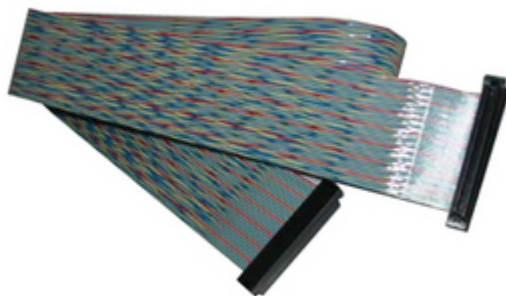
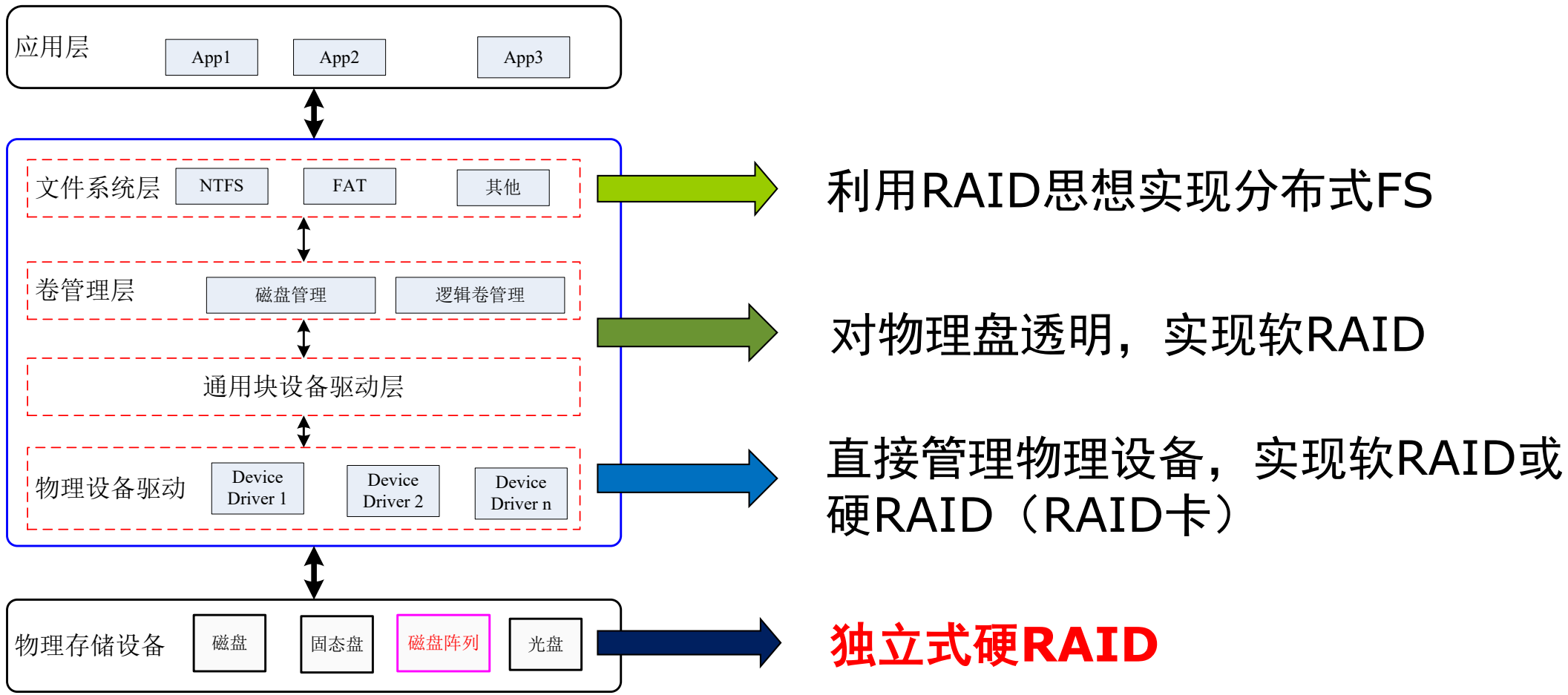


