CS532 Operating System Foundations

Karen Karavanic

Winter 2020

Portland State
UNIVERSITY

General Questions about I/O

Why care about I/O?

How can we integrate I/O into a computer?

Any general mechanisms?

How can we make I/O efficient?

Basic Assumption

I/O is slow

That is, operations are much slower than CPU

Multiple orders of magnitude slower

But it can be capable of great throughput

Basic Assumption

I/O is slow

That is, operations are much slower than CPU

Multiple orders of magnitude slower

But it can be capable of great throughput

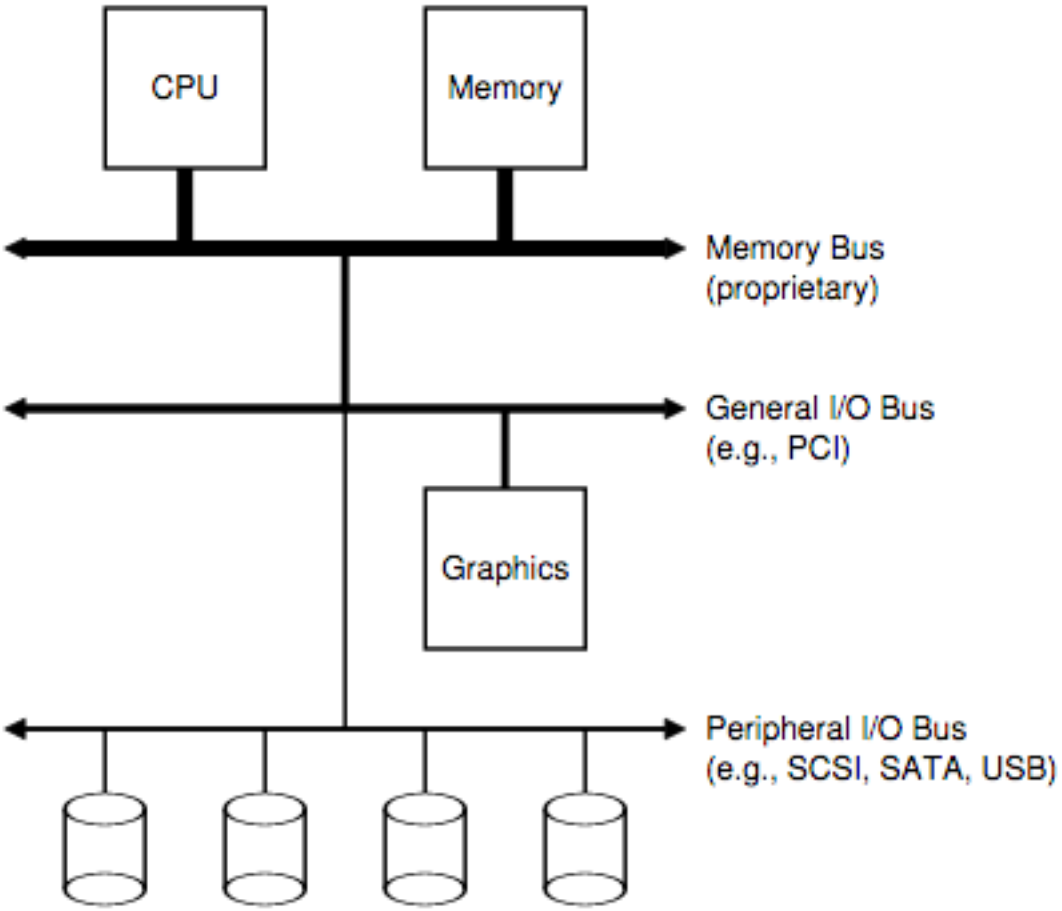


Figure 36.1: **Prototypical System Architecture**

Physics and Cost

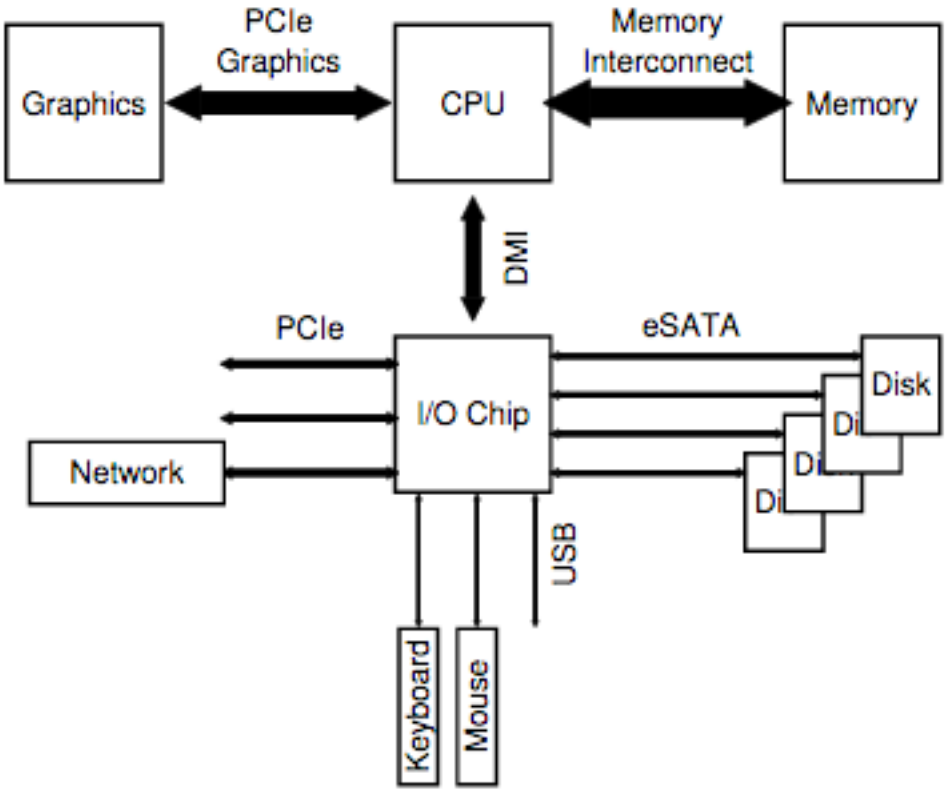
Faster bus == shorter bus

Shorter bus == fewer devices plugged in

Faster bus == expensive

So we build a hierarchical structure to allow for varied tradeoffs

Intel Kaby Lake (Z270 Chipset, 2017)



Simplified I/O Device

