操作系统

Operating system

胡燕

大连理工大学

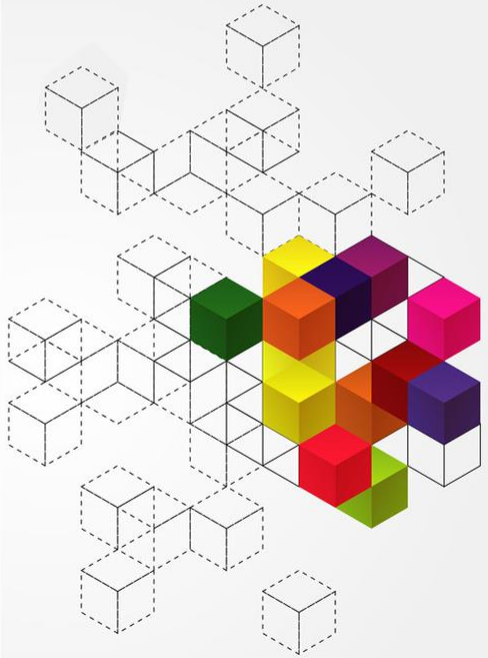
内容纲要

12.1 大容量存储简介

一、磁盘物理结构

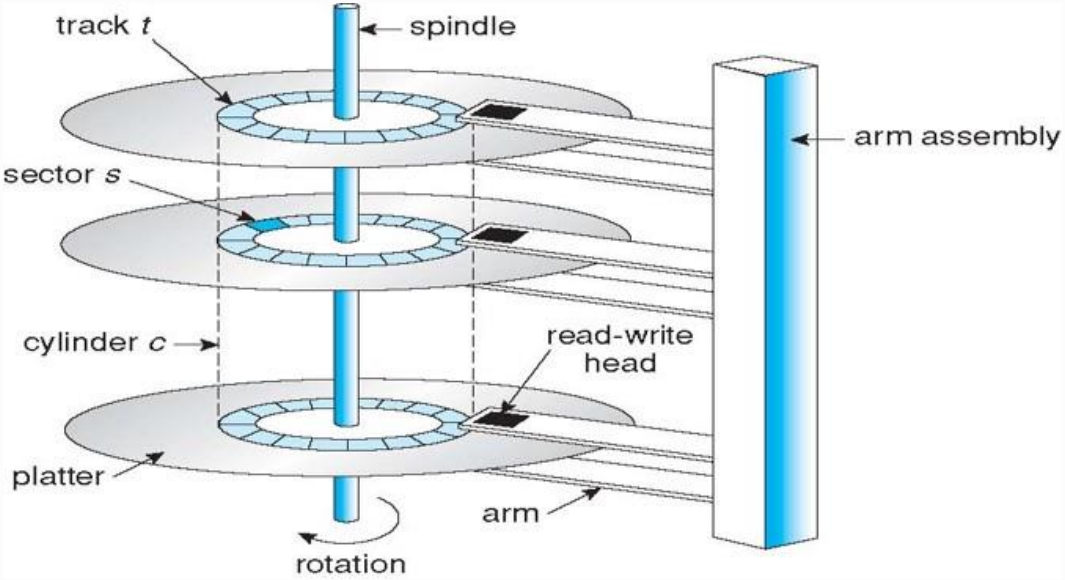
二、磁盘参数

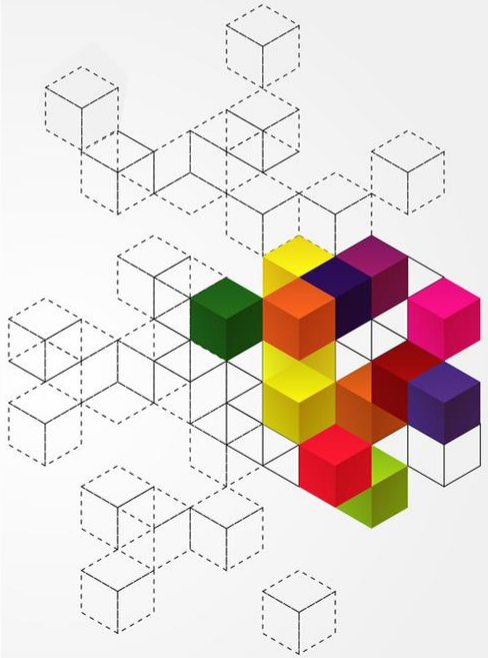
三、大容量存储技术发展



一、磁盘物理结构

- **机械磁盘结构示意图**
  - 柱面（Cylinder），磁道（track），扇区（sector）



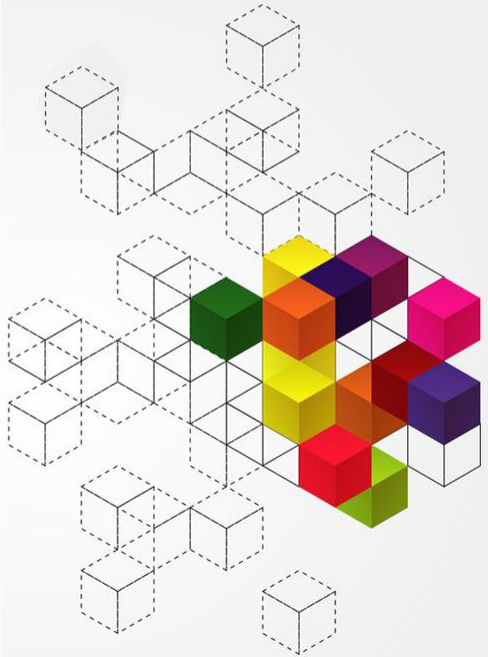


二、磁盘参数

- **现代机械磁盘的典型参数值**
  - 理论传输速率：6Gb/s
  - 实际（有效）传输速率：1Gb/s
  - 寻道时间：3ms~12ms
  - 旋转延迟：1/(RPM*60)
  - 平均旋转延迟=0.5*旋转延迟

Spindle [rpm]	Average latency [ms]
4200	7.14
5400	5.56
7200	4.17
10000	3
15000	2

(From Wikipedia)



三、大容量存储技术发展

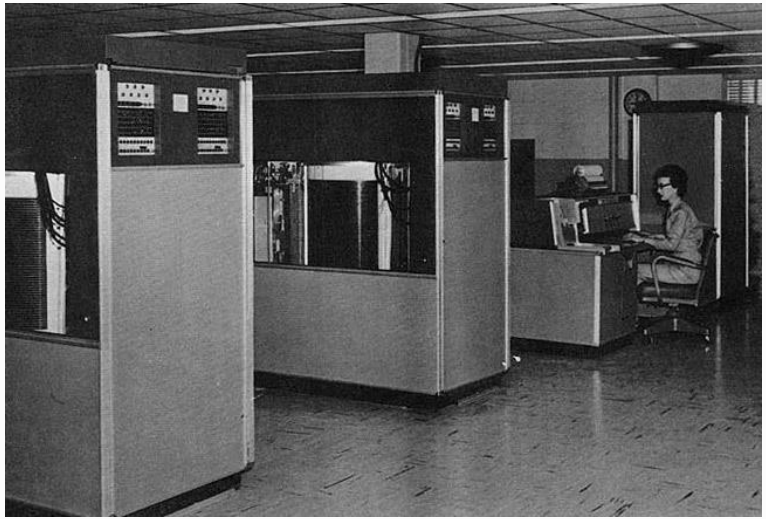
- **史上最早的商用磁盘**
  - the IBM Model 350 disk storage system， 1956
  - 容量： 5M (7 bit)
  - 50个24英寸磁碟（Platters）
  - 访问时间 < 1 second

三、大容量存储技术发展

- **史上最早的商用磁盘**
  - the IBM Model 350 disk storage system， 1956
  - 容量： 5M (7 bit)
  - 50个24英寸磁碟（Platters）
  - 访问时间 < 1 second

三、大容量存储技术发展

- **IBM推动早期硬盘技术的创新**
  - RAMAC：最早使用硬盘存储的商用计算机，上面装载了Model 350 disk storage system
  - 需要整个房间放置该计算机，其硬盘系统有两个冰箱那么大



