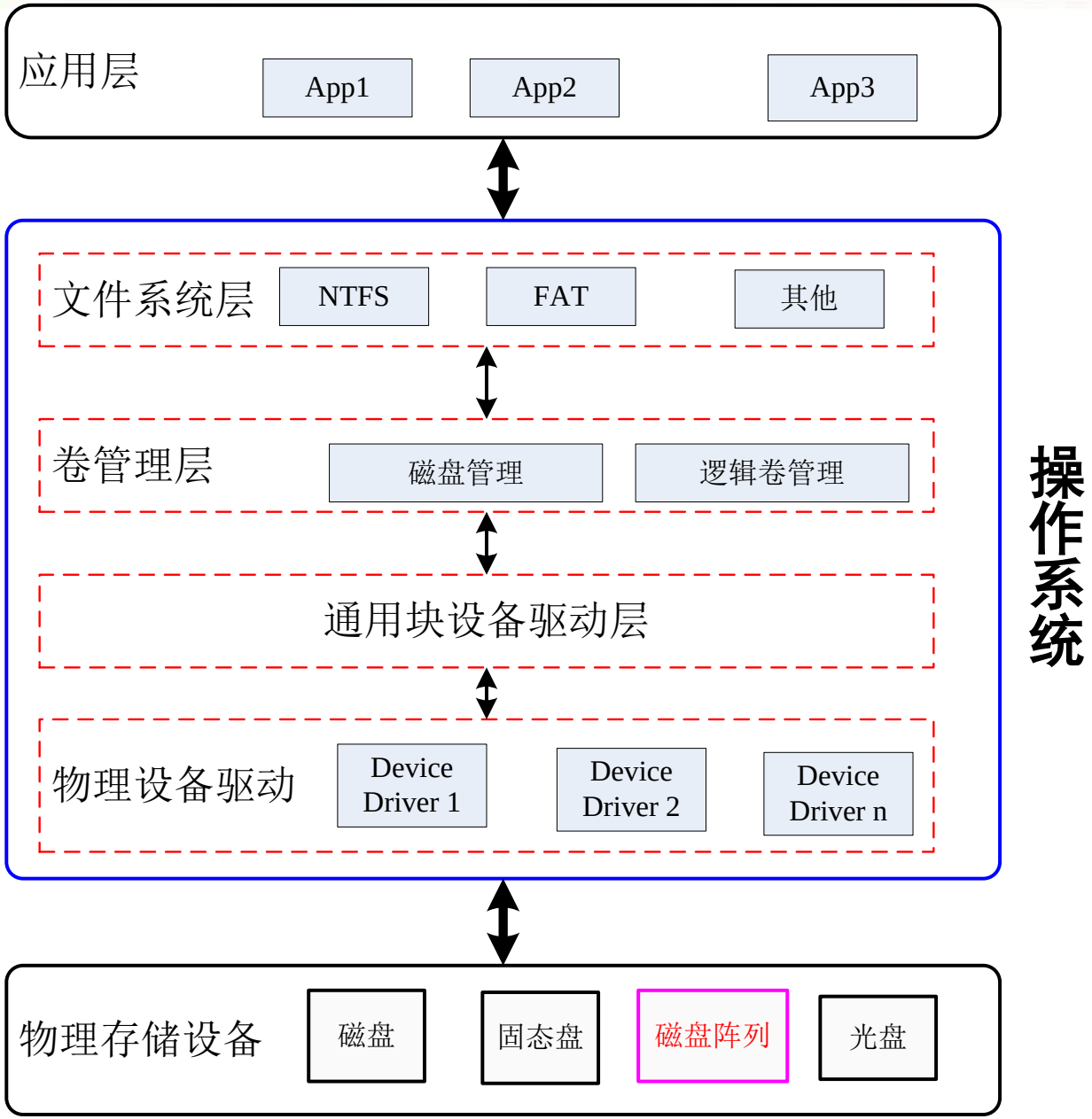
RAID技术原理

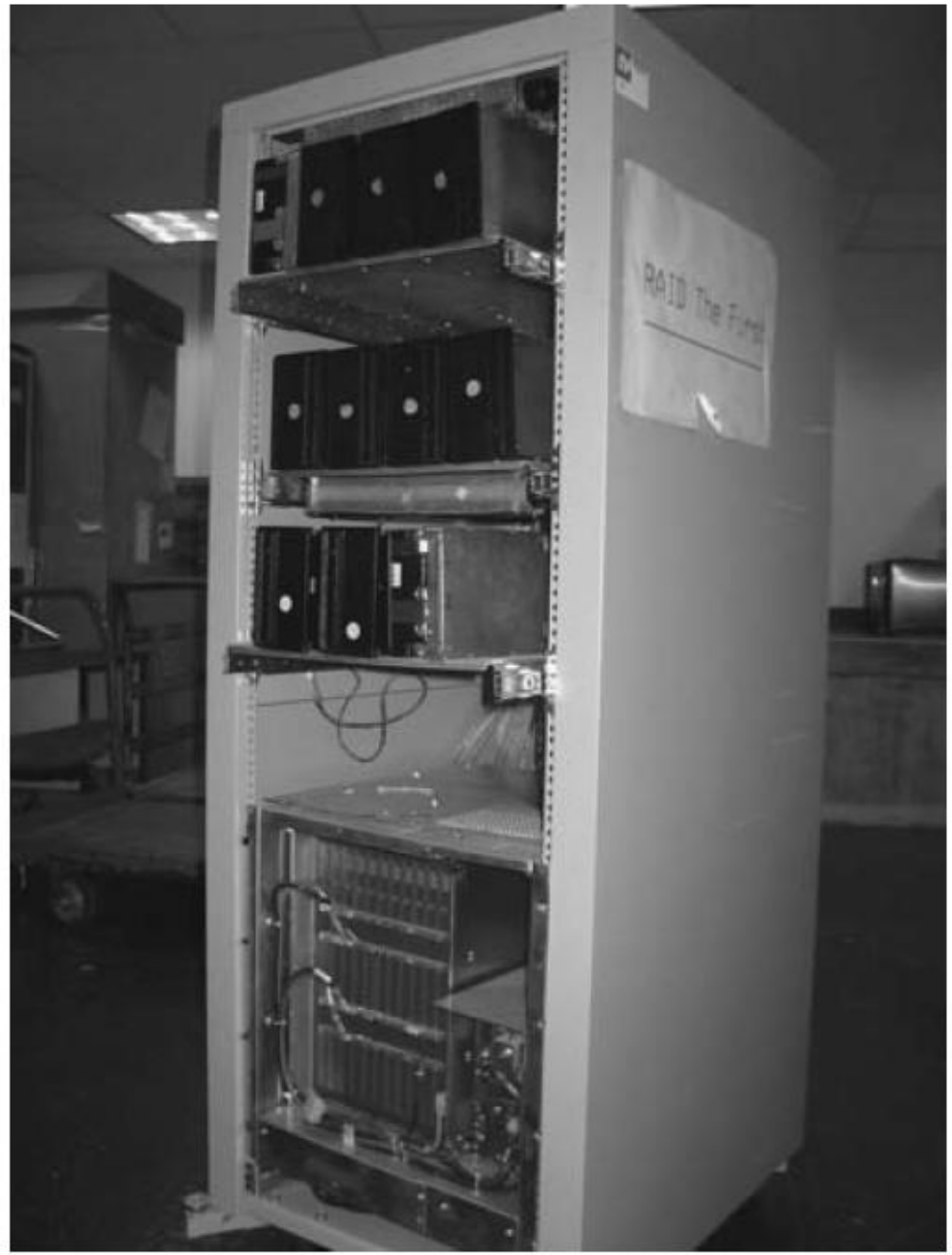
传统计算机存储


应用层
App1
App2
App3
文件系统层
NTFS
FAT
其他
卷管理层
磁盘管理
逻辑卷管理
通用块设备驱动层
物理设备驱动
Device Driver 1
Device Driver 2
Device Driver n
操作系统
物理存储设备
磁盘
固态盘
磁盘阵列
光盘


Storage


09-09-07 16:39

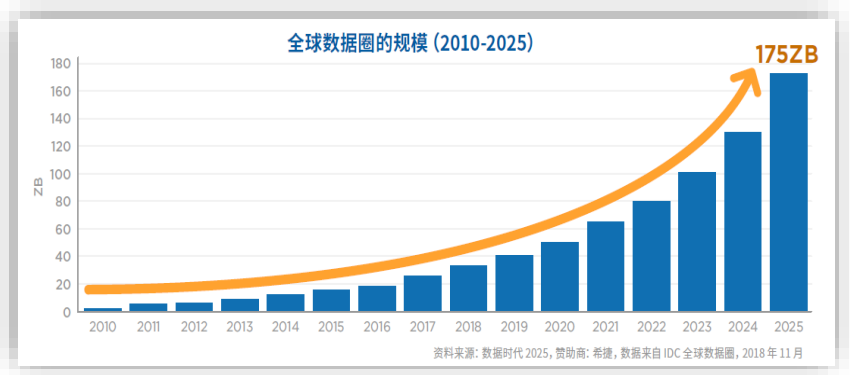
09-09-07 16:42


RAID The First

Why RAID？

- **存储容量需求不断增加：**

  数据总量18个月翻一番，2028年将达到393 ZB



- **存储数据可靠性要求越来越高：**

  24×7的持续服务，EB级数据中心平均

  每天约20块盘故障

数据中心硬盘年故障率统计 2022

品牌	硬盘型号	数量	年故障率
日立	HUH721212ALE604	13165	0.56%
希捷	ST4000DM000	18246	3.45%
东芝	MG07ACA14TA	38182	1.01%
…	…	…	…
平均年故障率：**1.37%**			

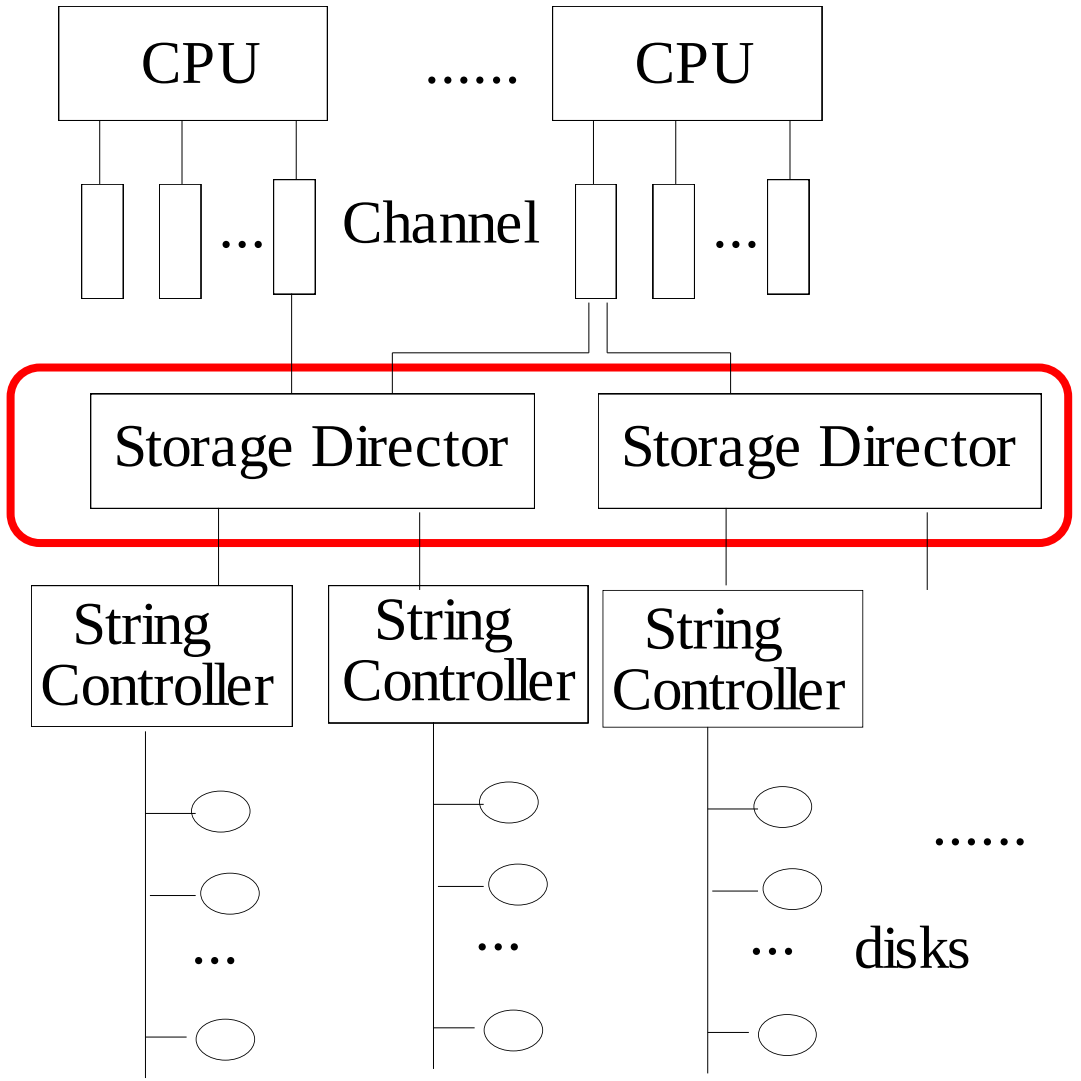
数据来源：BACKBLAZE（2023年1月）

- **存取速度要求越来越快：**

  速度要求达TBps

RAID起源

RAID设计借用大型机中Striping和Interleaving的概念



IBM大型机多路磁盘I/O系统

RAID(独立冗余盘阵列)

- Redundant Arrays of Independent (Inexpensive) Disks
  - **D.A. Patterson**, **G. Gibson**, and R.H. Katz, "A Case for Redundant Arrays of Inexpensive Disks(RAID)", 1988.

A case for redundant arrays of inexpensive disks (RAID)

DA Patterson, G Gibson, RH Katz - Proceedings of the 1988 ACM ..., 1988 - dl.acm.org

... use of extra disks containing redundant information to recover the original information when a disk fails. Our acronym for these Redundant Arrays of Inexpensive Disks is RAID. To ...

☆ 保存 99 引用 被引用次数: 4737 相关文章 所有 123 个版本 »

RAID定义

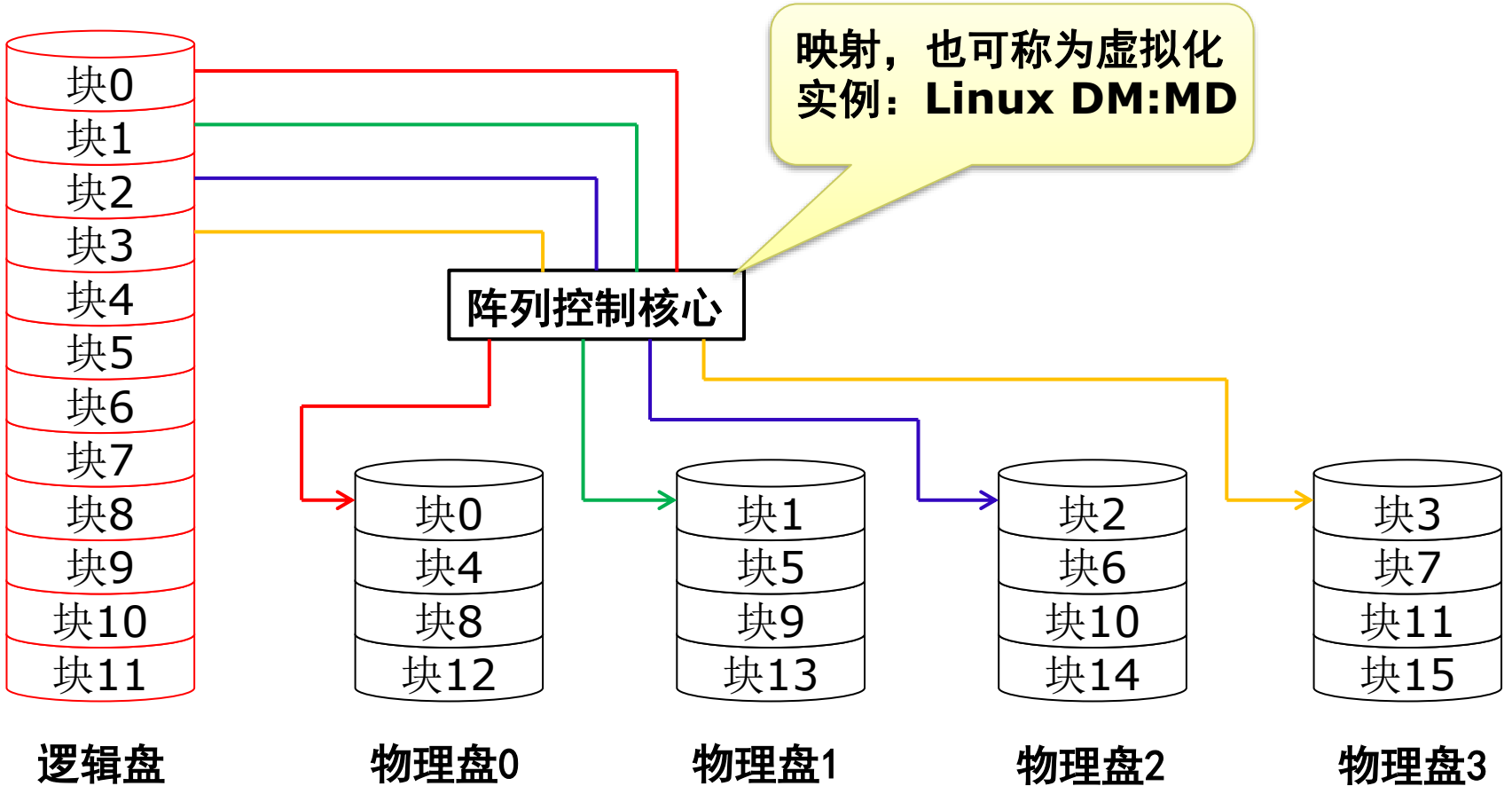
- 盘阵列：

  把多块独立的存储盘按某种方式组织起来形成一个或多个逻辑盘，从而提供比单个存储盘更高性能和更高可靠性的存储技术。

  - 条带化（Striping）
  - 镜像（Mirroring）
  - 奇偶校验（Parity）

RAID：构成一个逻辑盘

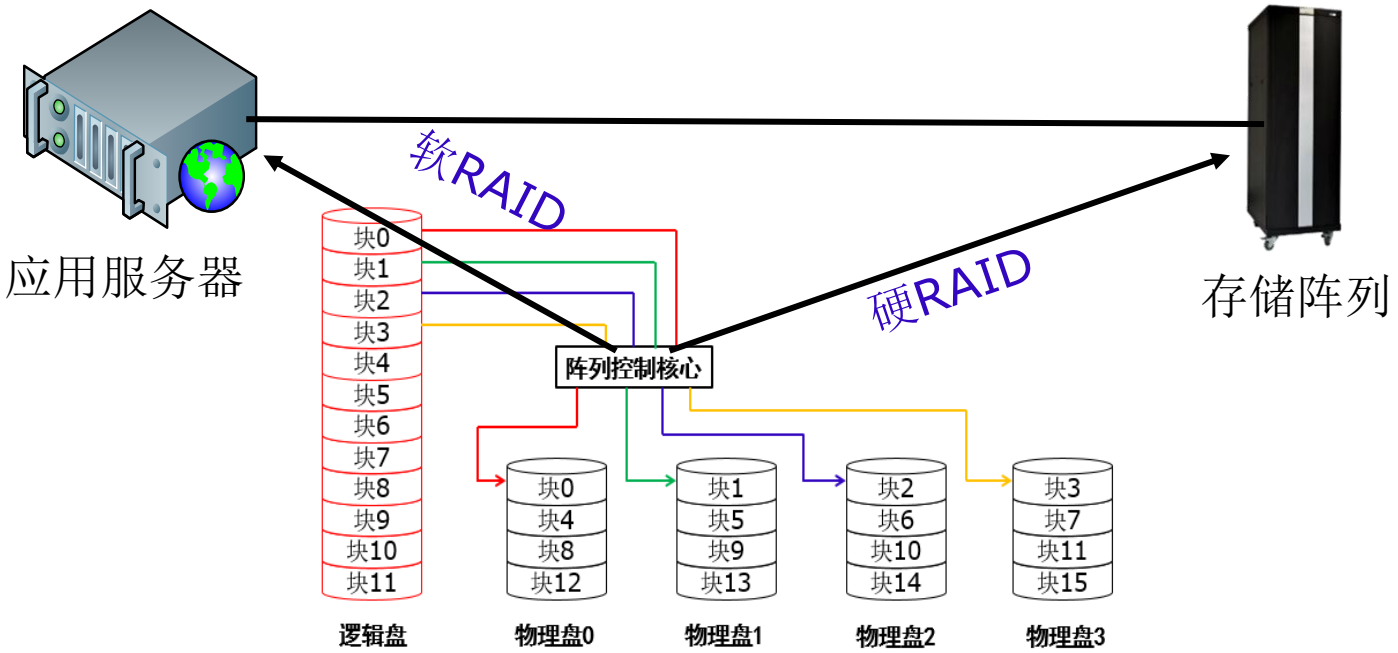
把多块独立的盘按某种方式组织起来形成一个盘组，且在操作系统下视为一个逻辑盘的存储设备



RAID分类

软件式与硬件式(Software RAID & Hardware RAID)

区分的标准是看实现RAID是否占用主机的资源（CPU，Memory）



RAID基本原理与特征

- 数据以条块化（striping）分布于多个磁盘
- 存储容量扩展，I/O性能提升
- 冗余机制获得较高的数据可用性
  - 可用性（Availability）：即使某些部件故障仍能够为用户提供服务
- 通过冗余信息实现数据恢复（Reconstructed & Rebuild)

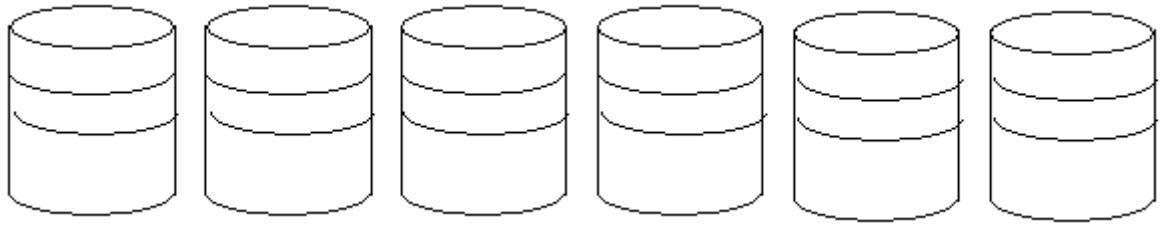
- 不足之处：
  - ⇒ 容量损失：存储冗余信息
  - ⇒ 带宽损失：冗余信息的读写
  - ⇒ 计算资源损失：冗余信息的更新，恢复

RAID 级别（Level）

- RAID 0：数据分割，无容错能力
- RAID 1：镜像(双拷贝)
- RAID 2：海明码，不具有商业生命力
- RAID 3：并行，位交叉，单校验盘
- RAID 4：并行，块交叉，单校验盘
- RAID 5：独立，块交叉，循环校验盘
- RAID 6：容双盘错，块交叉，近年被广泛重视

RAID 0

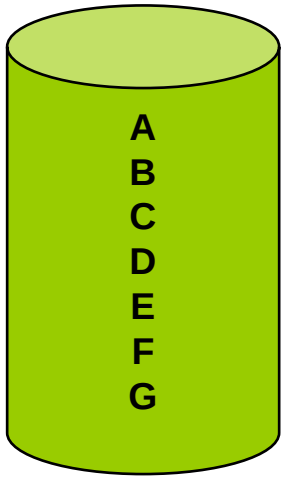
- 多个盘构成阵列，数据分块轮流存储，提高容量
- 解决单盘无法并行工作的问题，多盘同时操作，提高性能