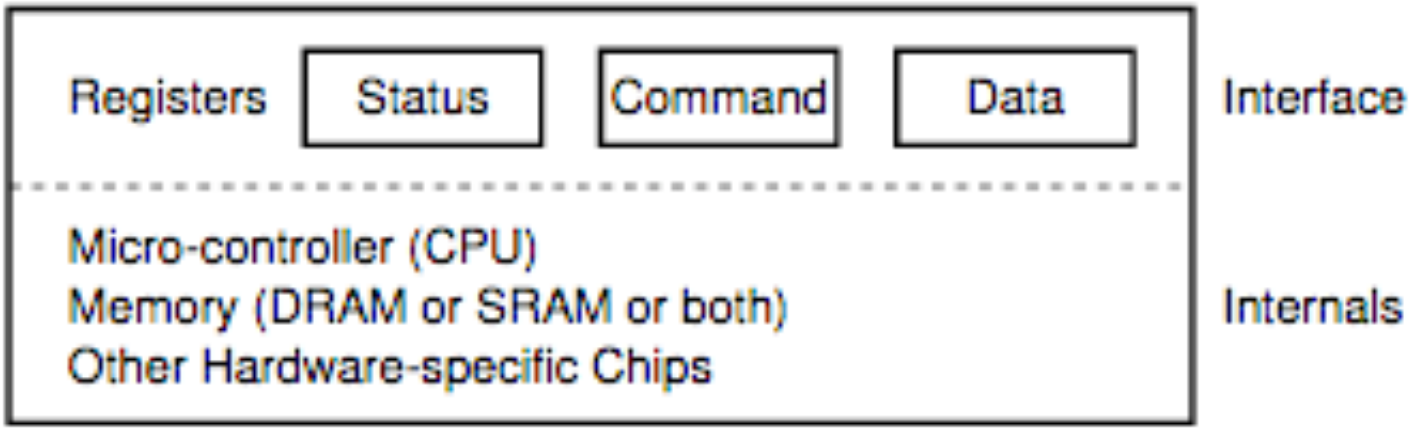


Figure 36.3: **A Canonical Device**

Programmed I/O

```
While (STATUS == BUSY)
    ; // wait until device is not busy
Write data to DATA register
Write command to COMMAND register
    (Doing so starts the device and executes the command)
While (STATUS == BUSY)
    ; // wait until device is done with your request
```

Interrupts

`while()` polling can be slow

Solution: interrupts

Interrupt: asynch hardware-level signal

Overlap computation with I/O operations

Allows OS to schedule another runnable process while waiting for I/O

PIO sometimes better than Interrupts

Issue: sometimes I/O devices are very fast

Issue: sometimes too many interrupts can be detrimental

Solution: poll for a little while, then block

(example: network packet bursts)

Coalescing of Interrupts

Some I/O devices have buffers and are therefore able to queue up results of multiple I/O operations.

