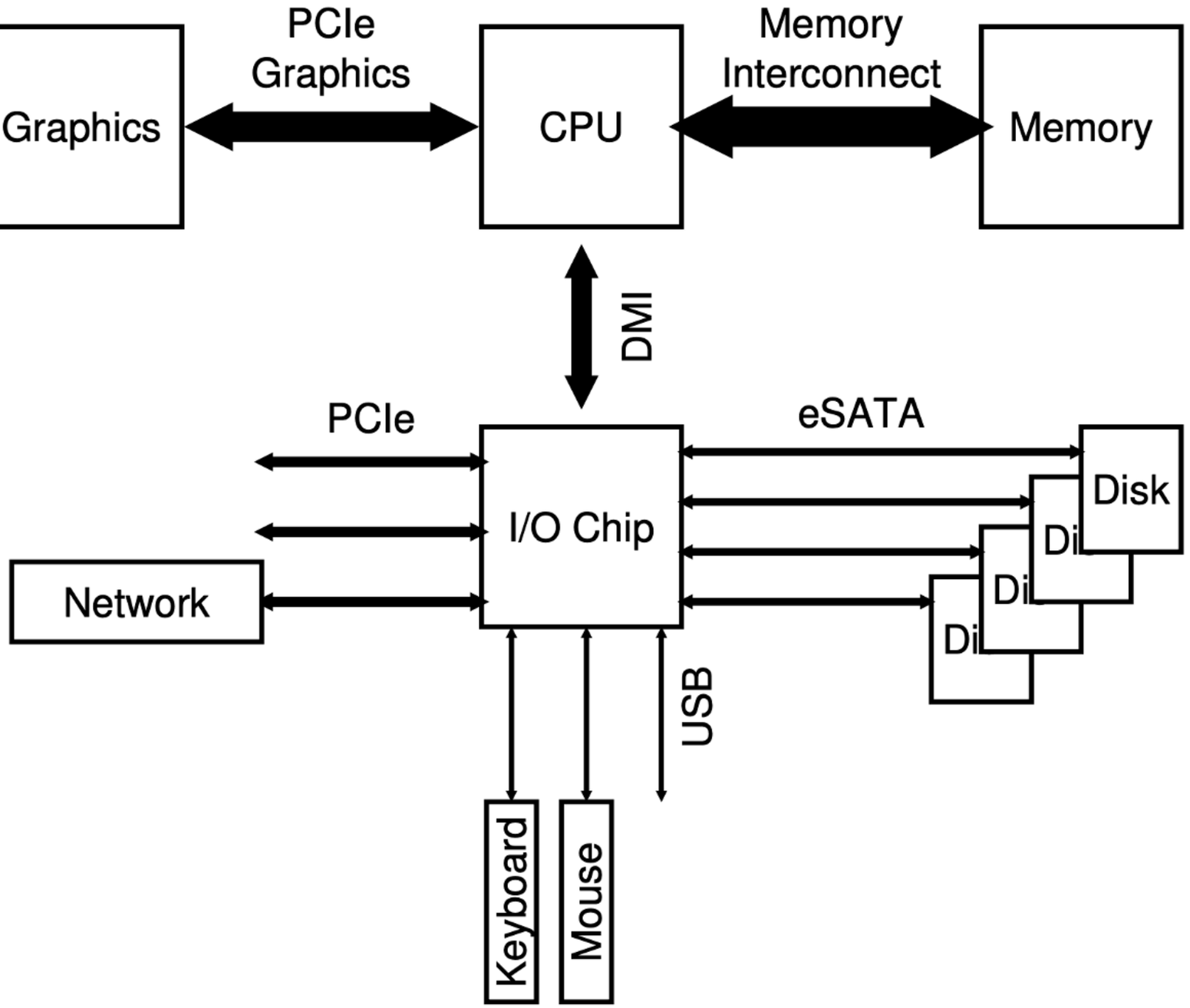
设备管理

Section 5: Part II

2024

典型的系统架构

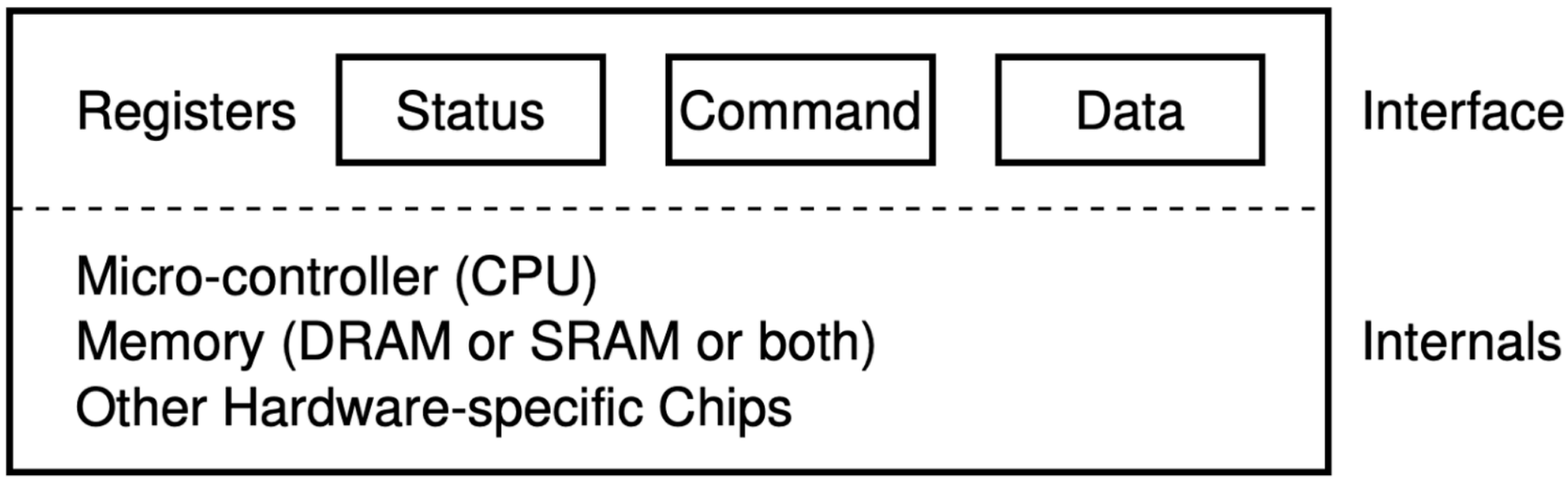
通过通用总线和多个外围总线将不同 I/O 设备连接到计算机



设备的接口

设备提供的接口 (interface) 通常由若干寄存器组成

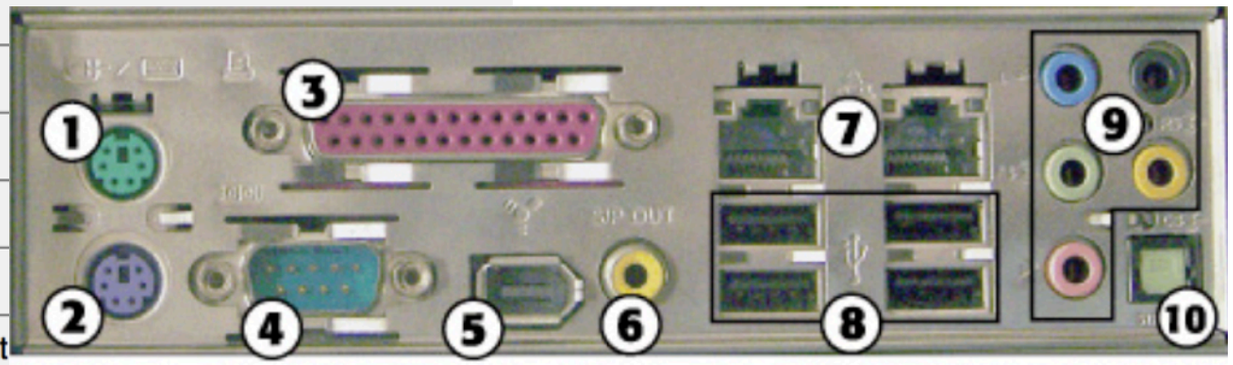
- 状态寄存器 (status register): 设备当前的状态
- 命令寄存器 (command register): 指示设备执行特定的任务
- 数据寄存器 (data register): 将数据传递给设备或从设备获取数据



与设备的交互

- 通过端口映射的方式实现 CPU 与设备的通信 (Port-Mapped I/O)
  - 为设备的每个寄存器分配一个 I/O 端口号
  - 通过专门的 I/O 指令进行读写 (e.g., `in` and `out` on x86)

Port range	Summary
0x0000-0x001F	The first legacy DMA controller, often used for transfers to floppies.
0x0020-0x0021	The first Programmable Interrupt Controller
0x0022-0x0023	Access to the Model-Specific Registers of Cyrix processors.
0x0040-0x0047	The PIT (Programmable Interval Timer)
