Storage: HDD, SSD and RAID

Johan Montelius

KTH

2017
Give me two reasons why we would like to have secondary storage?
Give me two reasons why we would like to have secondary storage?
Gigabyte Z170 Gaming

- 2 PCIe x16/x4
- 4 PCIe x1
- 2 USB 3.1
- 6 USB 3.0
- 4 USB 2.0
- 6 SATA-III
- 2 SATA Express
- 1 M.2
- 1 gigabit Ethernet
- 4 DDR4 SDRAM
Computer architecture

CPU
Computer architecture

CPU

SDRAM

memory bus up to 160 Gb/s
Computer architecture

CPU

SDRAM

GPU

PCIe x16 up to 128 Gb/s

memory bus up to 160 Gb/s
Computer architecture

- CPU
  - PCIe x16 up to 128Gb/s
  - PCIe x4
  - memory bus up to 160 Gb/s

- GPU
  - PCIe x16 up to 128Gb/s

- SDRAM
  - USB up to 10Gb/s
  - SATA up to 6Gb/s
  - SAS up to 12Gb/s

- Control Hub
  - PCIe x4
Computer architecture

- **CPU**
  - PCIe x16 up to 128 Gb/s
  - memory bus up to 160 Gb/s

- **GPU**
  - PCIe x16 up to 128 Gb/s

- **SDRAM**
  - PCIe x4

- **Control Hub**
  - USB up to 10 Gb/s
  - SATA up to 6 Gb/s
  - SAS up to 12 Gb/s
  - BIOS
  - keyboard
  - audio
  - network

- **BIOS**
- **Keyboard**
- **Audio**
- **Network**
Computer architecture

- **CPU**
  - PCIe x16 up to 128 Gb/s
  - Memory bus up to 160 Gb/s

- **SDRAM**

- **GPU**
  - PCIe x16 up to 128 Gb/s

- **Control Hub**
  - PCIe x4
  - USB up to 10 Gb/s
  - SATA up to 6 Gb/s
  - SAS up to 12 Gb/s

- **BIOS**
- **keyboard**
- **audio**
- **network**
Computer architecture

- **CPU**
  - PCIe x16 up to 128 Gb/s
  - memory bus up to 160 Gb/s
  - < 1 ns

- **SDRAM**
  - < 10 ns

- **Control Hub**
  - PCIe x4
  - USB up to 10 Gb/s
  - SATA up to 6 Gb/s
  - SAS up to 12 Gb/s

- **GPU**
  - PCIe x16 up to 128 Gb/s

- **BIOS**
  - keyboard
  - audio
  - network
Computer architecture

- **CPU**
  - PCIe x16 up to 128 Gb/s
  - memory bus up to 160 Gb/s

- **GPU**
  - PCIe x16 up to 128 Gb/s

- **SDRAM**
  - < 10 ns

- **Control Hub**
  - PCIe x4
  - USB up to 10 Gb/s
  - SATA up to 6 Gb/s
  - SAS up to 12 Gb/s
  - 10 µs - 10 ms

- **BIOS**
- keyboard
- audio
- network

- **< 1 ns**
- **< 10 ns**
70 percent of the code of an operating system is code for device drivers.
70 percent of the code of an operating system is code for device drivers.
70 percent of the code of an operating system is code for device drivers.
70 percent of the code of an operating system is code for device drivers.
70 percent of the code of an operating system is code for device drivers.
70 percent of the code of an operating system is code for device drivers.
70 percent of the code of an operating system is code for device drivers.
70 percent of the code of an operating system is code for device drivers.
70 percent of the code of an operating system is code for device drivers.
70 percent of the code of an operating system is code for device drivers.
70 percent of the code of an operating system is code for device drivers.
70 percent of the code of an operating system is code for device drivers.
70 percent of the code of an operating system is code for device drivers.
70 percent of the code of an operating system is code for device drivers.
70 percent of the code of an operating system is code for device drivers.
how to interact with a device

driver

device
how to interact with a device

- A register to read the status of the device.

driver

status

device
A register to read the status of the device.
A register to instruct the device to read or write.
how to interact with a device

- A register to read the status of the device.
- A register to instruct the device to read or write.
- A register that holds the data.
how to interact with a device