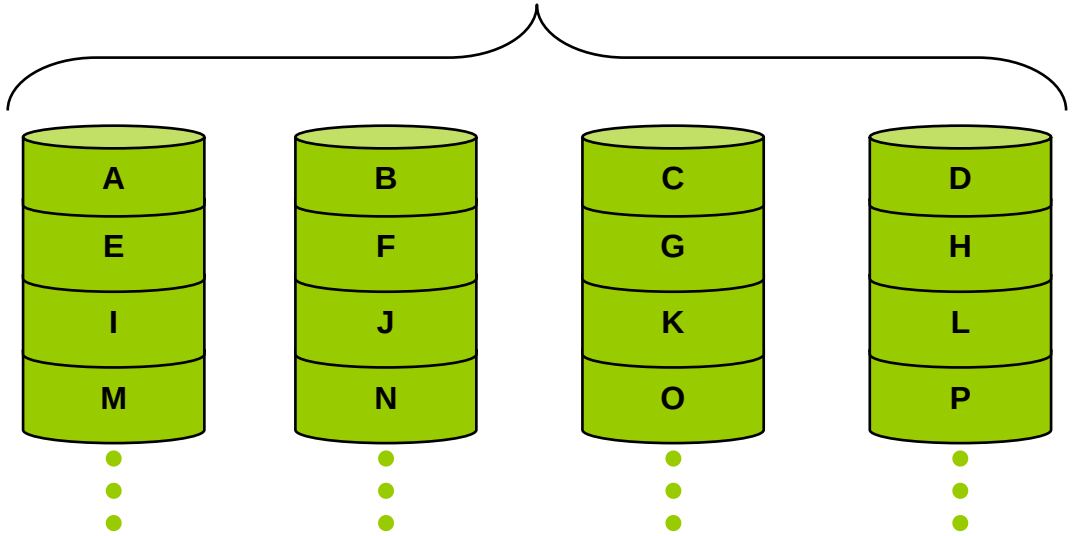
RAID 0：数据分块，无冗余校验

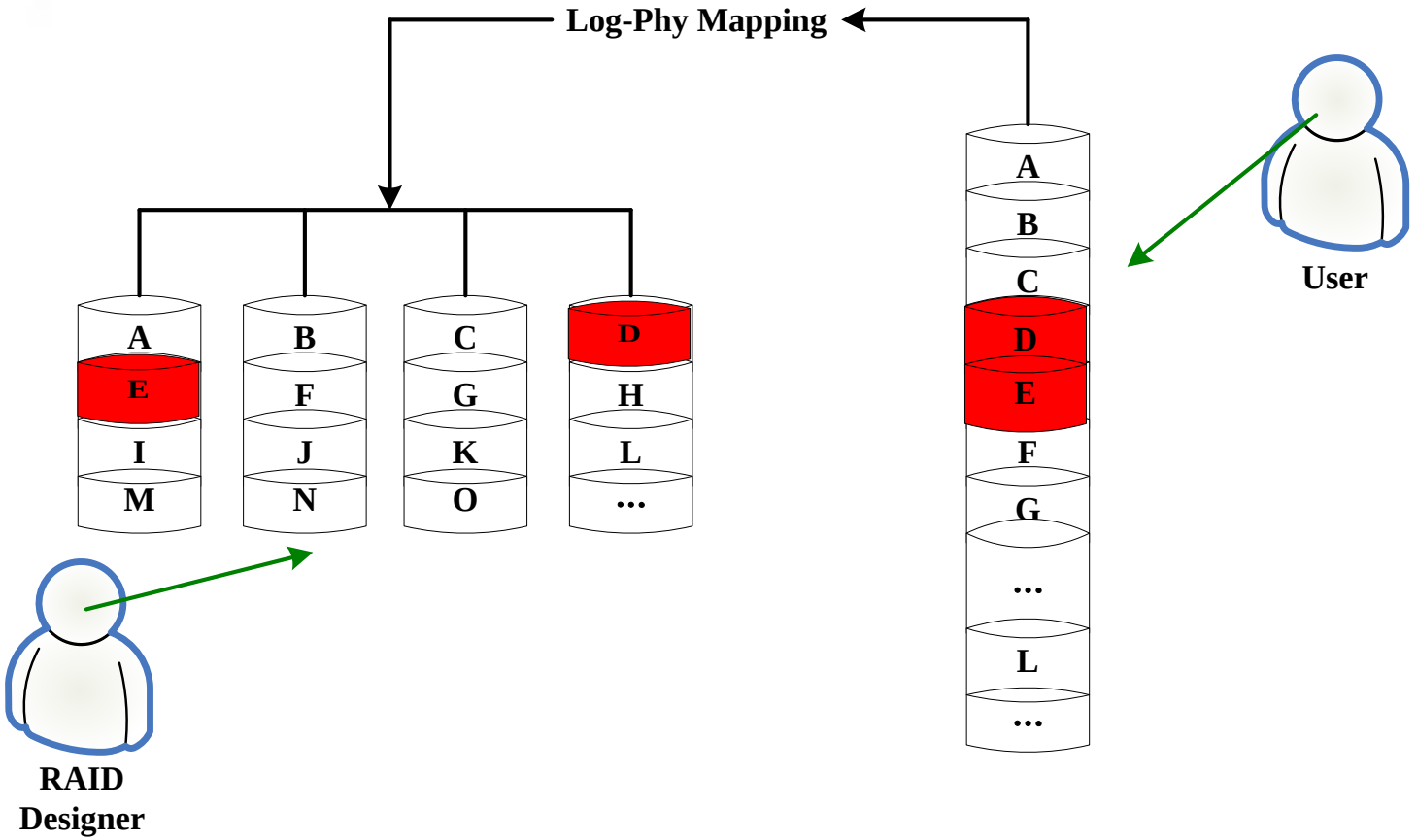
RAID 0 Striping

RAID array



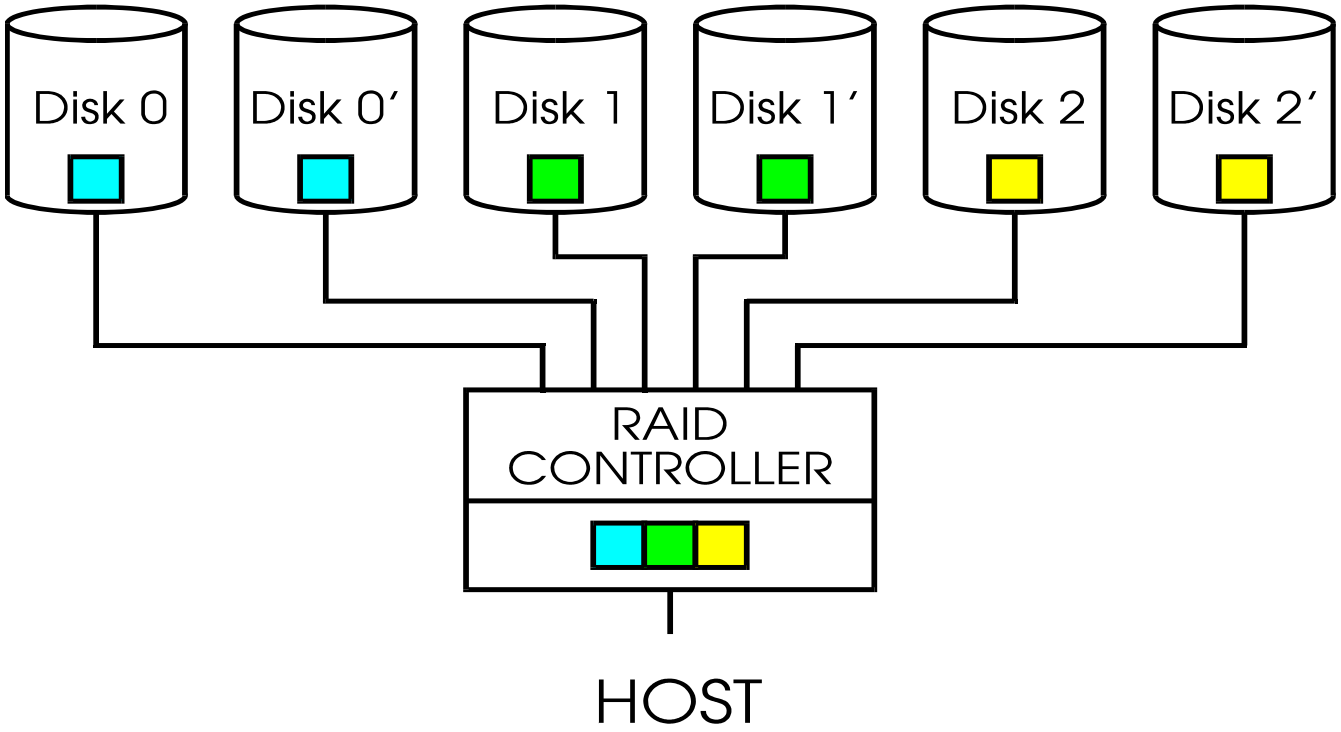
Data striped across member disks





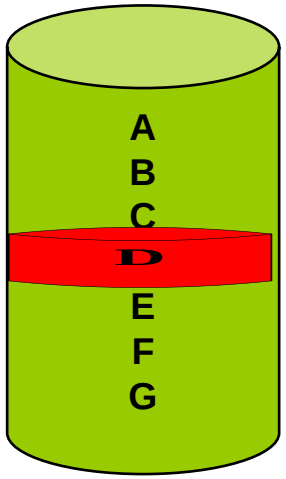
存在的问题：可靠性降低，无容错能力。由N个盘构成的盘阵的故障率为单盘的N倍

RAID 1

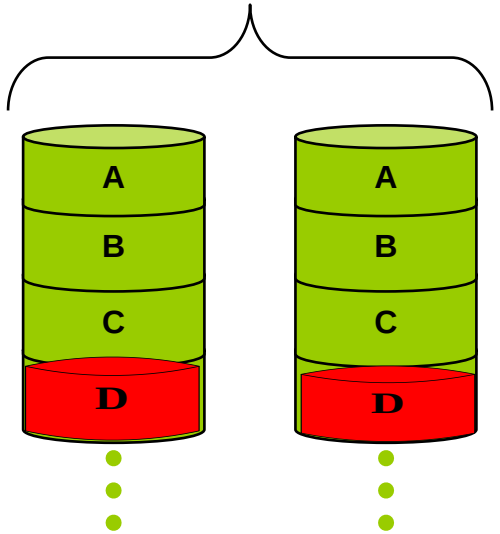


RAID 1 Mirroring

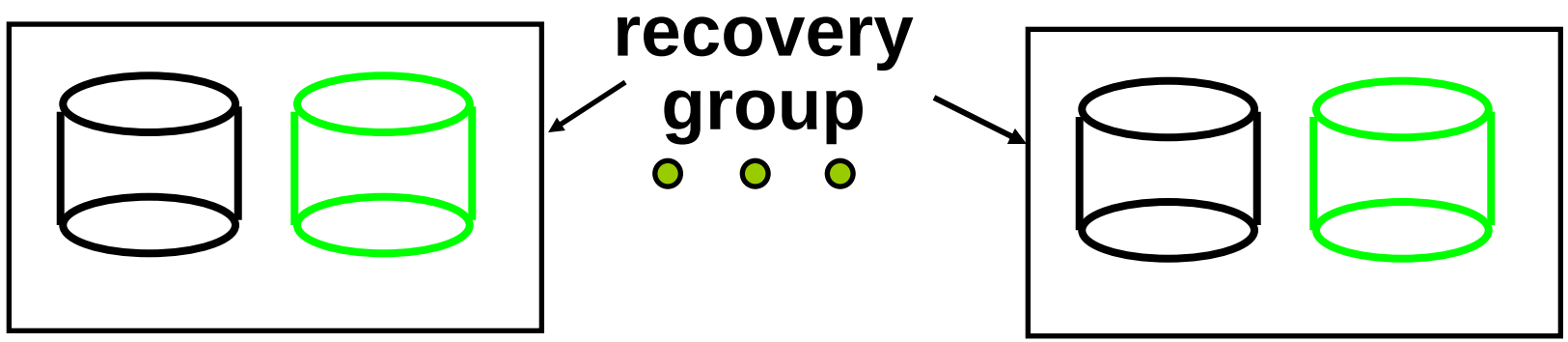
RAID array



Data mirrored across member disks



RAID 1: Disk Mirroring

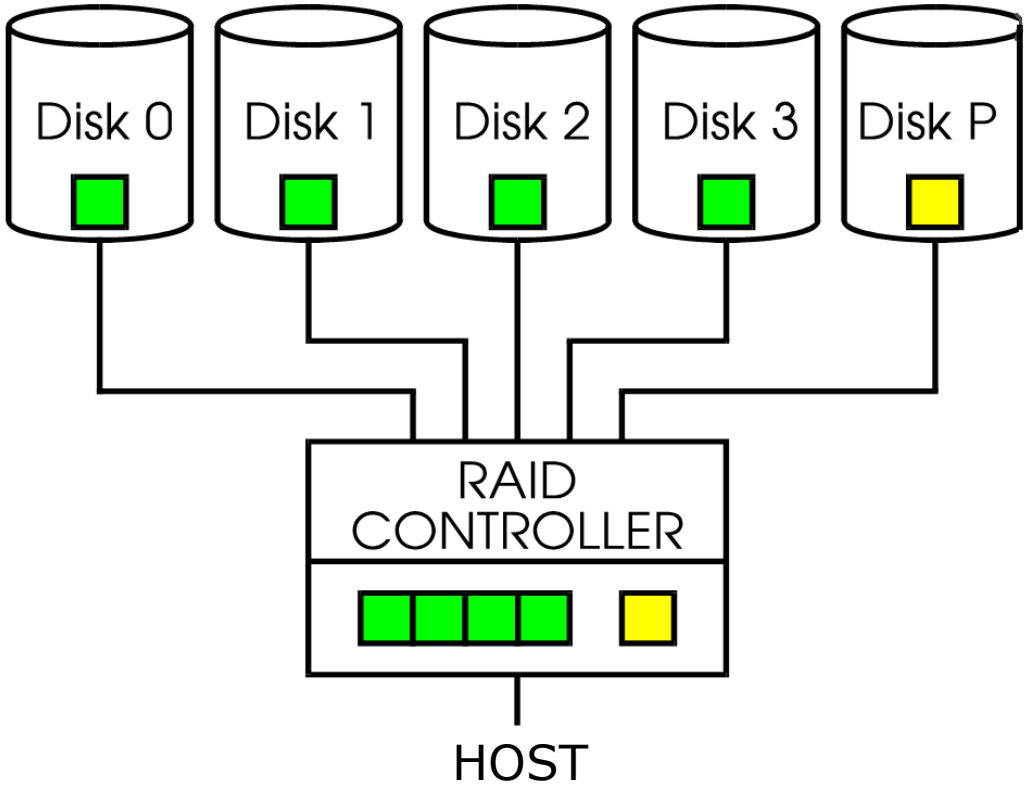


- 每个盘上的数据均在镜像盘上有一个完整的复制副本；
  高可用性
- 写入带宽缩减:

  Logical write = two physical writes

  Reads may be optimized
- 代价较高的解决方案：一半的容量损失

RAID 3、4



奇偶校验

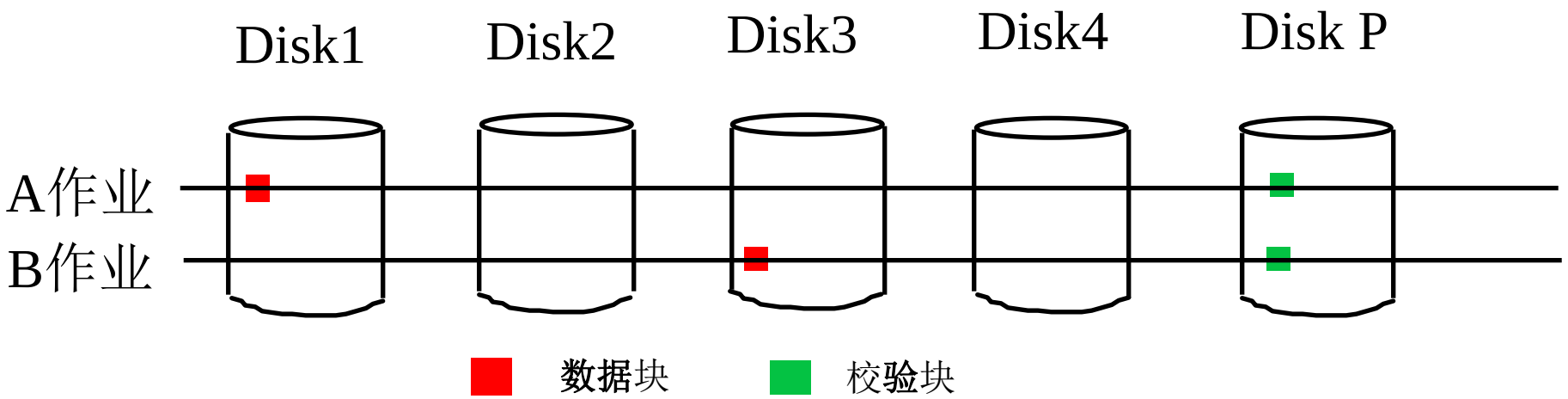
区别：

RAID 3： 位交叉，细粒度

RAID 4： 块交叉，粗粒度

适用于大组，大文件顺序传送

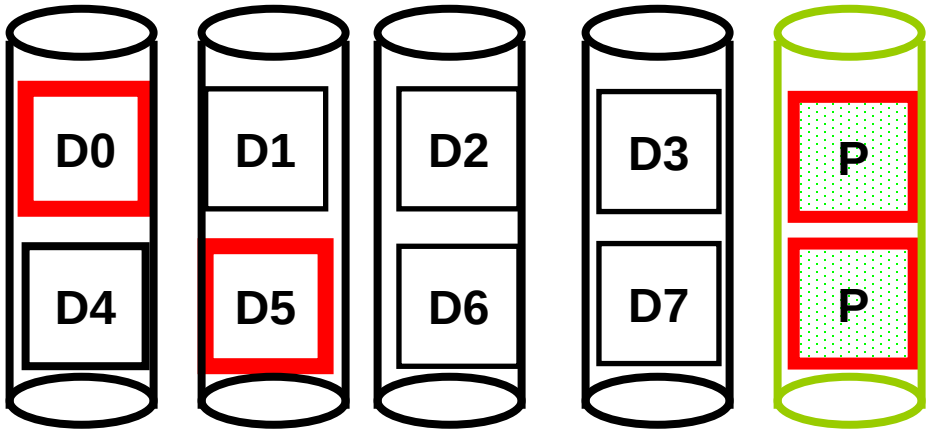
RAID 4 校验瓶颈问题



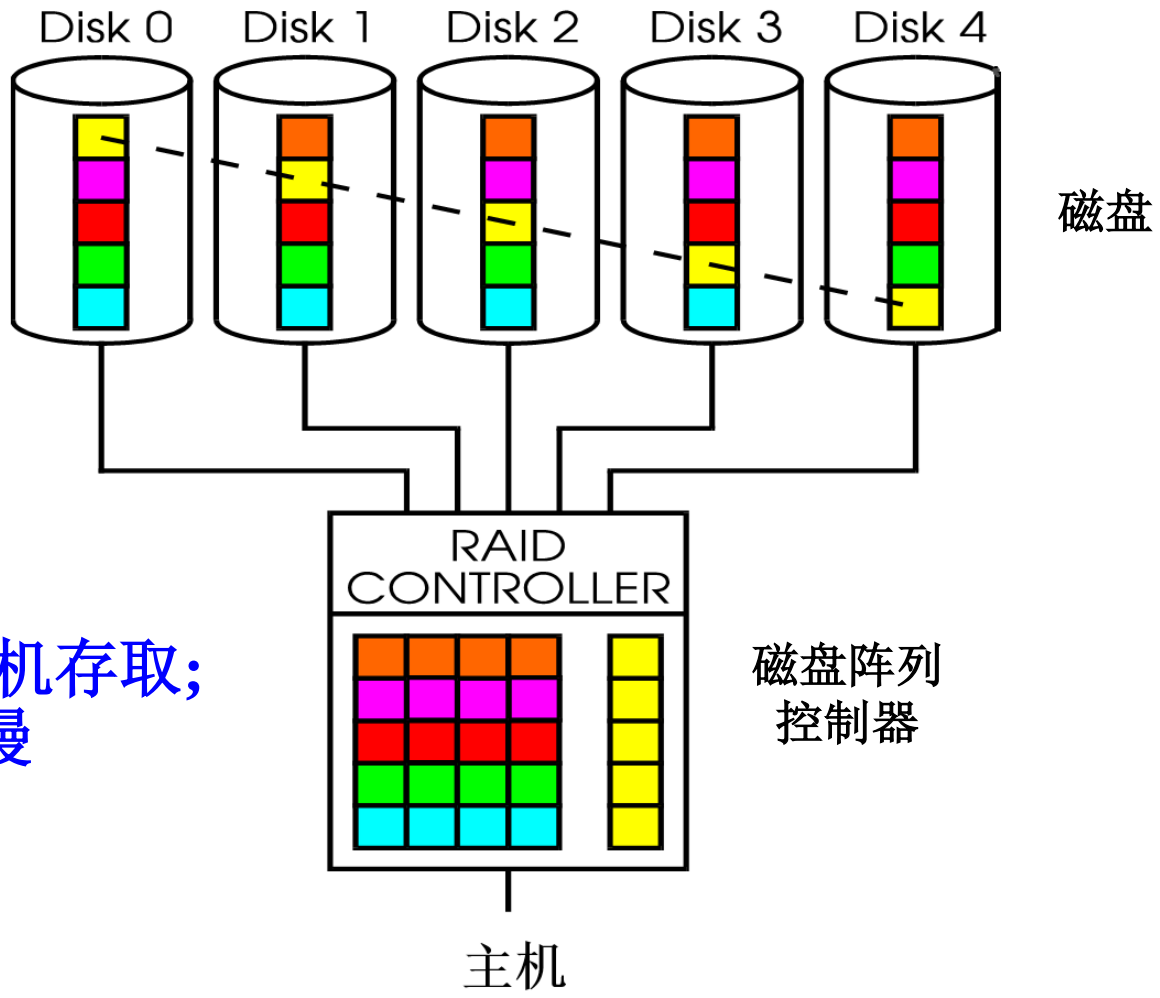
在同一时刻DISK P只能有一个I/O操作，因此作业 B 只能在作业A完成后才能开始

RAID4改进→RAID 5

- RAID 4 对读比较有效
- 并发写操作受限于校验盘:
  Write to D0, D5, both also write to P disk



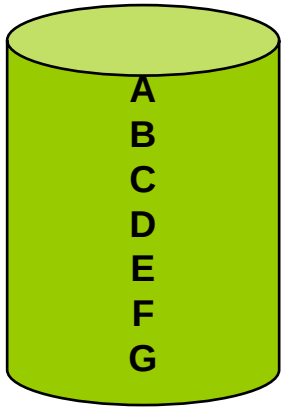
RAID 5



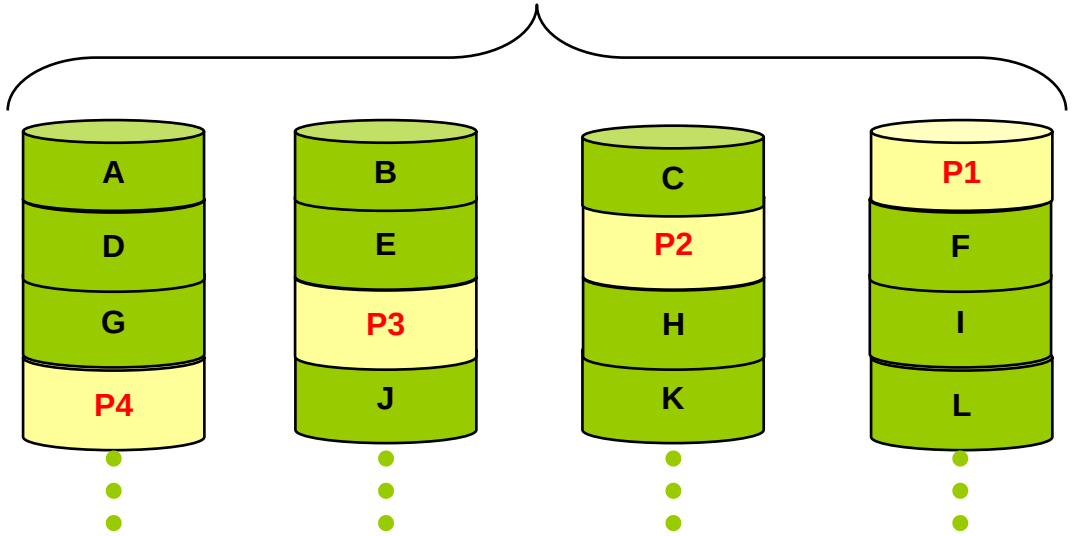
适用于小组，随机存取；
写速度缓慢

RAID 5: Rotate Parity

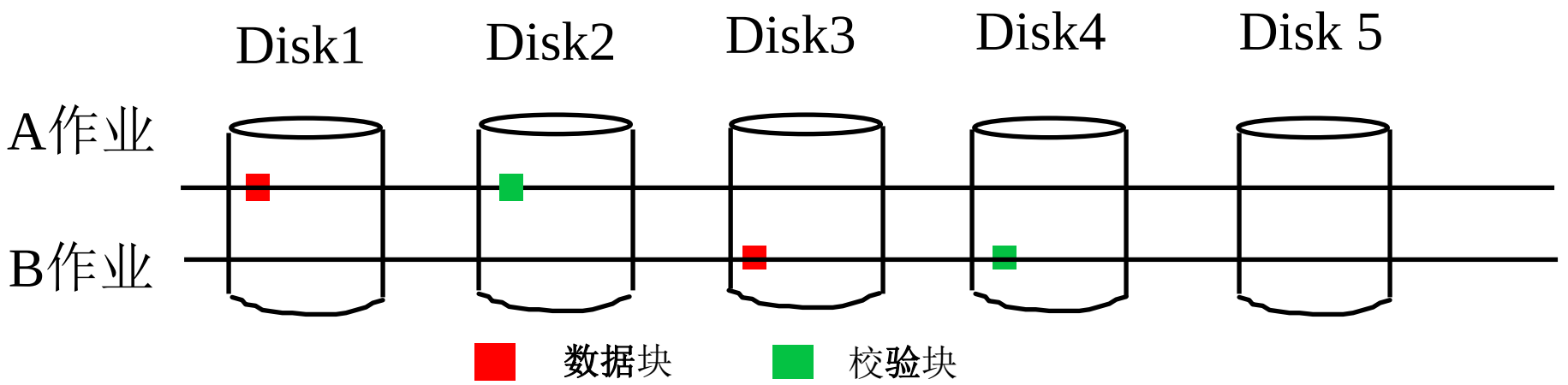
RAID array



Data striped across disks, with parity rotating



RAID 5



在同一时刻可以同时完成作业A及作业B

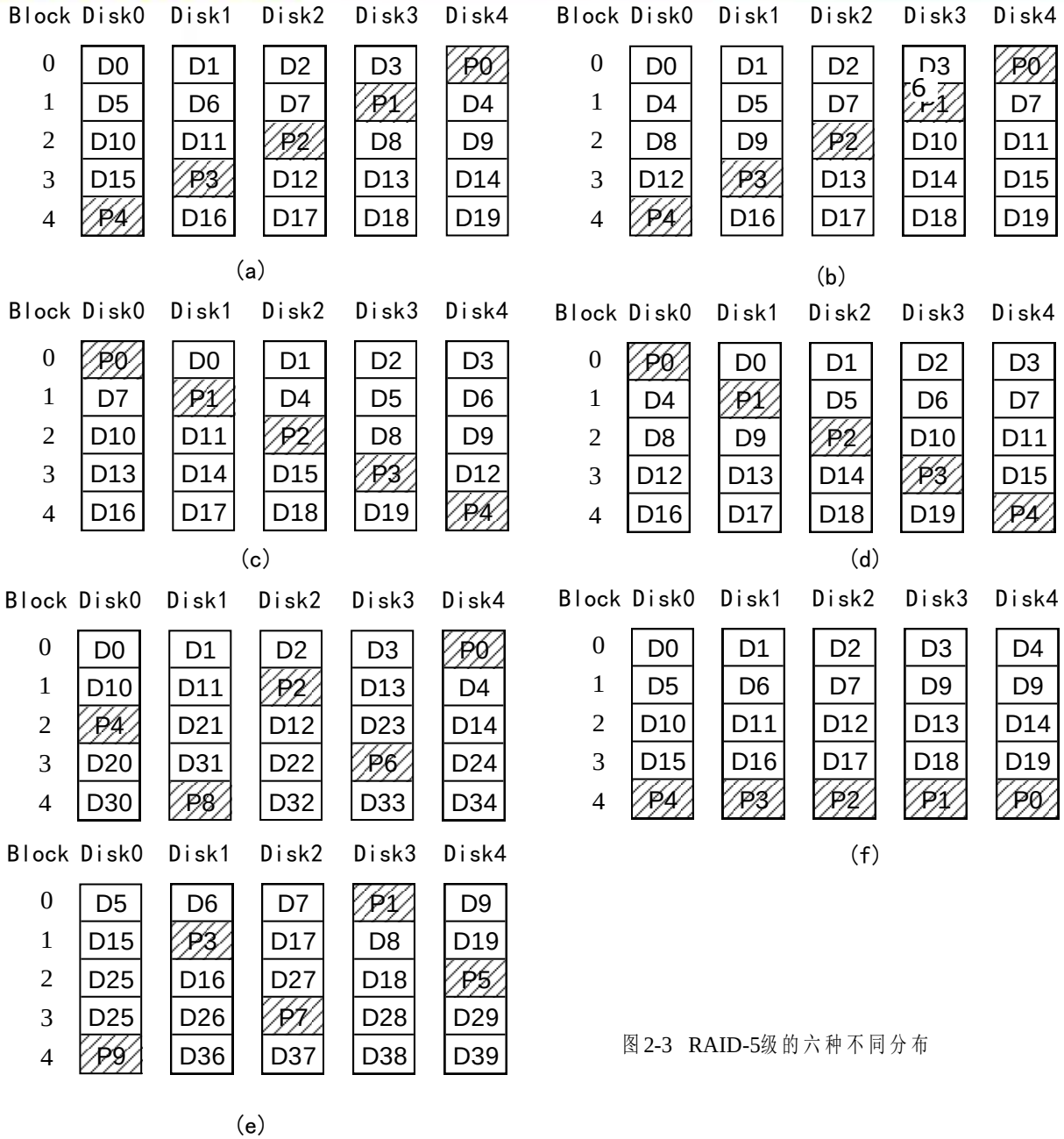
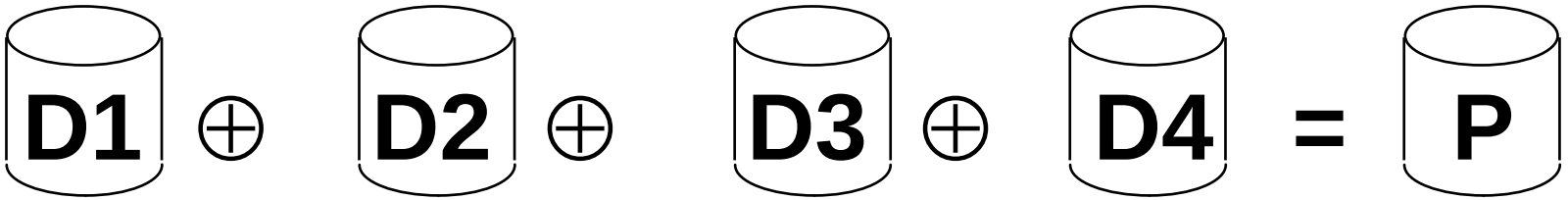


图2-3 RAID-5级的六种不同分布

a左对称
b左不对称
c右对称
d右不对称
e扩展左对称
f平面左对称

RAID 4、5 操作的例子：写数据D1新值



方法1：RCW

读取D2、D3、D4

计算新的校验码

$P_{新值} = D1_{新值} \oplus D2_{旧值} \oplus D3_{旧值} \oplus D3_{旧值}$

写数据$D1_{新值}$和检验码$P_{新值}$

共需5次I/O

方法2：RMW

(1) 读数据$D1_{旧值}$和校验码$P_{旧值}$

(2)计算新校验码

$$P_{新值} = D1_{旧值} \oplus P_{旧值} \oplus D1_{新值}$$

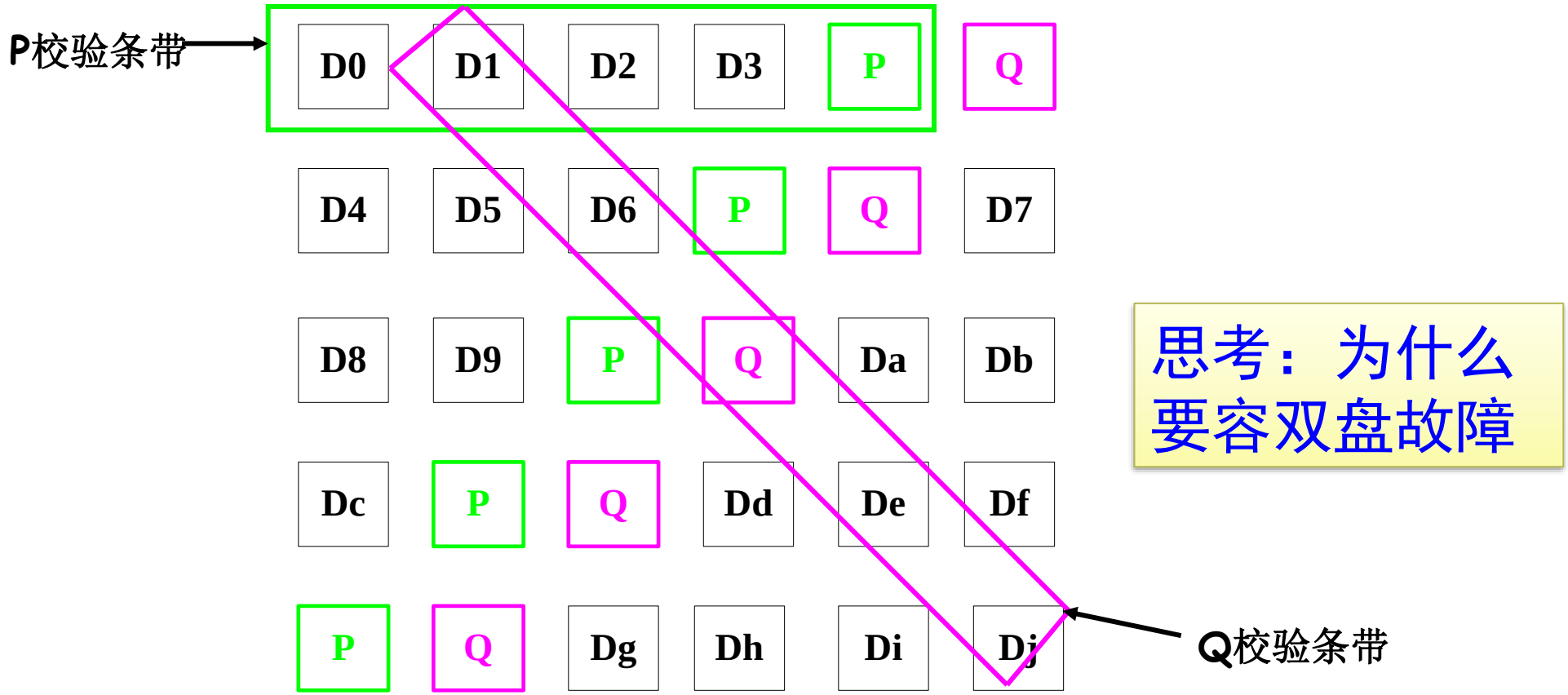
(3) 写数据$D1_{新值}$ 和校验码$P_{新值}$

共需4次I/O

原理?