0x0060-0x0064	The "8042" PS/2 Controller or its predecessors, dealing with keyboards and mice.
0x0070-0x0071	The CMOS and RTC registers
0x0080-0x008F	The DMA (Page registers)
0x0092	The location of the fast A20 gate register
0x00A0-0x00A1	The second PIC
0x00C0-0x00DF	The second DMA controller, often used for soundblasters
0x00E9	Home of the Port E9 Hack. Used on some emulators to directly send text
0x0170-0x0177	The secondary ATA harddisk controller.
0x01F0-0x01F7	The primary ATA harddisk controller.
0x0278-0x027A	Parallel port
0x02F8-0x02FF	Second serial port



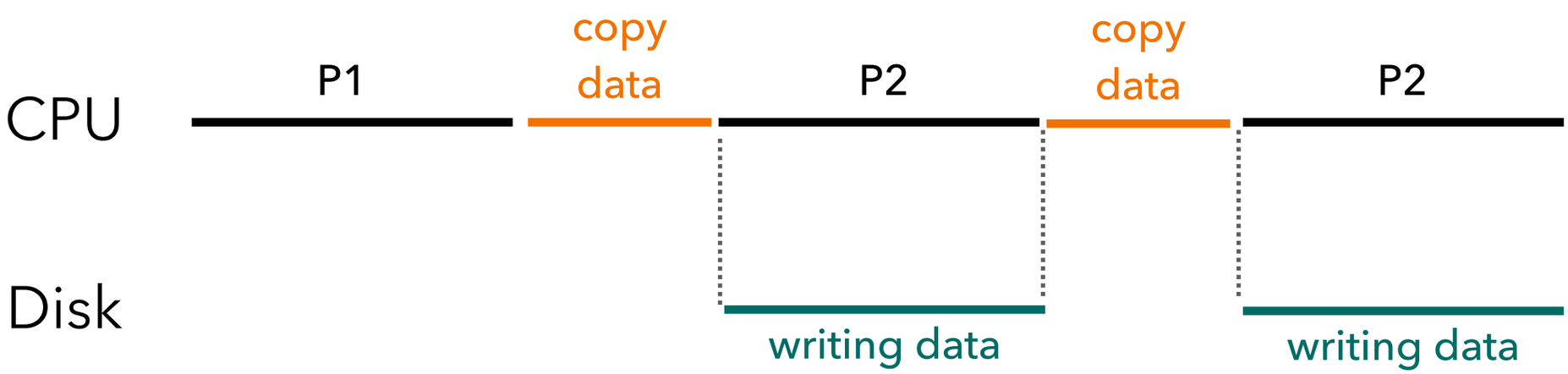
1. PS/2 mouse port
2. PS/2 keyboard port
3. Parallel port
4. Serial port
5. IEEE 1394a port
6. SPDIF coaxial digital audio port
7. Ethernet ports
8. USB ports
9. 1/8-inch mini-jack audio ports
10. SPDIF optical digital audio port