RAID & SCSI

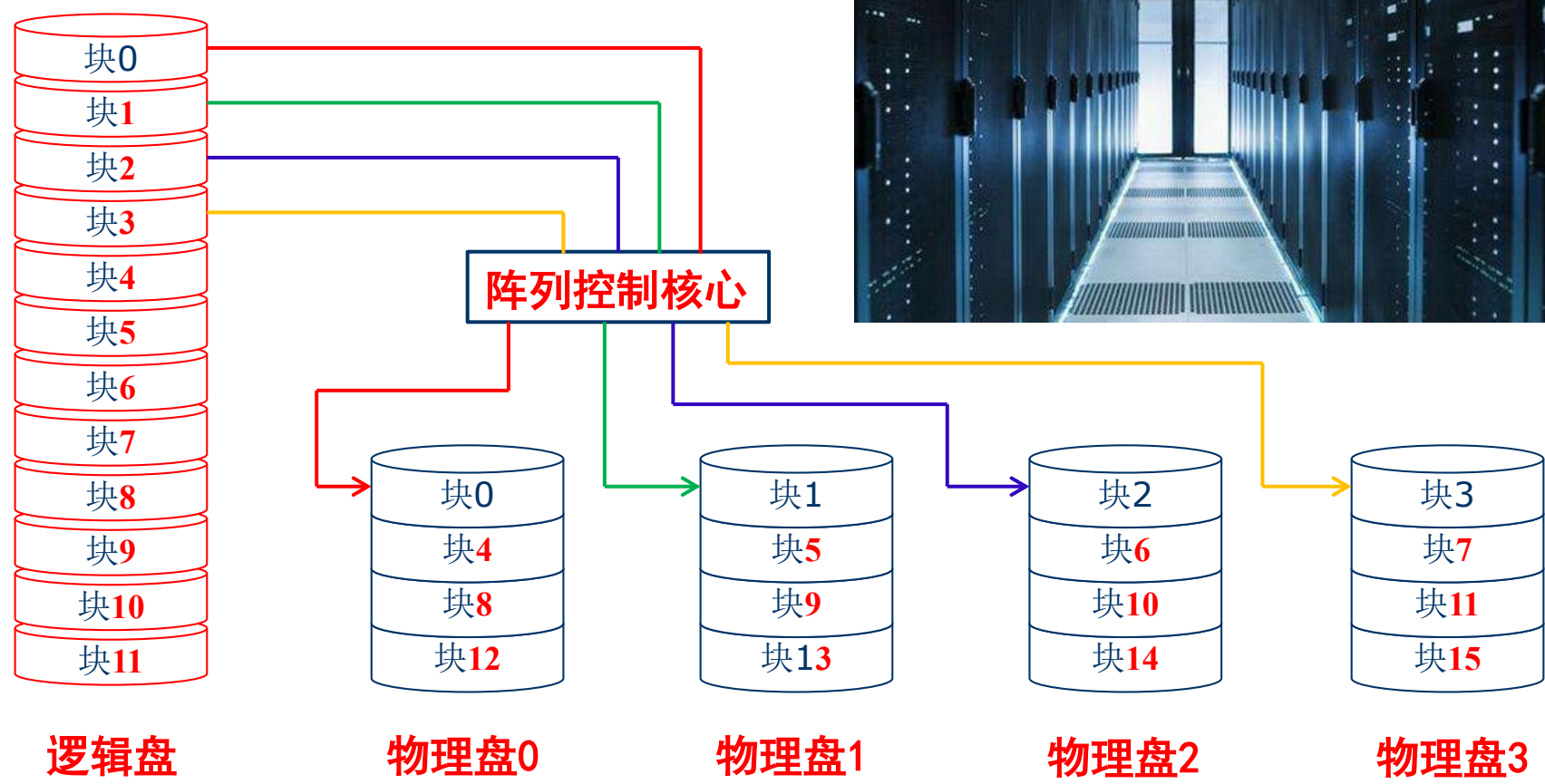


RAID的实现