RAID 6：P、Q校验

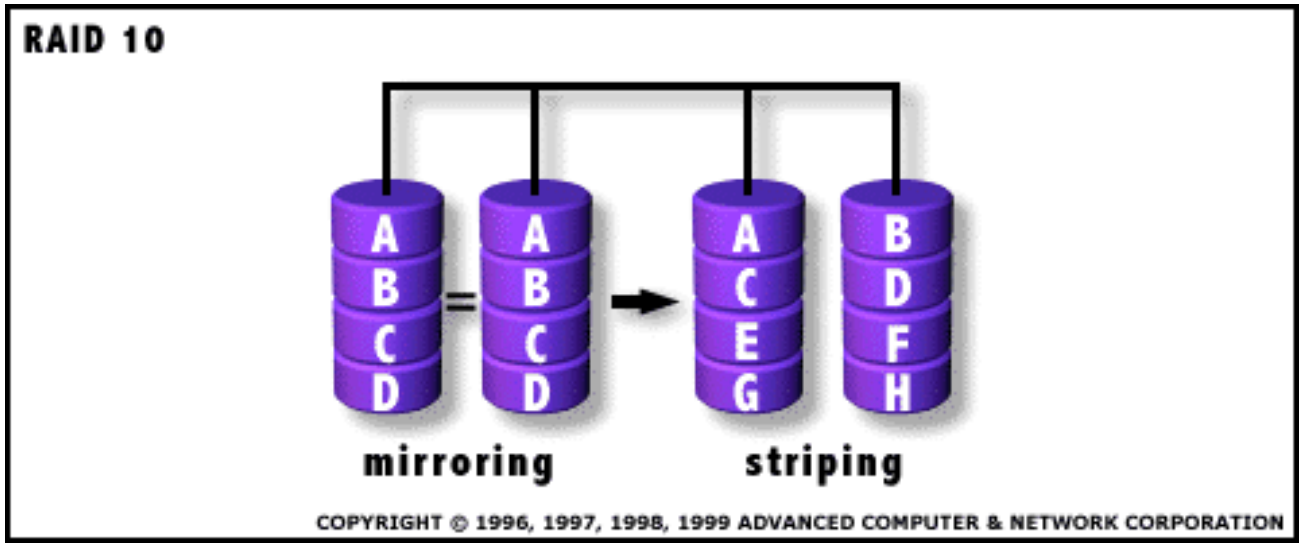
容双盘错



思考：为什么要容双盘故障

组合级别:

RAID10，RAID50...

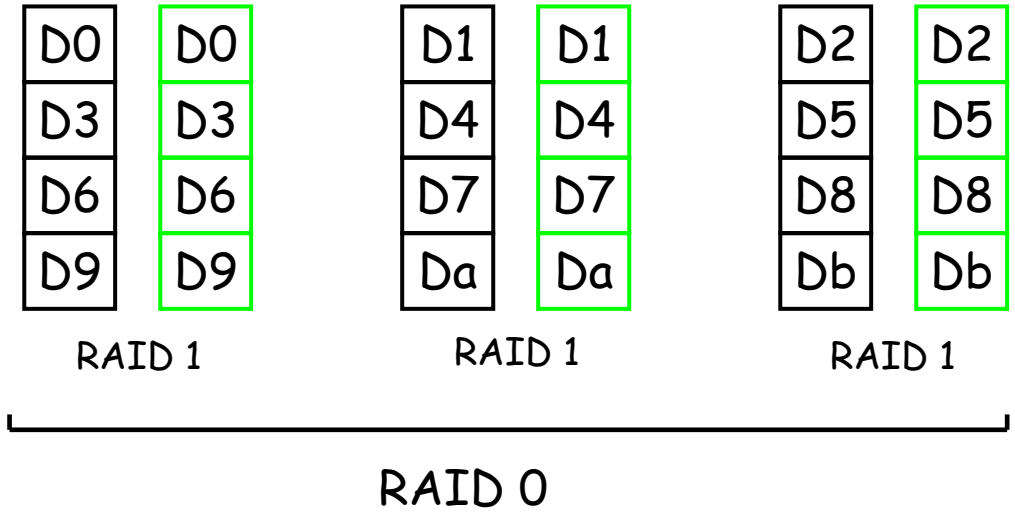


具有**RAID1**的高可用性

较高的读取性能

写性能缩减

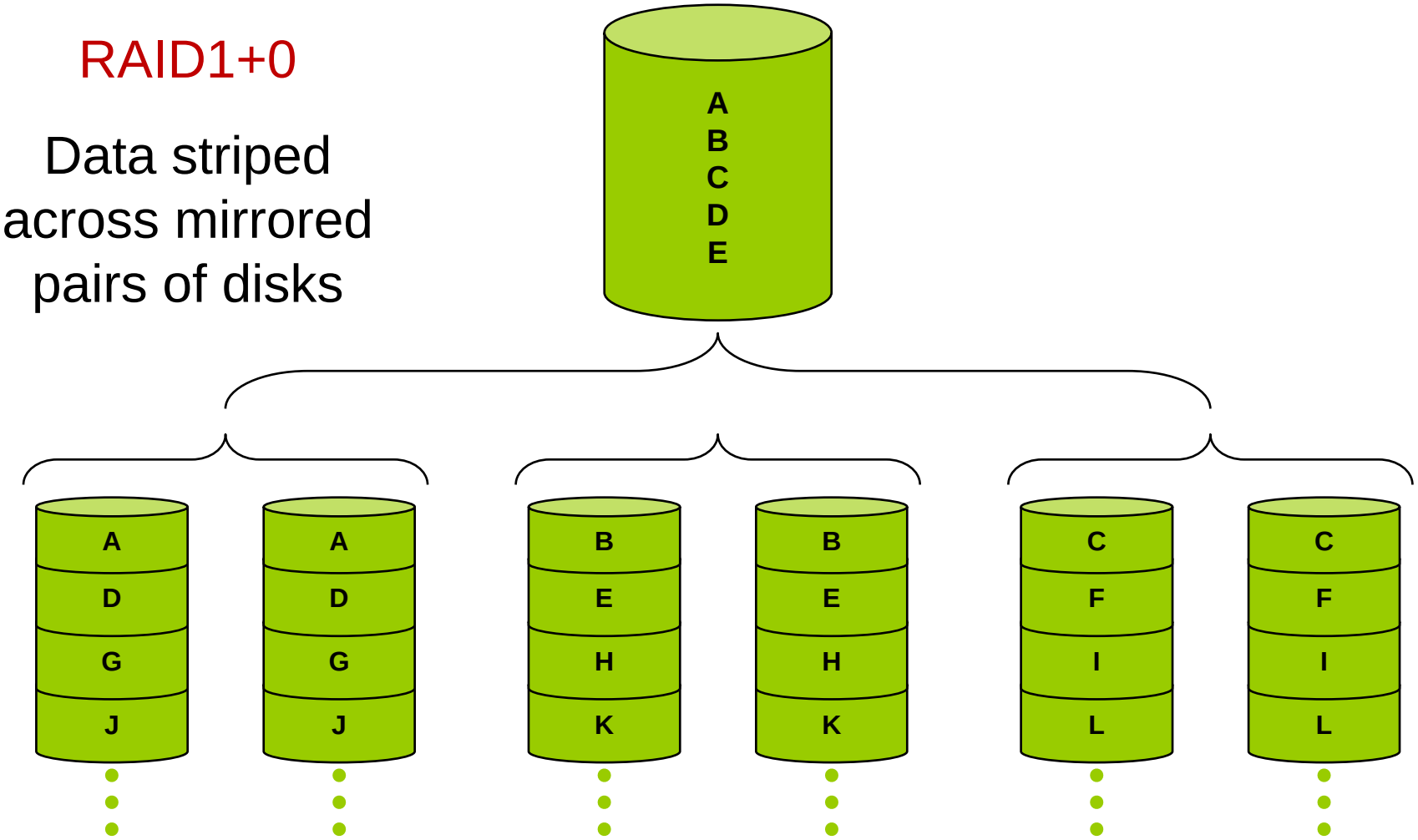
冗余高，代价大



RAID 10: Striping & Mirroring

RAID1+0

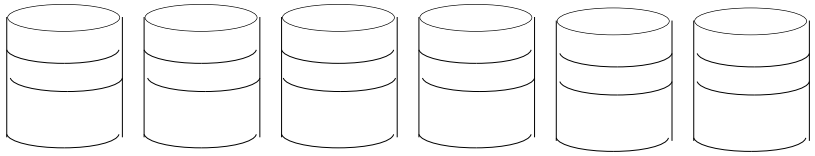
Data striped across mirrored pairs of disks



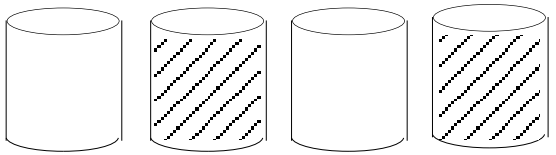
RAID 0＋1 :
data mirrored between 2 groups of striped disks

RAID级别小结

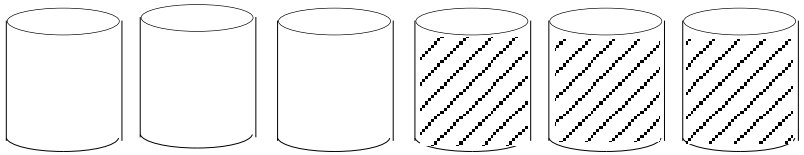
级别无高低贵贱之分



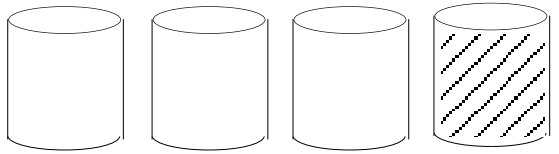
(a) RAID 0： 数据分块，无校验



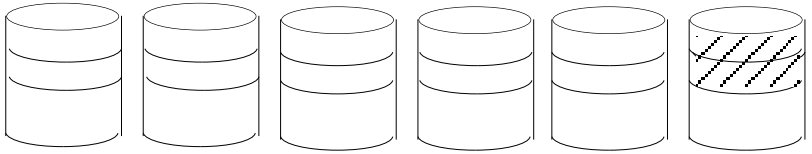
(b) RAID 1： 镜像



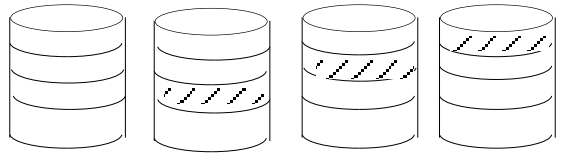
(c) RAID 2： 位交叉，海明码纠错



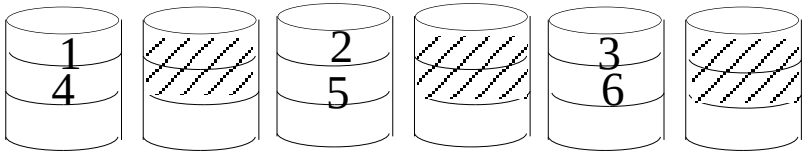
(d) RAID 3： 位交叉，奇偶校验



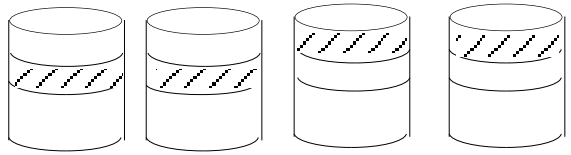
(e) RAID 4：块交叉，固定校验盘



(f) RAID 5： 块交叉，校验信息分散存放



(g) RAID 10：分块与镜像结合



(h) RAID 6：纠双错阵列

适合的才是最好的

RAID技术小结

- RAID技术从0到5很好地解决了阻碍计算机发展的几个重要问题

  1. 通过多个盘构成阵列，解决了容量问题
  2. 通过冗余算法，解决了存储设备的可靠性问题
  3. 通过多盘并行，解决IO性能问题

- 不足：

  存取速度仍是处理器的瓶颈

  1.STRIPING技术并不能有效地提高速度

  分块的大小与磁盘的机械特性有关

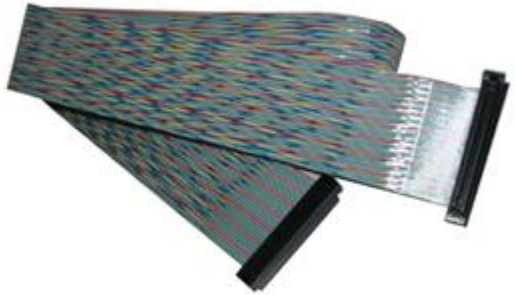
  文件的大小也不同

  2.并行I/O提高速度的能力有限

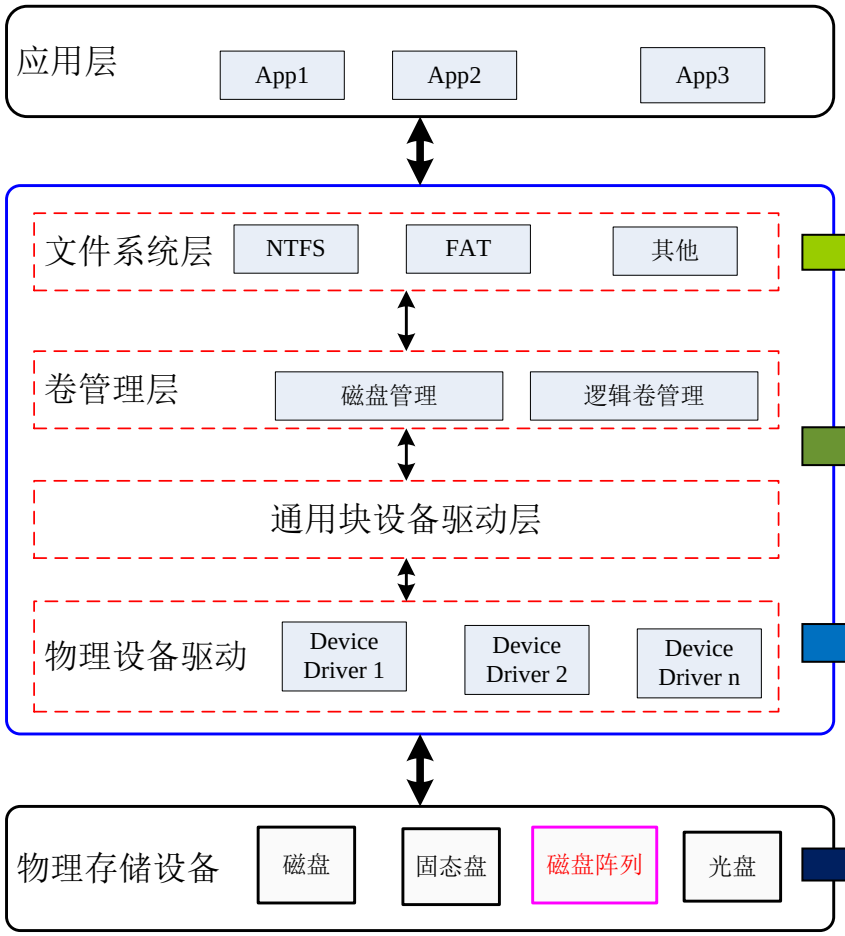
  多次读写后,数据散列在各个盘上, 磁盘的机械运动成为阻碍速度提高的主要因素

并非备份解决方案，无法抵御文件损坏、病毒感染或人为误删等逻辑错误

RAID实现



RAID的实现



RAID是一种存储设备，是一项技术，更是一种思想

利用RAID思想实现分布式FS

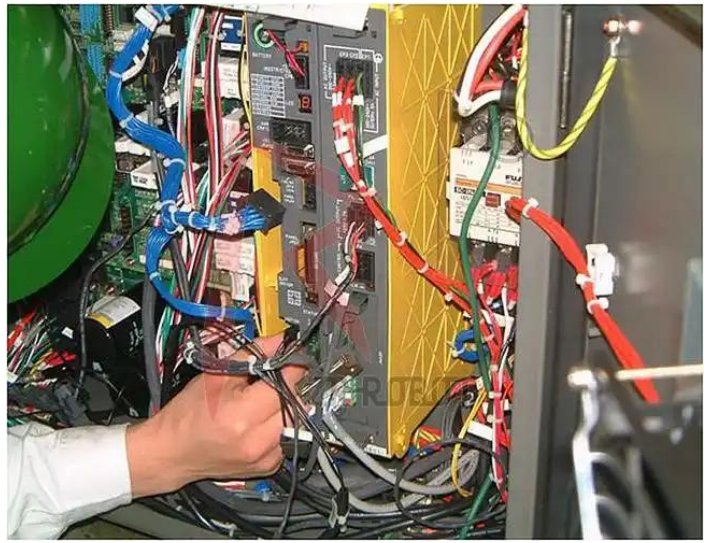
对物理盘透明，实现软RAID

直接管理物理设备，实现软RAID或硬RAID（RAID卡）

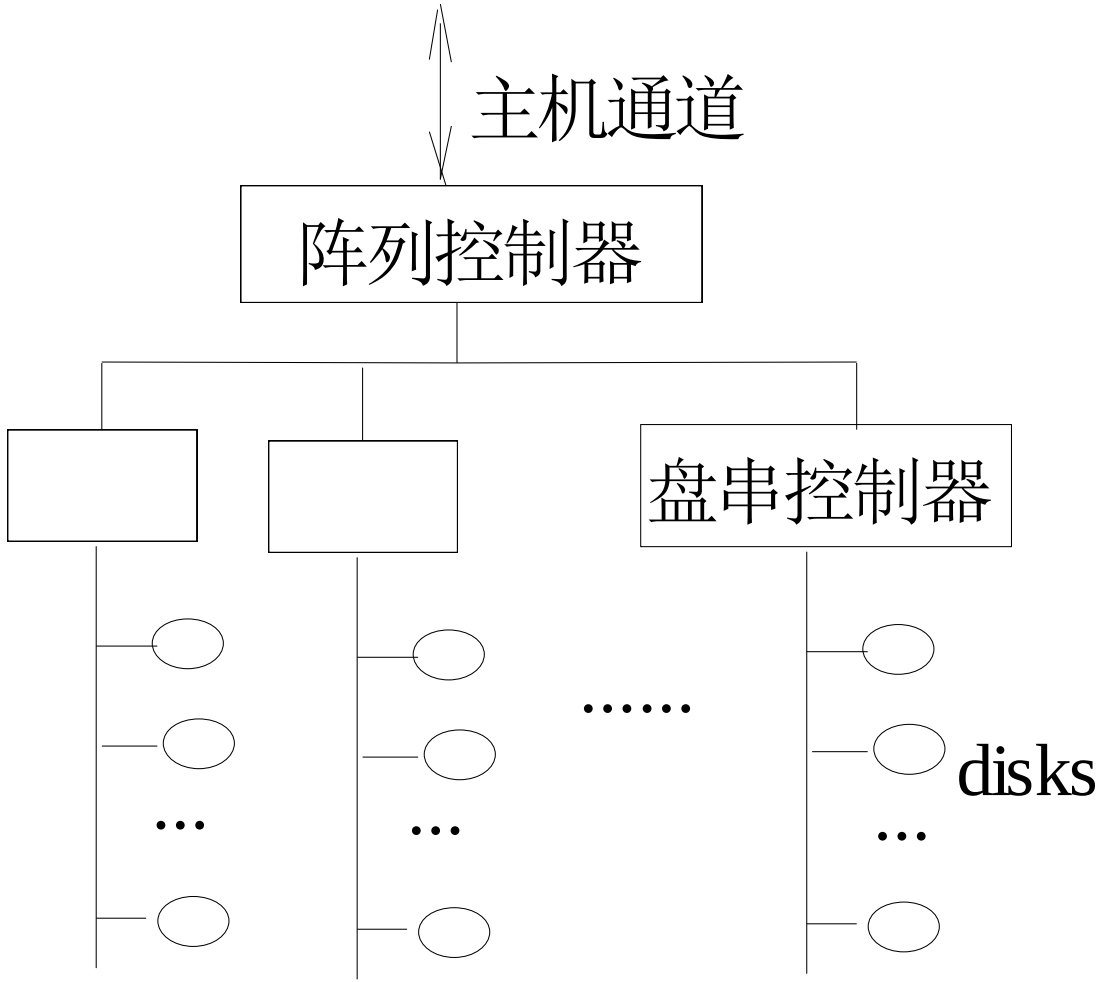
独立式硬RAID

RAID逻辑上构成一个盘设备

RAID逻辑上构成一个盘设备

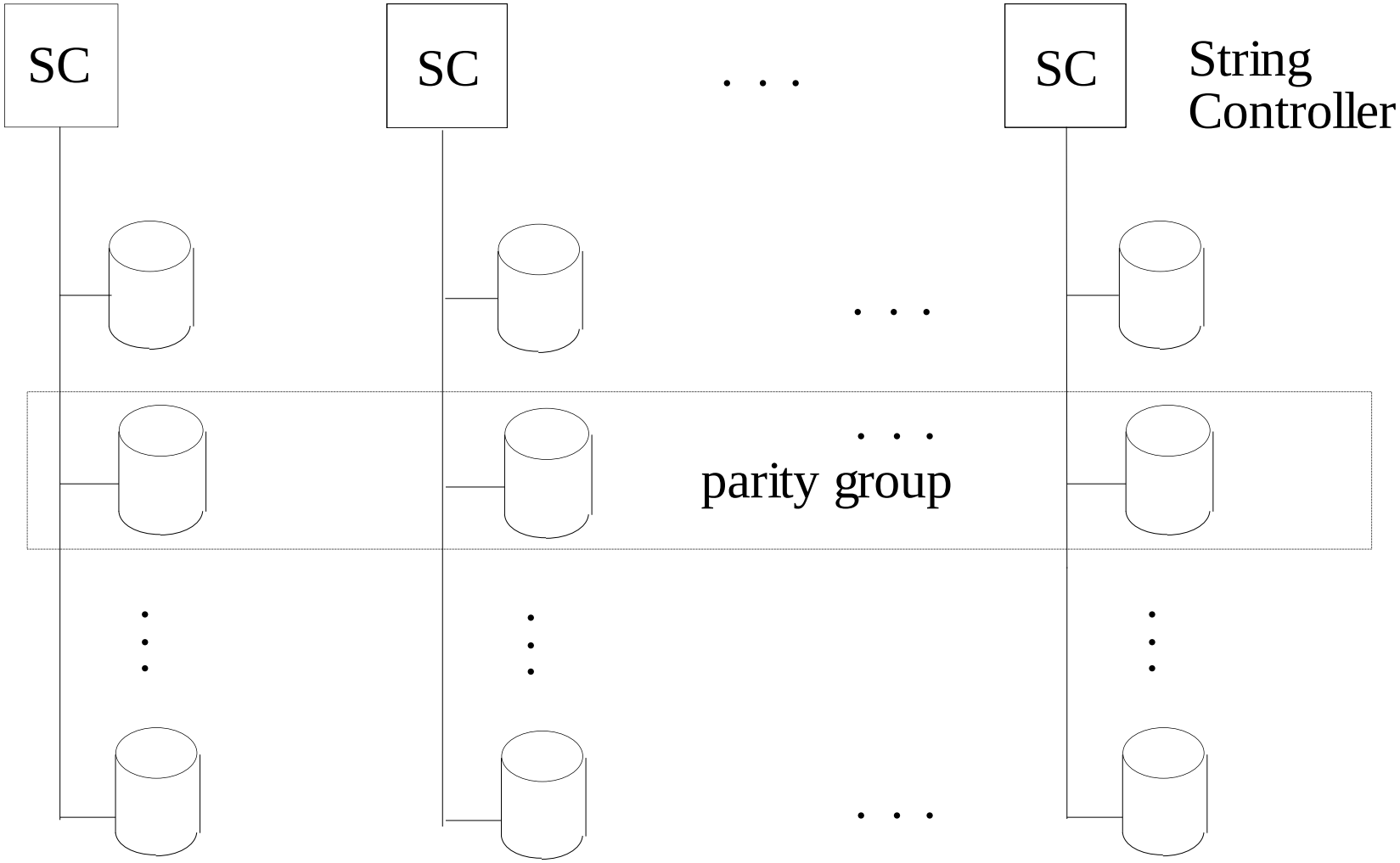


磁盘阵列控制器设计

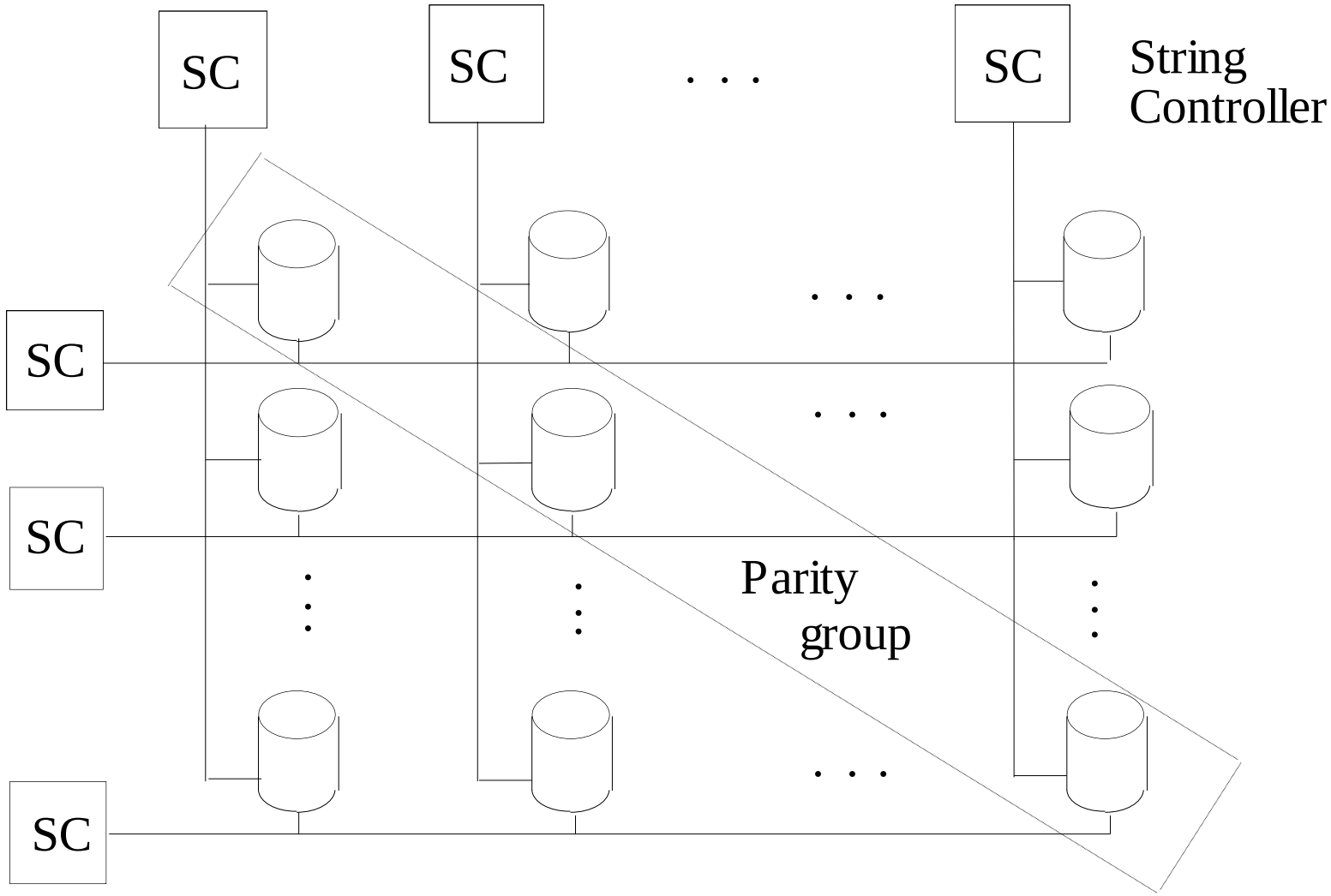


集中控制式阵列

盘阵列正交结构示意图



Crosshatch盘阵列结构



RAID命令处理过程中的关键问题

- I/O分解

磁盘阵列接收到主机的I/O请求命令，派生出对应于各个磁盘上的子I/O命令

I/O命令的形式？

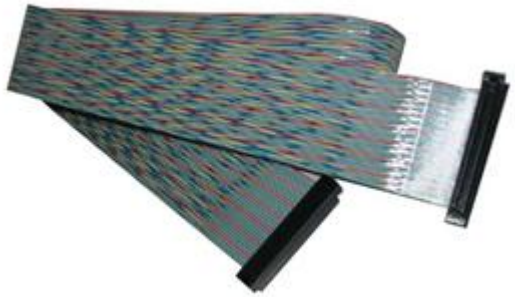
SCSI协议
支撑存储发展的脊梁

- SCSI：Small Computer System Interface，计算机与外部设备（特别是存储设备）间系统级接口的标准；
- SCSI标准定义了命令、通信协议、实体电器特性等（物理层，连接层，通信层，应用层）