In this case the device can raise just one *coalesced* interrupt

CPU needs to know to handle multiple I/O operations per interrupt.

After the Interrupt…

After the interrupt, the CPU still needs to move data from device to memory.

Data movement can be slow and waste valuable CPU cycles

Is there a better way?

DMA: Direct Memory Access

Allow the I/O device to directly write/read data to/from memory.

All data transferred at I/O speeds

Then raise an interrupt when finished

Meanwhile, CPU is free to do other work

DMA might be orchestrated by DMA Controller, I/O chip or even a device itself.

How to Communicate w Devices?

Two Common Approaches

I/O instructions

- e.g., Intel's `in` and `out` instructions
- privileged

Memory-mapped I/O

- Specific memory locations are used as communication channels for devices

Gazillions of Devices

- Disks, Graphics Cards, Displays, Keyboards, NICs, SSDs, etc.
- Many manufacturers
- Many versions
- Many configurations

How do we deal with the complexity?

Device Drivers

Goals

- Allow for variety of devices
- Keep the OS sane and consistent

Device Driver

- Bit of software that encapsulates management of specific device
- Must follow OS-defined interface
- Must be installed

File System Stack (Linux)

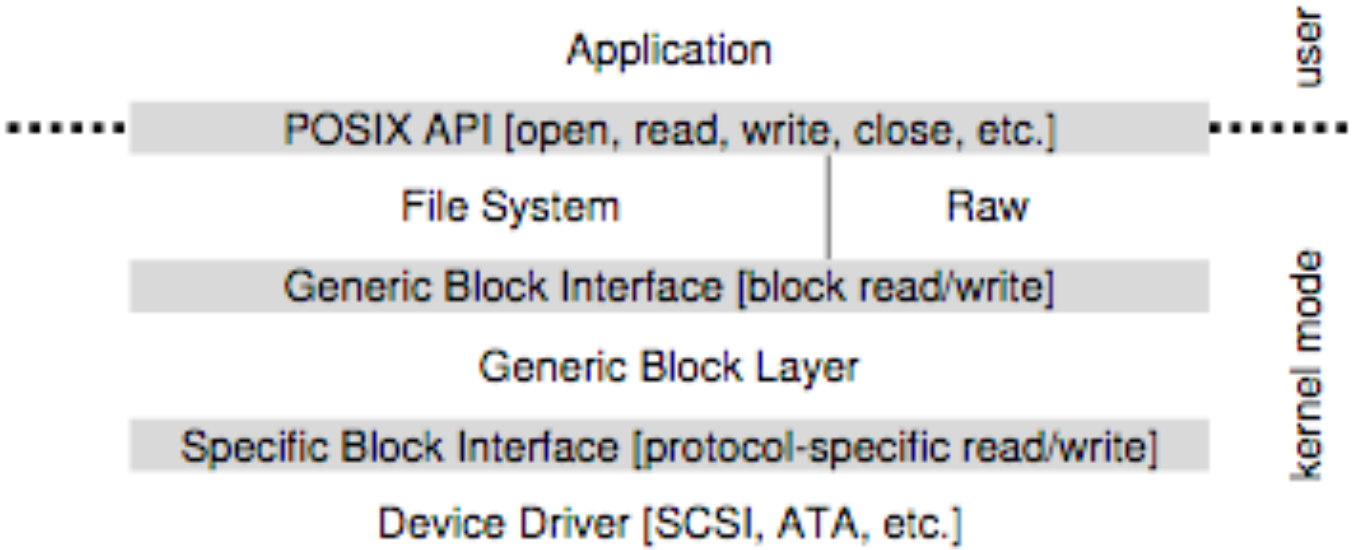


Figure 36.4: **The File System Stack**

Device Drivers

- Take up >70% of kernel code
  - Even when very few devices are actually installed
- Often responsible for many system crashes
- Drive kernel developers insane
- Sometimes hide too many device-specific details

Hard Disk Drives (HDDs)

- Still the most common type of storage device
- Quirky physical device
- Inspired wonderful CS algorithms
- De riguer for OS courses
- The rise of Solid State Storage Devices (SSDs)