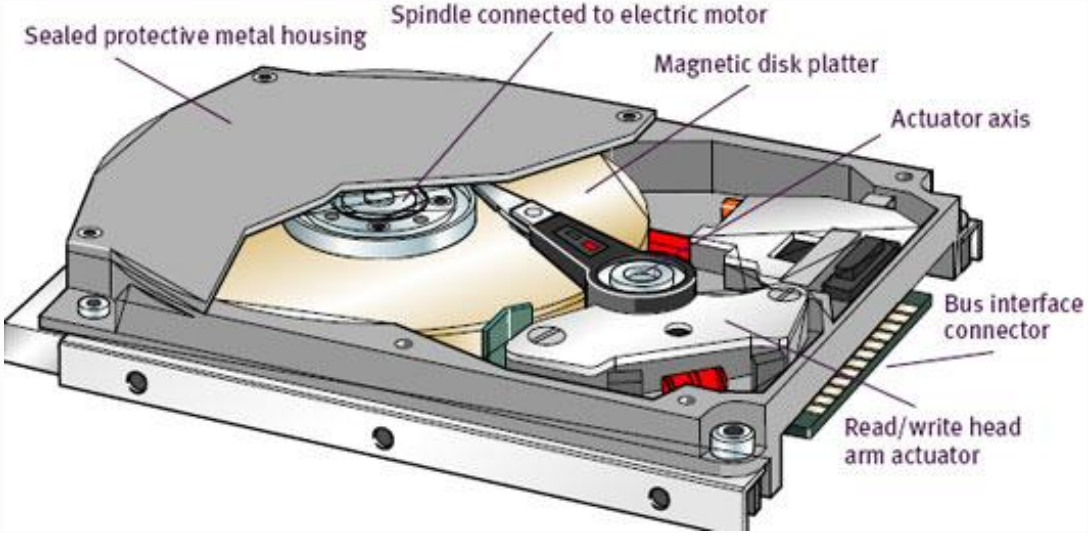
RAMAC：Random Access Method for Accounting and Control

三、大容量存储技术发展

- **IBM早期推出的可移动存储：IBM 1311磁盘**
  - Removable storage

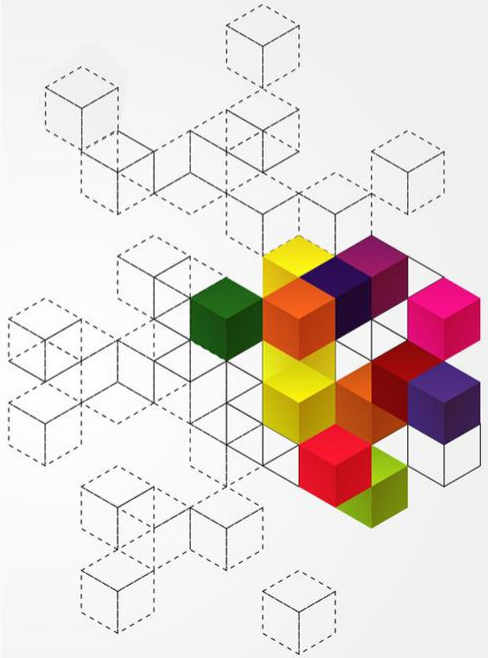
三、大容量存储技术发展

- **现代磁盘样式结构**



本讲小结

- IO子系统功能概述

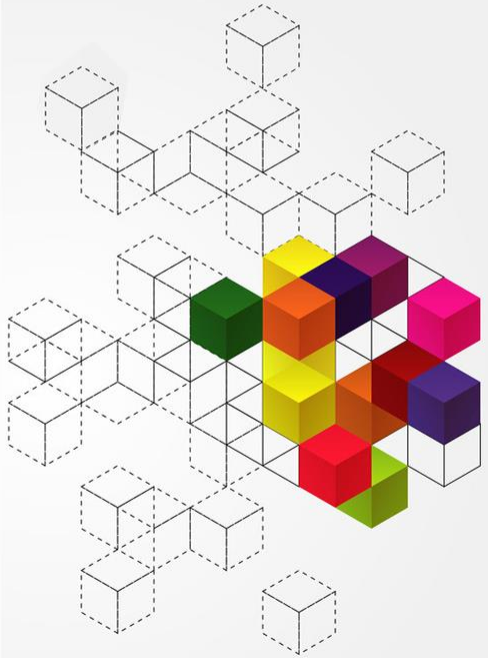


内容纲要

12.2 磁盘调度算法

一、 磁盘调度背景

二、 磁盘调度算法

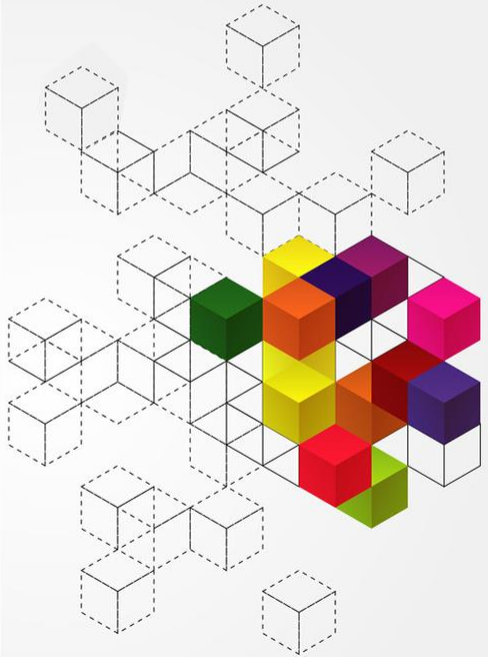


一、磁盘调度背景

为什么需要磁盘调度

- **根据磁盘特性进行高效数据访问**
- **充分利用磁盘数据传输带宽**

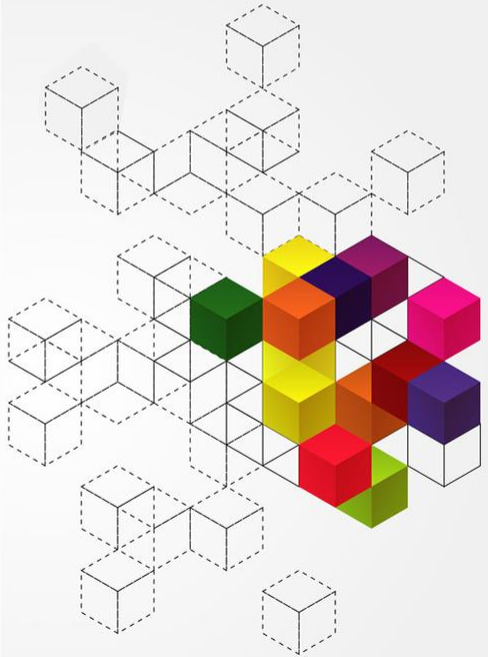
适用对象：机械磁盘



一、磁盘调度背景

磁盘调度的优化指标：寻道时间

- 可由操作系统控制
- 调度算法目标：最小化磁头寻道距离

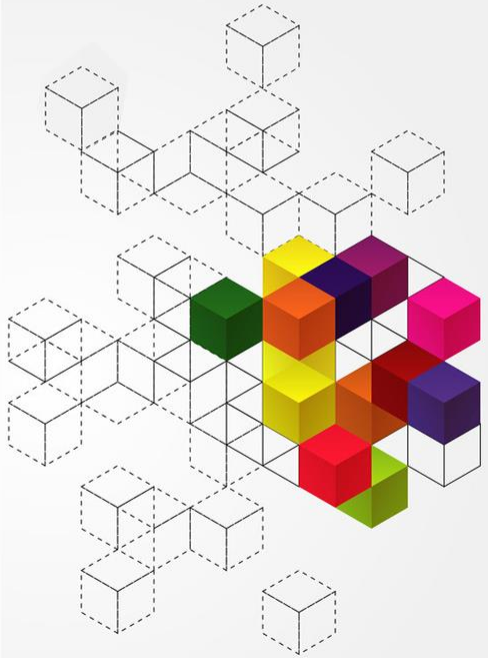


二、磁盘调度算法

磁盘调度算法输入：IO请求队列

98, 183, 37, 122, 14, 124, 65, 67

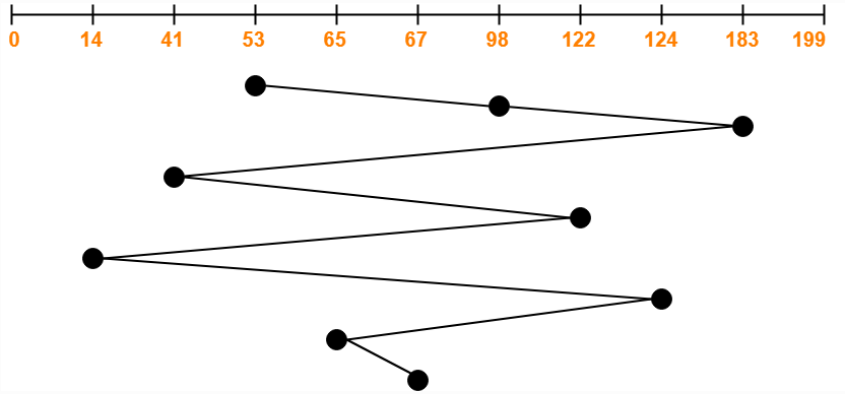
每个数字表示一个具体的IO请求所访问磁盘块所处的磁道号
不同的磁盘调度算法，服务IO请求的顺序不同，算法效果也因此不同