与设备的交互

- 通过端口映射的方式实现 CPU 与设备的通信 (Port-Mapped I/O)
  - 为设备的每个寄存器分配一个 I/O 端口号
  - 通过专门的 I/O 指令进行读写 (e.g., `in` and `out` on x86)
- 通过内存映射的方式实现 CPU 与设备的通信 (Memory-Mapped I/O)
  - 为设备的每个寄存器分配一个内存地址
  - 直接使用内存 load 和 store 操作进行访问
    - 需要注意缓存的问题
  - 硬件将对该地址的内存操作转向设备寄存器

与设备的交互

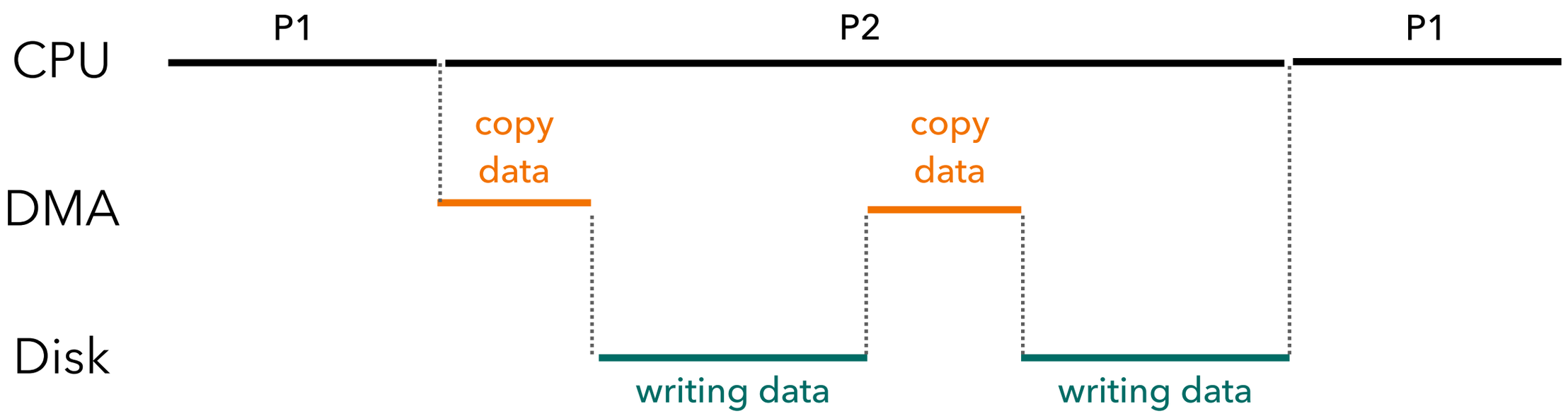
- 在向 I/O 设备发出指令后需要知道设备是否已完操作 (或遇到错误)
  - 通过轮询 (polling) 的方式
  - 通过中断 (interrupt) 的方式
    - 在 I/O 操作运行时允许 CPU 执行其它计算任务
    - 但在数据传输时 CPU 仍需向设备寄存器不断写数据
      (e.g., transfer 1GB data to a device)



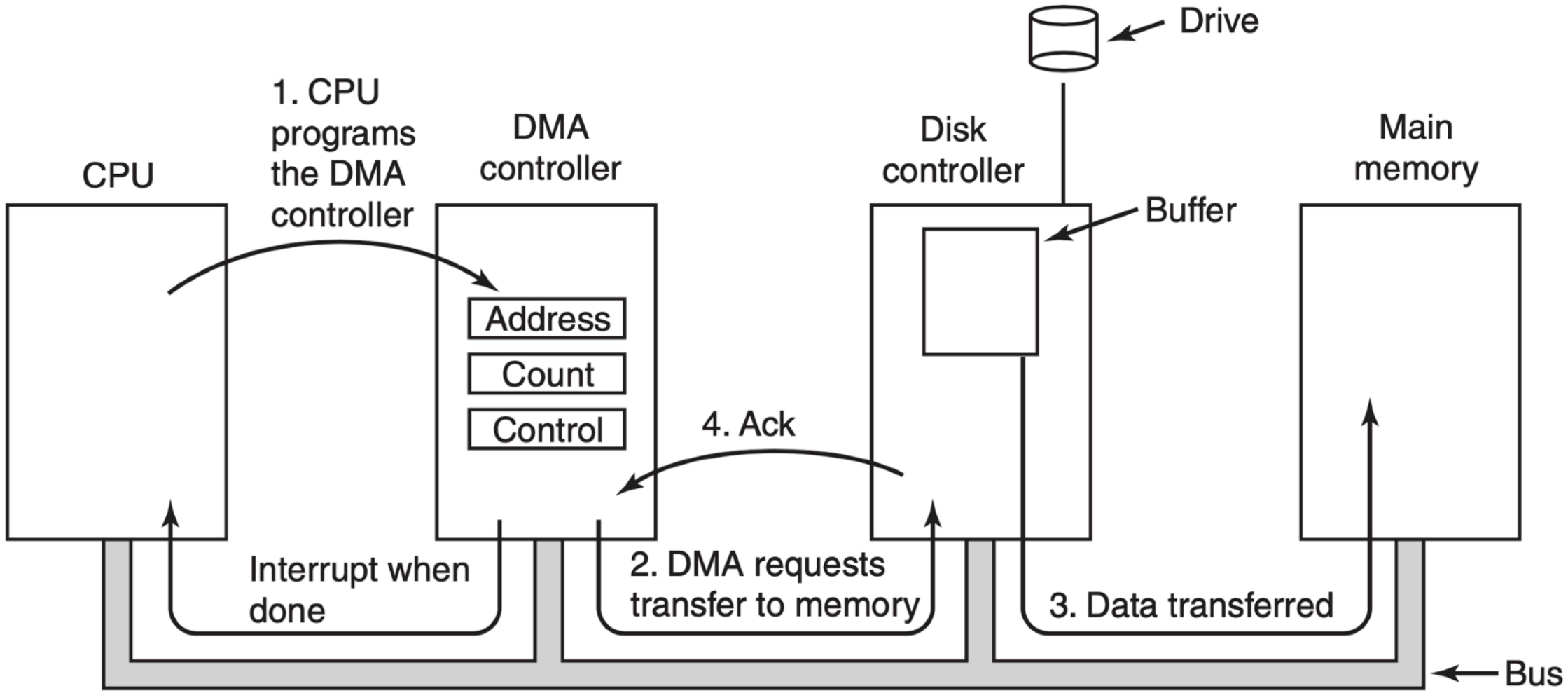
Direct Memory Access (DMA)