SCSI历史

- SCSI协议V1版本（1986）：5MB/s，8位总线带宽，最多连接7个设备，25针

- SCSI协议V2版本：20MB/s（Fast-Wide），16位数据带宽、15个设备，50针或68针。高主频的SCSI存储设备陆续出现并成为市场的主流产品，也使得SCSI技术牢牢地占据了存储市场



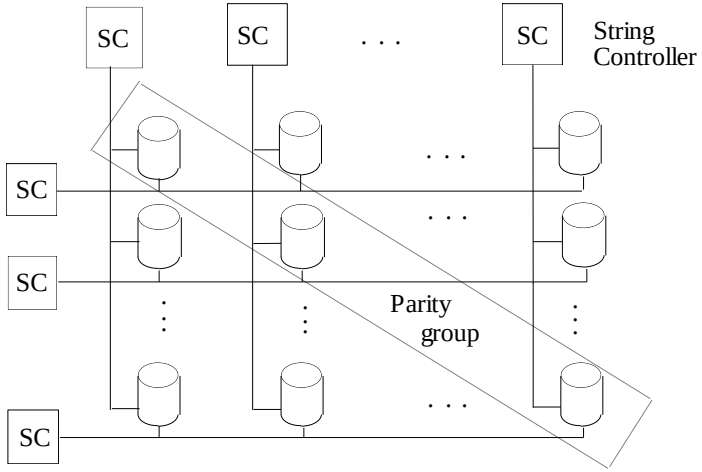
SCSI历史

- SCSI-3协议：320MB/s或更高，增加了能满足特殊设备协议所需要的命令集，使得SCSI协议既适应传统的并行传输设备，又能适应最新出现的一些串行设备的通信需要，如光纤通道协议（FCP）、串行存储协议（SSP）、串行总线协议等
- 串行SCSI——SAS
  - 第一代SAS　3.0Gbps
  - 第二代SAS　6.0Gbps
  - 第三代SAS　12.0Gbps

版本	年份	速度	特点
SCSI-1	1986	5 MB/s	初始版本，支持8位数据传输
SCSI-2	1990	10 MB/s	支持16位数据传输，增加同步传输模式
Fast SCSI	1991	20 MB/s	提高数据传输速度
Wide SCSI	1992	40 MB/s	支持32位数据传输，提高带宽
Ultra SCSI	1994	40 MB/s	提高数据传输速度，支持更小的数据包
Ultra2 SCSI	1996	80 MB/s	进一步提高数据传输速度
Ultra3 SCSI	1998	160 MB/s	继续提高数据传输速度
Ultra160 SCSI	2000	160 MB/s	支持16位数据传输，提高带宽
Ultra320 SCSI	2002	320 MB/s	进一步提高数据传输速度
Ultra640 SCSI	2004	640 MB/s	继续提高数据传输速度
SAS (Serial Attached SCSI)	2003	3 Gbps	使用串行传输技术，提高速度和可靠性
SAS 2.0	2009	6 Gbps	进一步提高数据传输速度
SAS 3.0	2013	12 Gbps	继续提高数据传输速度

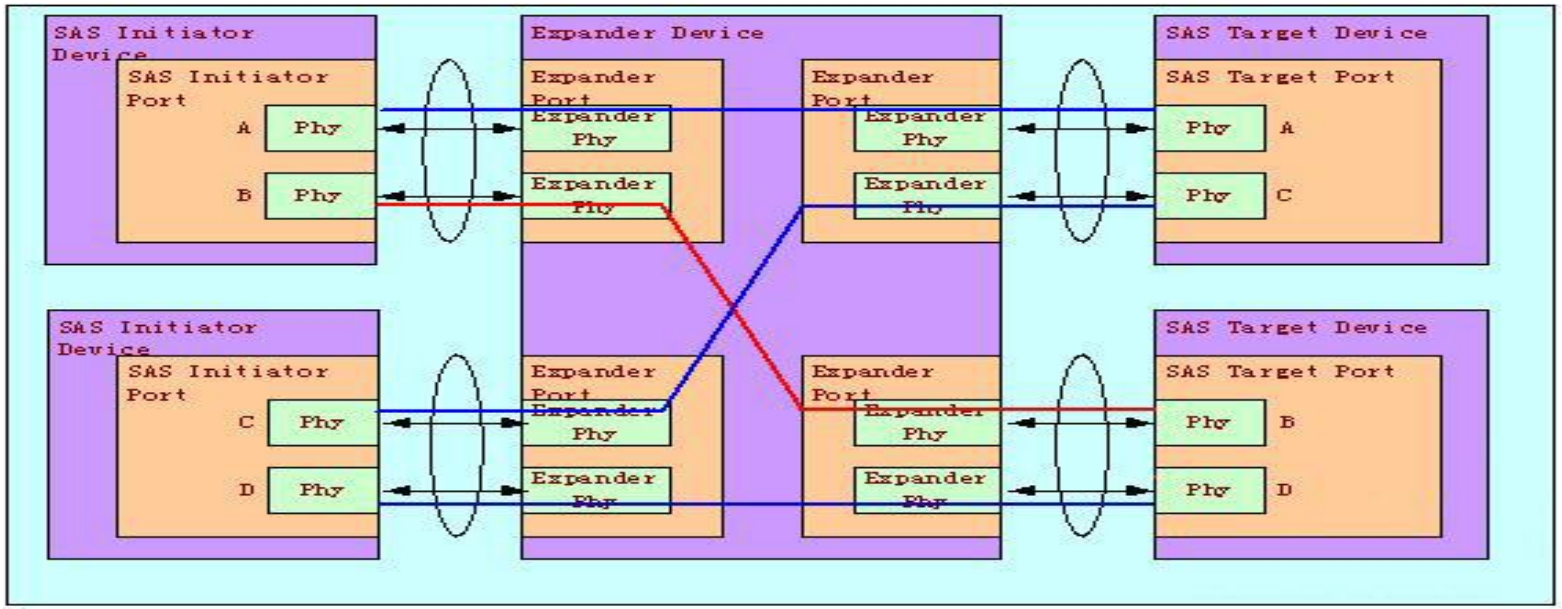
SAS(Serial Attached SCSI)简介

- 产生
  - 并行SCSI发展到Ultra320，没有提升空间
  - 低端的SATA性能、可靠性受限
- 特点
  - SCSI向下兼容性、串行点对点互连、双端口、寻址性和向小型化的扩展能力于一身
  - 可提供大数量设备、高带宽、可扩展性支持



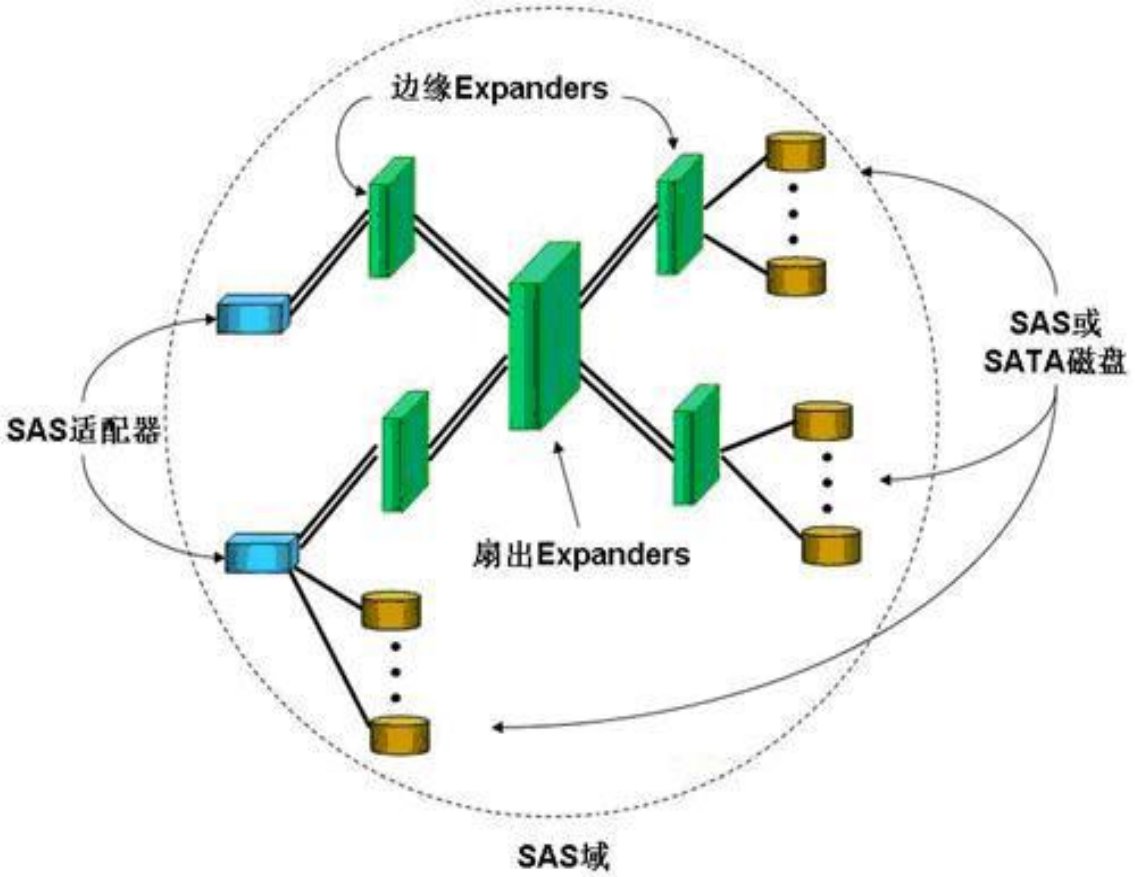
SAS连接

- SAS域
  - 一个SAS域主要由SAS初始设备(SAS Initiator Device)，扩展设备 (Expander Device)，以及SAS目标设备(SAS Target Device)组成



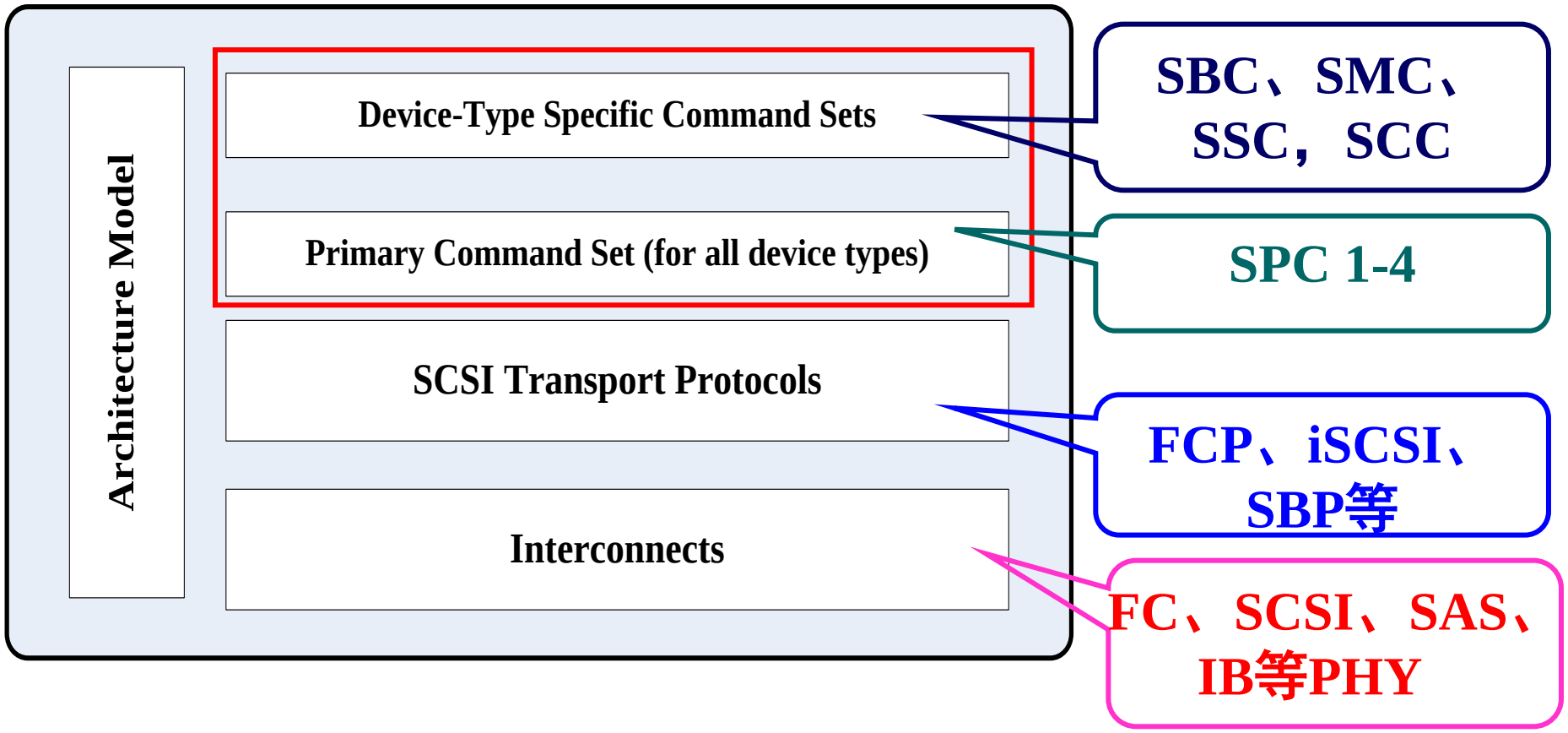
SAS连接

- 连接实例

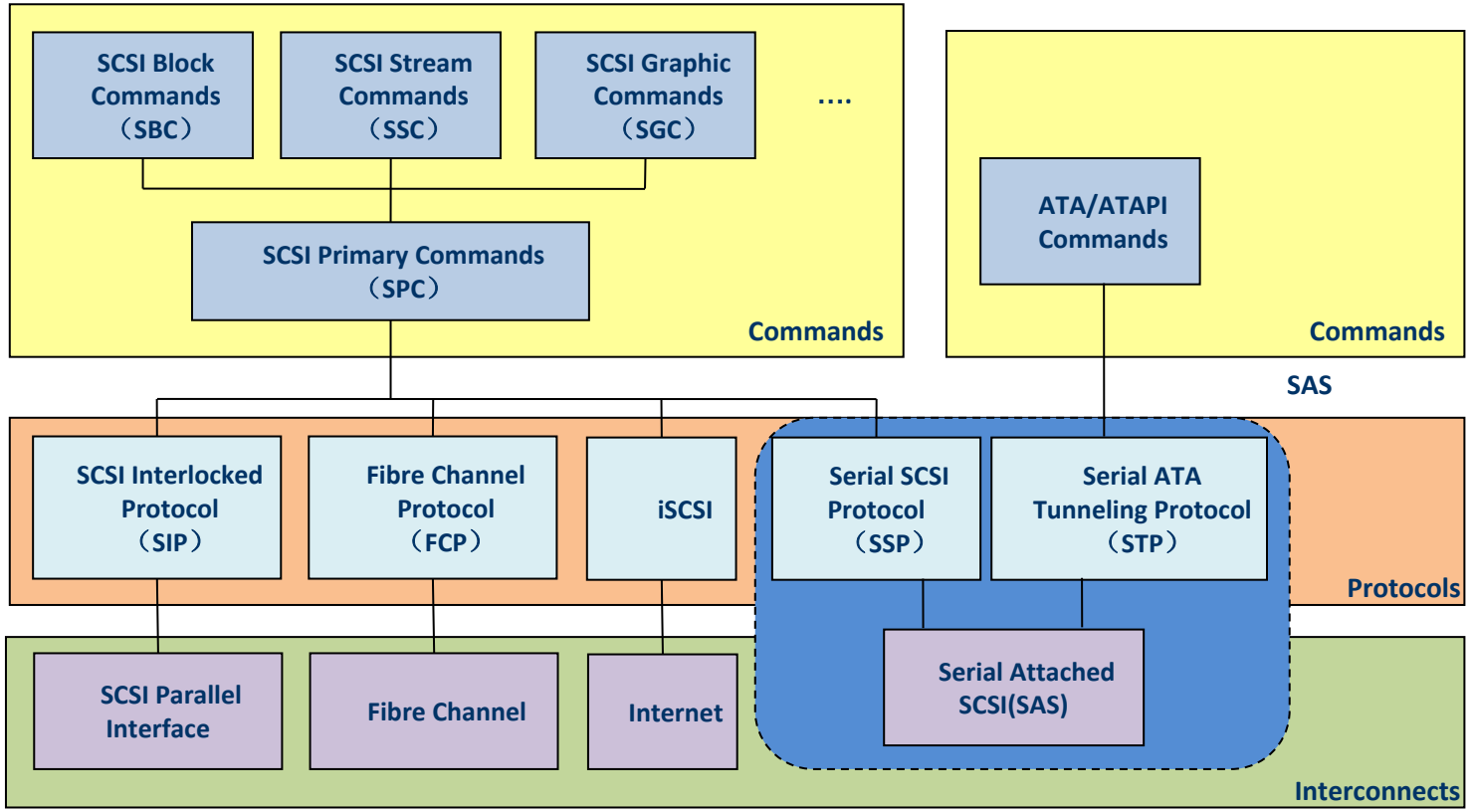


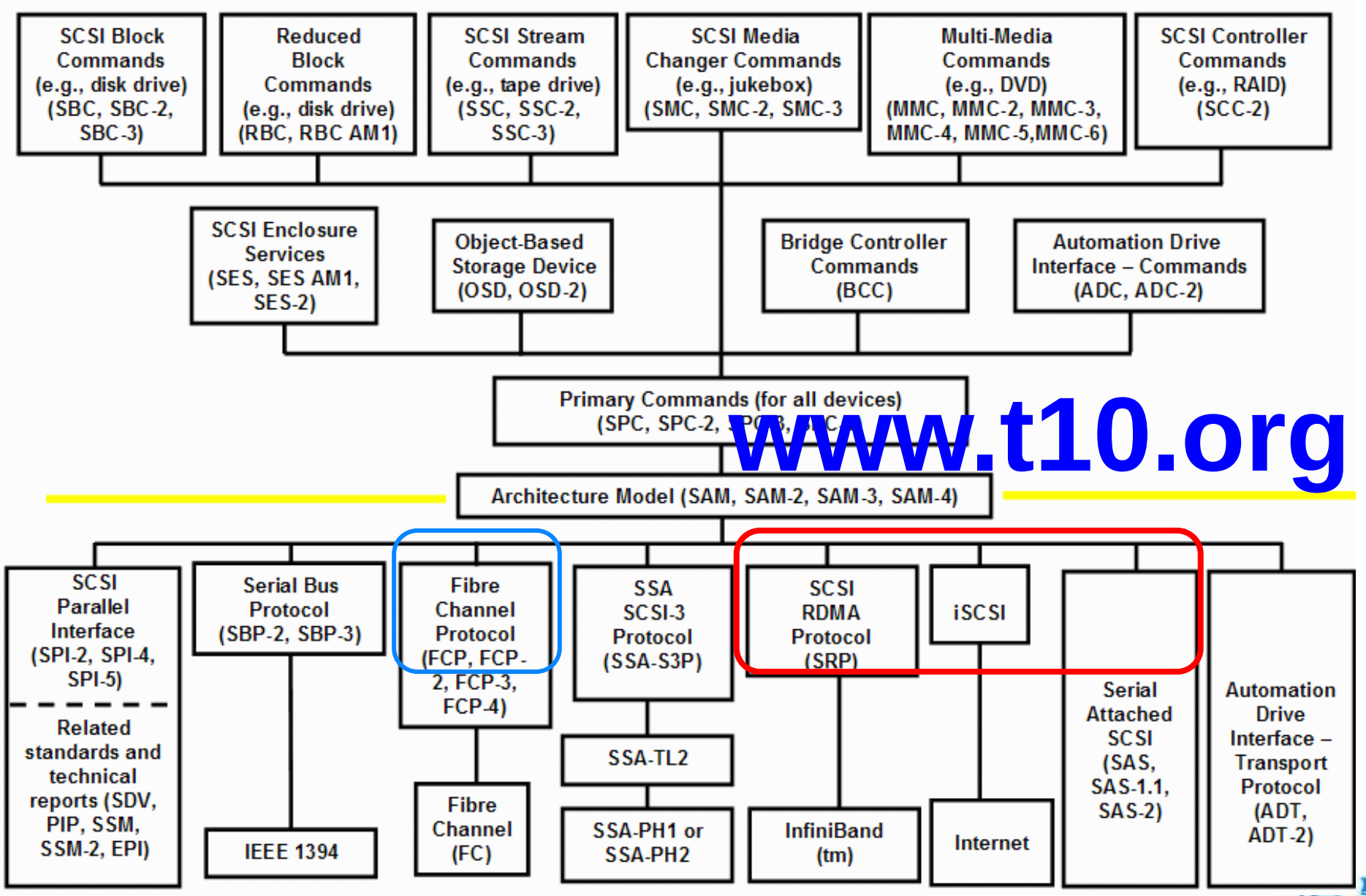
理论上，每个“边缘Expander”可以支持128个端口，每个SAS域可以有128个“边缘Expander”，因此，每个SAS域中最多可以有128 x 128 = 16384个端口。当然，内部互联至少要占用若干个端口。

SAM-3结构


Architecture Model
Device-Type Specific Command Sets
Primary Command Set (for all device types)
SCSI Transport Protocols
Interconnects
SBC、SMC、SSC，SCC
SPC 1-4
FCP、iSCSI、SBP等
FC、SCSI、SAS、IB等PHY

SAS在SAM中的位置

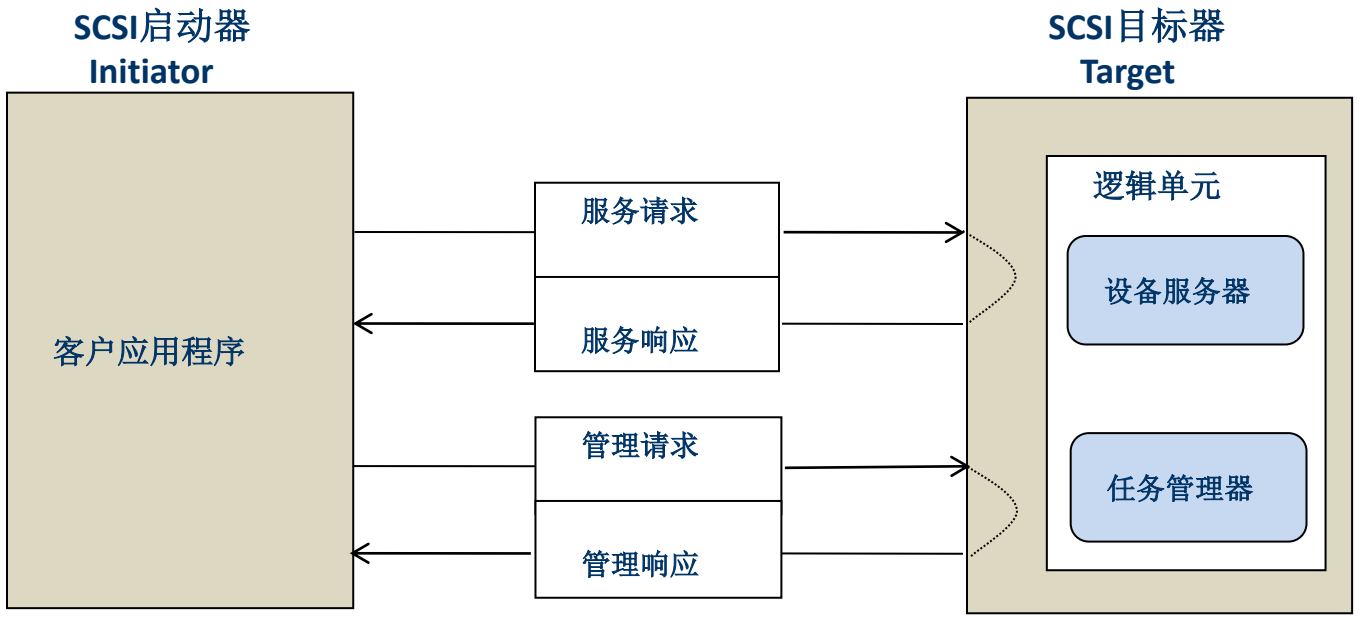



SCSI Block Commands (e.g., disk drive) (SBC, SBC-2, SBC-3)
Reduced Block Commands (e.g., disk drive) (RBC, RBC AM1)
SCSI Stream Commands (e.g., tape drive) (SSC, SSC-2, SSC-3)
SCSI Media Changer Commands (e.g., jukebox) (SMC, SMC-2, SMC-3
Multi-Media Commands (e.g., DVD) (MMC, MMC-2, MMC-3, MMC-4, MMC-5,MMC-6)
SCSI Controller Commands (e.g., RAID) (SCC-2)
SCSI Enclosure Services (SES, SES AM1, SES-2)
Object-Based Storage Device (OSD, OSD-2)
Bridge Controller Commands (BCC)
Automation Drive Interface – Commands (ADC, ADC-2)
Primary Commands (for all devices)
www.t10.org
Architecture Model (SAM, SAM-2, SAM-3, SAM-4)
SCSI Parallel Interface (SPI-2, SPI-4, SPI-5)
Related standards and technical reports (SDV, PIP, SSM, SSM-2, EPI)
Serial Bus Protocol (SBP-2, SBP-3)
IEEE 1394
Fibre Channel Protocol (FCP, FCP-2, FCP-3, FCP-4)
Fibre Channel (FC)
SSA SCSI-3 Protocol (SSA-S3P)
SSA-TL2
SSA-PH1 or SSA-PH2
SCSI RDMA Protocol (SRP)
InfiniBand (tm)
iSCSI
Internet
Serial Attached SCSI (SAS, SAS-1.1, SAS-2)
Automation Drive Interface – Transport Protocol (ADT, ADT-2)