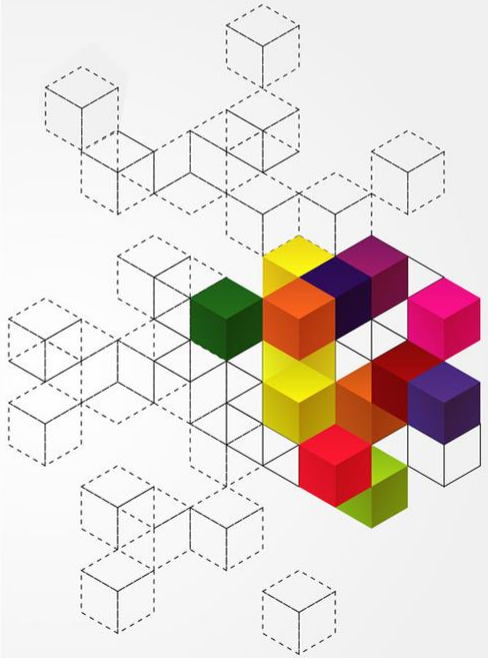
二、磁盘调度算法 – (1)FCFS算法

FCFS算法

请求队列=98, 183, 37, 122, 14, 124, 65, 67



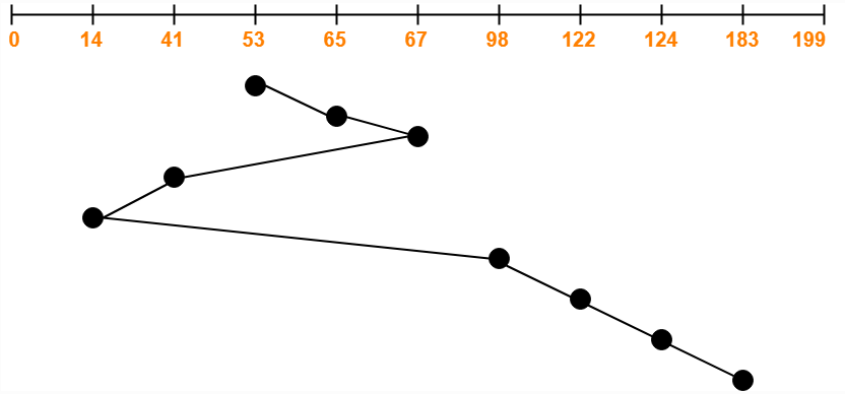
Total head movements incurred while servicing these requests
= (98 – 53) + (183 – 98) + (183 – 41) + (122 – 41) + (122 – 14) + (124 – 14) + (124 – 65) + (67 – 65)
= 45 + 85 + 142 + 81 + 108 + 110 + 59 + 2
= 632



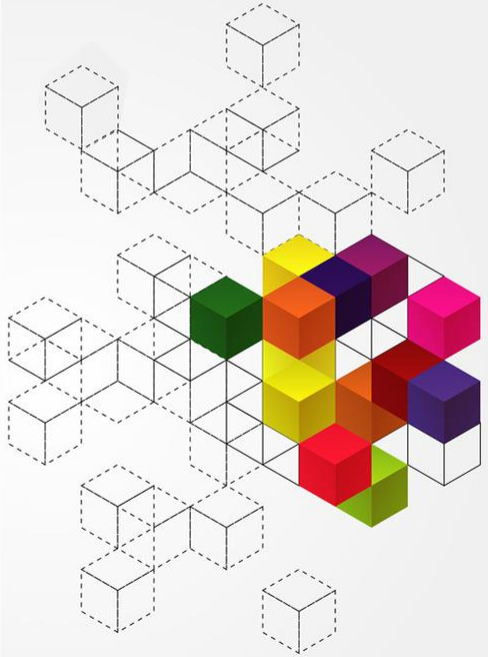
二、磁盘调度算法 – (2)SSTF算法

SSTF算法

请求队列=98, 183, 37, 122, 14, 124, 65, 67



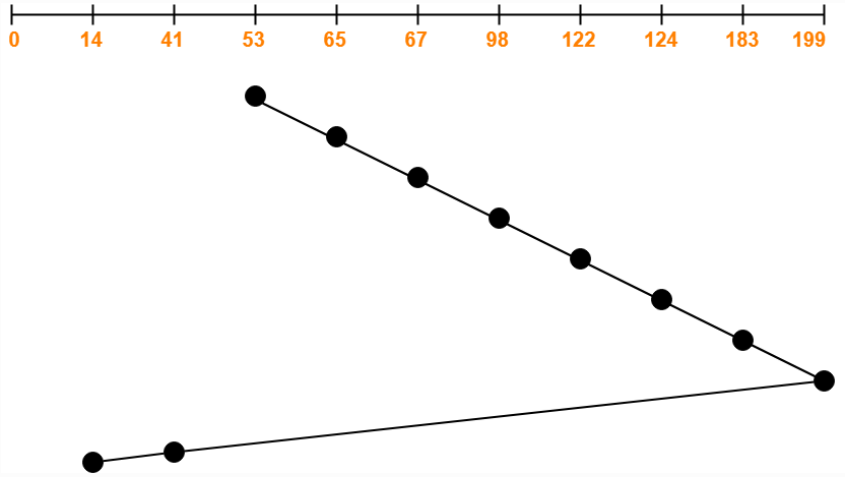
Total head movements incurred while servicing these requests
= (65 – 53) + (67 – 65) + (67 – 41) + (41 – 14) + (98 – 14) + (122 – 98) + (124 – 122) + (183 – 124)
= 12 + 2 + 26 + 27 + 84 + 24 + 2 + 59
= 236



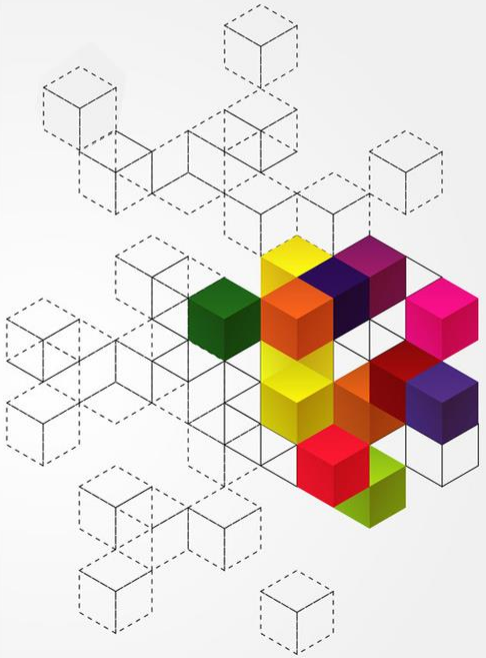
二、磁盘调度算法 – (3)SCAN算法

SCAN算法

请求队列=98, 183, 37, 122, 14, 124, 65, 67



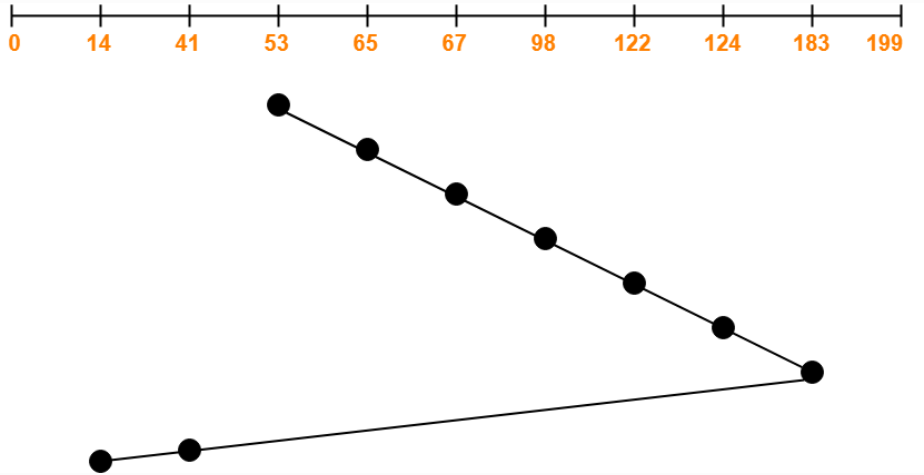
Total head movements incurred while servicing these requests
= (199 – 53) + (199 – 14)
= 331