直接内存访问 (Direct Memory Access, DMA): 依靠一个特殊的硬件控制器来协调内存和设备寄存器之间的数据传输

- 操作系统首先向 DMA 发送指令 (where the data lives in, how much data to copy, and which device to send it to …)
- 随后由 DMA 和 I/O 设备进行交互
- 当 I/O 操作结束时，DMA 控制器向 CPU 发送中断信号



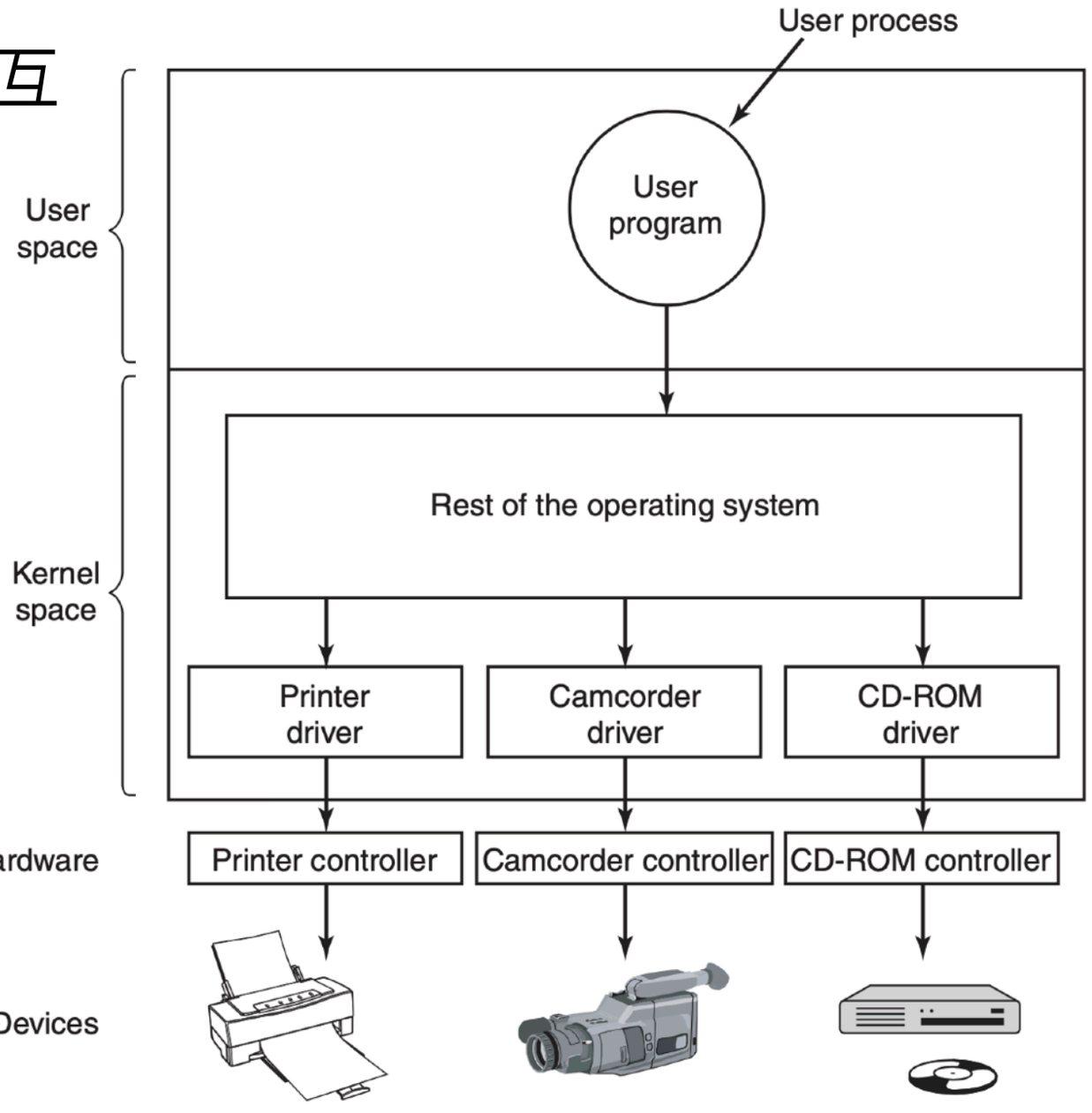
Direct Memory Access (DMA)