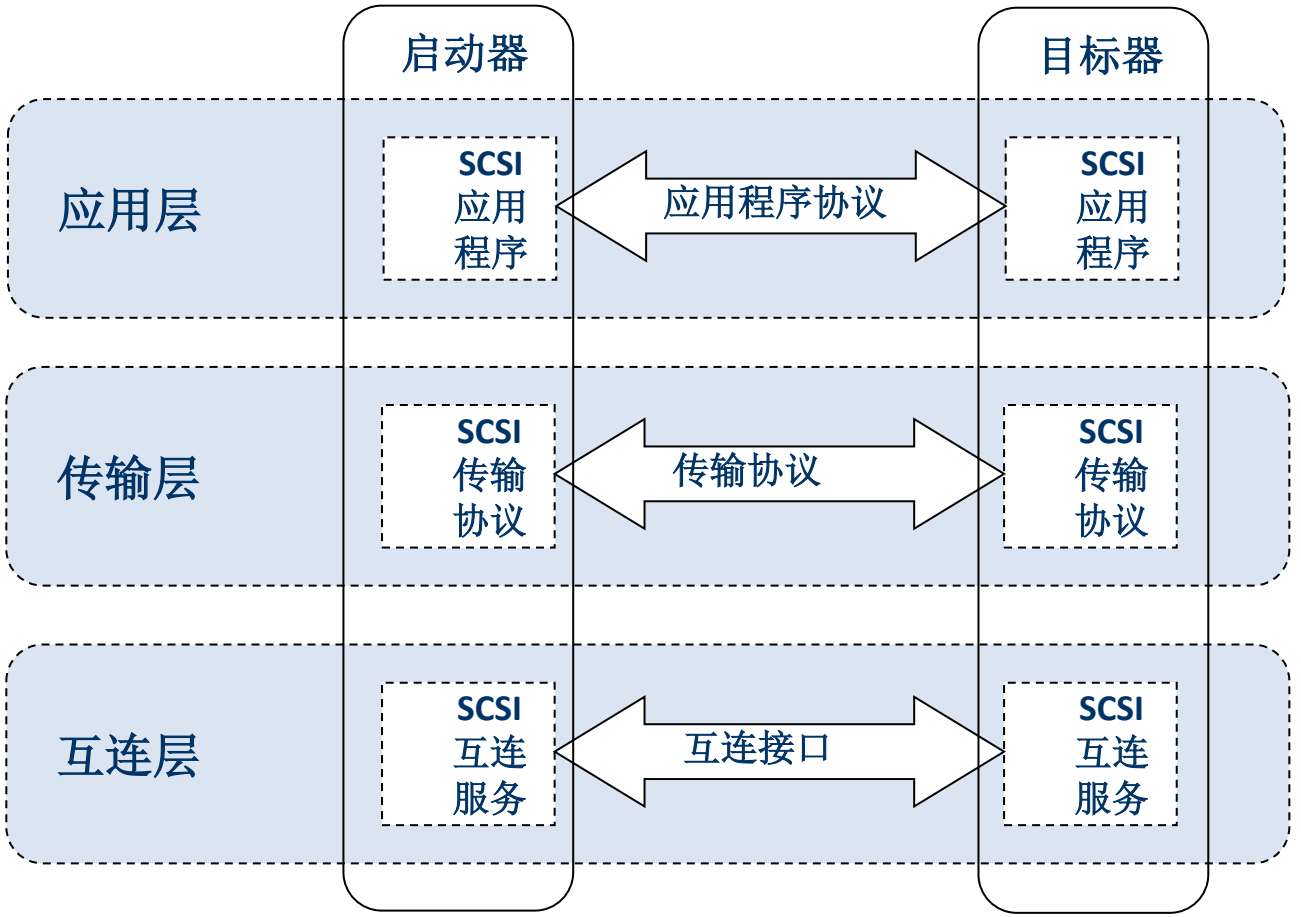
SCSI在系统中的地位

SCSI通信服务模型

- “客户-服务器”模型

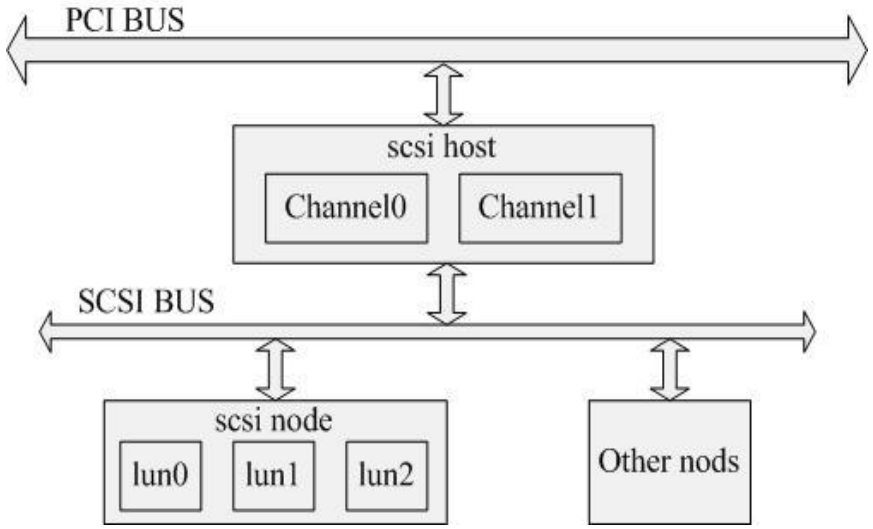


SCSI通信模型


启动器
目标器
应用层
SCSI
应用
程序
应用程序协议
SCSI
应用
程序
传输层
SCSI
传输
协议
传输协议
SCSI
传输
协议
互连层
SCSI
互连
服务
互连接口
SCSI
互连
服务

对SCSI设备的访问通过三元组：

- 总线（Bus, Channel）
- 目标（Target, ID）
- LUN（Logical Unit Number）



Linux系统SCSI中间层（SCSI Middle Level）定义了“scsi device”的数据结构

SCSI CDB (Command Descriptor Block)

SCSI命令

Typical CDB for 6-byte commands

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	miscellaneous CDB information			(MSB)				
2	LOGICAL BLOCK ADDRESS (if required)							
3								(LSB)
4	TRANSFER LENGTH (if required) PARAMETER LIST LENGTH (if required) ALLOCATION LENGTH (if required)							
5	CONTROL							

21bits

TEST UNIT READY(0x00)
INQUIRY(0x12)
READ（0x08）
REPORT LUNS(0xA0)
WRITE（0x0A）

SCSI命令

Typical CDB for 10-byte commands

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	miscellaneous CDB information			SERVICE ACTION (if required)				
2	(MSB)							
3				LOGICAL BLOCK ADDRESS (if required)				
4								
5								(LSB)
6	miscellaneous CDB information							
7	(MSB)			TRANSFER LENGTH (if required) PARAMETER LIST LENGTH (if required) ALLOCATION LENGTH (if required)				
8								(LSB)
9	CONTROL							

32bits

READ_CAPACITY（0x25）
WRITE_10（0x2a）
VERIFY（0x2f）

READ_10（0x28）
WRITE_VERIFY（0x2e）
SYNCHRONIZE_CACHE（0x35）

SCSI命令

Typical CDB for 12-byte commands

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	miscellaneous CDB information			SERVICE ACTION (if required)				
2	(MSB)							
3				LOGICAL BLOCK ADDRESS (if required)				
4								
5								(LSB)
6	(MSB)							
7				TRANSFER LENGTH (if required) PARAMETER LIST LENGTH (if required) ALLOCATION LENGTH (if required)				
8								
9								(LSB)
10	miscellaneous CDB information							
11	CONTROL							

32bits

READ_12（0xa8）　　WRITE_12（0xaa）
WRITE_VERIFY_12（0xae）　　SEARCH_HIGH_12（0xb0）
SEARCH_EQUAL_12（0xb1）　　SEARCH_LOW_12（0xb2）

SCSI命令

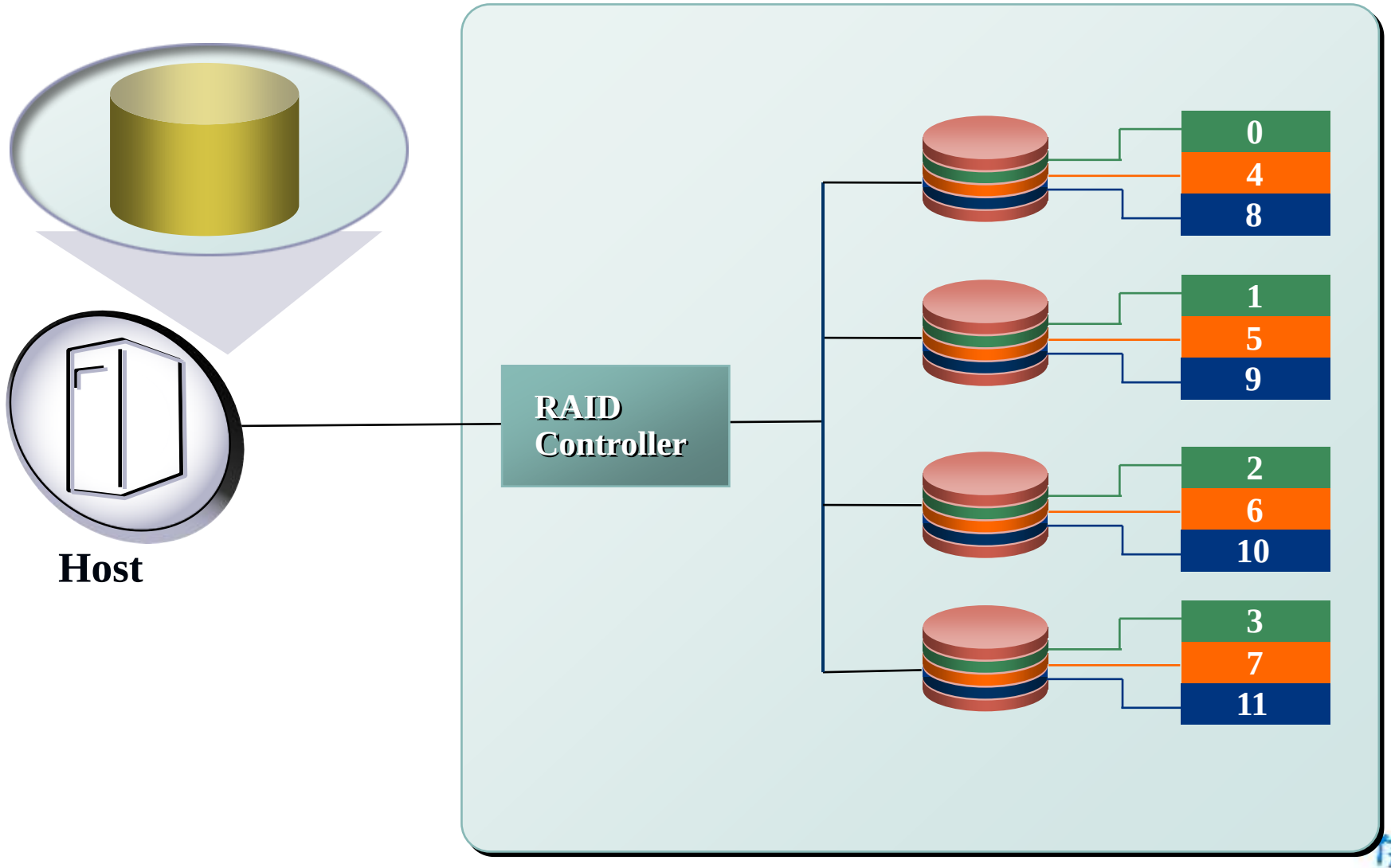
Typical CDB for long LBA 16-byte commands

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	miscellaneous CDB information							
2	(MSB)							
3								
4								
5								
6				LOGICAL BLOCK ADDRESS				
7								
8								
9								(LSB)
10	(MSB)							
11				TRANSFER LENGTH (if required) PARAMETER LIST LENGTH (if required) ALLOCATION LENGTH (if required)				
12								
13								(LSB)
14	miscellaneous CDB information							
15	CONTROL							

64bits

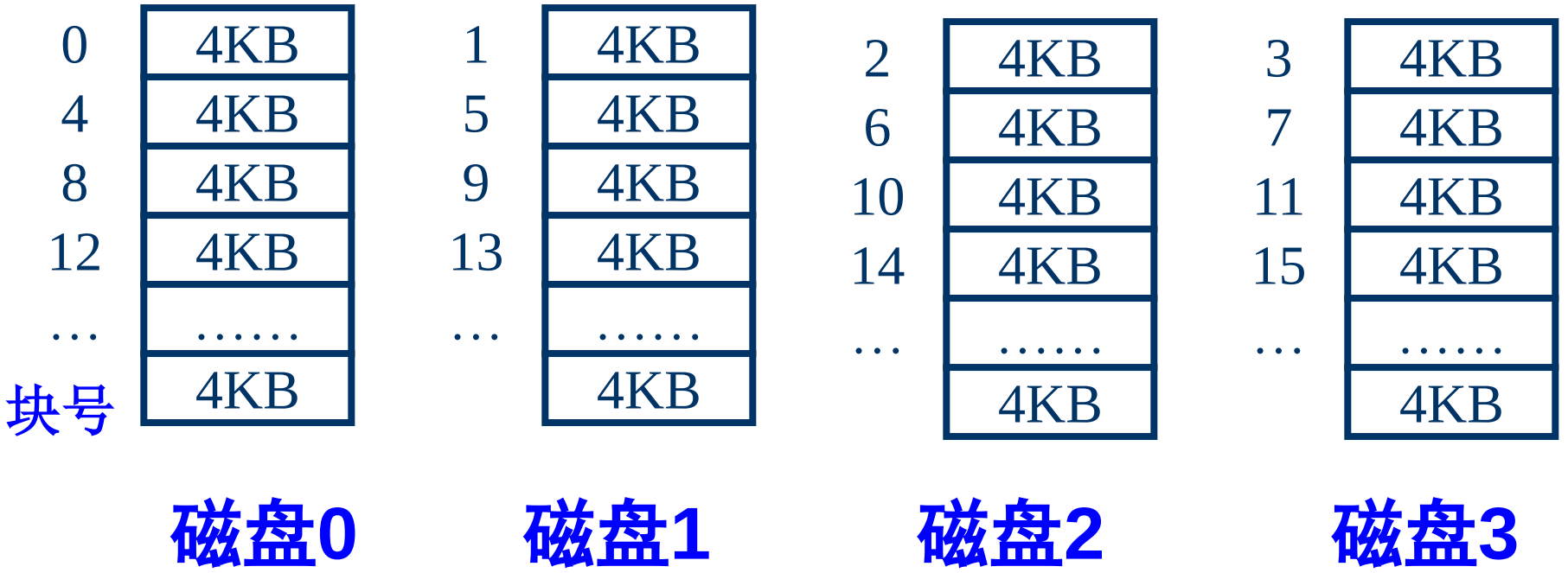
READ_16(0x88)　　write_16(0x8a)
Readcapacity_16(0x9e)

命令分解举例（RAID 0)

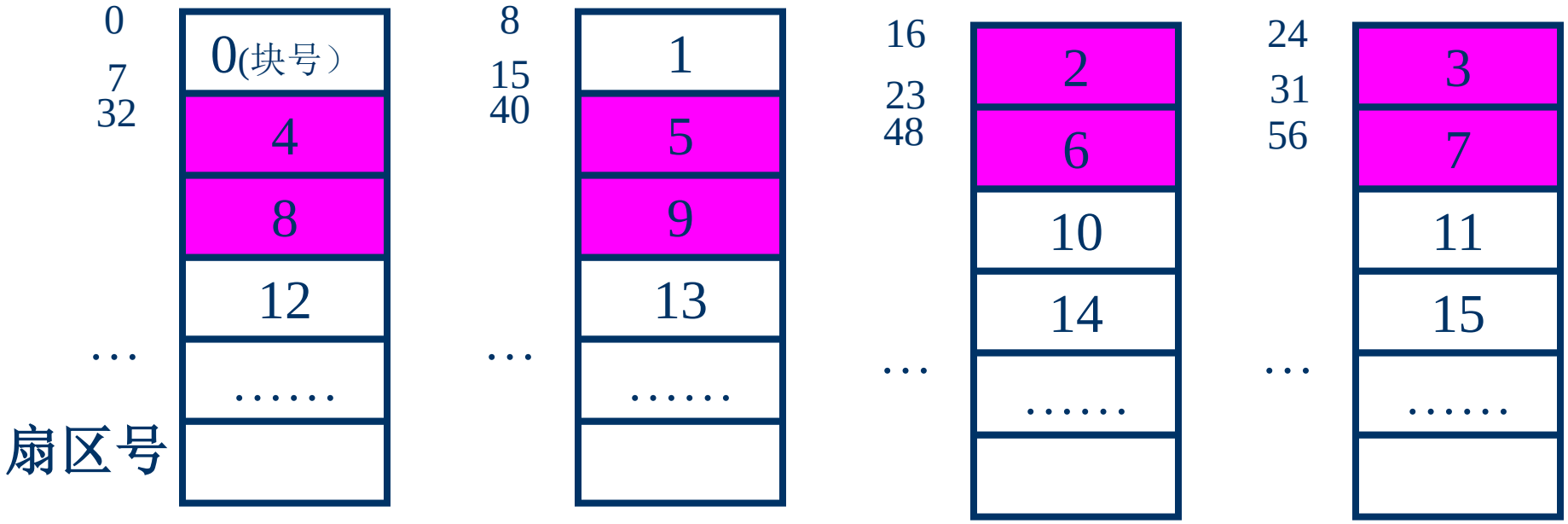


命令分解举例（RAID 0)

设阵列由4个磁盘构成，每个磁盘容量为C，则主机用户看到的阵列为一个容量为4C的大磁盘；数据分块大小为4KB，构成的阵列结构如下：



设主机请求读32KB的数据，起始地址为16（单位:扇区），SCSI命令为：（0x08, 0x00, 0x00, 0x10, 0x40, 0x00）

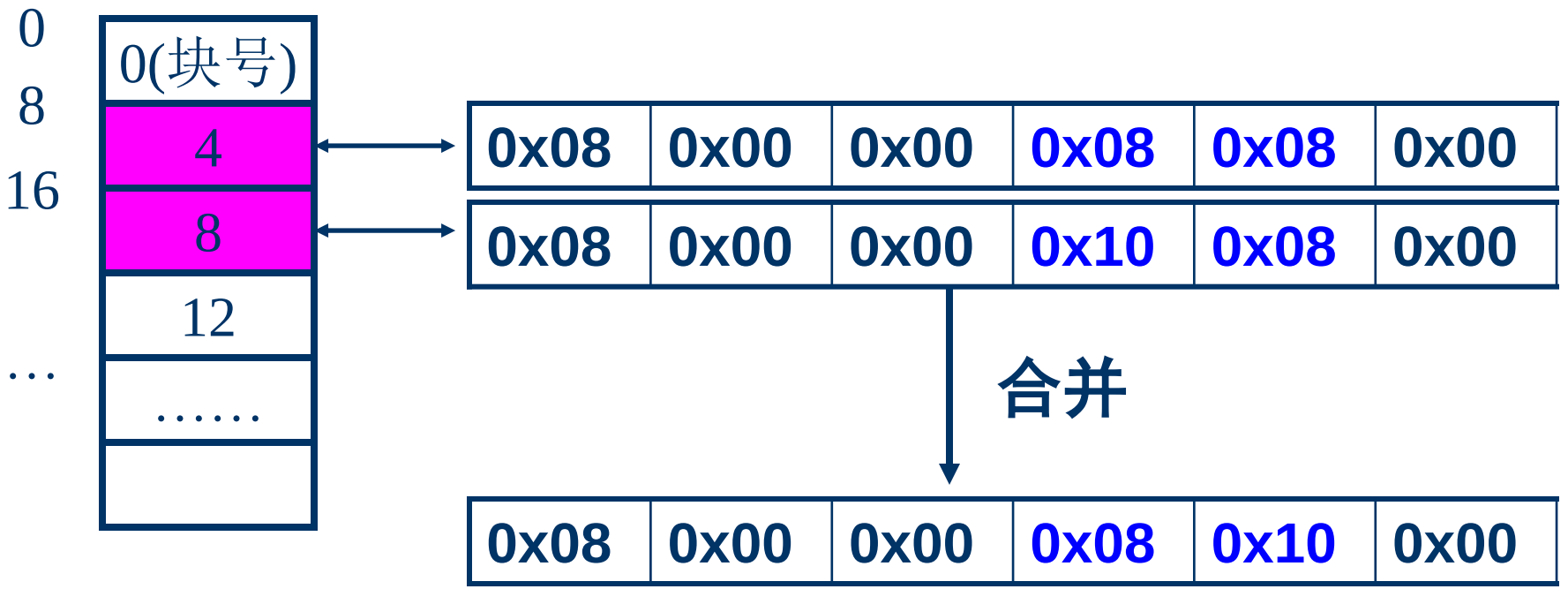


要读的数据在磁盘上的分布

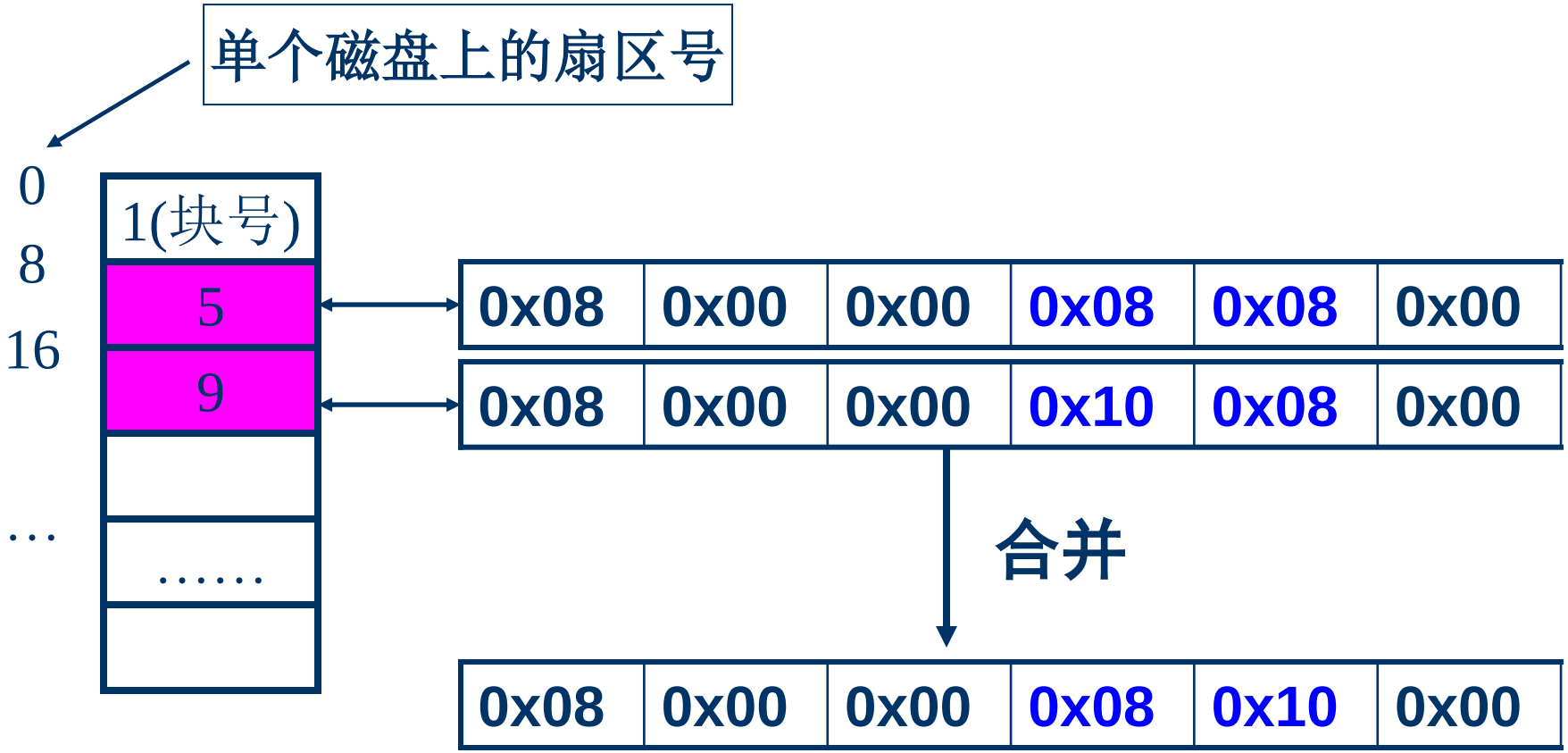
实际上主机要读的数据就是上图中的2～9块，并且是按照2，3，4，…，9的顺序。

对应单个磁盘上的子命令：

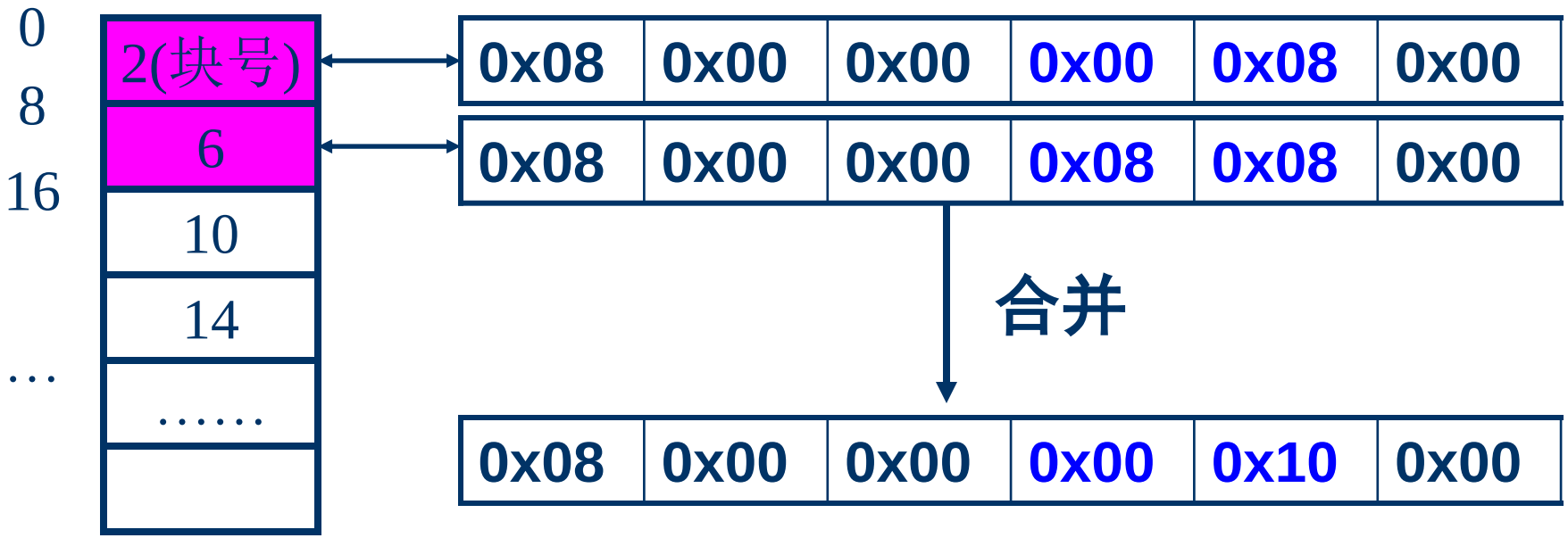
磁盘1：



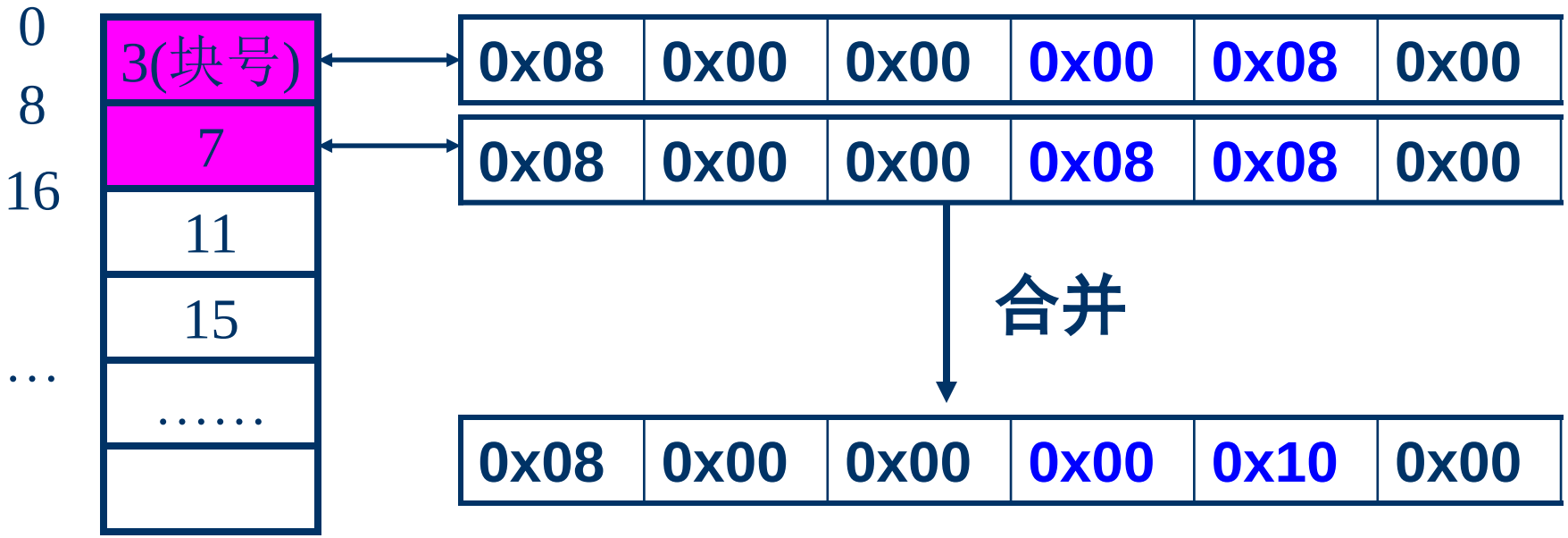
磁盘2


单个磁盘上的扇区号
0
8
16
…
1(块号)
5
9
……
0x08 0x00 0x00 0x08 0x08 0x00
0x08 0x00 0x00 0x10 0x08 0x00
合并
0x08 0x00 0x00 0x08 0x10 0x00