Incredible SSDs

1 TB USB 3.0 Premium Flash Drive 200MB/sec R,100MB/sec W Thumb Drive Memory Stick Pen Drive Keychain Design (1

★★☆☆☆ ˅17

$19.99

Get it **Fri, Mar 8 - Sat, Mar 9**
FREE Shipping on orders over $25
shipped by Amazon

SSD characteristics

Random reads and writes perform better than HDDs

Device	Random Reads (MB/s)	Random Writes (MB/s)	Sequential Reads (MB/s)	Sequential Writes (MB/s)
Samsung 840 Pro SSD	103	287	421	384
Seagate 600 SSD	84	252	424	374
Intel SSD 335 SSD	39	222	344	354
Seagate Savvio 15K.3 HDD	2	2	223	223

Figure 44.4: **SSDs And Hard Drives: Performance Comparison**

Flash-based SSDs

No mechanical/moving parts, just transistors

Quiet, small

Unlike DRAM, it retains data when off

NAND-based Flash

Invented: Fujio Masuoka at Toshiba (1984)

Has some interesting properties…

Flash-based SSDs properties

To write a given chunk (flash page), you first must erase a bigger chunk (!)

Writing a flash memory location repeatedly causes that location to wear out (!)

SSDs: SLC, MLC, TLC, QLC

Each cell is a transistor

SLC – one bit per transistor, fastest, least dense, most expensive

MLC – two bits per transistor

TLC – three bits per transistor (2017)

QLC – most capacity density, cheaper (2018)

SSDs: leakage

All SSDs store data in electrical charges

Slowly leak over time if left w/o power

Drives lose data after 1-2 years (depending on ambient temperature)

Not a good solution yet for data archival

SSDs: banks and planes

Plane – collection of banks

Bank – contains many blocks of 128+K

Block – contains pages of 4K each

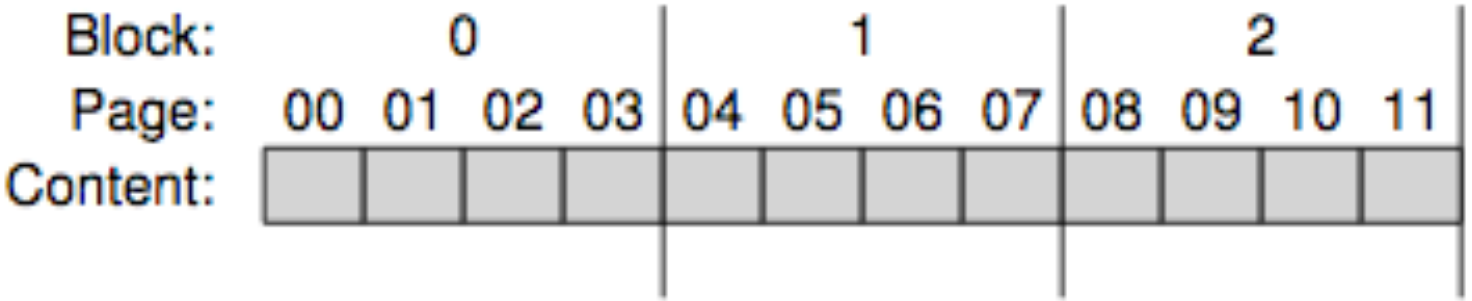


Figure 44.1: **A Simple Flash Chip: Pages Within Blocks**

SSDs: basic operations

Read a page: Takes 10s of usecs, same for any page at any time

Erase a block: sets every bit in the block to 1. so first make a copy! Takes msecs

Program a page: change some bits to 0. Takes 100s of usecs. Must erase block first!

SSDs: example update

Page 0	Page 1	Page 2	Page 3
00011000	11001110	00000001	00111111
VALID	VALID	VALID	VALID

Page 0	Page 1	Page 2	Page 3
11111111	11111111	11111111	11111111
ERASED	ERASED	ERASED	ERASED

Page 0	Page 1	Page 2	Page 3
00000011	11111111	11111111	11111111
VALID	ERASED	ERASED	ERASED

Whoops!

We lost the data for pages 1, 2 and 3!

Solution: when writing any page must first copy the other pages to memory.

Write cost: 30usecs to copy, 1000 usecs to flash, 100usecs to program

So updates ~100x slower than reads

SSDs: latency comparison

Device	Read (μs)	Program (μs)	Erase (μs)
SLC	25	200-300	1500-2000
MLC	50	600-900	~3000
TLC	~75	~900-1350	~4500

Figure 44.2: **Raw Flash Performance Characteristics**

Flashing is not just expensive…

It also wears out the transistors over time

When designing systems with SSDs the performance and reliability of writing is usually the central focus.