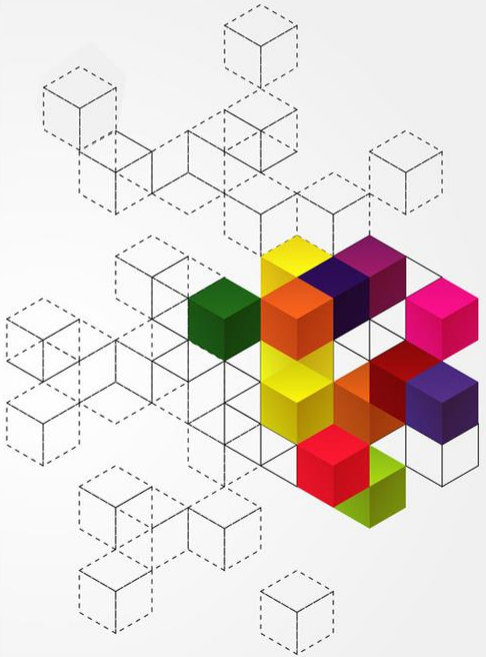
二、磁盘调度算法 – (4)LOOK算法

LOOK算法

请求队列=98, 183, 37, 122, 14, 124, 65, 67



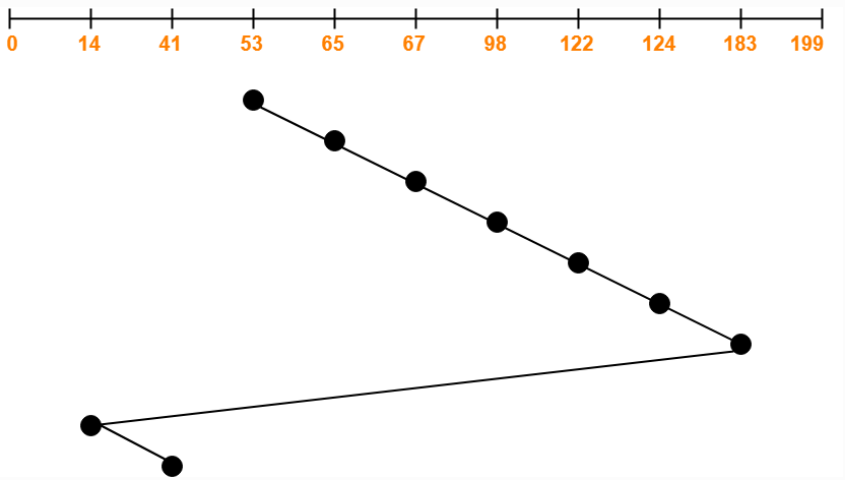
Total head movements incurred while servicing these requests
= (183 – 53) + (183 – 14)
= 299



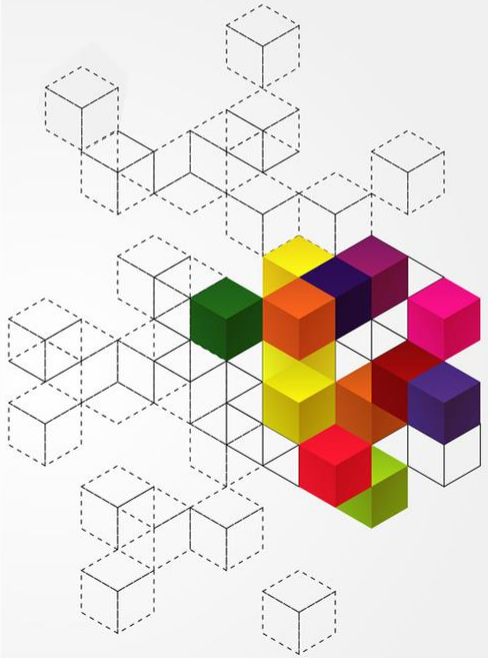
二、磁盘调度算法 – (5)C-LOOK算法

C-LOOK算法

请求队列=98, 183, 37, 122, 14, 124, 65, 67

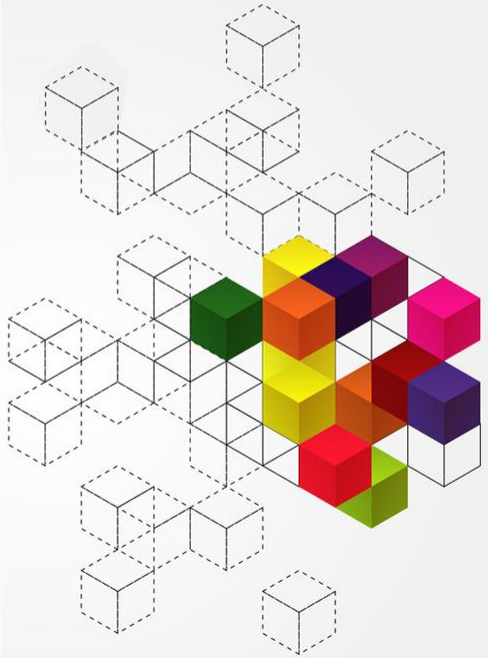


Total head movements incurred while servicing these requests
= (183 – 53) + (183 – 14) + (41 – 14)
= 326