- A register to read the status of the device.
- A register to instruct the device to read or write.
- A register that holds the data.
- I/O-bus could be separate from memory bus (or the same).
A register to read the status of the device.
A register to instruct the device to read or write.
A register that holds the data.
I/O-bus could be separate from memory bus (or the same).
The driver will use either special I/O instructions or regular load/store instructions.
char read_from_device() {
    while (STATUS == BUSY) {} // do nothing, just wait
    COMMAND = READ;
    while (STATUS == BUSY) {} // do nothing, just wait
    return DATA;
}
int read_request(int pid, char *buffer) {

    while (STATUS == BUSY) {} 

    COMMAND = READ;

    interrupt -> process = pid;
    interrupt -> buffer = buffer;

    block_process(pid);

    scheduler();
}
int interrupt_handler() {

    int pid = interrupt->pid;
    *(interrupt->buffer) = DATA;

    ready_process(pid);
}

int interrupt_handler() {
    int pid = interrupt->pid;
    *(interrupt->buffer) = DATA;
    ready_process(pid);
}

This is very schematic, more complicated in real life.
The kernel is interrupt driven.
The kernel is interrupt driven.
Direct Memory Access

Allow devices to read and write to buffers in physical memory.

```c
int write_request (int pid, char* string, int size) {
    while (STATUS == BUSY) {}  
    memcpy (string , buffer , size)
    COMMAND = WRITE;
    blocked -> pid = pid;
    block_process (pid);
    scheduler();
}
```

DMA often limited to lower memory addresses.
Direct Memory Access

Allow devices to read and write to buffers in physical memory.

```c
int write_request(int pid, char *string, int size) {
    while (STATUS == BUSY) {}eos
    memcpy(string, buffer, size)
    COMMAND = WRITE;
    blocked->pid = pid;
    block_process(pid);
    scheduler();
}
```
Direct Memory Access

Allow devices to read and write to buffers in physical memory.

```c
int write_request(int pid, char *string, int size) {

    while (STATUS == BUSY) {}  

    memcpy(string, buffer, size)

    COMMAND = WRITE;

    blocked->pid = pid;

    block_process(pid);

    scheduler();
}
```
Each physical device is controlled by a *device driver* that provides the abstraction of a *character device* or *block device*.
Each physical device is controlled by a *device driver* that provides the abstraction of a *character device* or *block device*.

Block devices used as interface to disk drives that provide persistent storage.
Each physical device is controlled by a *device driver* that provides the abstraction of a *character device or block device*.

Block devices used as interface to disk drives that provide persistent storage.

All though all storage devices are presented using the same abstraction, they have very different characteristics.
Each physical device is controlled by a *device driver* that provides the abstraction of a *character device* or *block device*.

Block devices used as interface to disk drives that provide persistent storage.

All though all storage devices are presented using the same abstraction, they have very different characteristics.

*To understand the challenges and options of the operating system, you should know the basics of how storage devices work.*
Anatomy of a HDD

track/cylinder

sectors per track varies

sector size: 4K or 512 bytes

platters: 1 to 6

heads: one side or two sides

Only one head at a time is used (no parallel read).
Anatomy of a HDD

- track/cylinder
- sectors per track varies
- sector size: 4K or 512 bytes
- platters: 1 to 6
- heads: one side or two sides
- Only one head at a time is used (no parallel read).
Anatomy of a HDD

- track/cylinder
- sectors per track varies
- sector size: 4K or 512 bytes
- platters: 1 to 6
- heads: one side or two sides
- Only one head at a time is used (no parallel read).
Anatomy of a HDD

- track/cylinder
- sectors per track varies
- sector size: 4K or 512 bytes
- platters: 1 to 6
- heads: one side or two sides
- Only one head at a time is used (no parallel read).
Anatomy of a HDD

- Track/cylinder
- Sectors per track varies
- Sector size: 4K or 512 bytes
- Platters: 1 to 6
- Heads: one side or two sides

Only one head at a time is used (no parallel read).
Anatomy of a HDD

- track/cylinder

- sectors per track varies
- sector size: 4K or 512 bytes
- platters: 1 to 6
- heads: one side or two sides
- Only one head at a time is used (no parallel read).
Anatomy of a HDD

- track/cylinder
- sectors per track varies
Anatomy of a HDD

- track/cylinder
- sectors per track varies
- sector size: 4K or 512 bytes
Anatomy of a HDD