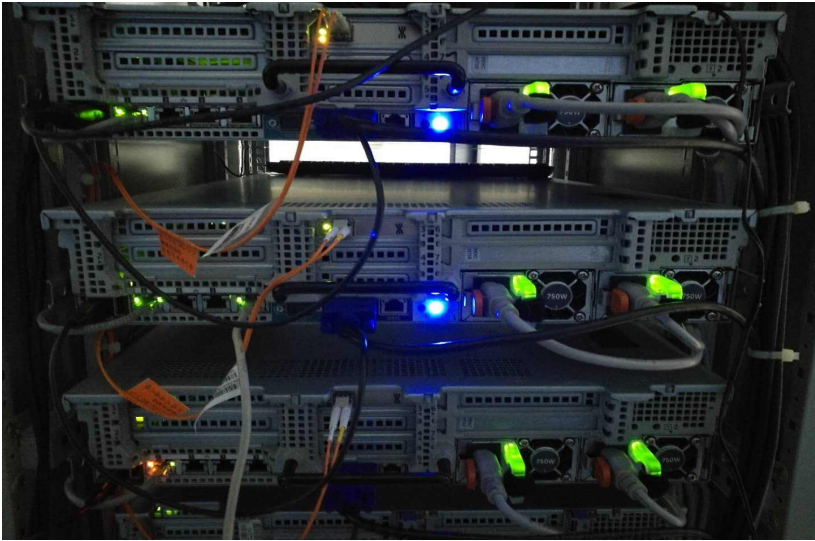
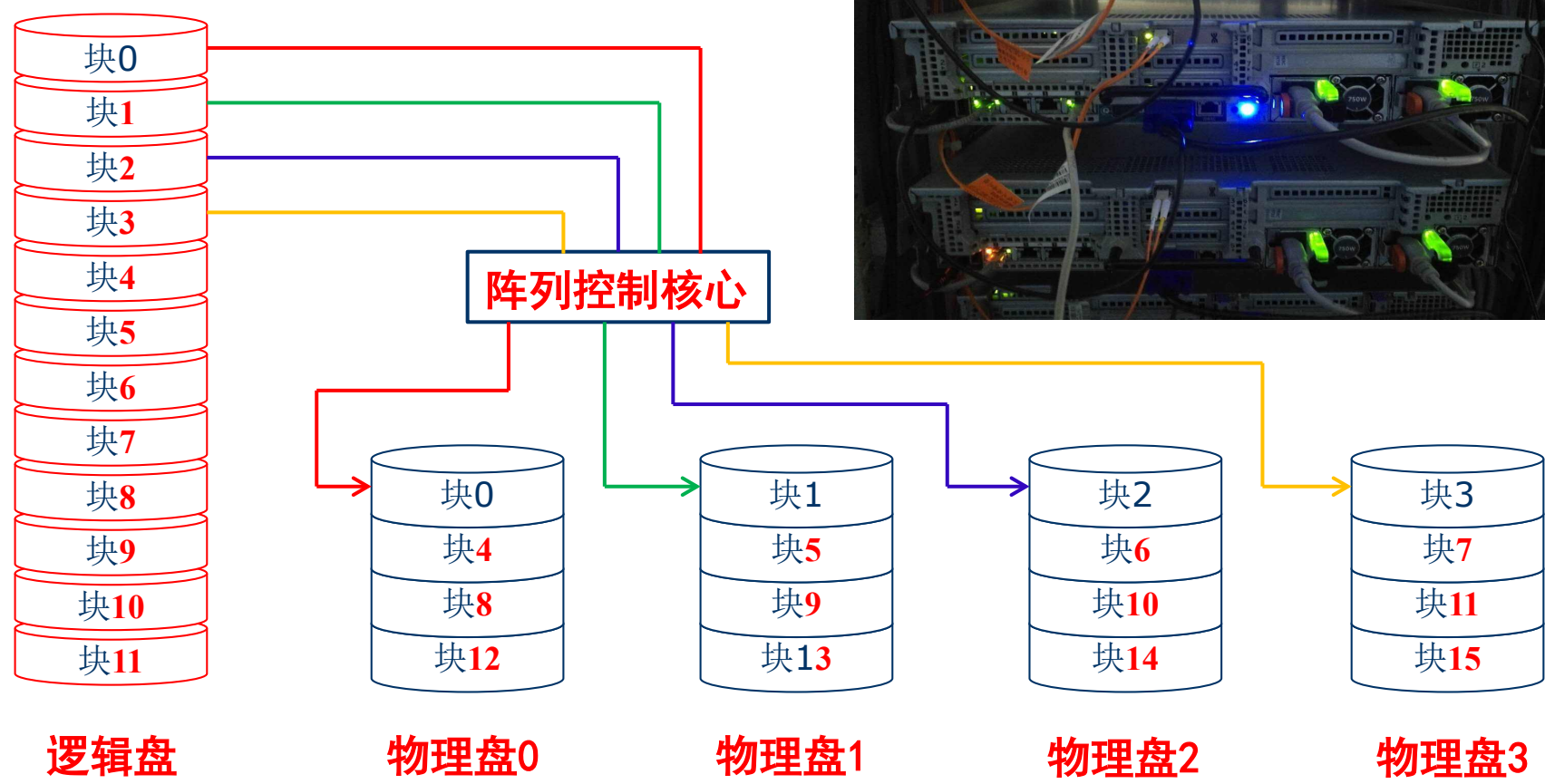