本讲小结

- 磁盘调度背景
- 磁盘调度算法
  - FCFS
  - SSTF
  - SCAN



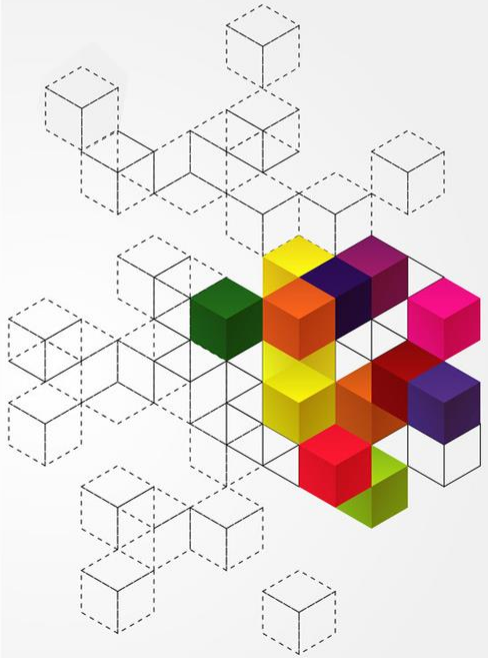
内容纲要

12.3 磁盘交换空间

一、 磁盘交换空间管理

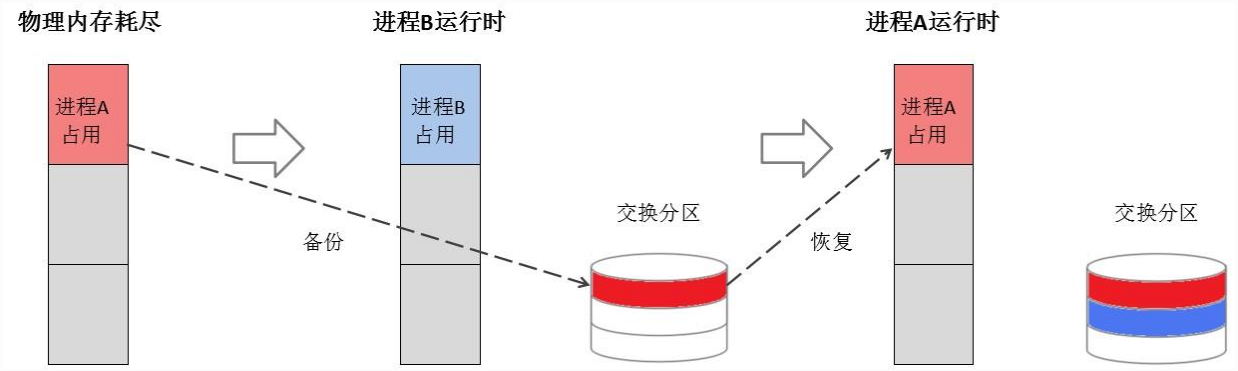
二、 Linux交换机制

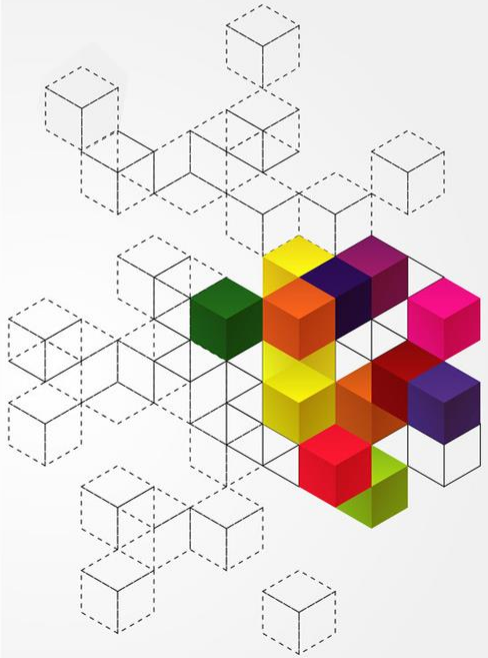
三、 Windows交换方式



一、磁盘交换空间

操作系统交换机制

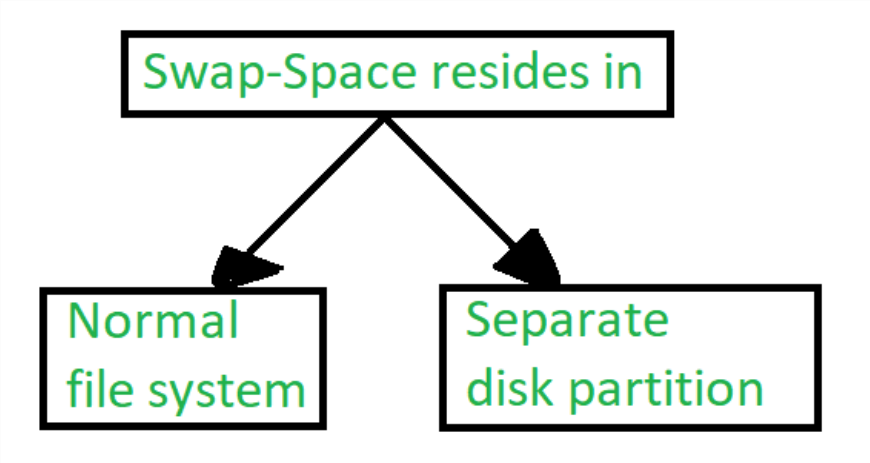




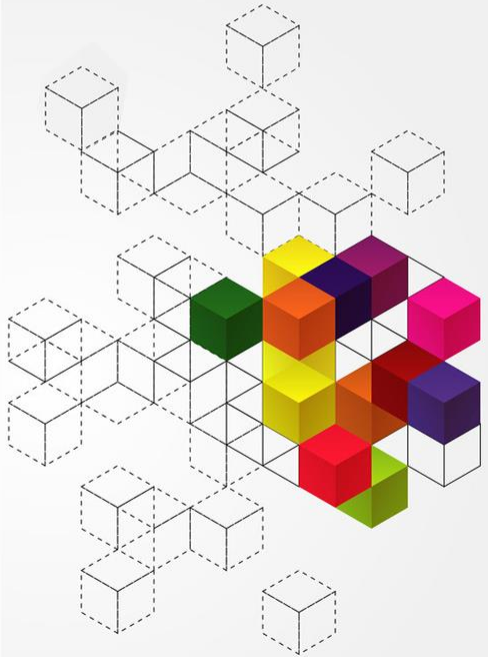
一、磁盘交换空间

磁盘交换空间的两种形式

- **交换分区**
- **交换文件**

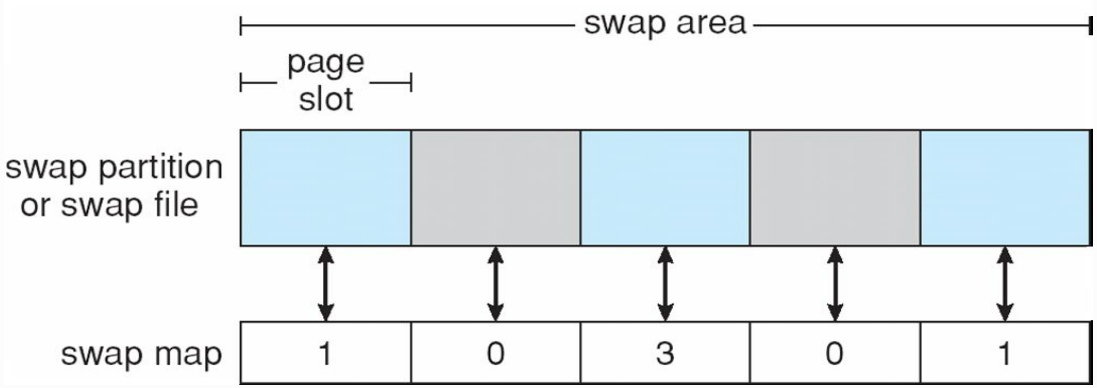


交换出去的进程映像在交换空间中连续存放

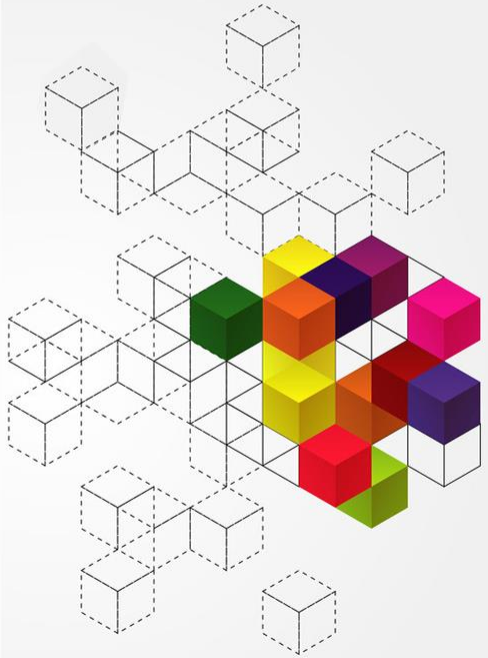


二、Linux交换机制

交换空间逻辑示意图



Each swap area consists of 4-KB **page slots**
Associated with each swap area is a swap-map (array of integers indicating how many different processes that page slot is mapped to)

