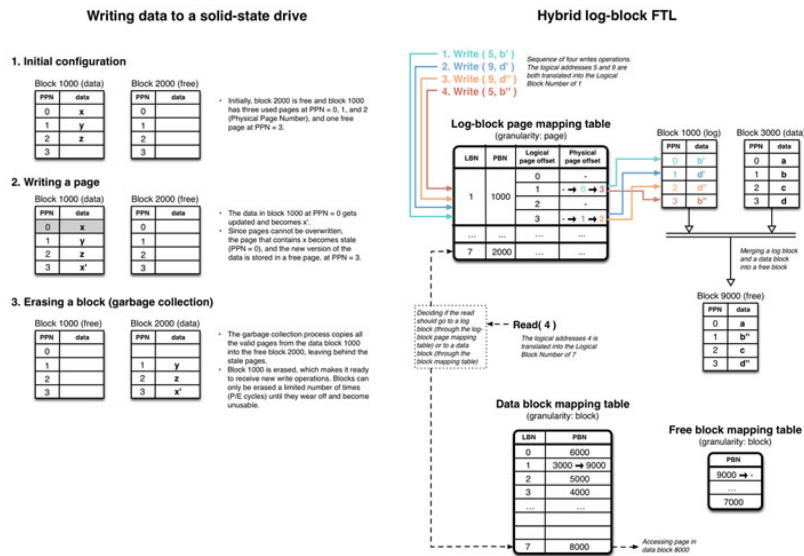
Disk Management

- Disk Structure
 - Disk Scheduling
 - RAID
 - Disk Block Management
 - Modern Secondary Storage Technologies
-

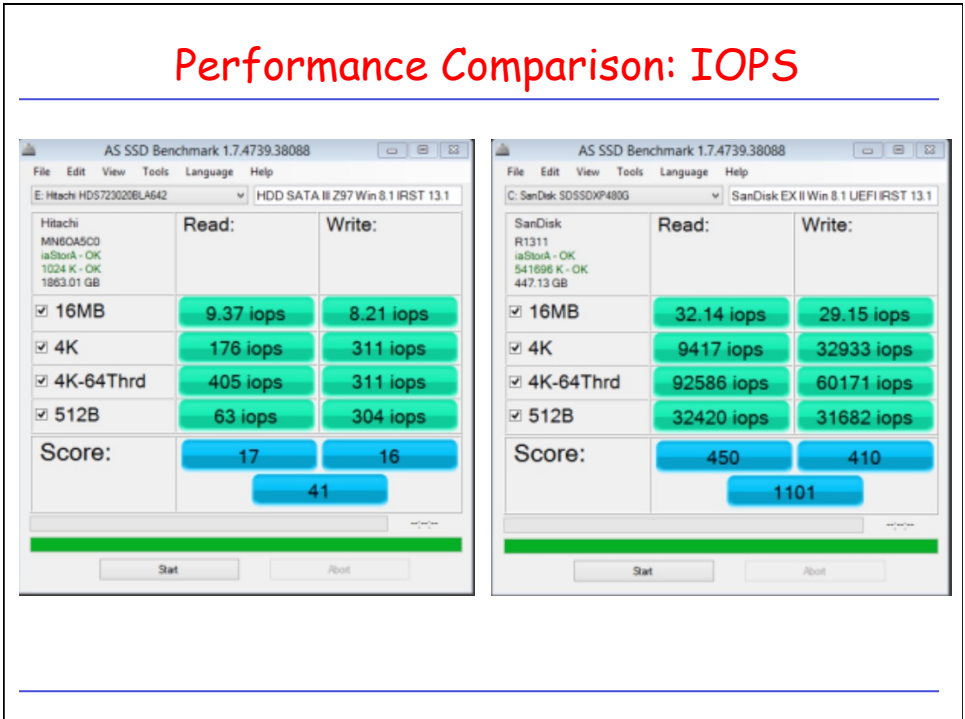
NAND Flash Memory

- Flash chips are arranged in 8kB **blocks**.
- Each **block** is divided into 512B **pages**.
- Flash memory does not support “overwrite” operations.
- Only supports a limited number of “erase” operations.
- This is handled in the Flash Translation Layer (FTL)