- track/cylinder
- sectors per track varies
- sector size: 4K or 512 bytes
- platters: 1 to 6
- heads: one side or two sides
Anatomy of a HDD

- track/cylinder
- sectors per track varies
- sector size: 4K or 512 bytes
- platters: 1 to 6
- heads: one side or two sides
Anatomy of a HDD

- track/cylinder
- sectors per track varies
- sector size: 4K or 512 bytes
- platters: 1 to 6
- heads: one side or two sides

*Only one head at a time is used (no parallel read).*
Historically sectors address by cylinder-head-sector (CHS), due to incompatible standars the limitation was:

- cylinder: 1024 (10-bits)
- heads: 16 (4-bits)
- sectors per cylinder: 63 (6-bits)
- number of sectors: 1 Mi

Largest disk assuming 512 Byte sectors: 512 MiByte

Today, sectors are addressed linearly 0..n, Linear Block Addressing (LBA):

- 28-bit or 48-bit address
- up to 256 Ti sectors

Largest disk assuming 4 KiByte sectors: 1 PiByte
Historically sectors address by cylinder-head-sector (CHS), due to incompatible standards the limitation was:
- cylinder: 1024 (10-bits)
Historically sectors address by cylinder-head-sector (CHS), due to incompatible standards the limitation was:

- cylinder: 1024 (10-bits)
- heads: 16 (4-bits)
**Historically sectors address by cylinder-head-sector (CHS), due to incompatible standards the limitation was:**

- cylinder: 1024 (10-bits)
- heads: 16 (4-bits)
- sectors per cylinder: 63 (6-bits)
Historically sectors address by cylinder-head-sector (CHS), due to incompatible standards the limitation was:

- cylinder: 1024 (10-bits)
- heads: 16 (4-bits)
- sectors per cylinder: 63 (6-bits)
- number of sectors: 1 Mi

Today, sectors are addressed linearly 0..n, Linear Block Addressing (LBA):

- 28-bit or 48-bit address
- up to 256 Ti sectors
- largest disk assuming 4 KiByte sectors: 1 PiByte
Historically sectors address by cylinder-head-sector (CHS), due to incompatibility standars the limitation was:

- cylinder: 1024 (10-bits)
- heads: 16 (4-bits)
- sectors per cylinder: 63 (6-bits)
- number of sectors: 1 Mi
- largest disk assuming 512 Byte sectors: 512 MiByte
Historically sectors address by cylinder-head-sector (CHS), due to incompatible standards the limitation was:

- cylinder: 1024 (10-bits)
- heads: 16 (4-bits)
- sectors per cylinder: 63 (6-bits)
- number of sectors: 1 Mi
- largest disk assuming 512 Byte sectors: 512 MiByte

Today, sectors are addressed linearly 0..n, Linear Block Addressing (LBA):

- 28-bit or 48-bit address
- up to 256 Ti sectors
- largest disk assuming 4 KiByte sectors: 1 PiByte

```bash
sudo hdparm /dev/sda
dmesg | grep ata2
```
Historically sectors address by cylinder-head-sector (CHS), due to incompatible standards the limitation was:

- cylinder: 1024 (10-bits)
- heads: 16 (4-bits)
- sectors per cylinder: 63 (6-bits)
- number of sectors: 1 Mi
- largest disk assuming 512 Byte sectors: 512 MiByte

Today, sectors are addresses linearly 0.. n, Linear Block Addressing (LBA):
Historically sectors address by cylinder-head-sector (CHS), due to incompatible standards the limitation was:

- cylinder: 1024 (10-bits)
- heads: 16 (4-bits)
- sectors per cylinder: 63 (6-bits)
- number of sectors: 1 Mi
- largest disk assuming 512 Byte sectors: 512 MiByte

Today, sectors are addresses linearly 0.. n, Linear Block Addressing (LBA):

- 28-bit or 48-bit address
Historically sectors address by cylinder-head-sector (CHS), due to incompatible standards the limitation was:

- cylinder: 1024 (10-bits)
- heads: 16 (4-bits)
- sectors per cylinder: 63 (6-bits)
- number of sectors: 1 Mi
- largest disk assuming 512 Byte sectors: 512 MiByte

Today, sectors are addresses linearly 0..n, Linear Block Addressing (LBA):