设备抽象

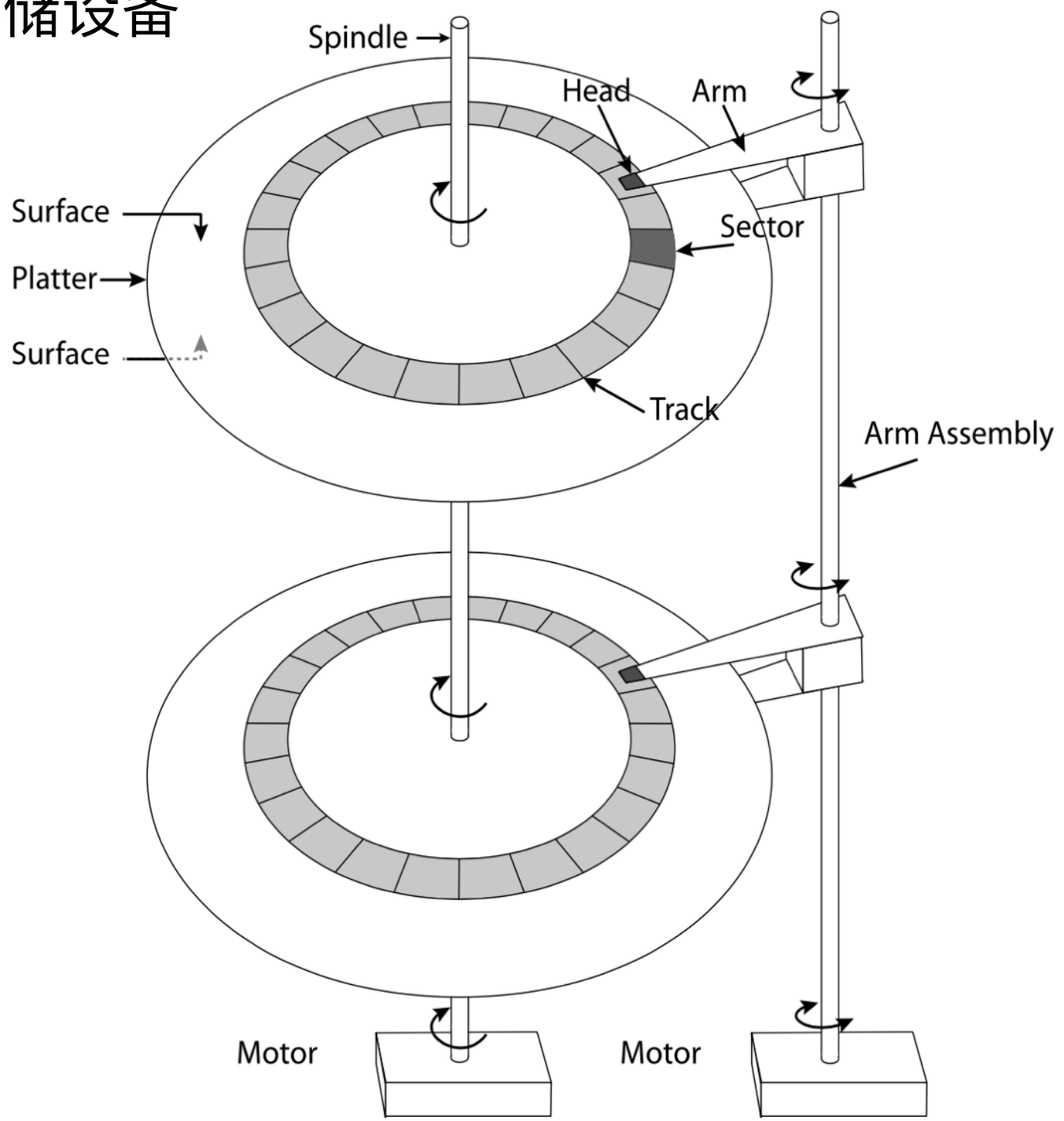
依靠设备驱动 (Device Driver) 实现对底层硬件的抽象

- 通过设备硬件接口与设备进行交互
- 向外暴露一组通用的 APIs
  - 通常包括 `read()`,`write()` 和 `ioctl()`
  - 只要实现了这些接口，就可以看作是一个 “设备”
- 现代操作系统中 70%+ 的代码都是驱动程序代码
  - 太多的设备、每个设备交互方式往往都不太一样