RAID是一种存储设备，是一项技术，更是一种思想

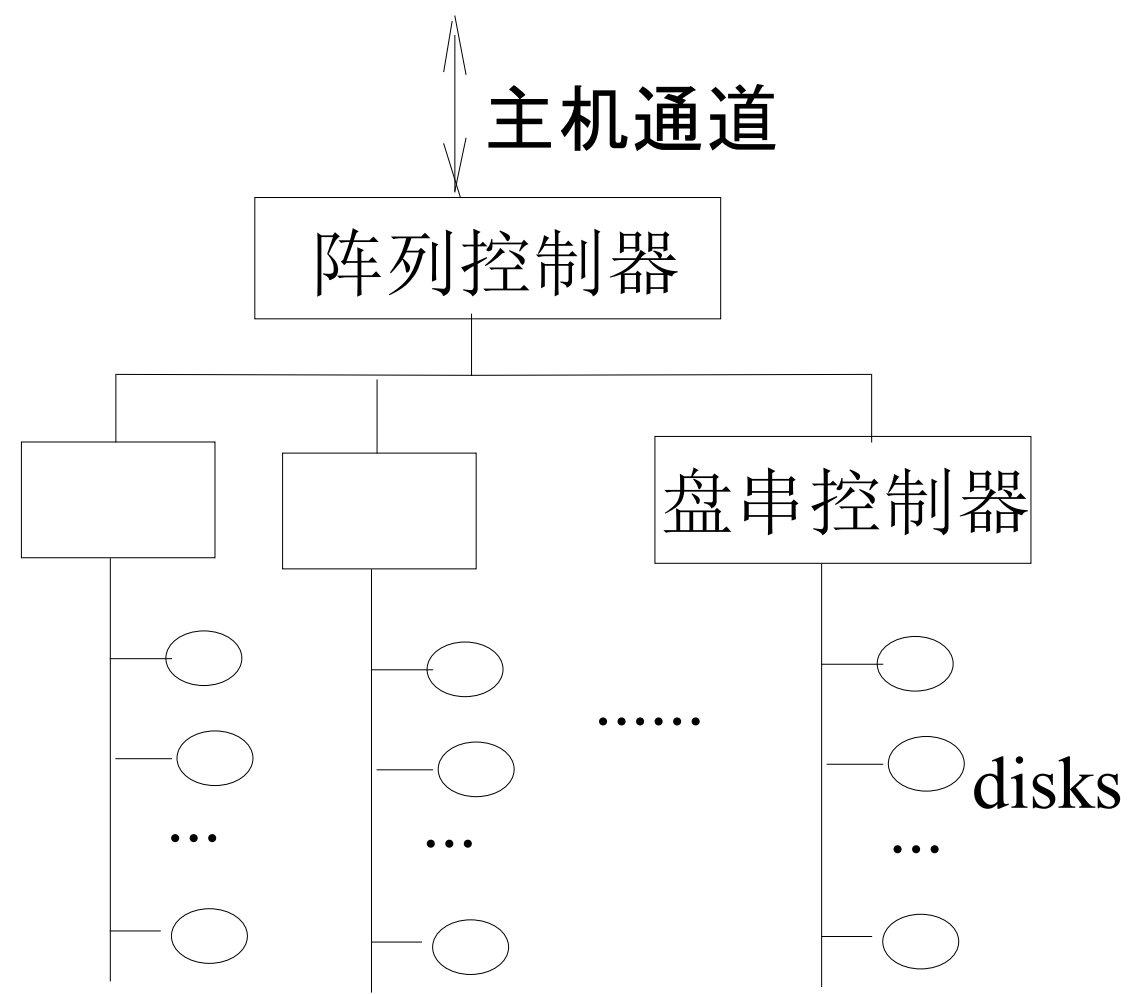
RAID逻辑上构成一个盘设备



RAID逻辑上构成一个盘设备

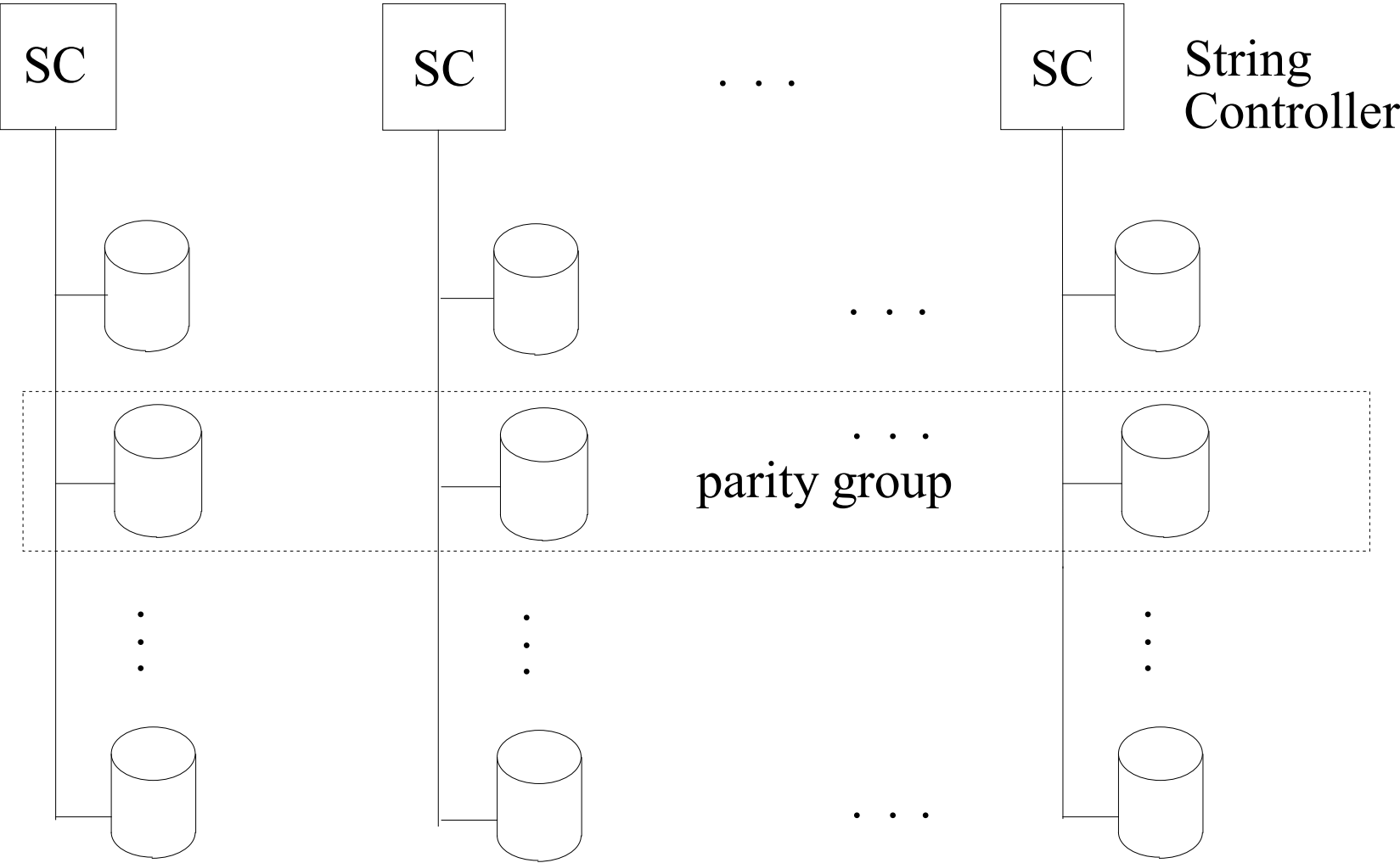