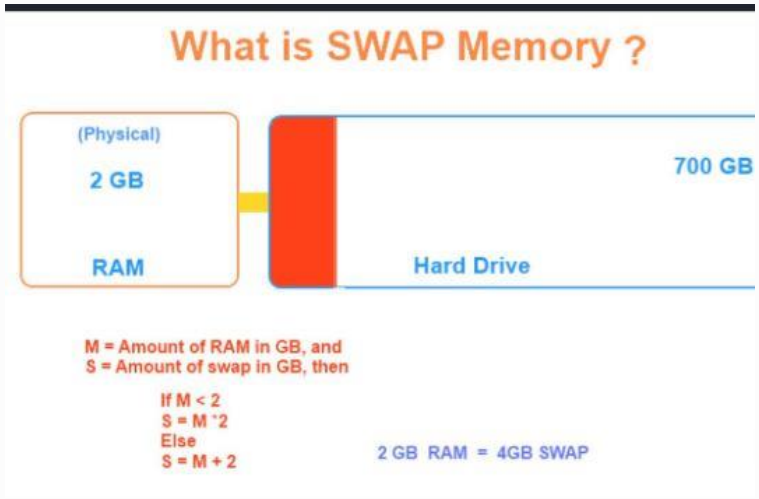


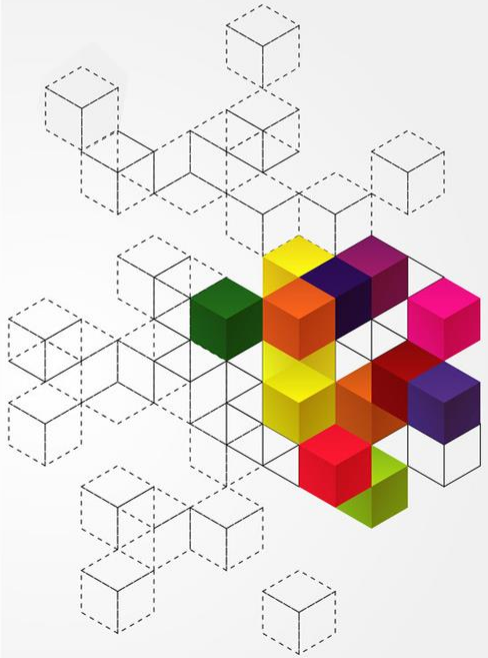
二、Linux交换机制

Linux交换空间

-Linux交换空间形式是专门的分区（swap分区）

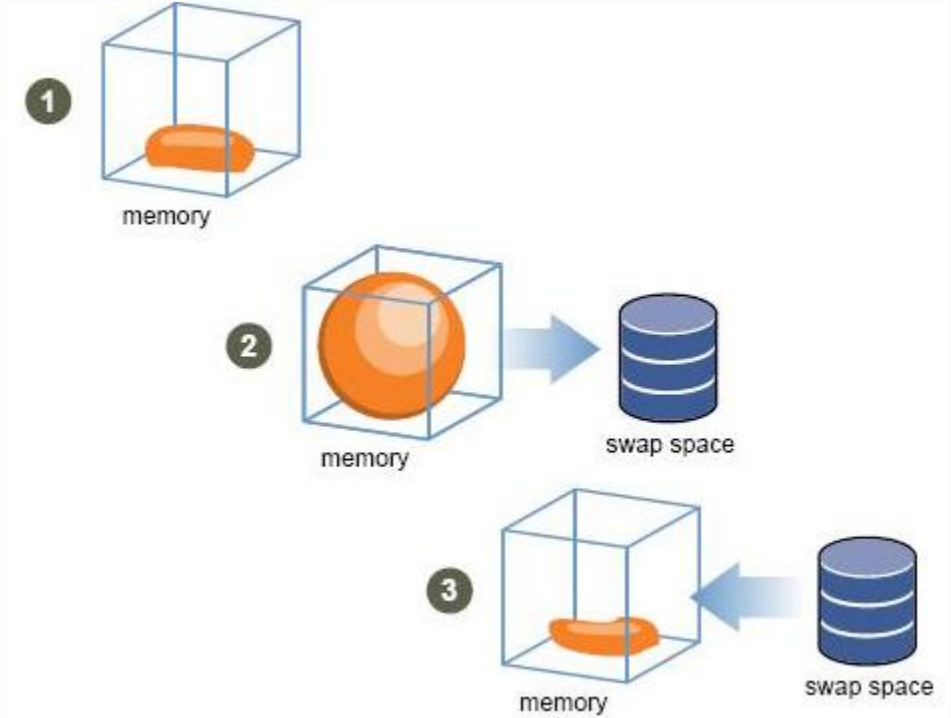
-大小通常为RAM的50%到100%

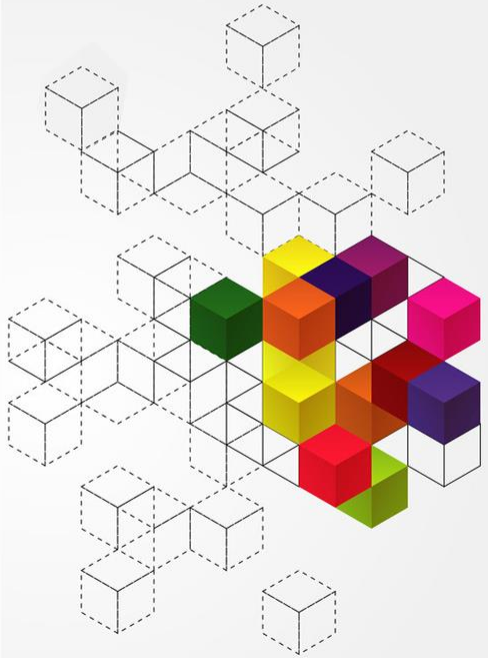




二、Linux交换机制

How linux swap works

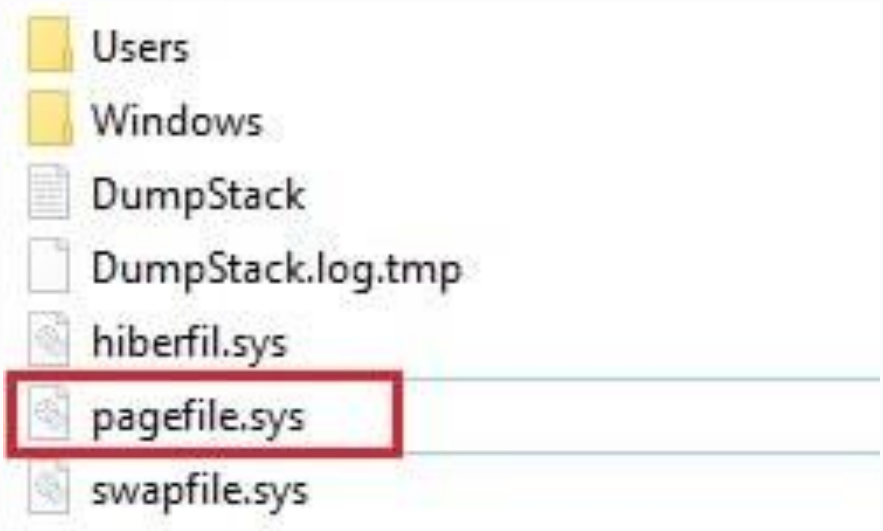


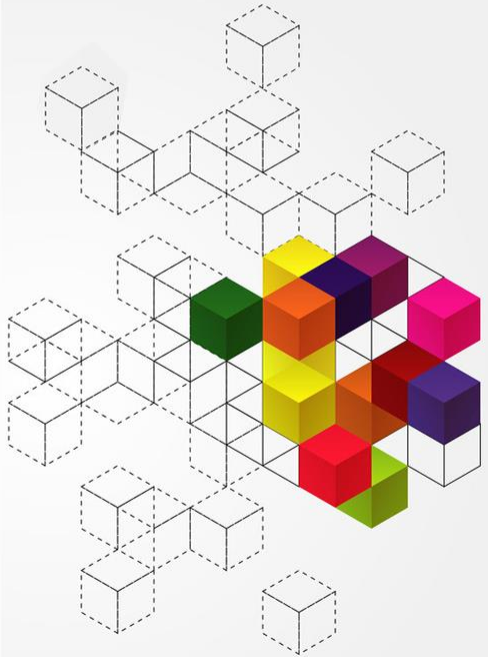


三、Windows交换方式

Windows交换空间

-交换空间是一个名为pagefile.sys的文件，交换的内容均放在该页文件中

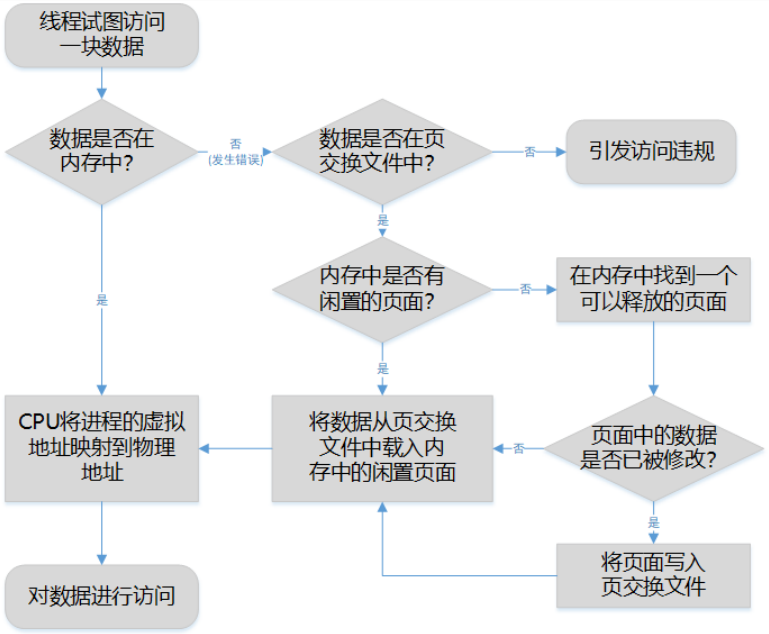


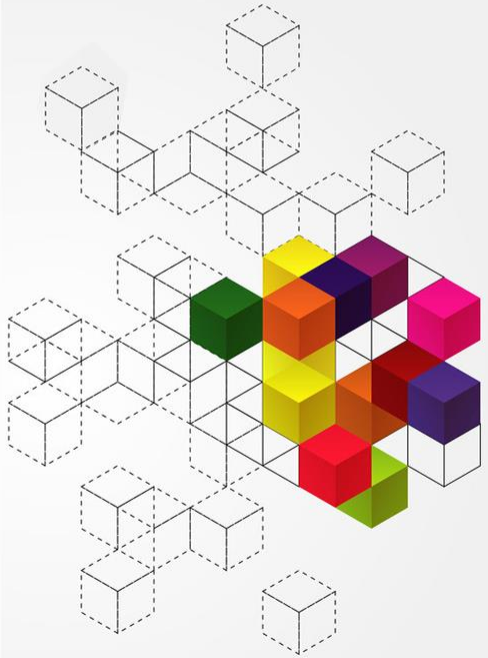