磁盘 Hard Disk

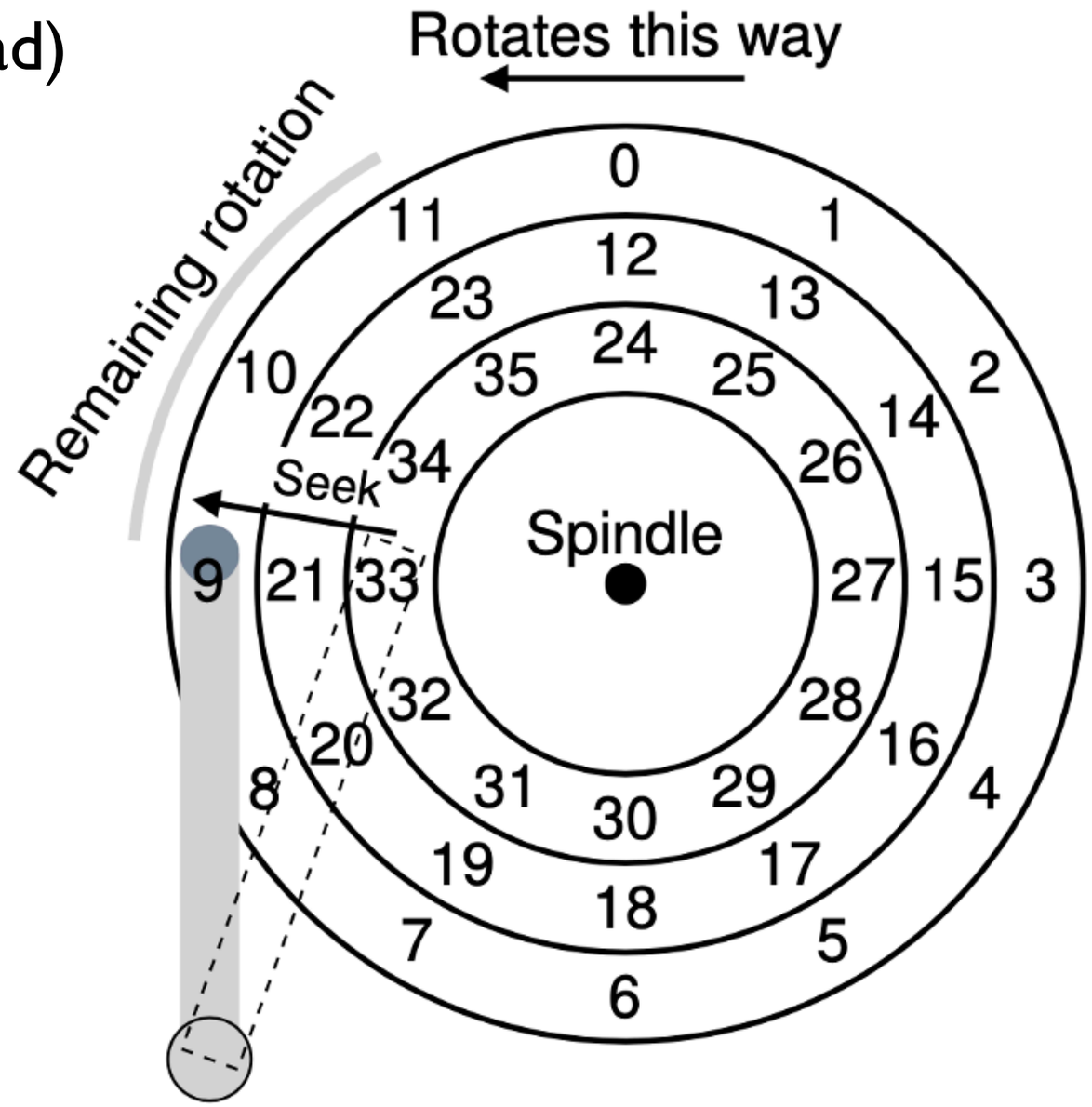
一种典型的大容量数据持久化存储设备



磁盘 Hard Disk

数据读写开销 $T_{I/O} = T_{seek} + T_{rotation} + T_{transfer}$

- 寻道时间 (seek time): 将磁头 (head) 移动到指定磁道 (track)
- 旋转延迟 (rotational delay): 等待扇区 (sector) 旋转到磁头下
- 数据传输 (data transfer): 向扇区读写数据



磁盘 Hard Disk

寻道时间和旋转延迟导致磁盘顺序读写性能优于随机读写

- 随机访问模式传输 4KB 数据

$T_{seek} = 4\ ms$ (average seek)

$T_{rotation} = 2\ ms$ (average rotation)

$T_{transfer} = 0.03\ ms$

$R_{I/O} = Size_{transfer} / T_{I/O}$

$= 4\ KB / 6.03\ ms = 0.65\ MB/s$

	Cheetah 15K.5
Capacity	300 GB
RPM	15,000
Average Seek	4 ms
Max Transfer	125 MB/s
Platters	4
Cache	16 MB
Connects via	SCSI

磁盘 Hard Disk

寻道时间和旋转延迟导致磁盘顺序读写性能优于随机读写

- 顺序访问模式传输 100MB 数据

$T_{seek} = 4\ ms$ (average seek)

$T_{rotation} = 2\ ms$ (average rotation)

$T_{transfer} = 0.8\ s$

$R_{I/O} = Size_{transfer} / T_{I/O}$

$= 100\ MB / 0.806\ s = 124\ MB/s$

	Cheetah 15K.5
Capacity	300 GB
RPM	15,000
Average Seek	4 ms
Max Transfer	125 MB/s
Platters	4
Cache	16 MB
Connects via	SCSI

磁盘 Hard Disk

近三十年来磁盘在数据容量上有非常大的提升，但在寻道时间和转速上的提升相对比较有限

Parameter	IBM 360-KB floppy disk	WD 3000 HLFS hard disk
Number of cylinders	40	36,481
Tracks per cylinder	2	255
Sectors per track	9	63 (avg)
Sectors per disk	720	586,072,368
Bytes per sector	512	512
Disk capacity	360 KB	300 GB
Seek time (adjacent cylinders)	6 msec	0.7 msec
Seek time (average case)	77 msec	4.2 msec
Rotation time	200 msec	6 msec
Time to transfer 1 sector	22 msec	1.4 μsec

磁盘调度

给定一组磁盘访问请求，决定以什么样的顺序进行响应 (disk schedule)

- 由于磁盘 I/O 成本很高，操作系统曾经在磁盘调度上发挥过重要作用
- 通过优化请求调度顺序来最小化磁头移动 (最大化磁盘 I/O 吞吐量)
  - First Come First Service (FCFS)
  - Shortest Seek Time First (SSTF)
  - Shortest Positioning Time First (SPTF)

磁盘调度

First Come First Service (FCFS)

按磁盘访问请求的到达顺序进行调度

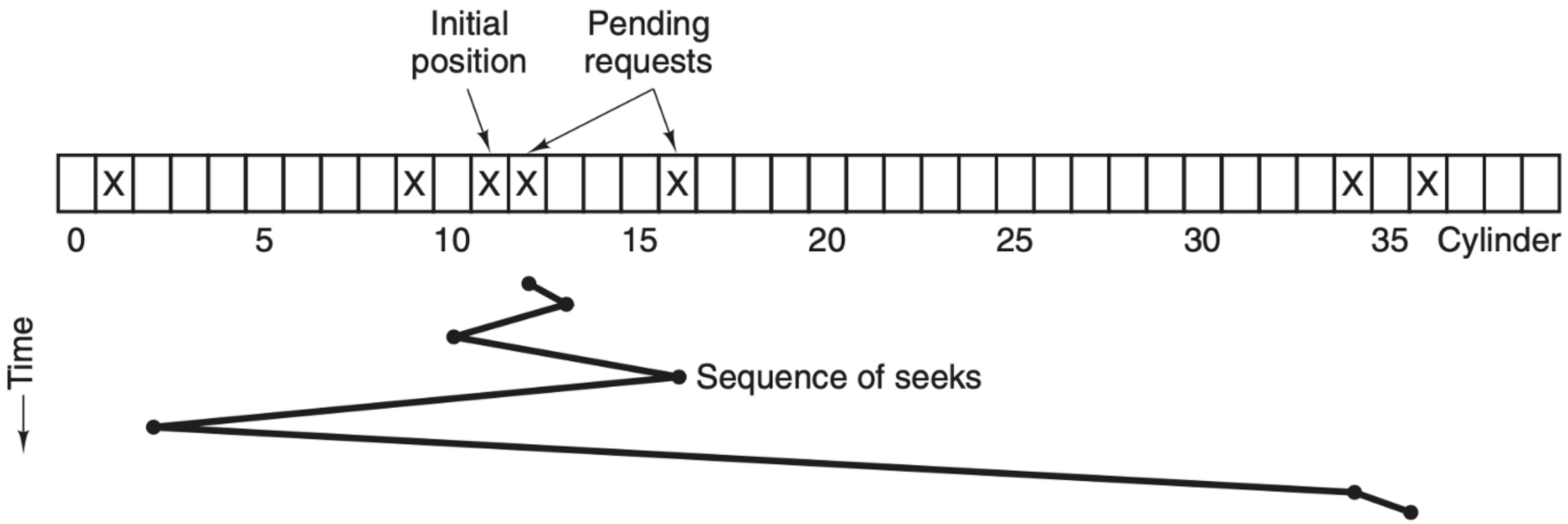
- 假设磁头 (arm head) 当前位于编号为 11 的磁道 (track)，后续对磁道访问请求的到达顺序为 1, 36, 16, 34, 9, 12
- 在 FCFS 下磁头总共需要移动 $10 + 35 + 20 + 18 + 25 + 3 = 111$ 个磁道