- 28-bit or 48-bit address
- up to 256 Ti sectors
Historically sectors address by cylinder-head-sector (CHS), due to incompatible standards the limitation was:
- cylinder: 1024 (10-bits)
- heads: 16 (4-bits)
- sectors per cylinder: 63 (6-bits)
- number of sectors: 1 Mi
- largest disk assuming 512 Byte sectors: 512 MiByte

Today, sectors are addresses linearly 0.. n, Linear Block Addressing (LBA):
- 28-bit or 48-bit address
- up to 256 Ti sectors
- largest disk assuming 4 KiByte sectors: 1 PiByte

> sudo hdparm /dev/sda
> dmesg | grep ata2
HDD - Hard Disk Drive

Seagate Desktop

- Total capacity: 2 TiByte
- Form factor: 3.5"
- Rotational speed: 7,200 rpm
- Connection: SATA III
- Cache size: 64 MiByte
- Read throughput: 156 MByte/s

ST2000DM001, aprx price, October 2016, 900:-
HDD - Hard Disk Drive

Seagate Desktop

- total capacity: 2 TiByte
Seagate Desktop

- total capacity: 2 TiByte
- form factor: 3.5"
HDD - Hard Disk Drive

Seagate Desktop

- total capacity: 2 TiByte
- form factor: 3.5"
- rotational speed: 7.200 rpm
Seagate Desktop

- total capacity: 2 TiByte
- form factor: 3.5"
- rotational speed: 7.200 rpm
- connection: SATA III
Seagate Desktop

- total capacity: 2 TiByte
- form factor: 3.5"
- rotational speed: 7,200 rpm
- connection: SATA III
- cache size: 64 MiByte
Seagate Desktop

- total capacity: 2 TiByte
- form factor: 3.5"
- rotational speed: 7,200 rpm
- connection: SATA III
- cache size: 64 MiByte
- read throughput: 156 MByte/s
HDD - Hard Disk Drive

Seagate Desktop

- total capacity: 2 TiByte
- form factor: 3.5"
- rotational speed: 7.200 rpm
- connection: SATA III
- cache size: 64 MiByte
- read throughput: 156 MByte/s

ST2000DM001, aprx price, October 2016, 900:-
HDD - Hard Disk Drive

Seagate Cheetah 15K

- Total capacity: 600 GiByte
- Form factor: 3.5"
- Rotational speed: 15,000 rpm
- Connection: SAS-3
- Cache size: 16 MiByte
- Read throughput: 204 MByte/s

ST3300657SS, aprx price, October 2016, 2.200:-
HDD - Hard Disk Drive

Seagate Cheetah 15K

- total capacity: 600 GiByte
HDD - Hard Disk Drive

Seagate Cheetah 15K

- total capacity: 600 GiByte
- form factor: 3.5"
HDD - Hard Disk Drive

Seagate Cheetah 15K

- total capacity: 600 GiByte
- form factor: 3.5"
- rotational speed: 15.000 rpm
Seagate Cheetah 15K

- total capacity: 600 GiByte
- form factor: 3.5"
- rotational speed: 15.000 rpm
- connection: SAS-3
HDD - Hard Disk Drive

Seagate Cheetah 15K

- total capacity: 600 GiByte
- form factor: 3.5"
- rotational speed: 15,000 rpm
- connection: SAS-3
- cache size: 16 MiByte
HDD - Hard Disk Drive

Seagate Cheetah 15K

- total capacity: 600 GiByte
- form factor: 3.5"
- rotational speed: 15.000 rpm
- connection: SAS-3
- cache size: 16 MiByte
- read throughput: 204 MByte/s
HDD - Hard Disk Drive

Seagate Cheetah 15K

- total capacity: 600 GiByte
- form factor: 3.5"
- rotational speed: 15.000 rpm
- connection: SAS-3
- cache size: 16 MiByte
- read throughput: 204 MByte/s

ST3300657SS, aprx price, October 2016, 2.200:-
access time

seek time: time to move arm to the right cylinder
rotation time: time to rotate the disk
read time: read one or more sectors
access time

- seek time: time to move arm to the right cylinder
access time

- **seek time:** time to move arm to the right cylinder
- **rotation time:** time to rotate the disk
access time

- seek time: time to move arm to the right cylinder
- rotation time: time to rotate the disk
- read time: read one or more sectors
HDD - shoot out