磁盘阵列控制器设计

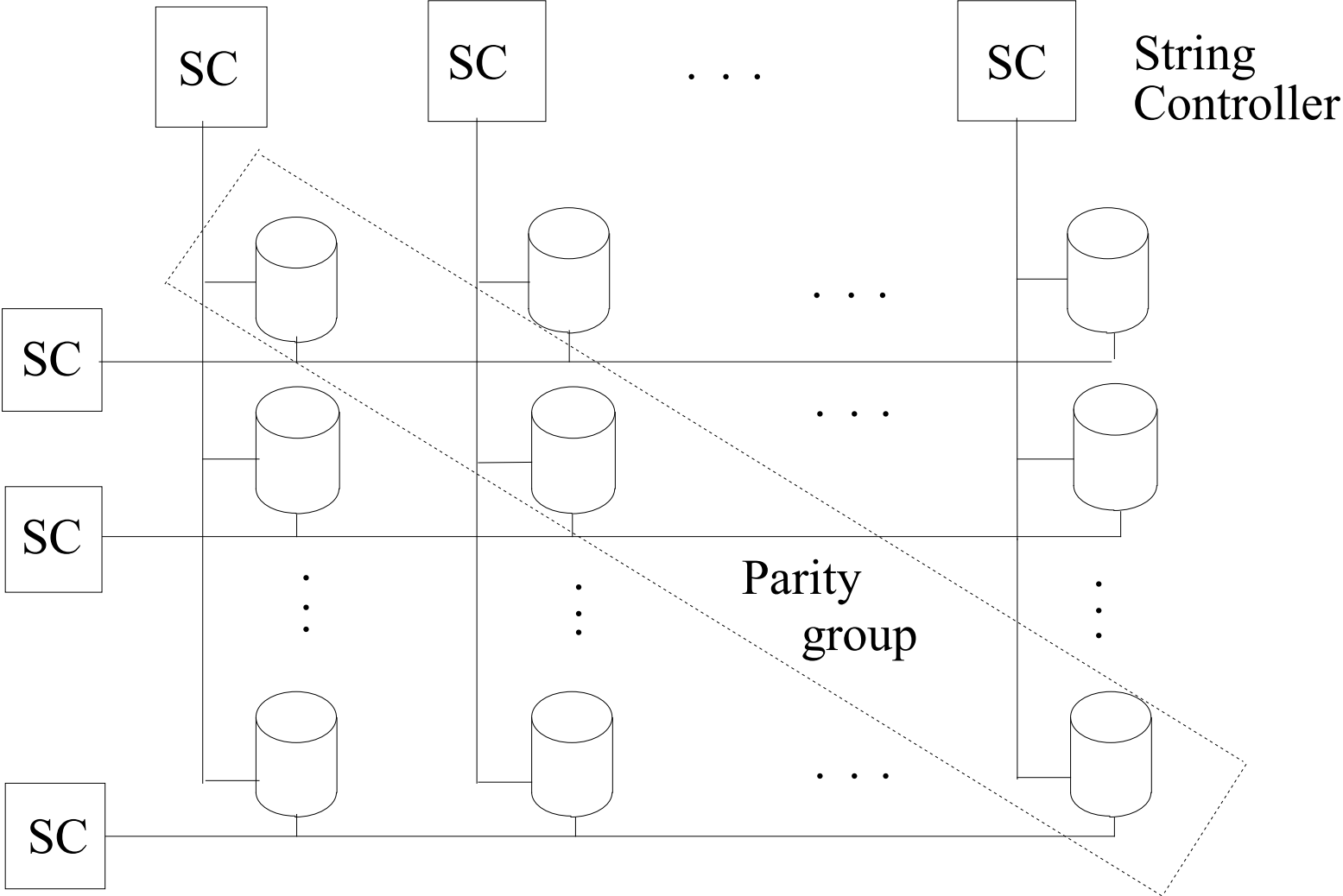


集中控制式阵列

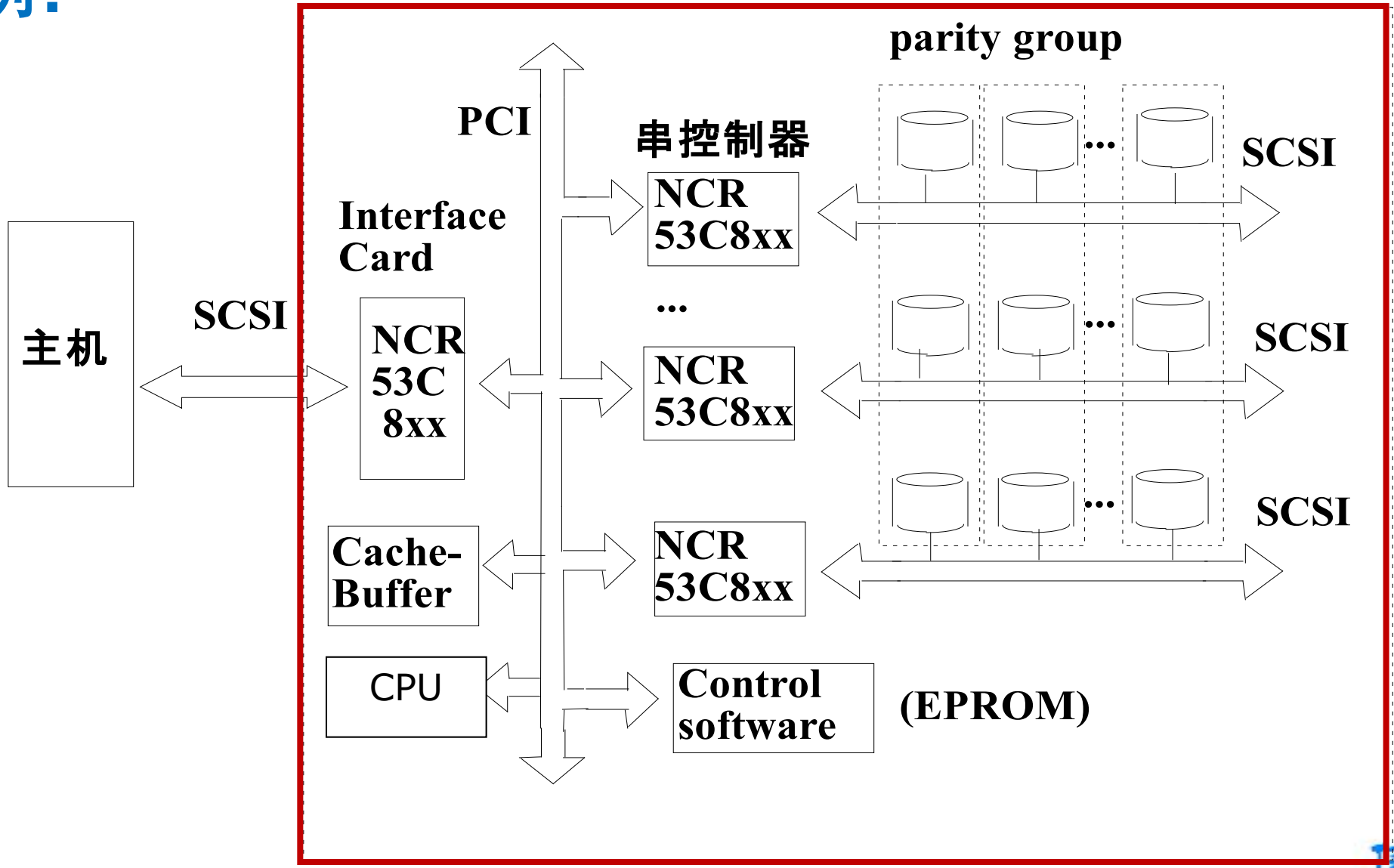
盘阵列正交结构示意图



Crosshatch盘阵列结构



实例：



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？

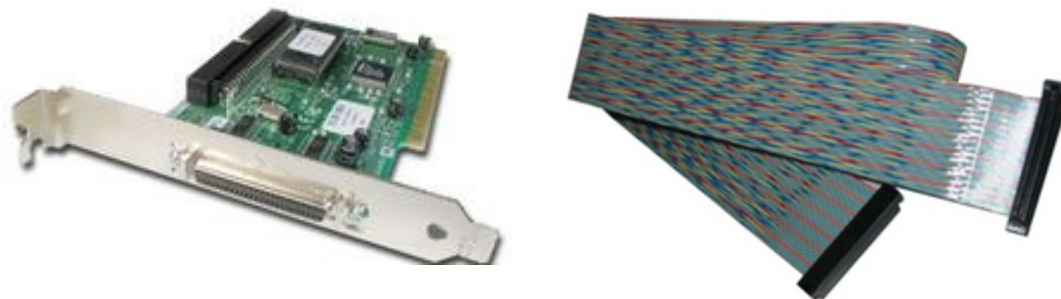
SCSI协议

支撑存储发展的脊梁

- **SCSI**: **S**mall **C**omputer **S**ystem **I**nterface, 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：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