磁盘3


0
8
16
…
2(块号)
6
10
14
……
0x08 0x00 0x00 0x00 0x08 0x00
0x08 0x00 0x00 0x08 0x08 0x00
合并
0x08 0x00 0x00 0x00 0x10 0x00

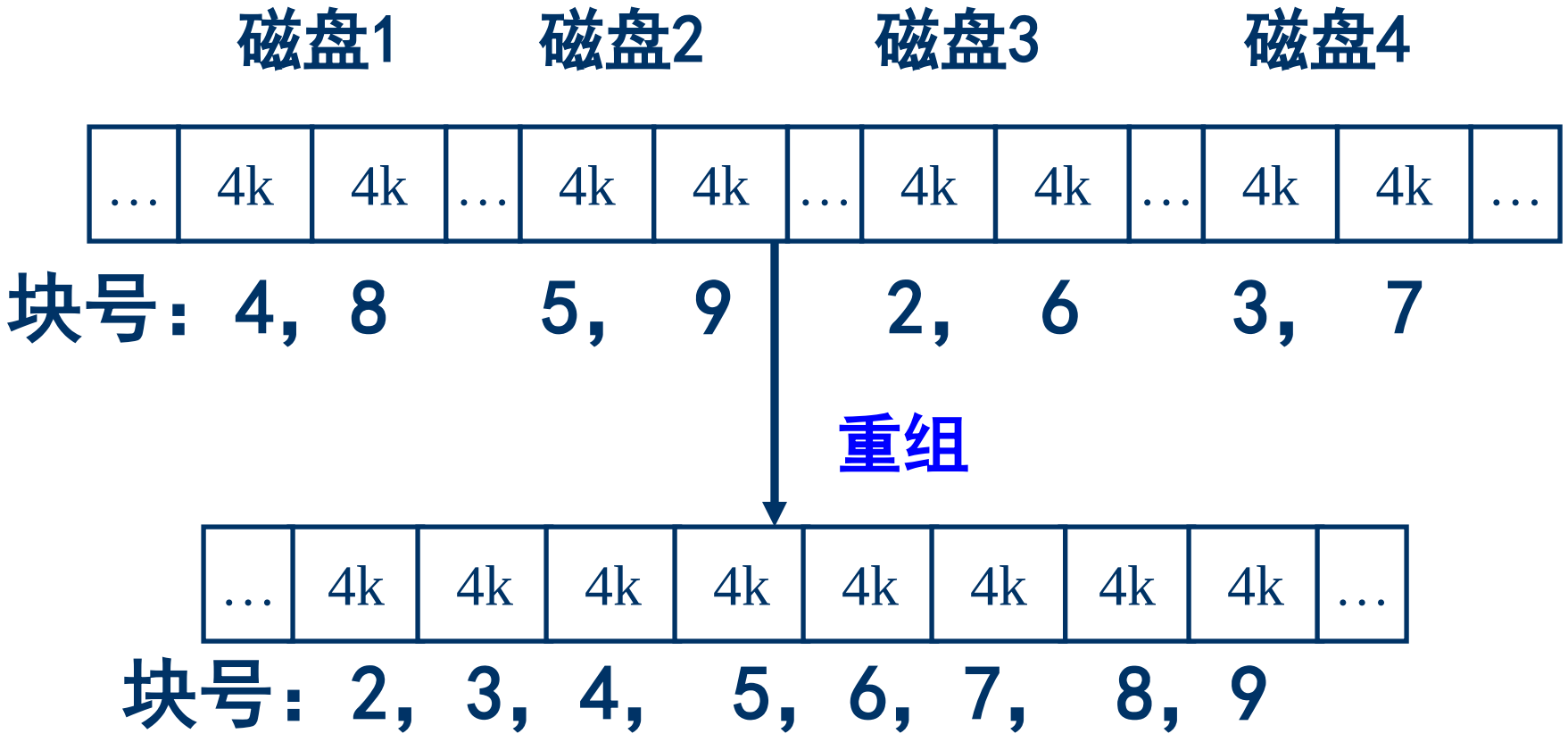
磁盘4


0
8
16
…
3(块号)
7
11
15
……
0x08 0x00 0x00 0x00 0x08 0x00
0x08 0x00 0x00 0x08 0x08 0x00
合并
0x08 0x00 0x00 0x00 0x10 0x00

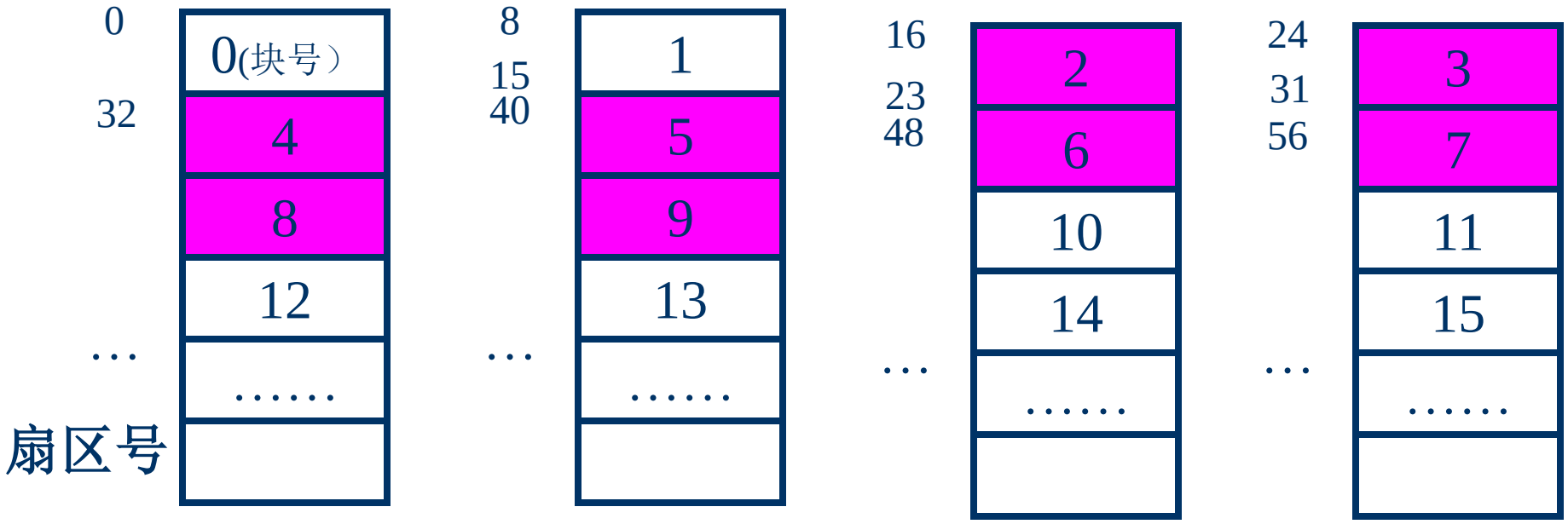
Next

数据在内存中的分布：

经过命令合并（减少I/O次数），采用普通DMA方式从磁盘读出数据在内存中分布如下：

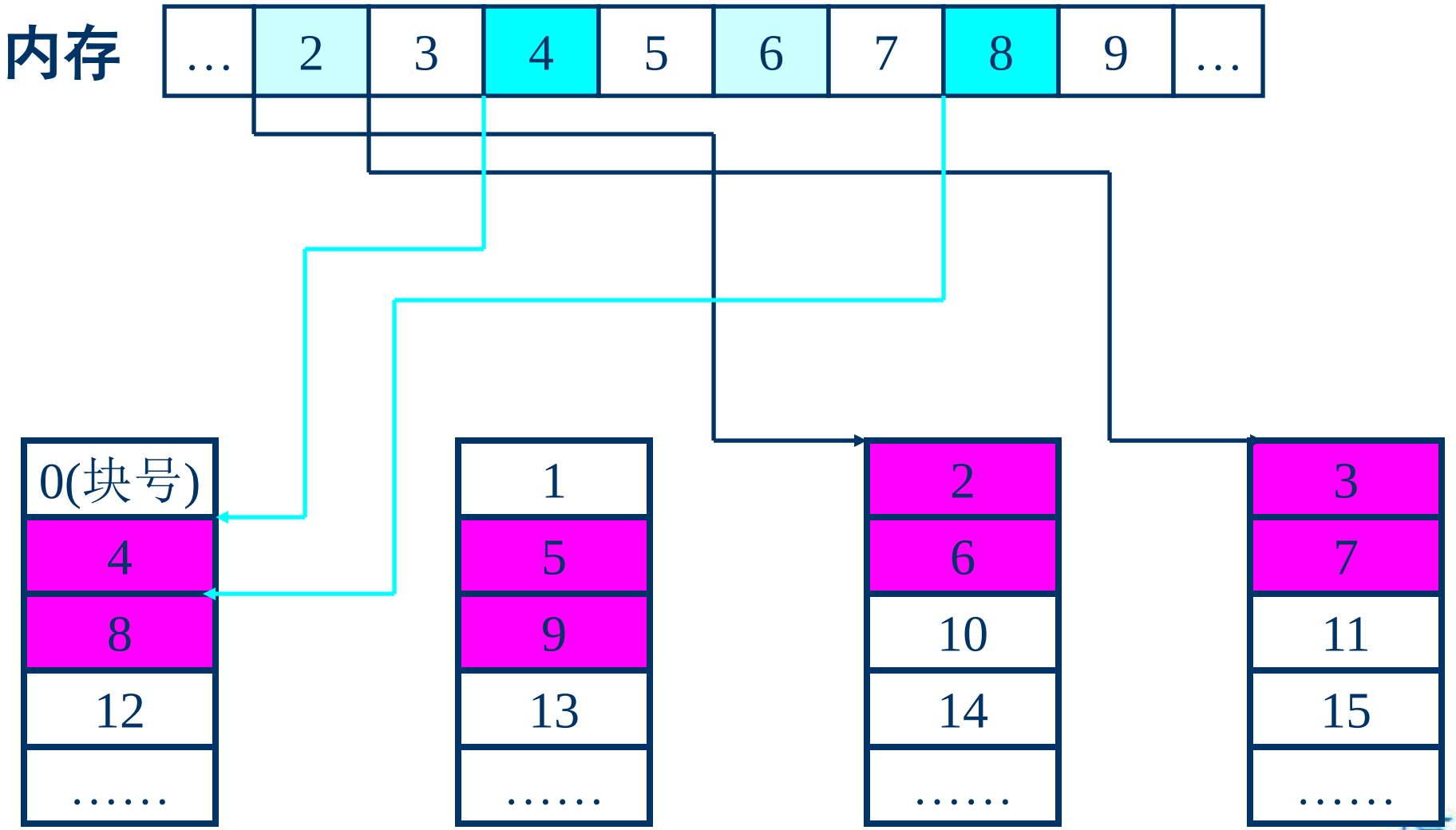


写过程：32KB的数据程度，起始地址为16（单位:扇区），SCSI命令为：(0x0A, 0x00, 0x00, x10, 0x40, 0x00)

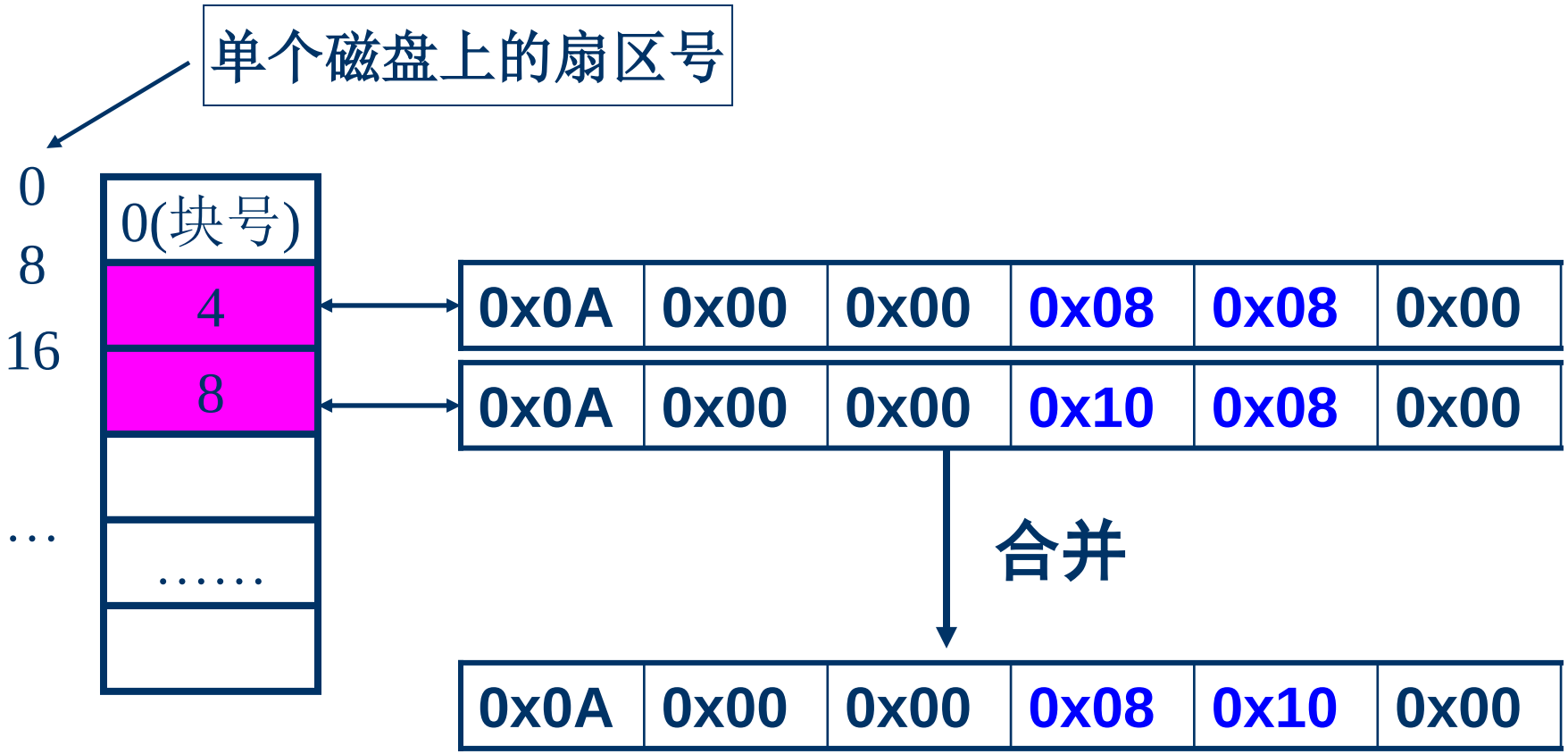


要写的数据在磁盘上的分布

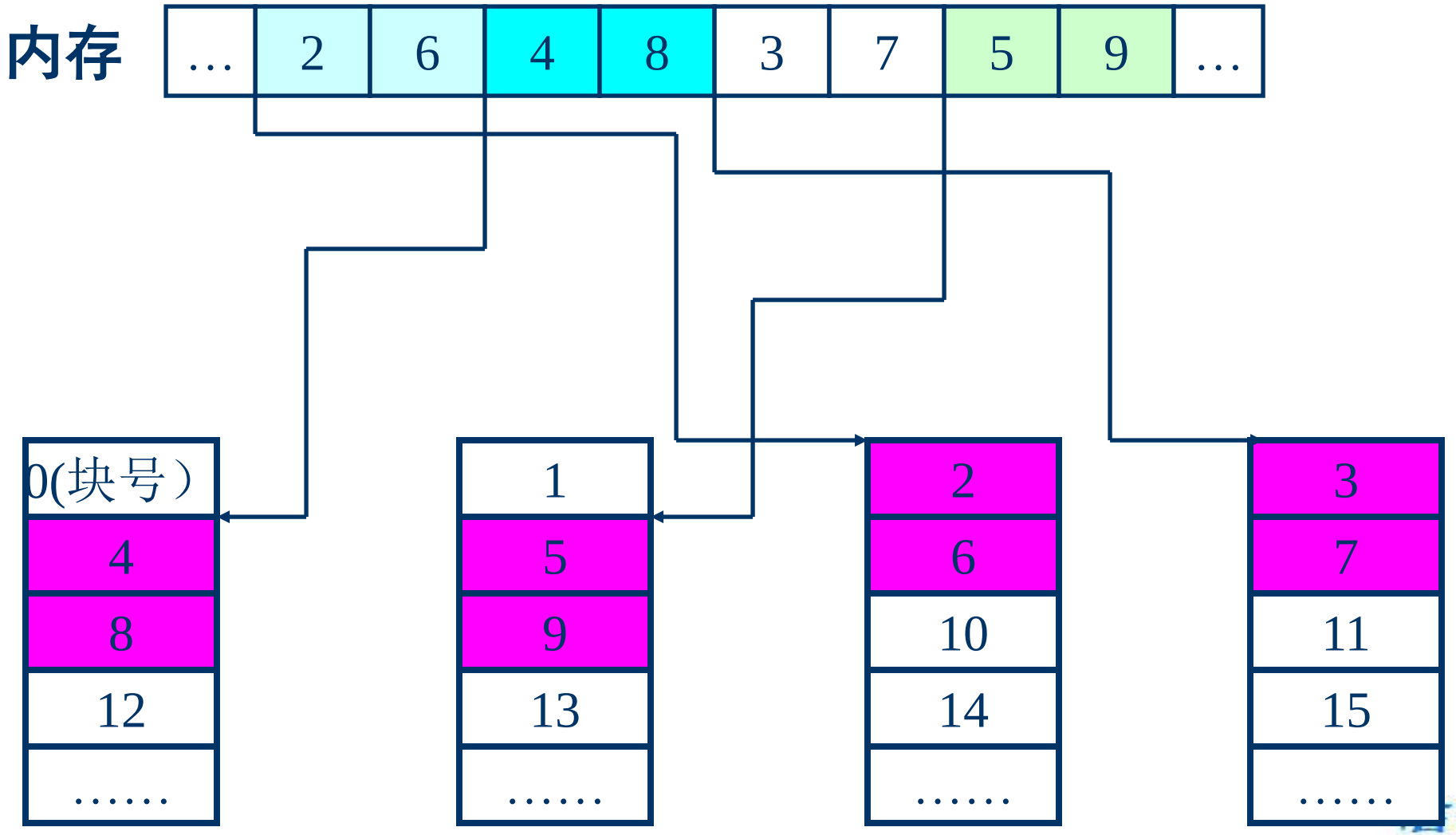
内存中分块数据—磁盘数据对应关系：



对应磁盘1上的写命令：


单个磁盘上的扇区号
0
8
16
…
0(块号)
4
8
……
0x0A 0x00 0x00 0x08 0x08 0x00
0x0A 0x00 0x00 0x10 0x08 0x00
合并
0x0A 0x00 0x00 0x08 0x10 0x00

分块重组后内存—磁盘对应关系：

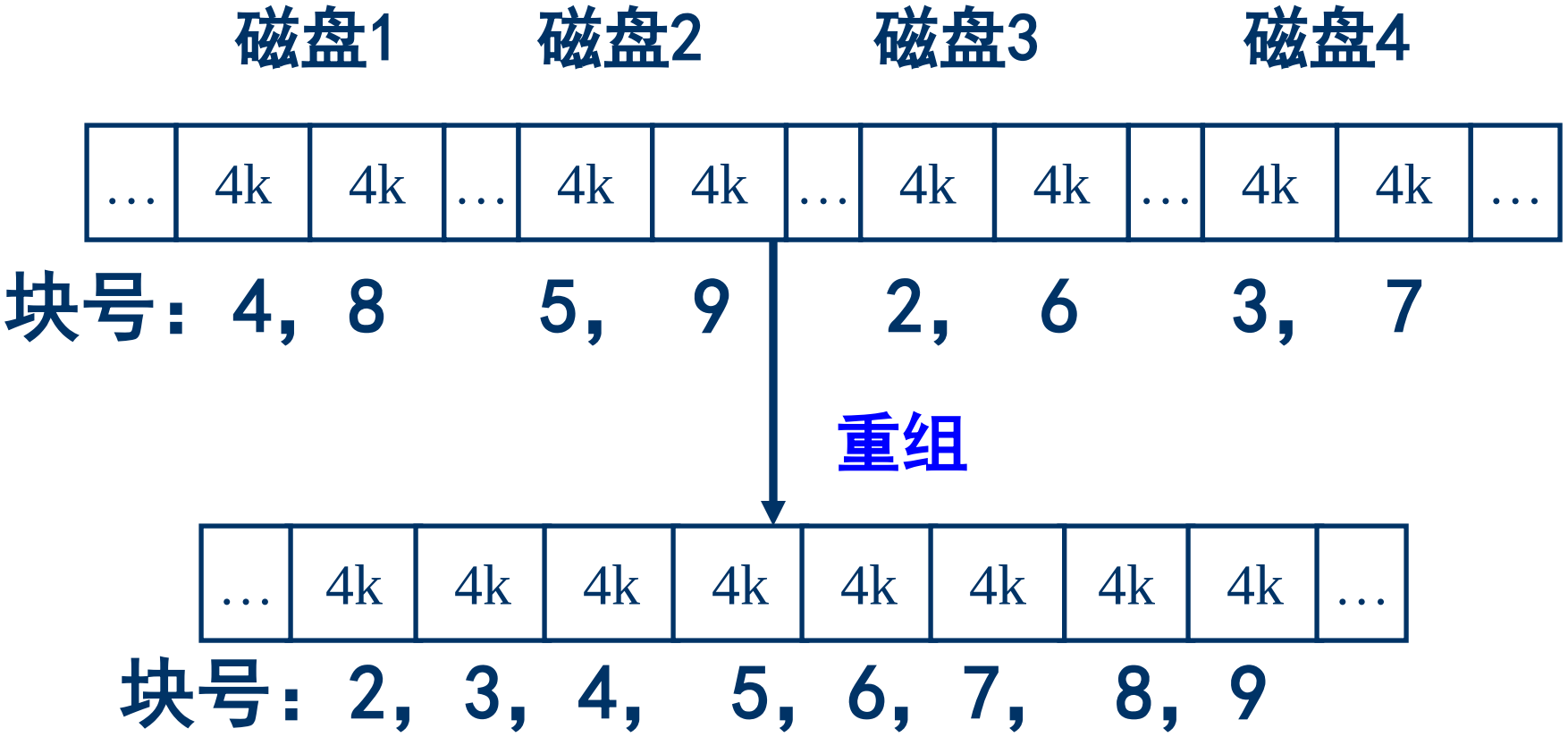


数据分块重组带来的问题—数据移动

分块前

重组后

读命令同样存在这个问题

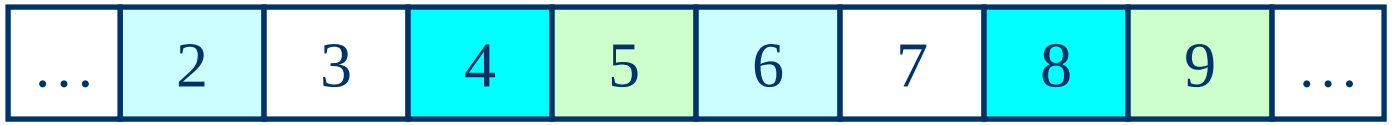


时间开销问题

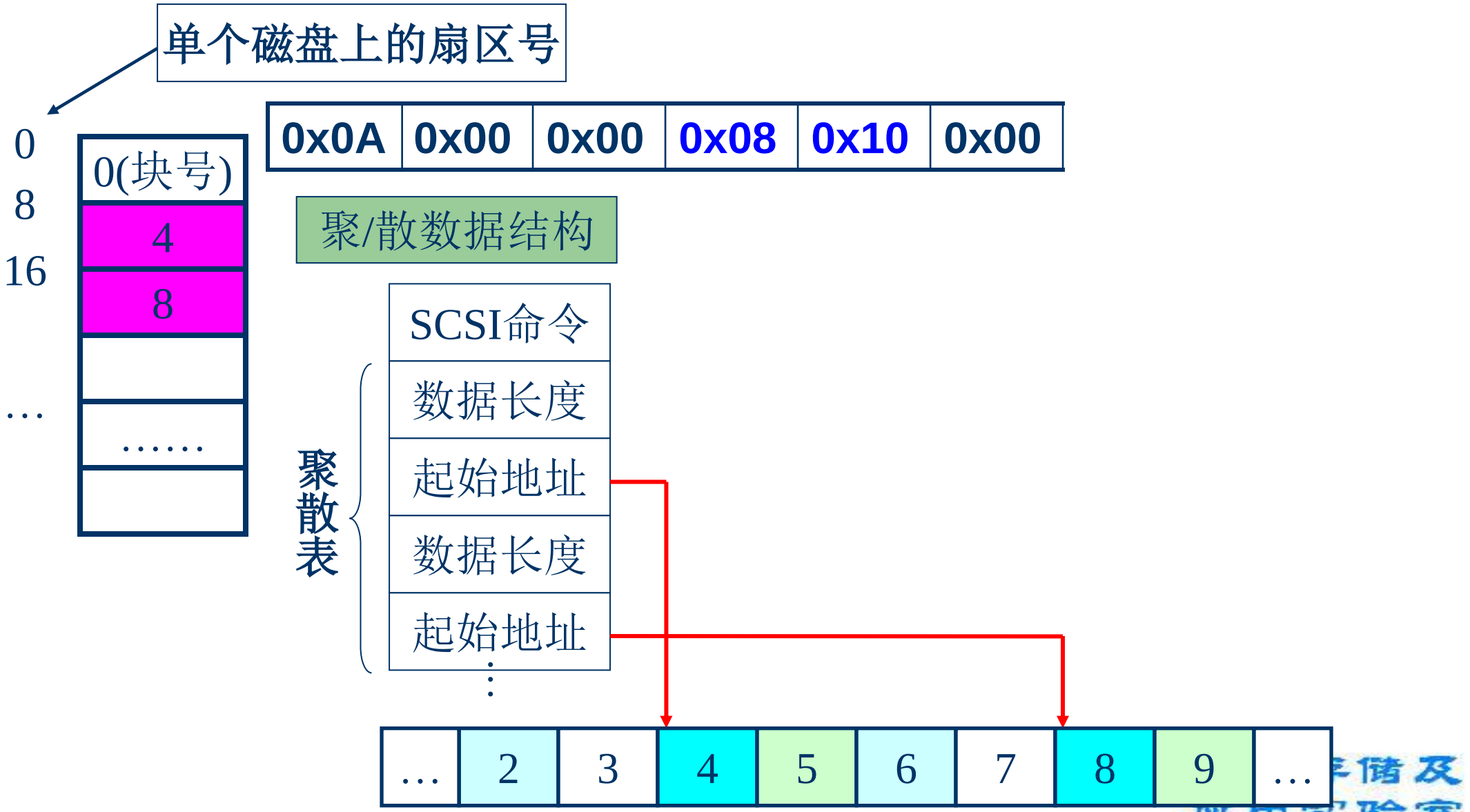
减少磁盘I/O操作的次数极大地提高阵列的性能，所以采用了I/O合并技术，I/O合并引起数据在内存中的移动，这种移动开销相对于磁盘I/O开销要小的多，但性能的影响还是相当大的（近25%）。如何消除这种不必要的开销？