磁盘调度

Shortest Seek Time First (SSTF)

优先调度和当前磁道距离最近的访问请求 (最小化寻道时间)

- 其实就是 Shortest Job First 的思想
- 在 SSTF 下磁头总共需要移动 $1 + 3 + 7 + 15 + 33 + 2 = 61$ 个磁道

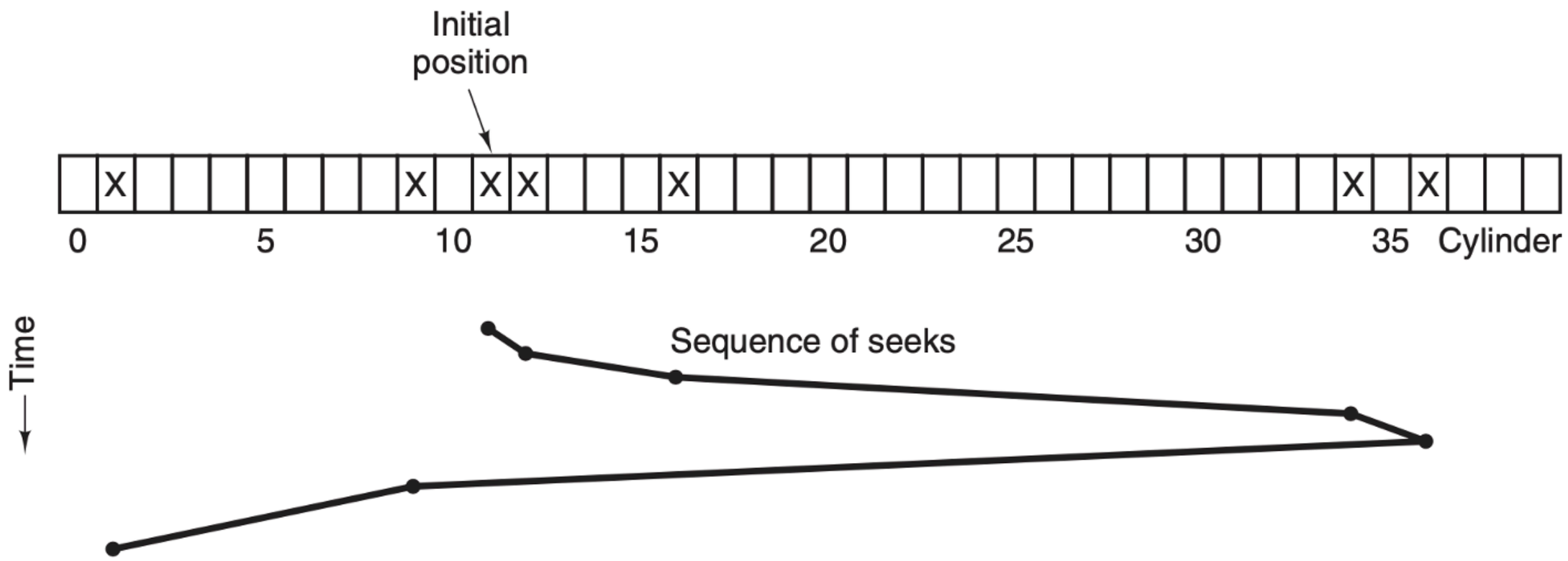


磁盘调度

Elevator (SCAN)

让磁头在磁道上来回移动，并依次处理遇到的访问请求

- 在 SCAN 下磁头总共需要移动 $1 + 4 + 18 + 2 + 27 + 8 = 60$ 个磁道
- 可以在此基础上实现很多变种方法 (例如，在每次扫描时是否冻结待处理请求、是否只从一个方向扫描)