Researchers have developed ways to mitigate some of these issues

SSD: wear out – why ?

When a flash block is erased and programmed it slowly accrues a little bit of extra charge.

Over time, as that charge builds up, it becomes increasingly difficult to differentiate between a 0 and a 1.

At some point the block becomes unusable.

How soon? 2010: 10K cycles for MLC, 100K cycles for SLC. Today, reportedly better.

SSD: wait, there's more!

Disturbance: reads and especially re-programs can sometimes cause bits in neighboring pages to become flipped.

Similar to "row hammer" attacks in memory

Building an SSD from Flash Chips

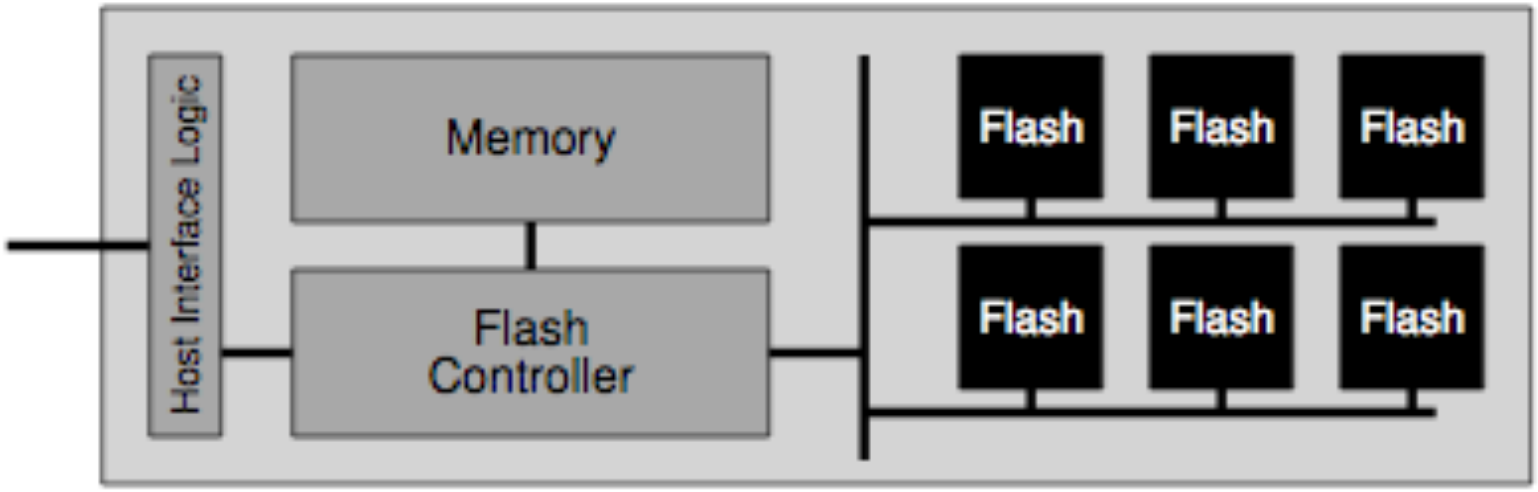


Figure 44.3: **A Flash-based SSD: Logical Diagram**

Building an SSD from Flash Chips

Host Interface Logic: allows reuse of existing disk-based OS software and bus interconnects

On board memory gives scratch space for reprogramming

Flash Controller implements various techniques to optimize the operations

Flash Controller Techniques

Parallel banks: used for redundancy

Spread: even/level the wear across all banks

Minimize disturbance: program pages within an erased block in order, from low page to high page.

Log Structuring of Blocks

As writes arrive, just keep them in order that they arrived, regardless of their logical location.

e.g. 100:0, 101:1, 2000:2, 2001:3

SSD pages 00,01,02,03

Must keep a map of logical:physical locations

How is the map persisted

Two techniques:

1. Write redundant map recovery information into the pages (cheaper)
2. Use asynchronous logging and checkpointing (faster)

Two problems with Log Structuring

Needs a map

Overwrites of data lead to holes and garbage collection

Some devices implement periodic de-fragmentation to coalesce data and do wear leveling

New algorithms developed each year.

SSDs: Summary

Not as quirky as HDDs

But still quirky

Much research/development has improved them, but much remains to be done.

The technology is changing rapidly!