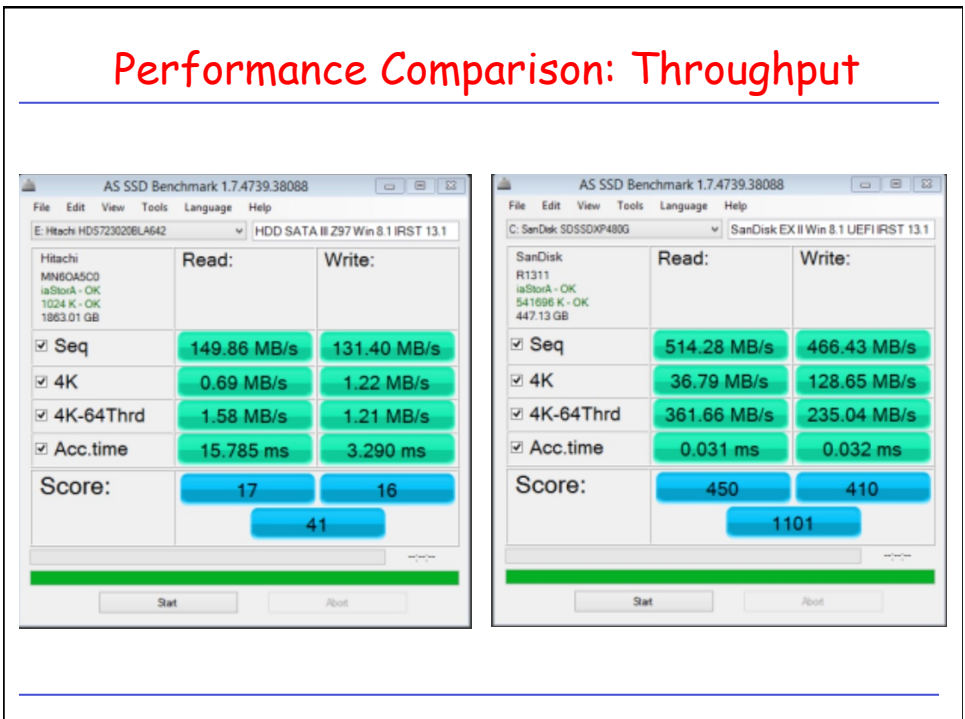
Flash Translation Layer (FTL)



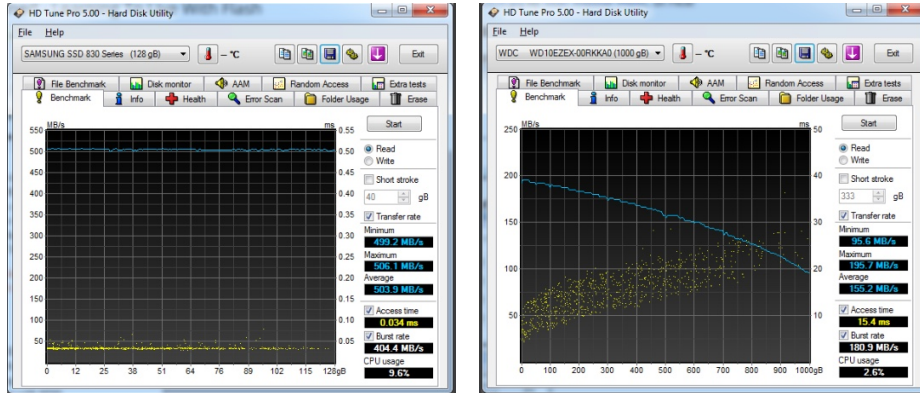
Performance Comparison: IOPS



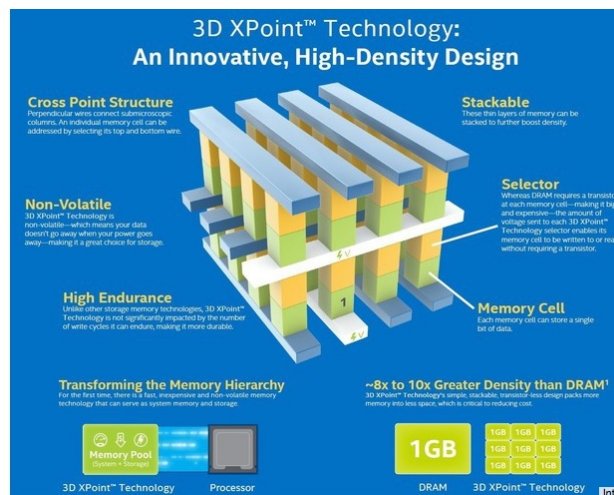
Performance Comparison: Throughput



Performance Comparison: Latency



Intel 3D XPoint (Optane)



Optane (cont)