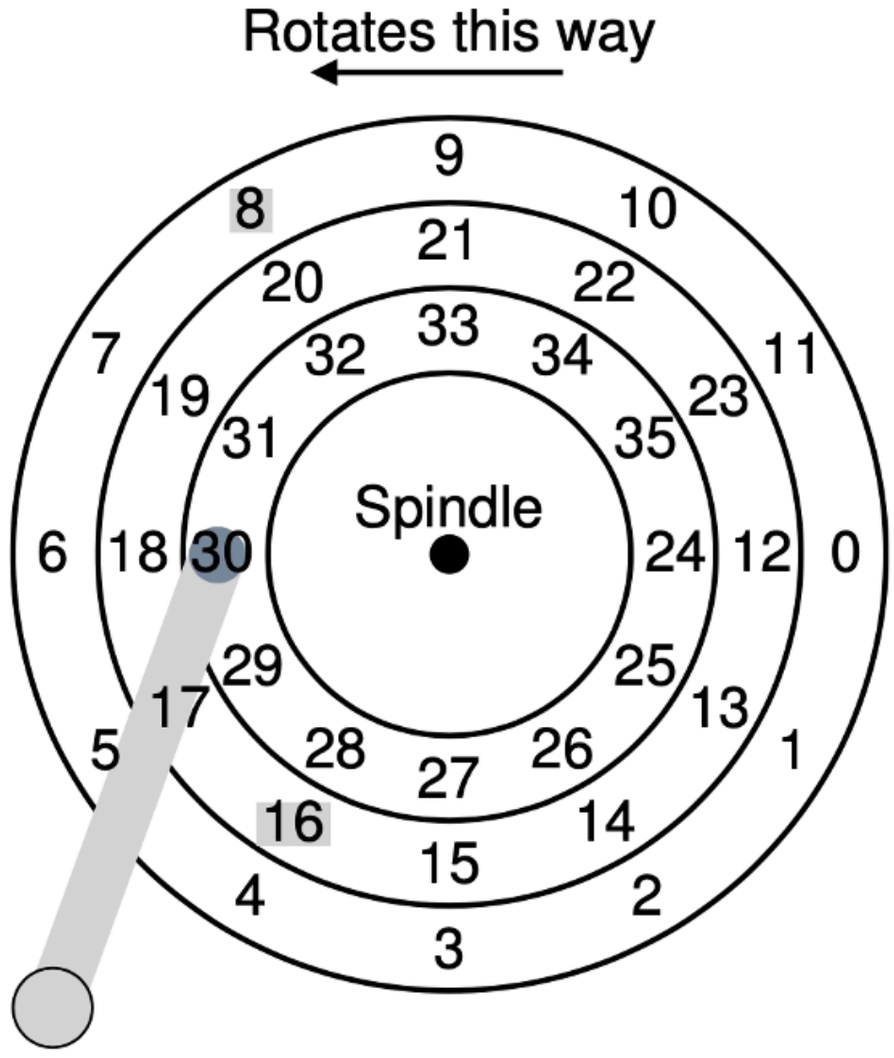


磁盘调度

Shortest Positioning Time First (SPTF)

同时考虑寻道时间 (seek time) 和旋转延迟 (rotational delay) 的开销

- 假设当前磁头位于 30 扇区 (sector)，后续请求为 16 和 8 扇区
  - 如果 seek time 开销明显高于 rotational delay → sector 16
  - 如果 seek time 开销仅略高于 rotational delay → sector 8
- 需要精确的知道磁盘的底层结构信息 (扇区布局、磁头移动速度等)



磁盘调度

- 现代操作系统通常把一系列磁盘访问请求直接发送给磁盘，由磁盘控制器负责调度
- 现代磁盘同时还拥有较大的缓存
  - 在读取一个扇区 (sector) 时将整个磁道 (track) 都缓存起来
    - 和缓存内存中数据不同，这里缓存的是没有被显示读取的数据
  - 在写一个扇区时，在数据写入缓冲区后就响应 (write back) 或等到数据确实写入扇区后才响应 (write through)

RAID *

Redundant Array of Inexpensive (Independent) Disks: 把多个潜在不可靠的磁盘虚拟成一块非常可靠且性能极高的虚拟磁盘

- 容量更大 (bigger): 毕竟由多个磁盘构成
- 速度更快 (faster): 更快的磁盘 I/O 访问速度
- 更加可靠 (more reliable): 部分磁盘坏了数据也不会丢失 (redundancy)

RAID-0: Striping *

将数据块分布在 N 个物理磁盘上

Disk 0	Disk 1	Disk 2	Disk 3
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

- **容量**：$N \times B$
- **速度**：$R_{I/O} = N \times X\ MB/s$
  - 多个物理磁盘可并行读写
- **可靠性**：没有改善
  - 任何一个数据块坏了都会导致数据丢失

RAID-1: Mirroring *

为每个物理磁盘都维护一个备份

Disk 0	Disk 1	Disk 2	Disk 3
0	0	1	1
2	2	3	3
4	4	5	5
6	6	7	7

- **容量**：$(N \times B) / 2$
- **速度**：
  - 随机读 $R_{I/O} = N \times X\ MB/s$
  - 顺序读 $R_{I/O} = N/2 \times X\ MB/s$
  - 每个写都要写两个磁盘块 $R_{I/O} = N/2 \times X\ MB/s$
- **可靠性**：
  - 可以容忍单个磁盘故障
  - 至多 $N / 2$ 个 (e.g., disks 1 and 3)

RAID-4: Saving Space With Parity *

只使用一个物理磁盘来实现容错 (用 1bit 的冗余来恢复丢失的 bit)

- 为每个 strip 维护一个 parity block
- 按 bit 做 XOR 操作
- 每个 strip 中 bit 1 的个数一定是偶数
- 通过对其余所有 bits 做 XOR 来恢复丢失的 bit

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
4	5	6	7	P1
8	9	10	11	P2
12	13	14	15	P3

C0	C1	C2	C3	P
0	0	1	1	XOR(0,0,1,1) = 0
0	1	0	0	XOR(0,1,0,0) = 1

Block0	Block1	Block2	Block3	Parity
00	10	11	10	11
10	01	00	01	10

RAID-4: Saving Space With Parity *

只使用一个物理磁盘来实现容错 (用 1bit 的冗余来恢复丢失的 bit)

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
*4	5	6	7	+P1
8	9	10	11	P2
12	*13	14	15	+P3

- **容量**：$(N - 1) \times B$
- **速度**：
  - 顺序读写：$R_{I/O} = (N - 1) \times X\ MB/s$
  - 随机读：$R_{I/O} = (N - 1) \times X\ MB/s$
  - 随机写：$R_{I/O} = X/2\ MB/s$
    - parity block 成为了瓶颈 (所有 random write 都涉及该磁盘的读写)
- **可靠性**：可以容忍单个磁盘故障

RAID-5: Rotating Parity *

将 parity block 均匀分布在物理磁盘中

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
5	6	7	P1	4
10	11	P2	8	9
15	P3	12	13	14
P4	16	17	18	19

- **容量**：$(N - 1) \times B$
- **速度**：
  - 顺序读写：$R_{I/O} = (N - 1) \times X\ MB/s$
  - 随机读：$R_{I/O} = N \times X\ MB/s$
  - 随机写：$R_{I/O} = X/2 \times N/2\ MB/s$
    - 虽然每个 logical write 需要 parity block 的两次 I/O，但可以并行发送 $N/2$ 个 write 请求
- **可靠性**：可以容忍单个磁盘故障

Summary

- 设备接口、以及与设备的交互 
  - Direct Memory Access (DMA)
  - 设备抽象和驱动
- 磁盘 (Hard Disk)
  - 磁盘的寻道时间和旋转延迟
  - 磁盘调度
  - RAID