聚/散技术

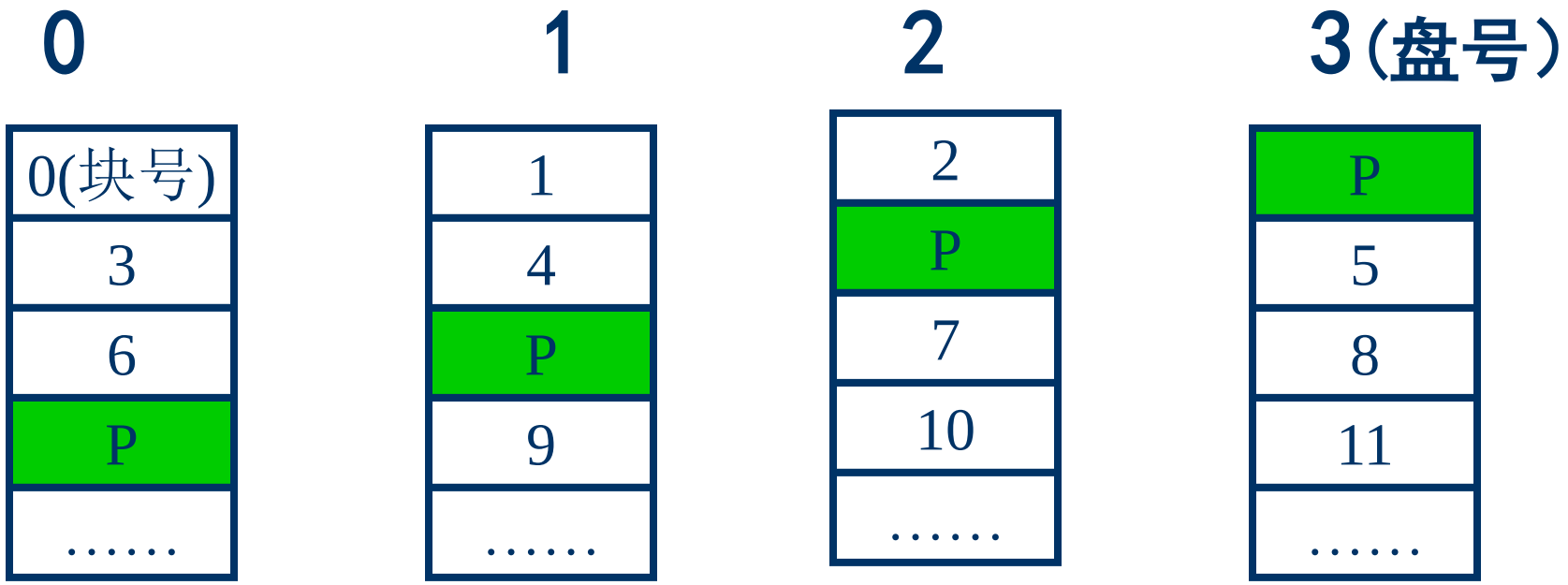
消除数据移动带来的额外开销，聚散技术的基本思想是将散落在内存的各块数据聚集起来传输。



对应磁盘1上的写命令聚散格式：

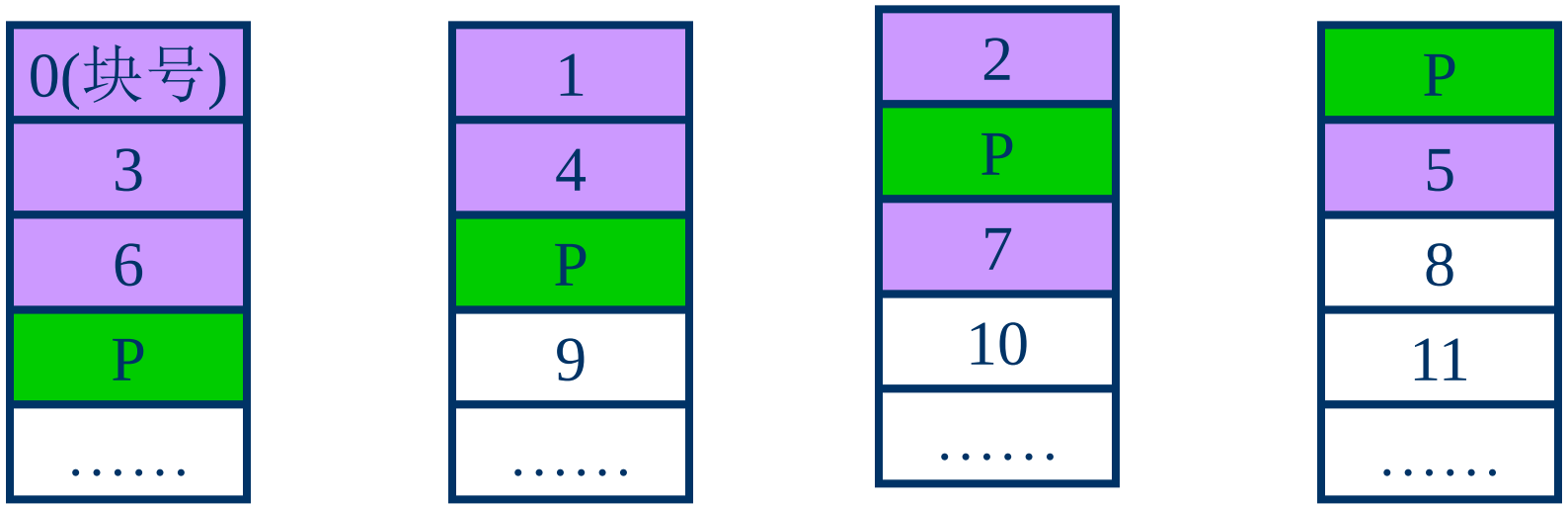


RAID5命令分解过程：



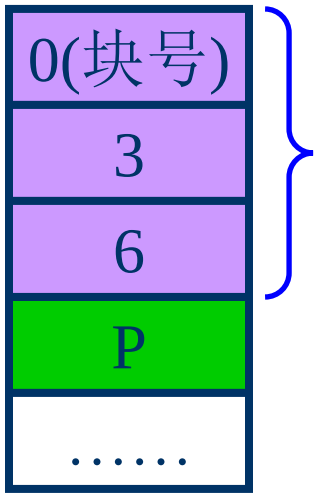
阵列参数：左不对称，4块盘，分块大小为2K

读命令：0x08, 数据长度16k，起始地址为0



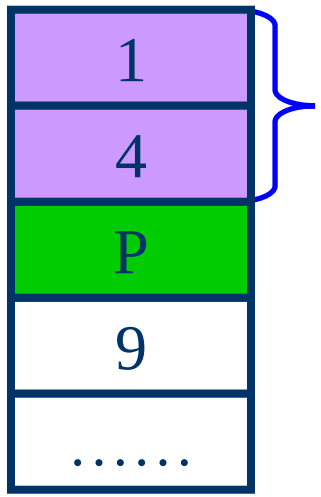
则要读的有效数据为图中的0～7块

0号盘上的子命令：



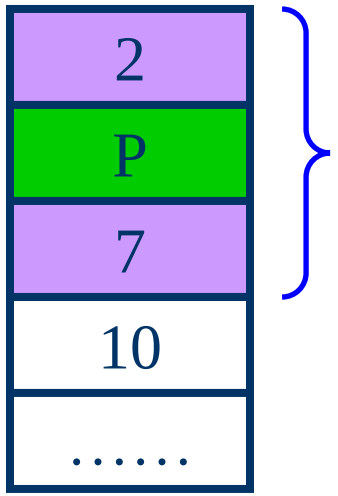
顺序读出0，3，6块，可以合并为一个读命令，数据长度为6k

1号盘上的子命令：



顺序读出1，4块，可以合并为一个读命令，数据长度为4k

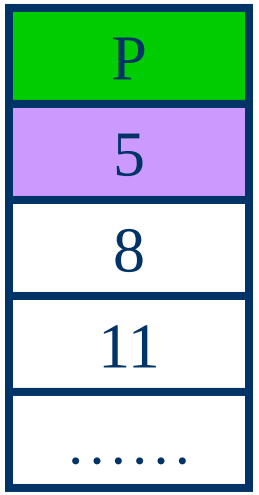
2号盘上的子命令：



方案1：
分别产生子命令读2号块和7号有效数据块

方案2：
为了减少I/O次数，把它们合并再一起读，把中间夹的校验块也读出来，数据总长度为6k，有效的为4k

3号盘上的子命令：



只有5号块，长度为2k

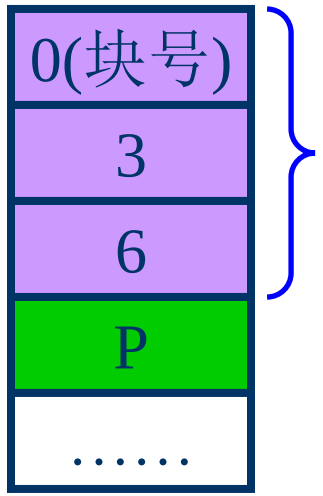
RAID5 块大小2KB

写命令：0x0a, 数据长度16k，起始地址为0

0(块号)	1	2	P	满条
3	4	P	5	满条
6	P	7	8	大写
P	9	10	11	
……	……	……	……	

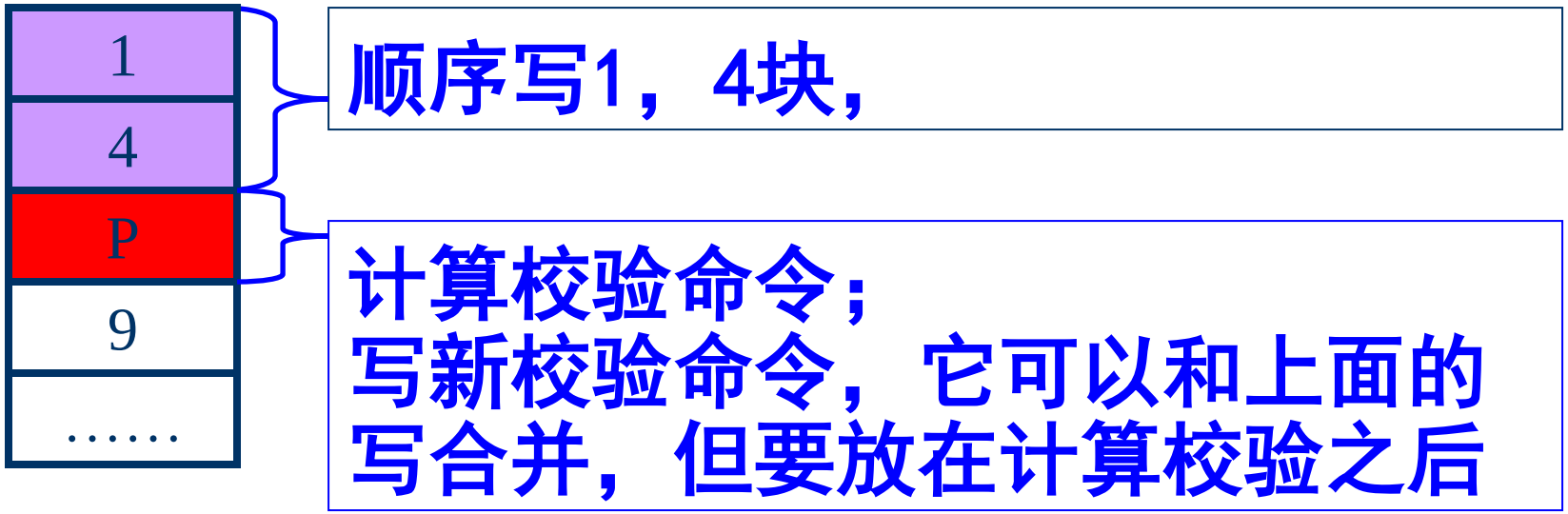
则要写的有效数据为图中的0～7块

0号盘上的子命令：



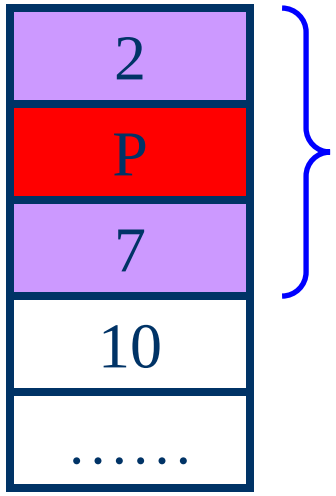
顺序写出0，3，6块，可以合并为一个读命令，数据长度为6k

1号盘上的子命令：



1. 计算校验P
2. 写数据和校验命令(合并为1条命令)

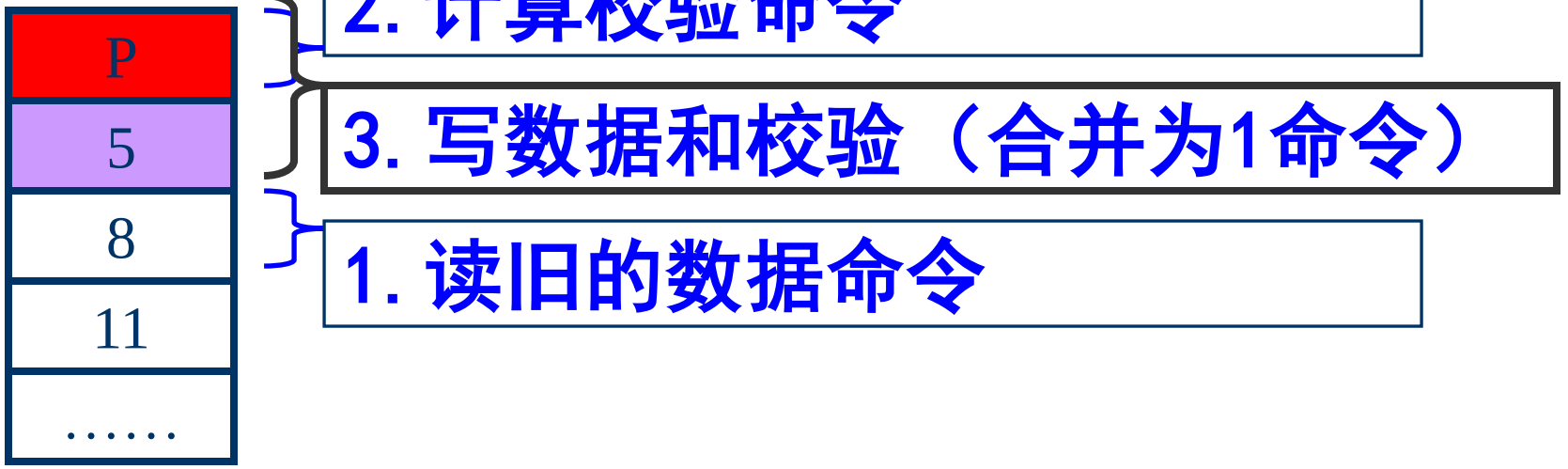
2号盘上的子命令：



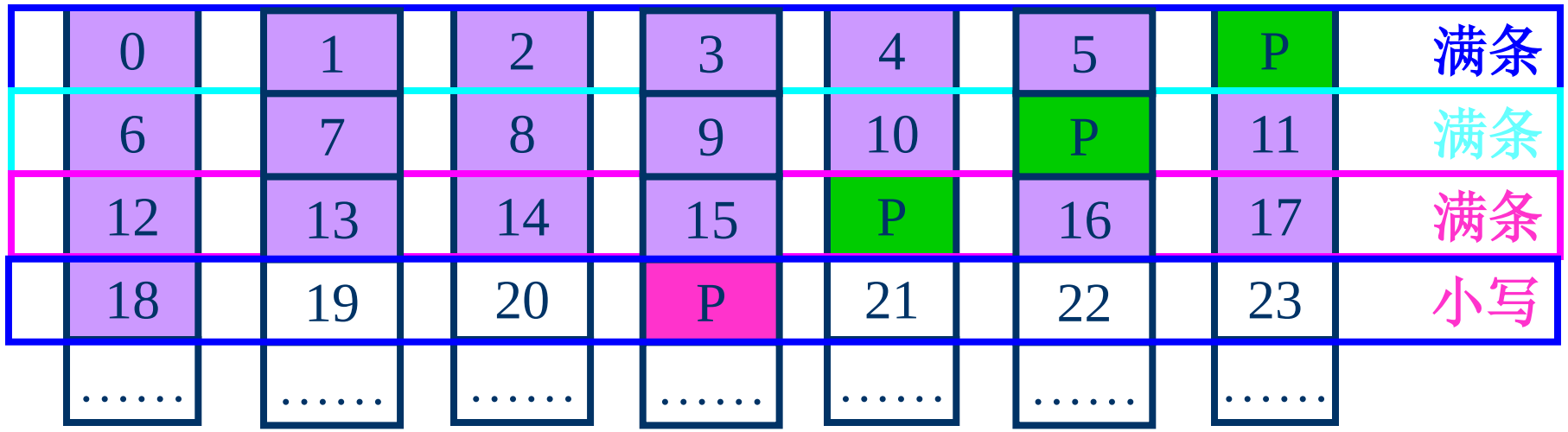
1. 计算校验
2. 写2，P，7块

1. 计算校验P
2. 写数据和校验命令(合并为1条命令)

3号盘上的子命令：



写命令：0x0a, 数据长度38k，起始地址为0



则要写的有效数据为图中的0～18块

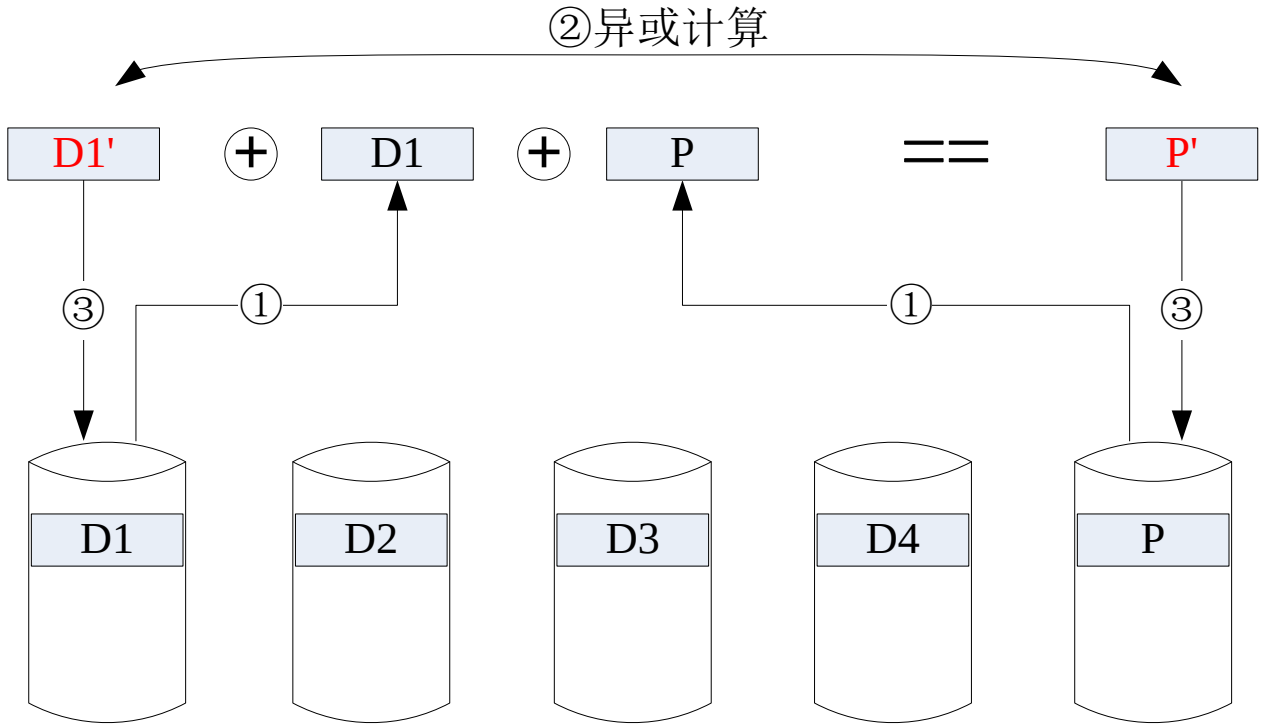
小写的过程(Read-Modify-Write, RMW)

- 读第18号块的旧数据
- 读图中红色的旧校验数据
- 计算校验：$P= D18_{old} \oplus D18 \oplus P_{old}$
- 写入新的数据D18和新的校验P

- P_{old}＝D1⊕D2⊕D3 ⊕...Dn
  P=D1⊕D1'⊕P_{old}
  P=D1⊕D1'⊕D1⊕D2⊕D3 ⊕...Dn
  P= D1'⊕D2⊕D3 ⊕...Dn = P_{new}

RAID5 Write Hole Problem

- 奇偶校验盘阵列（RAID5/6）中校验数据丢失（污染）问题（Parity Pollution）



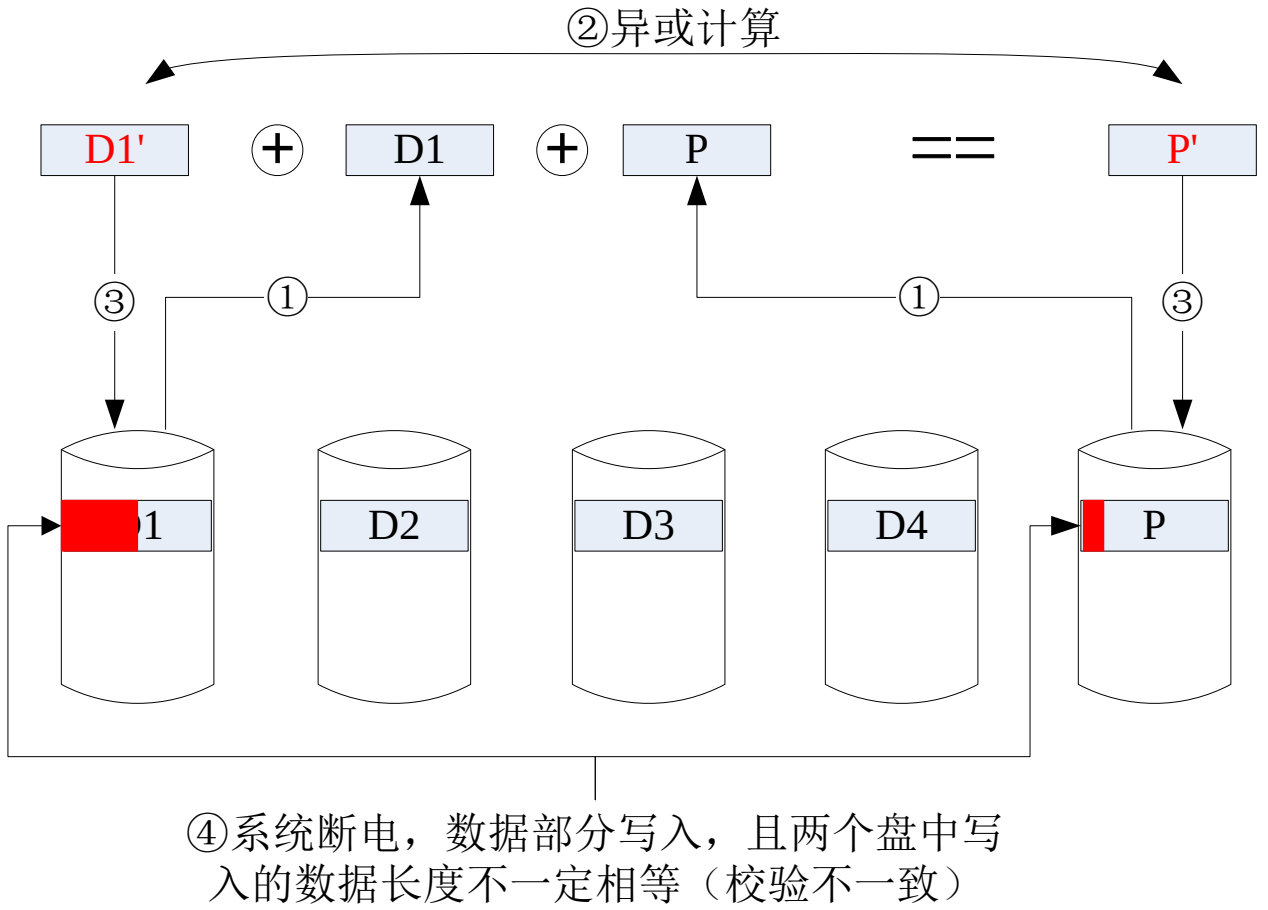
①：读取旧数据D1，旧校验P；

②：异或计算得到新的校验值P'；

③：写入新数据D1'和新校验值P’

RAID5 Write Hole Problem

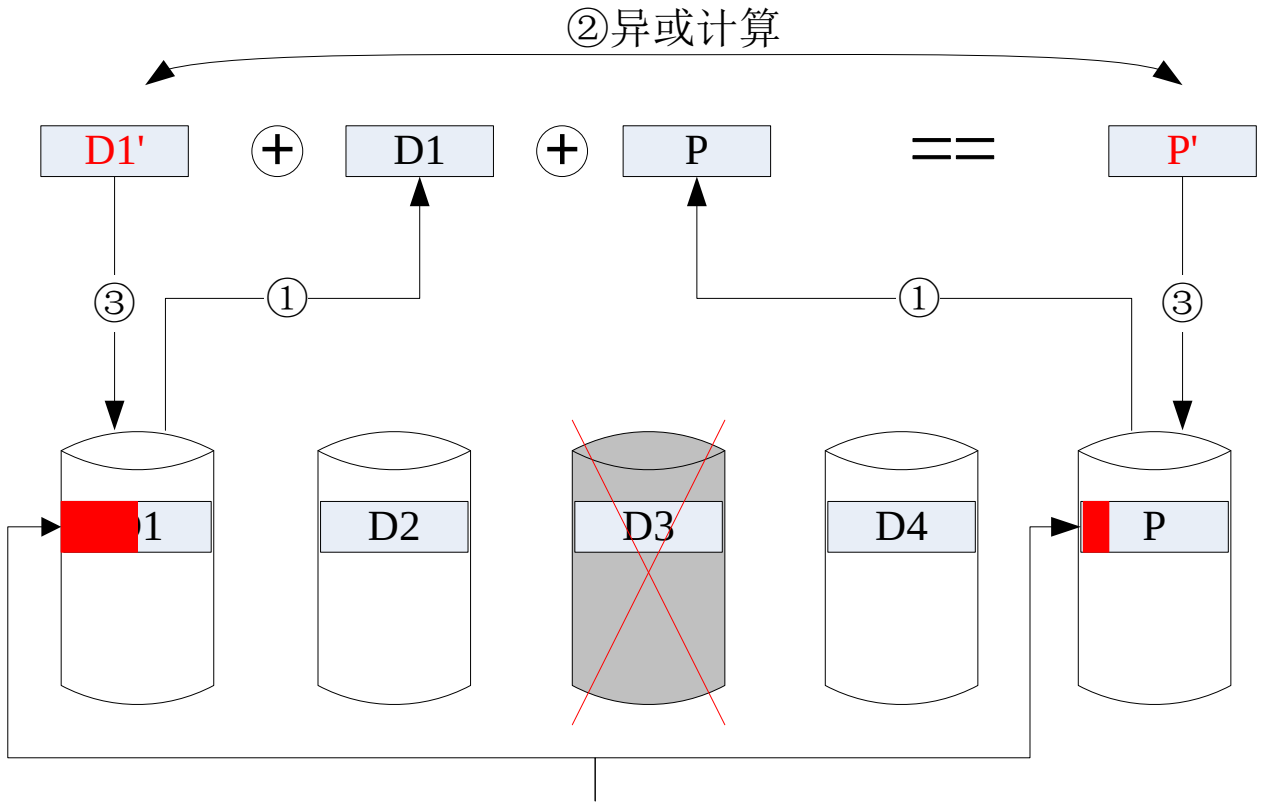
- 异常状态下（突然掉电）小写请求处理流程



⑤：系统重启后，再次写入D1‘，此时只有RCW (Read-Reconstruct-Write)机制才能恢复数据，否则会导致条带不一致现象，进而会丢失数据（潜在风险，再次发生磁盘故障时触发）

RAID5 Write Hole Problem

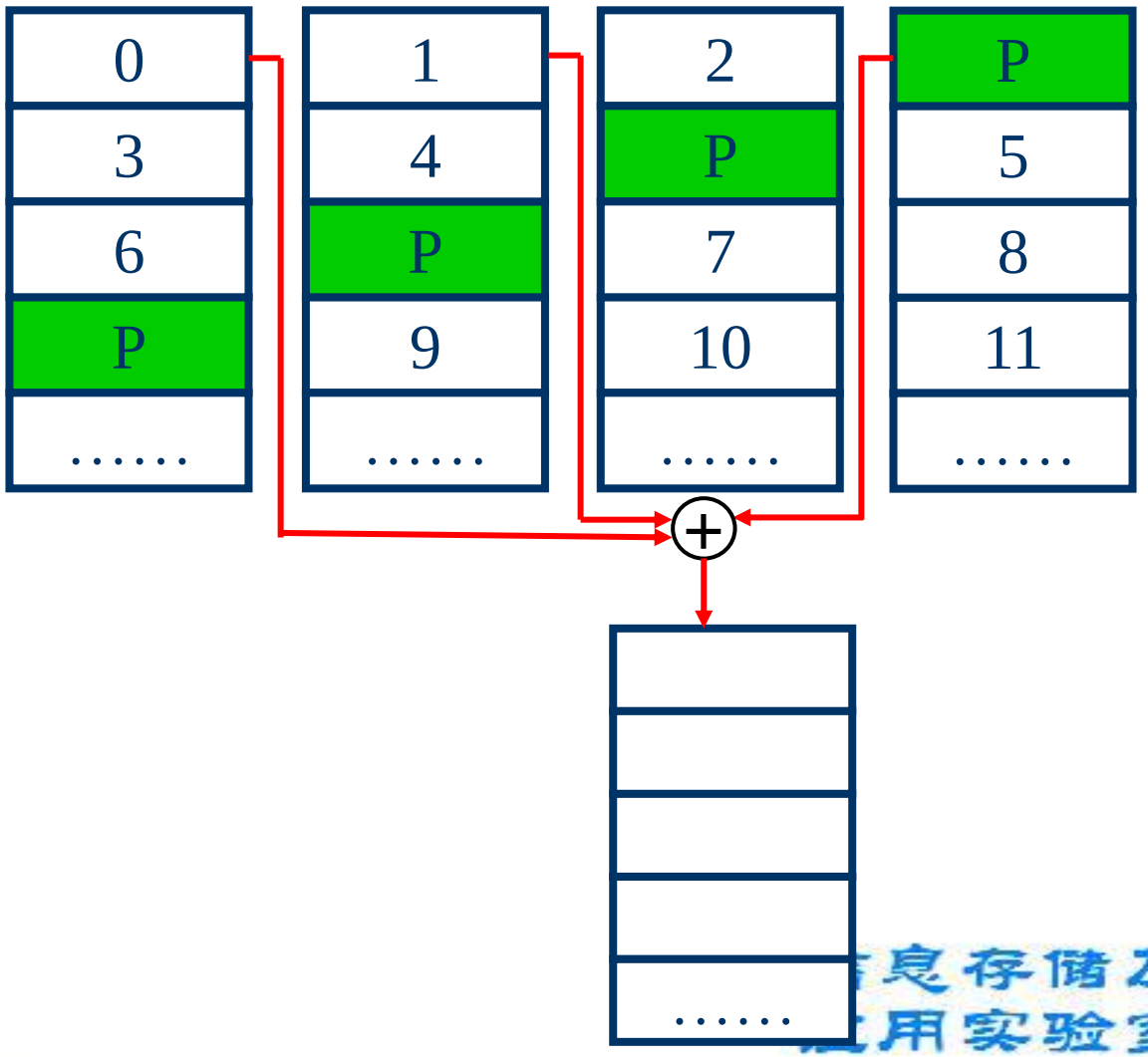
- 降级模式下（突然掉电）小写请求处理流程



⑤：系统重启后，如何恢复？？？

RAID5的重建与初始化

- 重建(Rebuild)：读出非故障盘上的数据，按照校验信息的计算方法计算得到故障盘上的数据，然后写入到新替换的盘上
- 重构（Reconstruct）：重建的微观操作过程