- Seagate Desktop
  - Rotation speed: 7200 rpm
  - Average seek time: < 10 ms
  - Average time to read a sector: 14 ms
  - Capacity: 2 TiByte
  - Approx. price: 900:-
  - Cost capacity: 0.44 SEK/GiByte

- Seagate Cheeta 15K
  - Rotation speed: 15000 rpm
  - Average seek time: < 4 ms
  - Average time to read a sector: 6 ms
  - Capacity: 600 GiByte
  - Approx. price: 2.200:-
  - Cost capacity: 3.70 SEK/GiByte
HDD - shoot out

- Seagate Desktop
  - rotation speed: 7200 rpm

- Seagate Cheeta 15K
  - rotation speed: 15000 rpm
HDD - shoot out

- Seagate Desktop
  - rotation speed: 7200 rpm
  - average seek time: < 10 ms

- Seagate Cheeta 15K
  - rotation speed: 15000 rpm
  - average seek time: < 4 ms
HDD - shoot out

- Seagate Desktop
  - rotation speed: 7200 rpm
  - average seek time: < 10 ms
  - average time to read a sector: 14ms

- Seagate Cheeta 15K
  - rotation speed: 15000 rpm
  - average seek time: < 4 ms
- **Seagate Desktop**
  - Rotation speed: 7200 rpm
  - Average seek time: < 10 ms
  - Average time to read a sector: 14 ms

- **Seagate Cheeta 15K**
  - Rotation speed: 15000 rpm
  - Average seek time: < 4 ms
  - Average time to read a sector: 6 ms
Seagate Desktop
- rotation speed: 7200 rpm
- average seek time: < 10 ms
- average time to read a sector: 14 ms
- capacity: 2 TiByte

Seagate Cheeta 15K
- rotation speed: 15000 rpm
- average seek time: < 4 ms
- average time to read a sector: 6 ms
- capacity: 600 GiByte
HDD - shoot out

- **Seagate Desktop**
  - rotation speed: 7200 rpm
  - average seek time: < 10 ms
  - average time to read a sector: 14ms
  - capacity: 2 TiByte
  - aprx. price: 900:-

- **Seagate Cheeta 15K**
  - rotation speed: 15000 rpm
  - average seek time: < 4 ms
  - average time to read a sector: 6ms
  - capacity: 600 GiByte
  - aprx. price: 2.200:-
HDD - shoot out

- Seagate Desktop
  - rotation speed: 7200 rpm
  - average seek time: < 10 ms
  - average time to read a sector: 14ms
  - capacity: 2 TiByte
  - aprx. price: 900:-

- Seagate Cheeta 15K
  - rotation speed: 15000 rpm
  - average seek time: < 4 ms
  - average time to read a sector: 6ms
  - capacity: 600 GiByte
  - aprx. price: 2.200:-
  - cost capacity: 3.70 SEK/GiByte
- **Seagate Desktop**
  - rotation speed: 7200 rpm
  - average seek time: < 10 ms
  - average time to read a sector: 14ms
  - capacity: 2 TiByte
  - aprx. price: 900:-
  - cost capacity: 0.44 SEK/GiByte

- **Seagate Cheeta 15K**
  - rotation speed: 15000 rpm
  - average seek time: < 4 ms
  - average time to read a sector: 6ms
  - capacity: 600 GiByte
  - aprx. price: 2.200:-
  - cost capacity: 3.70 SEK/GiByte
If a sector is 512 bytes, it takes 10ms to find and read a sector, and we want to read 512 MiBytes then .....?
If a sector is 512 bytes, it takes 10ms to find and read a sector, and we want to read 512 MiBytes then .....?

- Time to find first sector is less relevant.
If a sector is 512 bytes, it takes 10ms to find and read a sector, and we want to read 512 MiBytes then .....?

- Time to find first sector is less relevant.
- If sectors that belong to the same file are close to each other we minimize movement of arm.

Rotational speed should be high.

The density i.e. how many sectors in each track is important.

The communication with the drive should be fast.

Typical read and write performance is between 150 MiByte/s to 250 MiByte/s.
If a sector is 512 bytes, it takes 10ms to find and read a sector, and we want to read 512 MiBytes then .....?

- Time to find first sector is less relevant.
- If sectors that belong to the same file are close to each other we minimize movement of arm.
- Rotational speed should be high.
If a sector is 512 bytes, it takes 10ms to find and read a sector, and we want to read 512 MiBytes then .....?