三、Windows交换方式

Windows交换空间

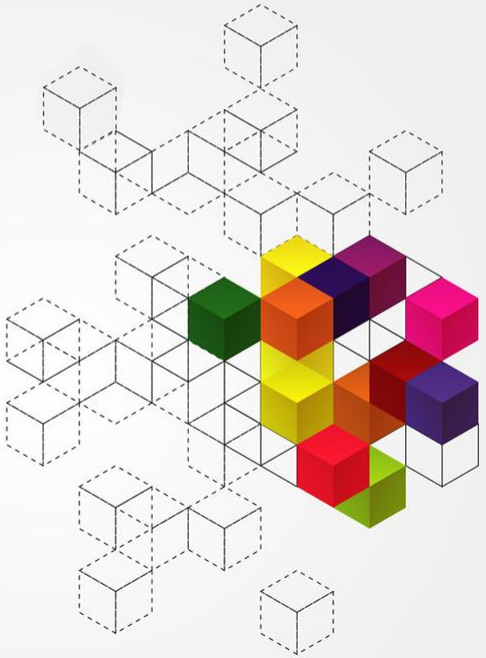
-交换空间是一个名为pagefile.sys的文件，交换的内容均放在该页文件中





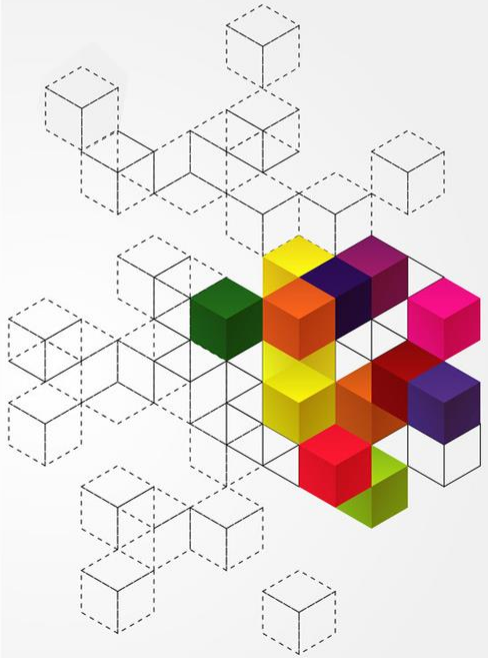
本讲小结

- 磁盘交换空间管理
- Linux交换机制
- Windows交换方式



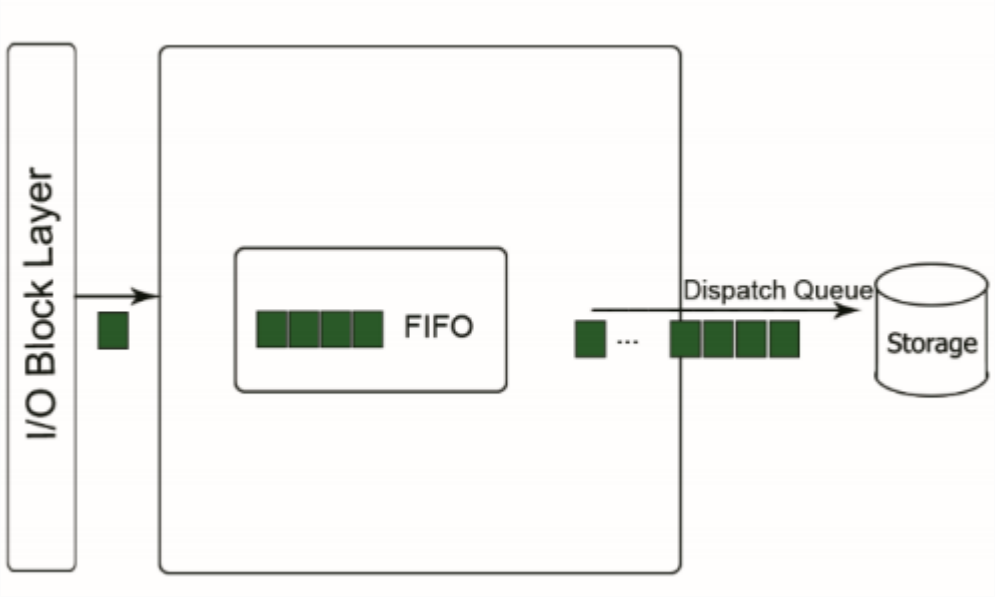
E1、Linux中的IO调度

磁盘调度，在Linux中被称为IO调度

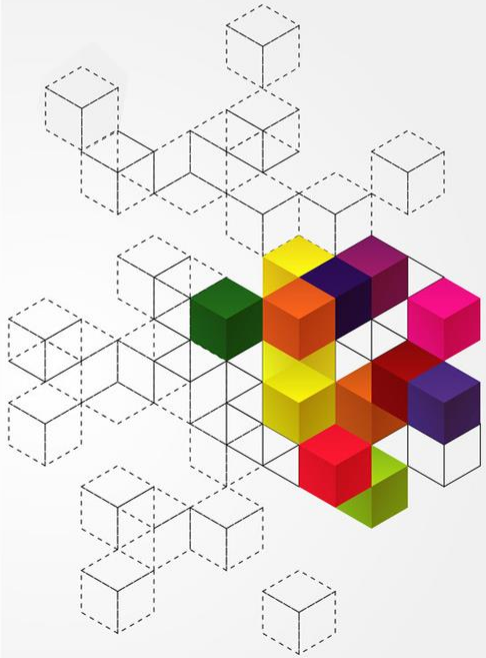


E1、Linux中的IO调度 - NOOP

NOOP调度：FIFO调度在Linux中的实现

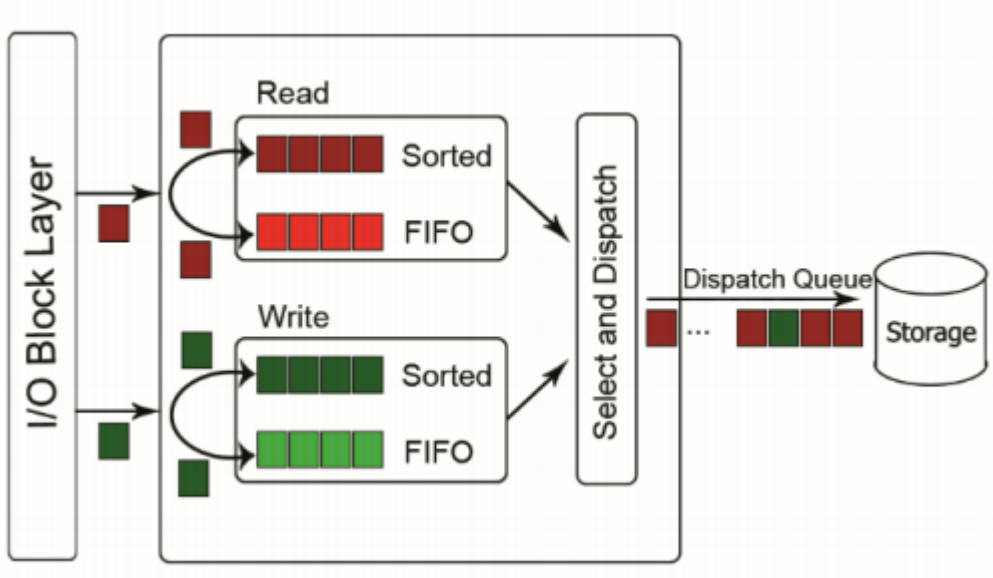


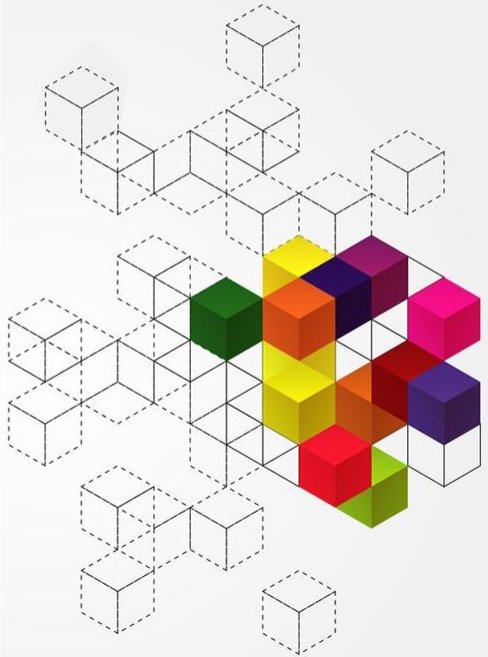
实现基本的邻近磁盘块的IO请求合并