校验信息的正确完整性问题

- 磁盘上初始信息是随机的
- 满条块写和大写(RCW)可以保证校验信息的正确性
- 若原始校验信息不正确，则小写不能保证校验信息的正确，例如：

1 0 1 0 1 1 1 0 （正确值1）
0 0 1 0 1 1 1 ? （应该是0）

根据小写计算得： 0 ⊕ 1 ⊕ 0 ＝ 1

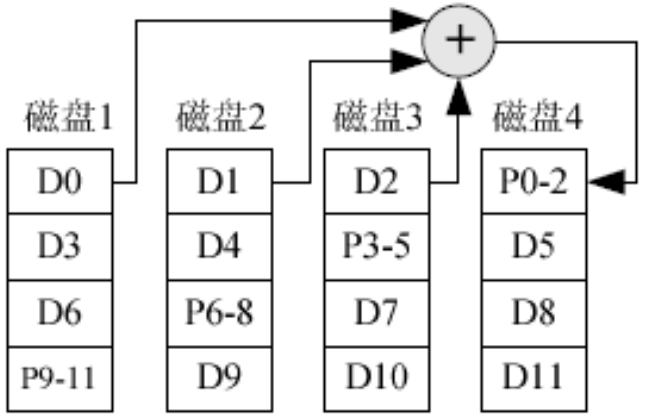
若刚写入的盘出现了故障，需要数据恢复：

恢复的数据 = 0 ⊕ 1 ⊕ 0 ⊕ 1 ⊕ 1 ⊕ 1 ⊕ 1 = 1

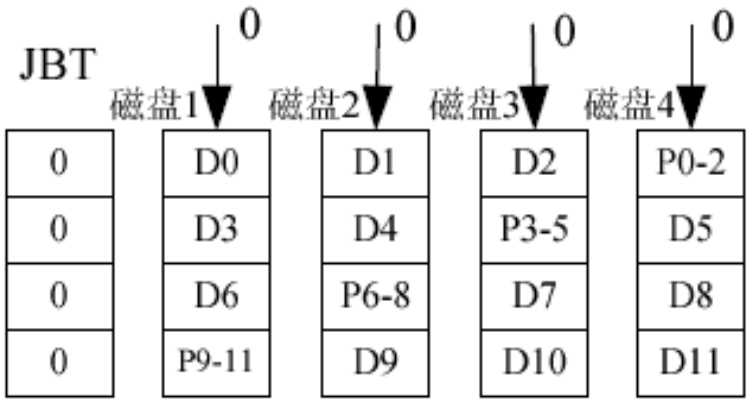
依据错误的校验信息进行重建得出的数据也是不正确的，所以需要**初始化**过程

RAID5 初始化过程

- 原则：保证校验信息一致正确
- 方法：
  - 计算校验
  - 全部写“0”



(a) 传统的数据同步过程



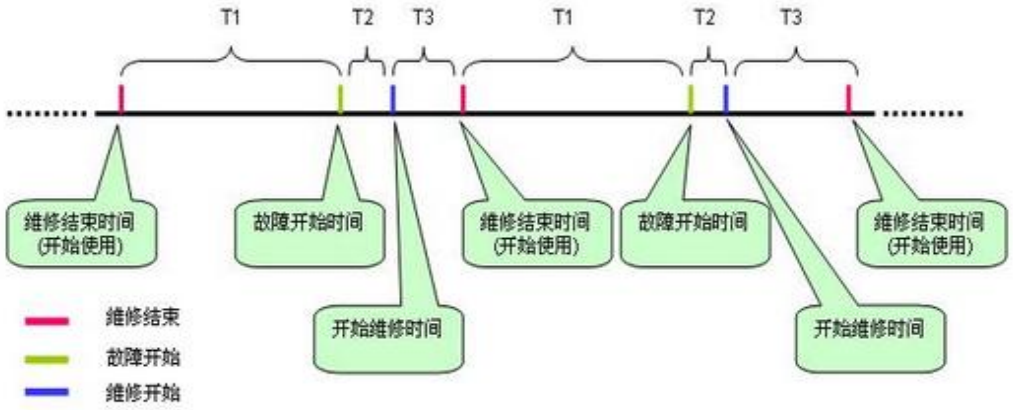
(b) 改进的数据同步过程

RAID高可靠、高可用技术

评估磁盘阵列可用性的方法

- 从SLA(Service-Level Agreement)角度看，系统在两种状态间切换：
  - 服务完成，即服务按照指定的SLA交付；
  - 服务中断，即提交的服务不满足SLA的要求。
- 评估指标：
  - 可靠性：评估服务完成，指从初始状态起，持续完成服务的时间，一般用MTTF评估；
  - 可用性：也评估服务完成，当系统从“完成”切换到“中断”时，同样认为系统可用性降低；传统的磁盘阵列可用性形式化定义为MTTF和MTTR的函数：

MTTF = T1
MTTR = T2 + T3
MTBF = T1+T2+T3



$$\text{可用性}= \frac{MTTF}{MTTF + MTTR} \times 100\%$$

5个9
99.999%：Downtime per Year 5.256 min

RAID高可靠、高可用技术

提高磁盘阵列可用性的相关技术

- 磁盘阵列数据重建算法研究
  - 数据布局的重新组织
  - 重建工作流优化
  - 重建顺序优化
  - 理论研究及其它

RAID高可靠、高可用技术

问题：

- 存储系统规模变大，磁盘故障成为经常性事件，RAID重建过程过长，系统可靠性成为问题
- 磁盘阵列在线重建时，用户应用请求和重建请求争夺磁盘资源，用户应用请求将延长重建时间，降低重建效率，而重建又会影响用户应用请求，降低用户应用的性能

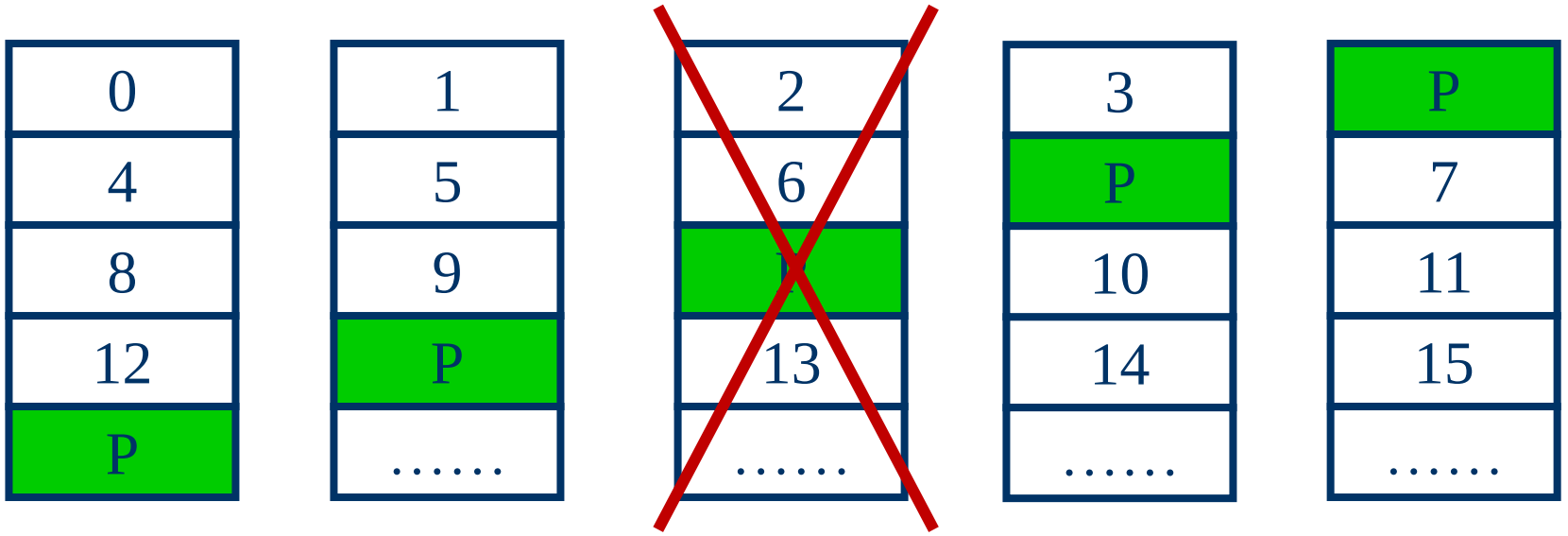
基于热度的多线程重建调度优化算法PRO

——FAST’ 07

基于I/O负载重定向的重建优化算法WorkOut

——FAST’ 09

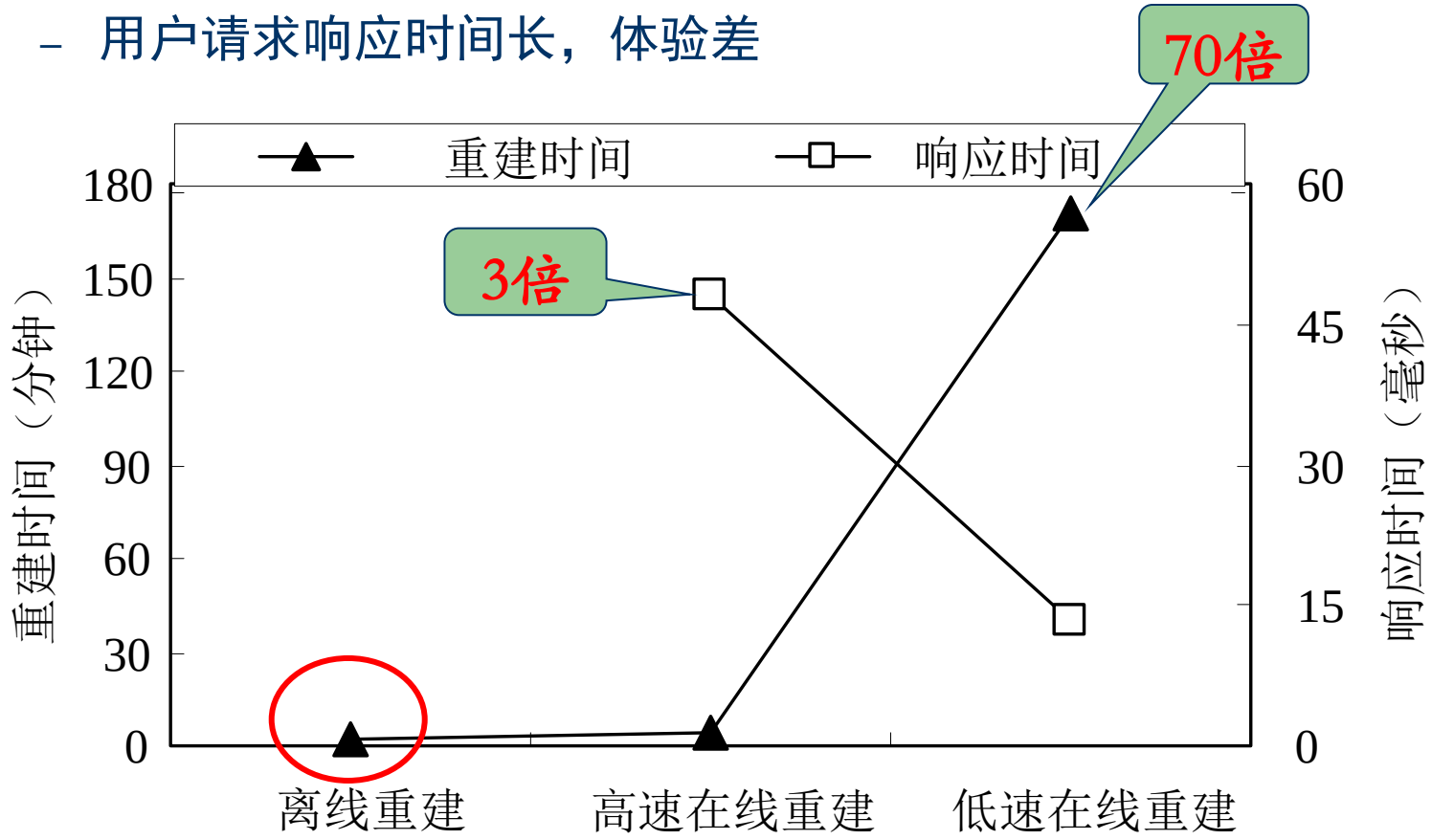
RAID5



- 重建IO
- 业务IO：
  - ✓ 读：读故障盘上的数据，读正常盘上的数据
  - ✓ 写：满条带写，非满条带写

重建过程的优化方法（PRO）

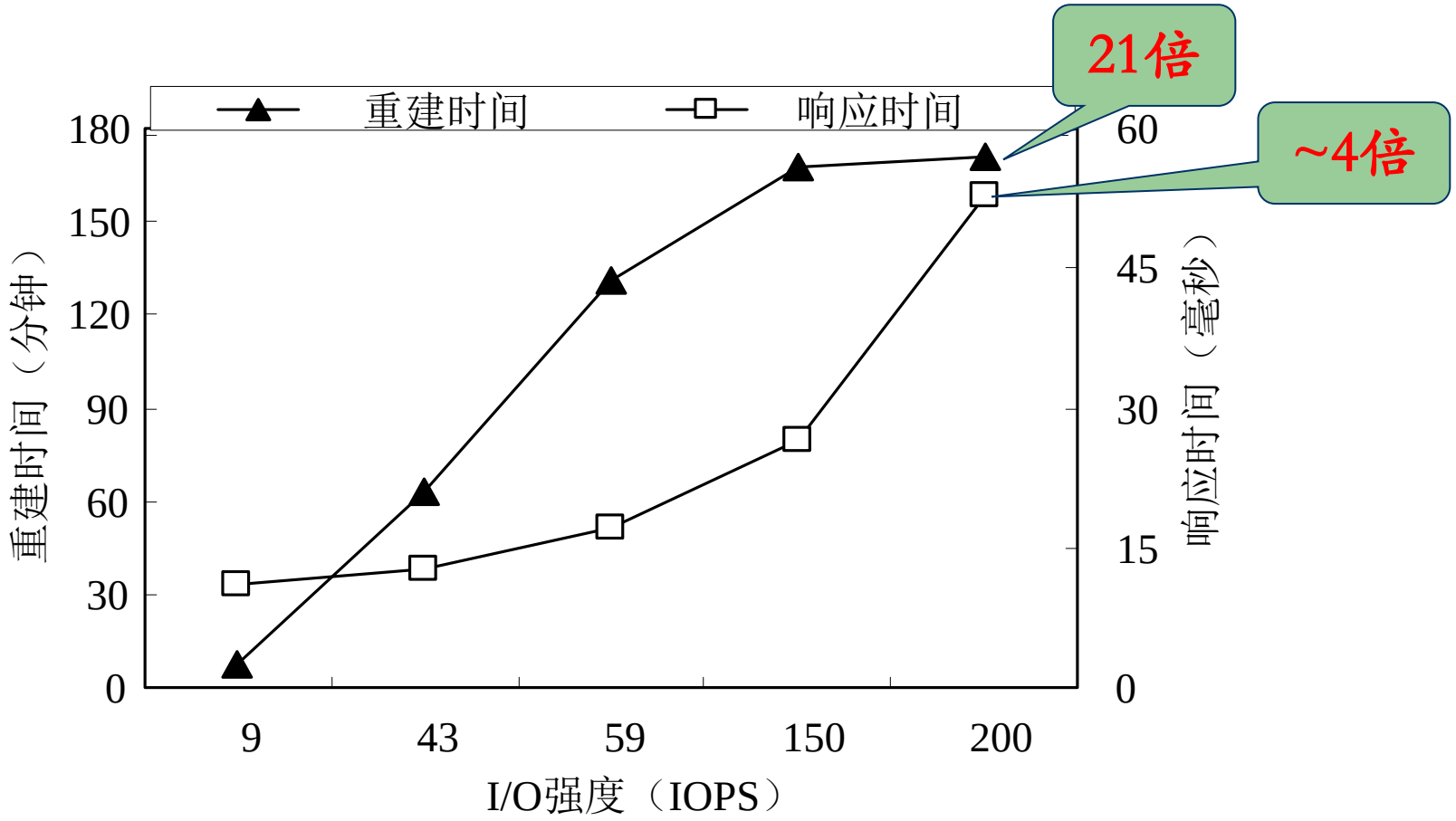
- 重建面临的挑战：重建I/O与用户I/O争用资源
  - 时间长，速度慢
  - 用户请求响应时间长，体验差



重建I/O和用户I/O的相互影响

重建过程的优化方法（PRO）

用户I/O强度对在线重建的影响



重建时间和用户I/O响应时间都随着用户请求的强度的增加而增加

重建过程的优化方法（PRO）

- 重建面临的挑战：**重建I/O与用户I/O争用资源**
  - 时间长，速度慢
  - 用户请求响应时间长，体验差
- **解决办法1**
  - **利用20－80原理**
  - **热点数据区优先重建，减少磁头移动**
  - **减少IO请求的延迟，加快重建速度**

Lei Tian，et al. "PRO: A Popularity-based Multi-threaded Reconstruction Optimization for RAID-Structured Storage Systems". In Proceedings of the 5th USENIX Conference on File and Storage (FAST'07). pp. 277~290, 2007

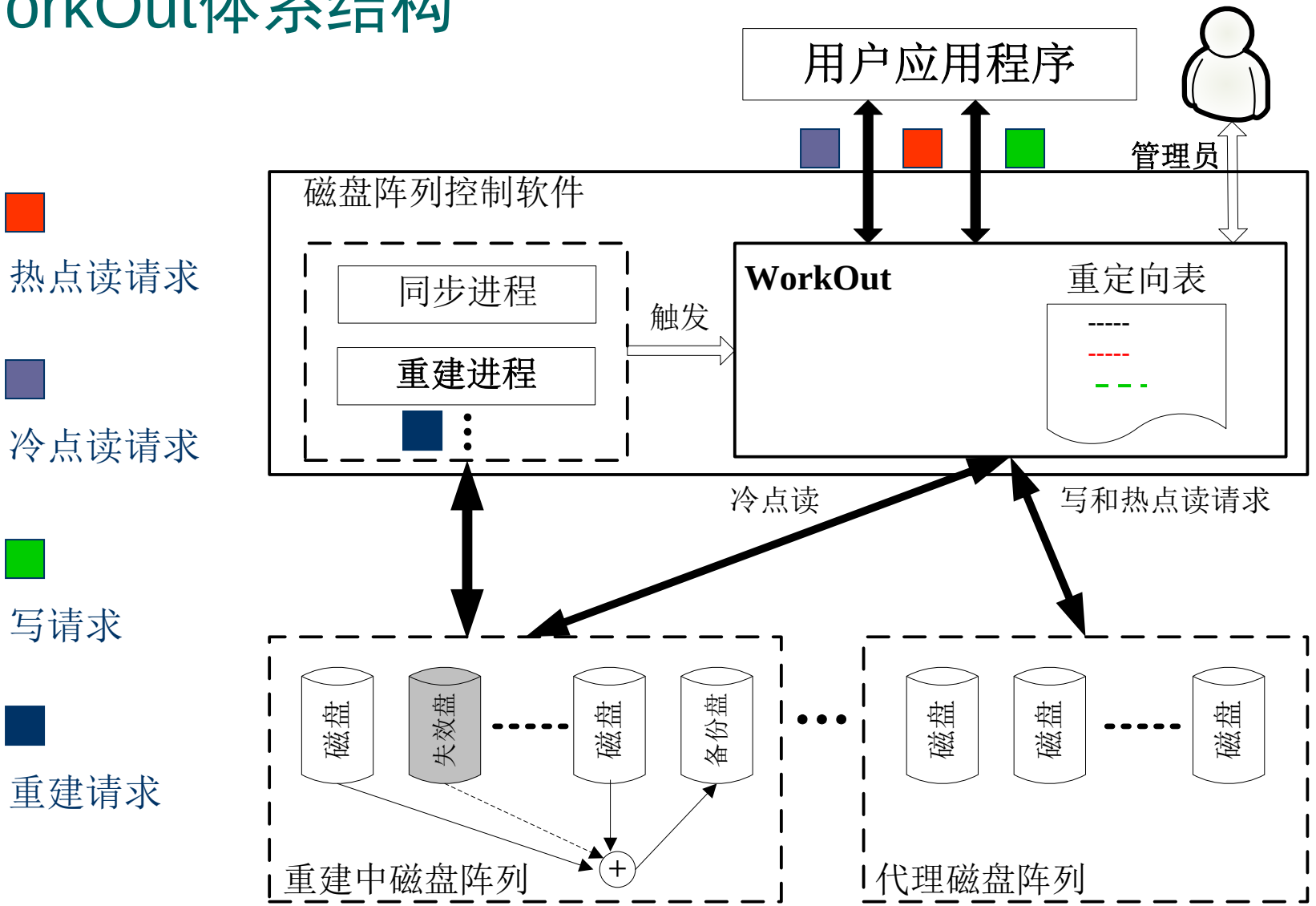
解决办法2：

I/O负载重定向的重建算法（WorkOut）

- *基本思想*
  - 通过将重建中磁盘阵列收到的所有写请求和热的读请求重定向到代理磁盘阵列中，从而减少重建中磁盘阵列的用户I/O负载强度，使重建进程得到更多的磁盘资源，进而提高了重建效率
- *目标*
  - 提高数据中心中存储系统的可靠性和可用性。即通过I/O负载重定向技术减少磁盘阵列的数据恢复时间，同时减少重建进程对用户I/O响应性能的影响

Suzhen Wu, et al. **Improving Availability of RAID-Structured Storage Systems by Workload Outsourcing**. *IEEE Transactions on Computers*. 60(1):64-79, 2011

WorkOut体系结构



解决办法3：

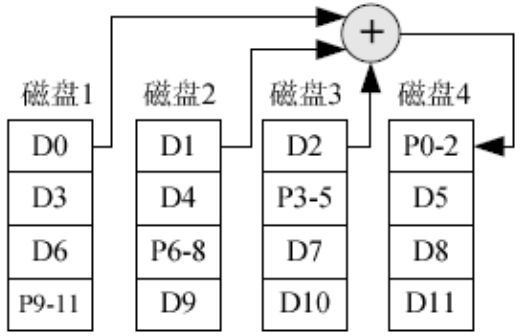
基于日志的重建

- 原理：
  - 初始化全部写0
  - 存储空间的空闲达50%，相当一部分空间一次都没修改过
  - 记录修改过的条块，重建的时候，对修改的条块重新计算，未修改的部分直接写0

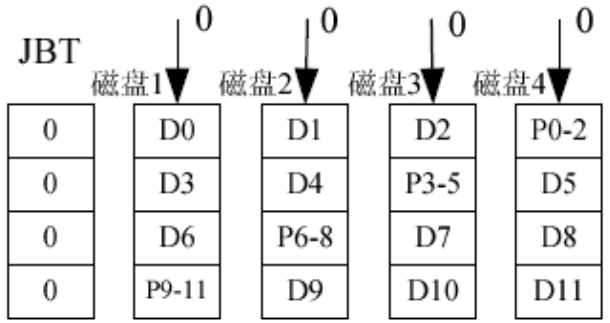
Suzhen Wu, et al. JOR: A Journal-guided Reconstruction Optimization for RAID-Structured Storage Systems. In Proceedings of the 15th International Conference on Parallel and Distributed Systems (ICPADS'09)

解决办法3：

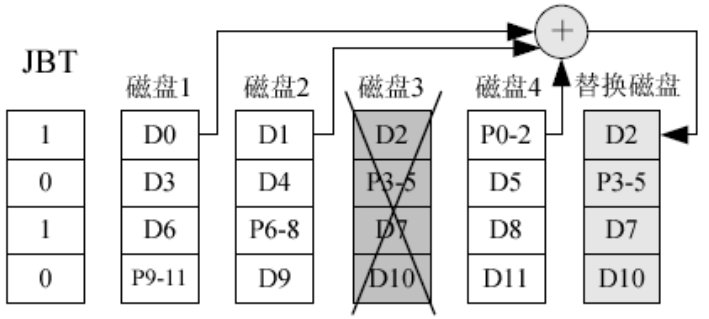
基于日志的重建



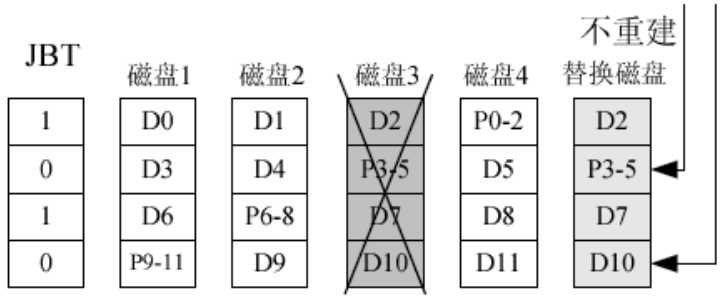
(a) 传统的数据同步过程



(b) 改进的数据同步过程



(a) 当bitmap=1时的重建过程



(b) 当bitmap=0时的重建过程

企业需求实例：

1.2TB的数据，30分钟内要求完成重建:

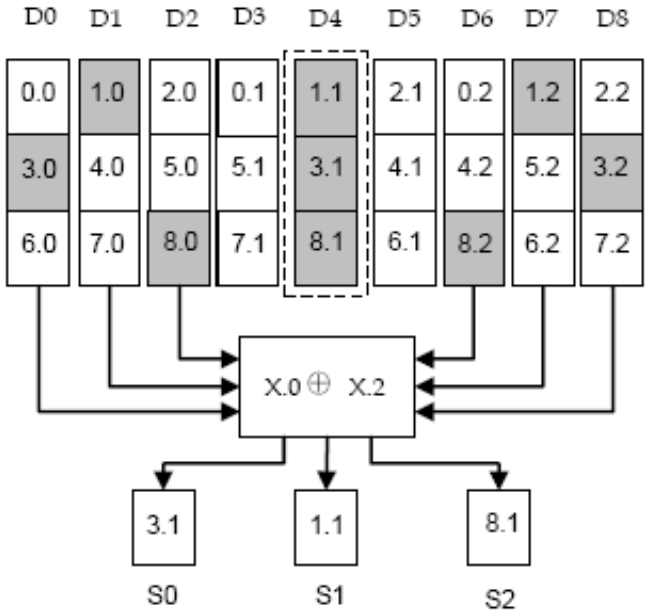
1.2TB/30*60s = 667MB/s

解决途径:

多设备并行重建

S²-RAID:
Parallel RAID Architecture for Fast Data Recovery

- The idea is to divide each large disk in the RAID into small partitions. Partitions on these disks form sub-arrays. The sub-arrays are skewed among the disks in the RAID in such a way that conflict-free parallelism is achieved during a RAID reconstruction when any disk fails.
- Recovered data that was on the failed disk is stored in parallel on multiple disks consisting of spare disks and available space of good disks.



Jiguang Wan, Jibin Wang, Changsheng Xie, Qing Yang, "S²-RAID: Parallel RAID Architecture for Fast Data Recovery," *IEEE Transactions on Parallel and Distributed Systems*, 20 Nov. 2013.