- Time to find first sector is less relevant.
- If sectors that belong to the same file are close to each other we minimize movement of arm.
- Rotational speed should be high.
- The density i.e. how many sectors in each track is important.
If a sector is 512 bytes, it takes 10ms to find and read a sector, and we want to read 512 MiBytes then .....?

- Time to find first sector is less relevant.
- If sectors that belong to the same file are close to each other we minimize movement of arm.
- Rotational speed should be high.
- The density i.e. how many sectors in each track is important.
- The communication with the drive should be fast.
read/write performance

If a sector is 512 bytes, it takes 10ms to find and read a sector, and we want to read 512 MiBytes then .....?

- Time to find first sector is less relevant.
- If sectors that belong to the same file are close to each other we minimize movement of arm.
- Rotational speed should be high.
- The density i.e. how many sectors in each track is important.
- The communication with the drive should be fast.
- Typical read and write performance is between 150 MiByte/s to 250 MiByte/s.
Historically, the Operating System was in complete control:

- it knew the layout cylinder-head-sector (CHS),
Historically, the Operating System was in complete control:
- it knew the layout cylinder-head-sector (CHS),
- could order data in segments that were close to each other and,
Historically, the Operating System was in complete control:

- it knew the layout cylinder-head-sector (CHS),
- could order data in segments that were close to each other and,
- would schedule disk operations to minimize arm movement.
Historically, the Operating System was in complete control:

- it knew the layout cylinder-head-sector (CHS),
- could order data in segments that were close to each other and,
- would schedule disk operations to minimize arm movement.

Today, the drive can often make a better decision:
Historically, the Operating System was in complete control:

- it knew the layout cylinder-head-sector (CHS),
- could order data in segments that were close to each other and,
- would schedule disk operations to minimize arm movement.

Today, the drive can often make a better decision:

- it knows, but might not reveal, the layout.
Historically, the Operating System was in complete control:

- it knew the layout cylinder-head-sector (CHS),
- could order data in segments that were close to each other and,
- would schedule disk operations to minimize arm movement.

Today, the drive can often make a better decision:

- it knows, but might not reveal, the layout.
- The operating system can help in grouping operations together, allowing the drive to decide what order they should be done (Native Command Queuing).
who’s in control

Historically, the Operating System was in complete control:

- it knew the layout cylinder-head-sector (CHS),
- could order data in segments that were close to each other and,
- would schedule disk operations to minimize arm movement.

Today, the drive can often make a better decision:

- it knows, but might not reveal, the layout.
- The operating system can help in grouping operations together, allowing the drive to decide what order they should be done (Native Command Queuing).

There is a reason why MS-DOS is called MS-DOS.
Samsung 850 EVO

- total capacity: 250 GiByte
Samsung 850 EVO

- total capacity: 250 GiByte
- form factor: 2.5"
SSD - Solid State Drive

Samsung 850 EVO

- total capacity: 250 GiByte
- form factor: 2.5"
- connection: SATA III
SSD - Solid State Drive

Samsung 850 EVO

- total capacity: 250 GiByte
- form factor: 2.5"
- connection: SATA III
- cache size: 64 MiByte
Samsung 850 EVO

- total capacity: 250 GiByte
- form factor: 2.5"
- connection: SATA III
- cache size: 64 MiByte
- random access: 30 $\mu$s
SSD - Solid State Drive

Samsung 850 EVO

- total capacity: 250 GiByte
- form factor: 2.5"
- connection: SATA III
- cache size: 64 MiByte
- random access: 30 $\mu$s
- read throughput: 540 MiByte/s
SSD - Solid State Drive

Samsung 850 EVO

- total capacity: 250 GiByte
- form factor: 2.5"
- connection: SATA III
- cache size: 64 MiByte
- random access: 30 $\mu$s
- read throughput: 540 MiByte/s

MZ-75E250B/EU, aprx price, October 2016, 1000:-
SanDisk Ultra SDXC

- form factor: SDXC
SanDisk Ultra SDXC

- form factor: SDXC
- capacity: 64 GiByte
SanDisk Ultra SDXC

- form factor: SDXC
- capacity: 64 GiByte
- read performance: 80 MiByte/s
SanDisk Ultra SDXC

- form factor: SDXC
- capacity: 64 GiByte
- read performance: 80 MiByte/s

aprx price, October 2016, 300:-
NAND - flash storage

memory bank
NAND - flash storage

memory bank
NAND - flash storage

memory bank
NAND - flash storage

memory bank

erase blocks ~256 KiByte
NAND - flash storage

memory bank

erase blocks ~256 KiByte

pages ~4KiByte
NAND - flash storage

You have constant time access to any page.