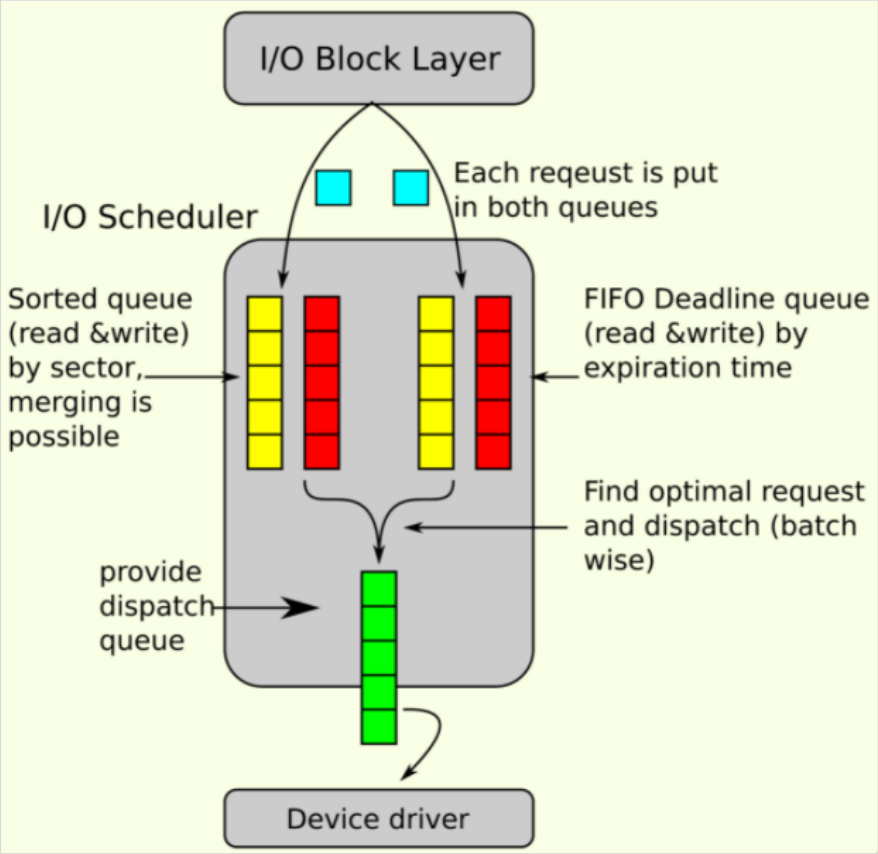
E1、Linux中的IO调度 - Deadline

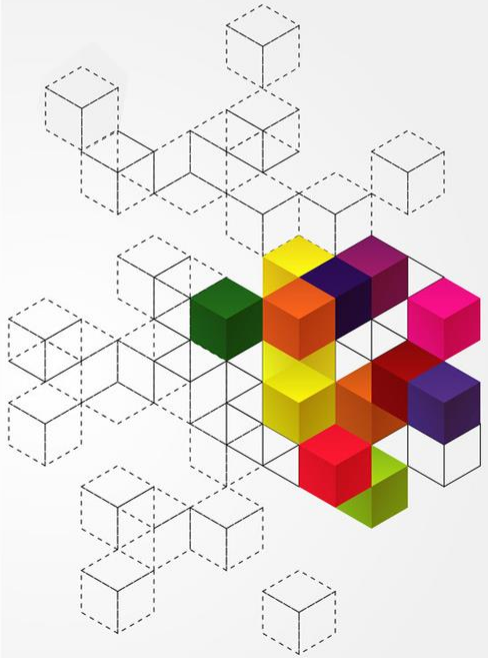
Deadline调度：





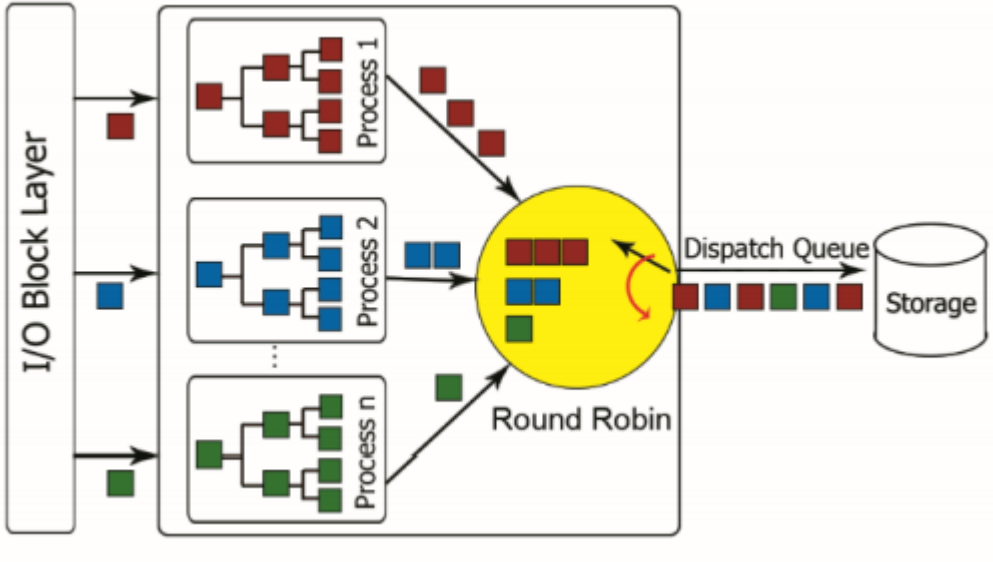
E1、Linux中的IO调度 - Deadline

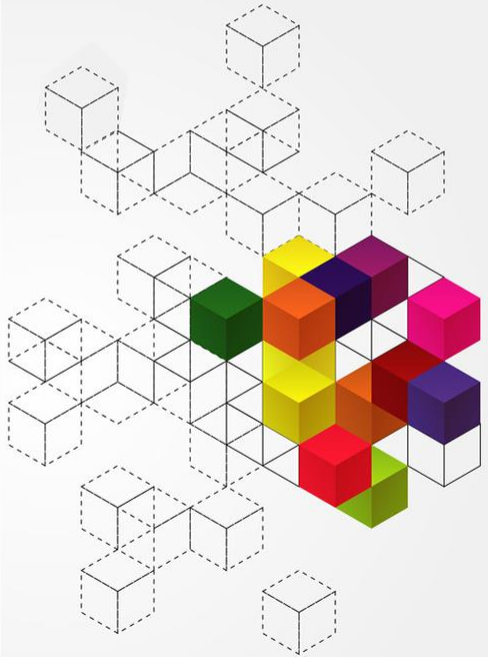




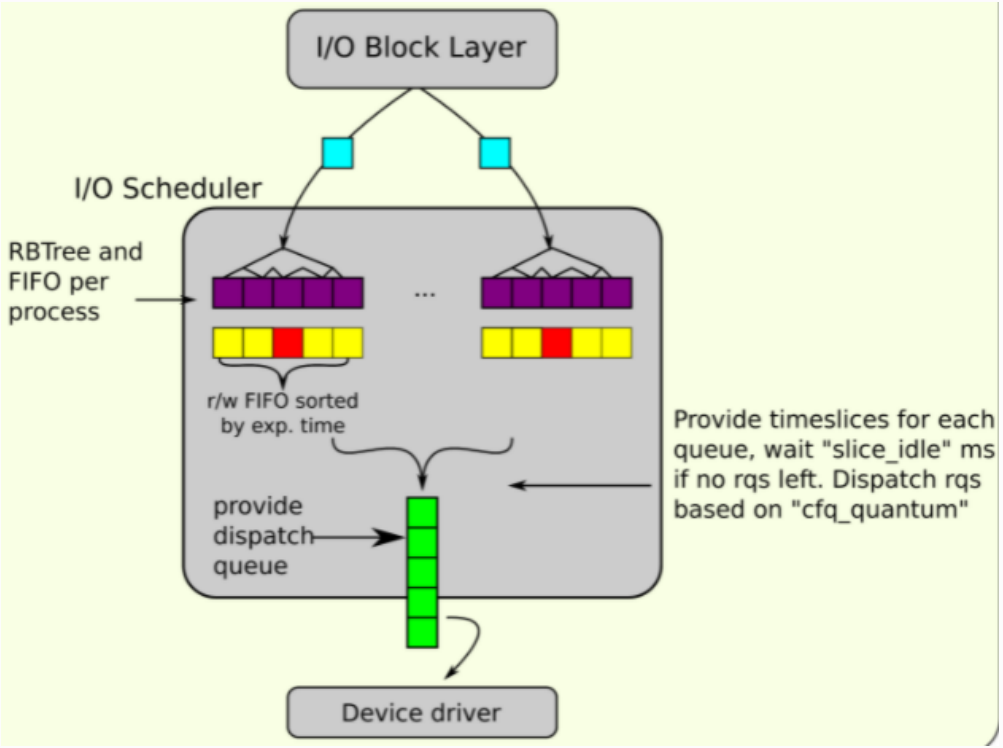
E1、Linux中的IO调度 - CFQ

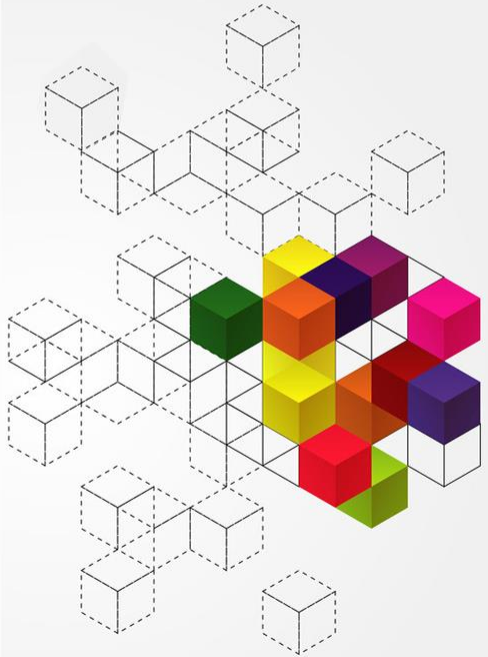
CFQ调度：





E1、Linux中的IO调度 - CFQ





核心思想：after ordering the queues to reduce disk seeking, it services these per-process I/O queues in a round-robin fashion.