- pages ~4KiByte
- memory bank
- erase blocks ~256 KiByte
NAND - flash storage

You have constant time access to any page.
You can only write to (or program) an erased page.
You have constant time access to any page.
You can only write to (or program) an erased page.
You can only erase a block.
<table>
<thead>
<tr>
<th>Drive</th>
<th>Capacity</th>
<th>Price</th>
<th>SEK/GiByte</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDD Desktop</td>
<td>2 TiByte</td>
<td>900:-</td>
<td>44 öre</td>
</tr>
<tr>
<td>HDD Performance</td>
<td>600 GiByte</td>
<td>2.200:-</td>
<td>3.70:-</td>
</tr>
<tr>
<td>SSD Desktop</td>
<td>250 GiByte</td>
<td>1000:-</td>
<td>4:-</td>
</tr>
</tbody>
</table>
Bus limitations

- SATA-III - 6 Gb/s, most internal HDD and SSD today

An SSD has a throughput of 500 MiByte/s which is a... b/s?
Bus limitations

- SATA-III - 6 Gb/s, most internal HDD and SSD today
- SAS-3 - 12 Gb/s, enterprise RAID HDD

An SSD has a read throughput of 500 MiByte/s which is a... b/s?
Bus limitations

- SATA-III - 6 Gb/s, most internal HDD and SSD today
- SAS-3 - 12 Gb/s, enterprise RAID HDD
- USB3.1 - 10 Gb/s, everything
Bus limitations

- SATA-III - 6 Gb/s, most internal HDD and SSD today
- SAS-3 - 12 Gb/s, enterprise RAID HDD
- USB3.1 - 10 Gb/s, everything
- PCI Express 3.0 x16 - 128 Gb/s, what is it used for?
Bus limitations

- SATA-III - 6 Gb/s, most internal HDD and SSD today
- SAS-3 - 12 Gb/s, enterprise RAID HDD
- USB3.1 - 10 Gb/s, everything
- PCI Express 3.0 x16 - 128 Gb/s, what is it used for?

An SSD has a read throughput of 500 MiByte/s which is a .... b/s?
SSD on the PCIe bus

Intel SSD 750 Series

- total capacity: 400 GiByte
Intel SSD 750 Series

- total capacity: 400 GiByte
- connection: PCI Express 3.0 x4
Intel SSD 750 Series

- total capacity: 400 GiByte
- connection: PCI Express 3.0 x4
- read performance: 2200 MByte/s
Intel SSD 750 Series

- total capacity: 400 GiByte
- connection: PCI Express 3.0 x4
- read performance: 2200 MByte/s
- write performance: 900 MByte/s
SSD on the PCIe bus

Intel SSD 750 Series

- total capacity: 400 GiByte
- connection: PCI Express 3.0 x4
- read performance: 2200 MByte/s
- write performance: 900 MByte/s

aprx price, October 2016, 4.599:-
Intel 600P Series 512GB

- total capacity: 512 GiByte
The M.2 connector

Intel 600P Series 512GB

- total capacity: 512 GiByte
- form factor: M-keyed
The M.2 connector

Intel 600P Series 512GB

- total capacity: 512 GiByte
- form factor: M-keyed
- connection: M.2 - PCI Express 3.0 x4
Intel 600P Series 512GB

- total capacity: 512 GiByte
- form factor: M-keyed
- connection: M.2 - PCI Express 3.0 x4
- read performance: 1775 MByte/s
The M.2 connector

Intel 600P Series 512GB

- total capacity: 512 GiByte
- form factor: M-keyed
- connection: M.2 - PCI Express 3.0 x4
- read performance: 1775 MByte/s
- write performance: 560 MByte/s
The M.2 connector

Intel 600P Series 512GB

- total capacity: 512 GiByte
- form factor: M-keyed
- connection: M.2 - PCI Express 3.0 x4
- read performance: 1775 MByte/s
- write performance: 560 MByte/s

aprx price, October 2016, 1799:-
SSD on the memory bus

HP NVDIMM 8GB
HP NVDIMM 8GB

- regular DRAM backed up by Flash
HP NVDIMM 8GB

- regular DRAM backed up by Flash
- total capacity: 8 GiByte
SSD on the memory bus

HP NVDIMM 8GB

- regular DRAM backed up by Flash
- total capacity: 8 GiByte
- form factor: DDR4 SDIM

28 / 33
HP NVDIMM 8GB

- regular DRAM backed up by Flash
- total capacity: 8 GiByte
- form factor: DDR4 SDIM
- bus speed: 2133 MHz

**aprx price, October 2016, ??**
Next year?

Intel Optane - 3D XPoint NVDIMM
Next year?

Intel Optane - 3D XPoint NVDIMM

in the pipe line
Next year?

Intel Optane - 3D XPoint NVDIMM

- in the pipe line
- total capacity: 512 GiByte
Next year?

Intel Optane - 3D XPoint NVDIMM

- in the pipeline
- total capacity: 512 GiByte
Increase capacity, performance and/or reliability

Redundant Array of Independent Disks
RAID

Multiple disks that can provide:
- capacity: looks like a 20 TiByte disk but is actually 10 2TiByte disks
- performance: spread a file across ten drives, read and write in parallel
- reliability: write the same file to several disks, if one crashes - not a problem
Increase capacity, performance and/or reliability

Redundant Array of Independent Disks
RAID

- Multiple disks that can provide:
  - Capacity: looks like a 20 TiByte disk but is actually 10 2TiByte disks
  - Performance: spread a file across ten drives, read and write in parallel
  - Reliability: write the same file to several disks, if one crashes - not a problem
Increase capacity, performance and/or reliability

Redundant Array of Independent Disks
RAID

- Multiple disks that can provide:
- capacity: looks like a 20 TiByte disk but is actually 10 2TiByte disks
Increase capacity, performance and/or reliability

Redundant Array of Independent Disks
RAID

- Multiple disks that can provide:
  - capacity: looks like a 20 TiByte disk but is actually 10 2TiByte disks
  - performance: spread a file across ten drives, read and write in parallel
Increase capacity, performance and/or reliability

Redundant Array of Independent Disks
RAID

- Multiple disks that can provide:
  - capacity: looks like a 20 TiByte disk but is actually 10 2TiByte disks
  - performance: spread a file across ten drives, read and write in parallel
  - reliability: write the same file to several disks, if one crashes - not a problem
Alternatives:
Alternatives:

- The cabinet that holds the disks present itself as one drive.
Alternatives:

- The cabinet that holds the disks present itself as one drive.
- A device driver in the kernel knows that we have several disks but the kernel presents it as one disk to the application layer.
the abstraction layer

Alternatives:

- The cabinet that holds the disks present itself as one drive.
- A device driver in the kernel knows that we have several disks but the kernel presents it as one disk to the application layer.
- The application layer knows that we have several disks but provides an API to other applications that looks a single drive.
RAID levels

- RAID 0: *stripe* files across several drives.
RAID levels

- RAID 0: *stripe* files across several drives.
- RAID 1: keep a complete *mirror copy* of each file.
RAID levels

- RAID 0: *stripe* files across several drives.
- RAID 1: keep a complete *mirror copy* of each file.
- RAID 2-6: spread a file plus parity information across several drives.
Summary

application layer, simple to understand

hardware - a complete mess
Summary

application layer, simple to understand

I/O and memory buses, protocols suchs as SATA, SCSI, USB etc

hardware - a complete mess
Summary

application layer, simple to understand

now it’s a bit structured
I/O and memory buses, protocols suchs as SATA, SCSI, USB etc

hardware - a complete mess
Summary

application layer, simple to understand

device drivers that know what they are doing

now it’s a bit structured

I/O and memory buses, protocols suchs as SATA, SCSI, USB etc

hardware - a complete mess
Summary

application layer, simple to understand

- all devices have a generic API
- device drivers that know what they are doing

now it’s a bit structured

I/O and memory buses, protocols suchs as SATA, SCSI, USB etc

hardware - a complete mess
Summary

application layer, simple to understand

system calls: open, read, write, lseek ...

all devices have a generic API
device drivers that know what they are doing

now it’s a bit structured
I/O and memory buses, protocols suchs as SATA, SCSI, USB etc

hardware - a complete mess