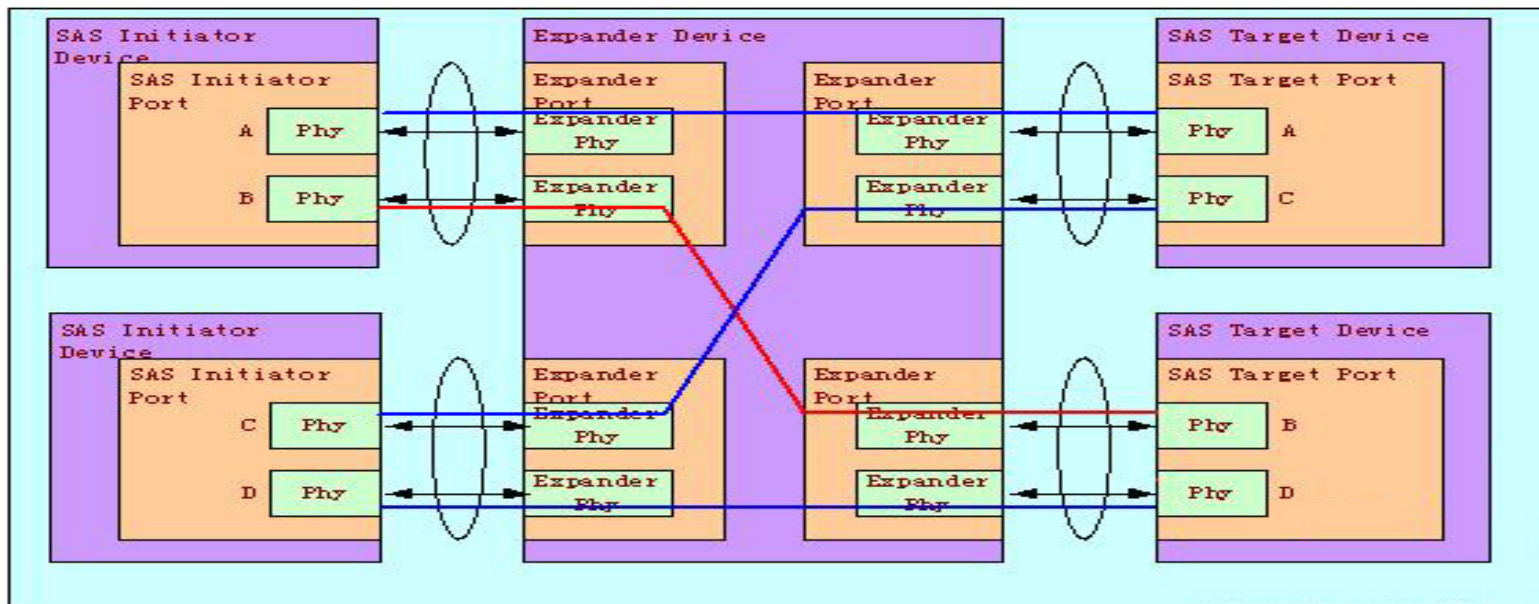
SAS(Serial Attached SCSI)简介

- 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

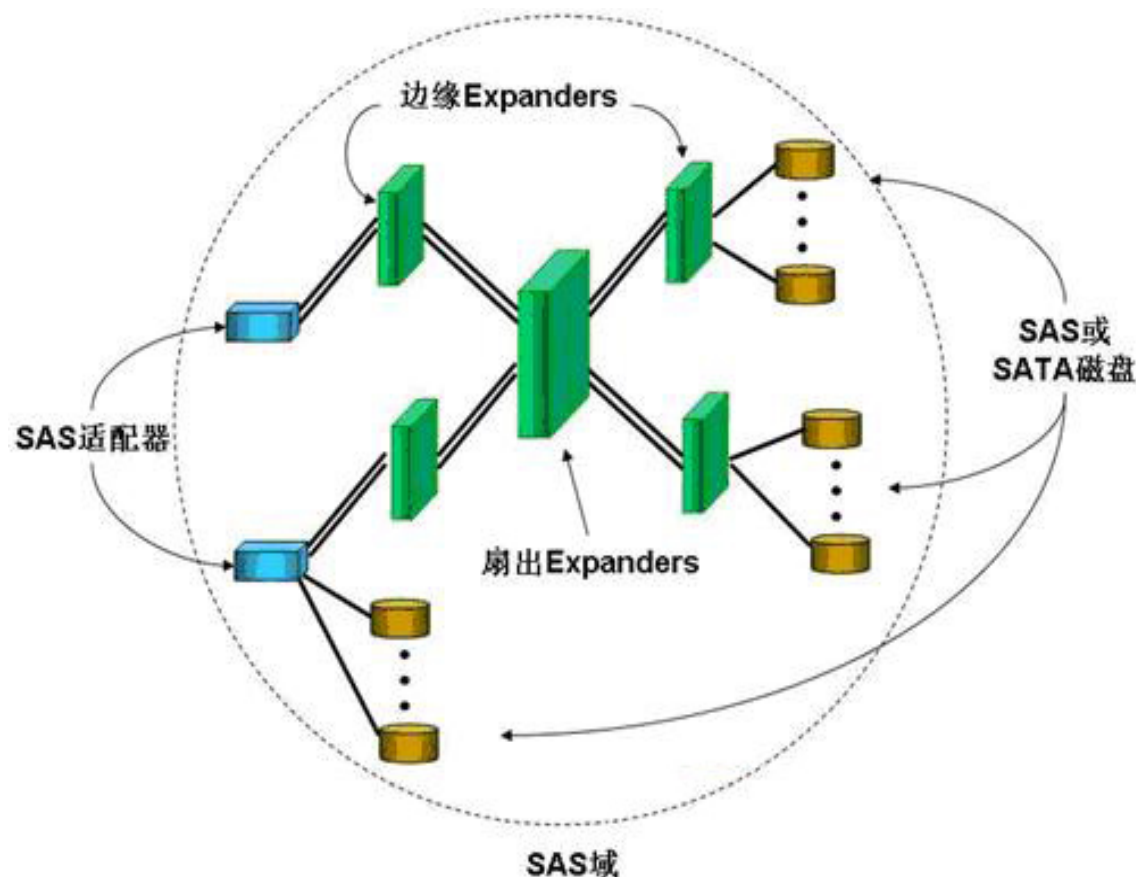
- SAS域

- 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

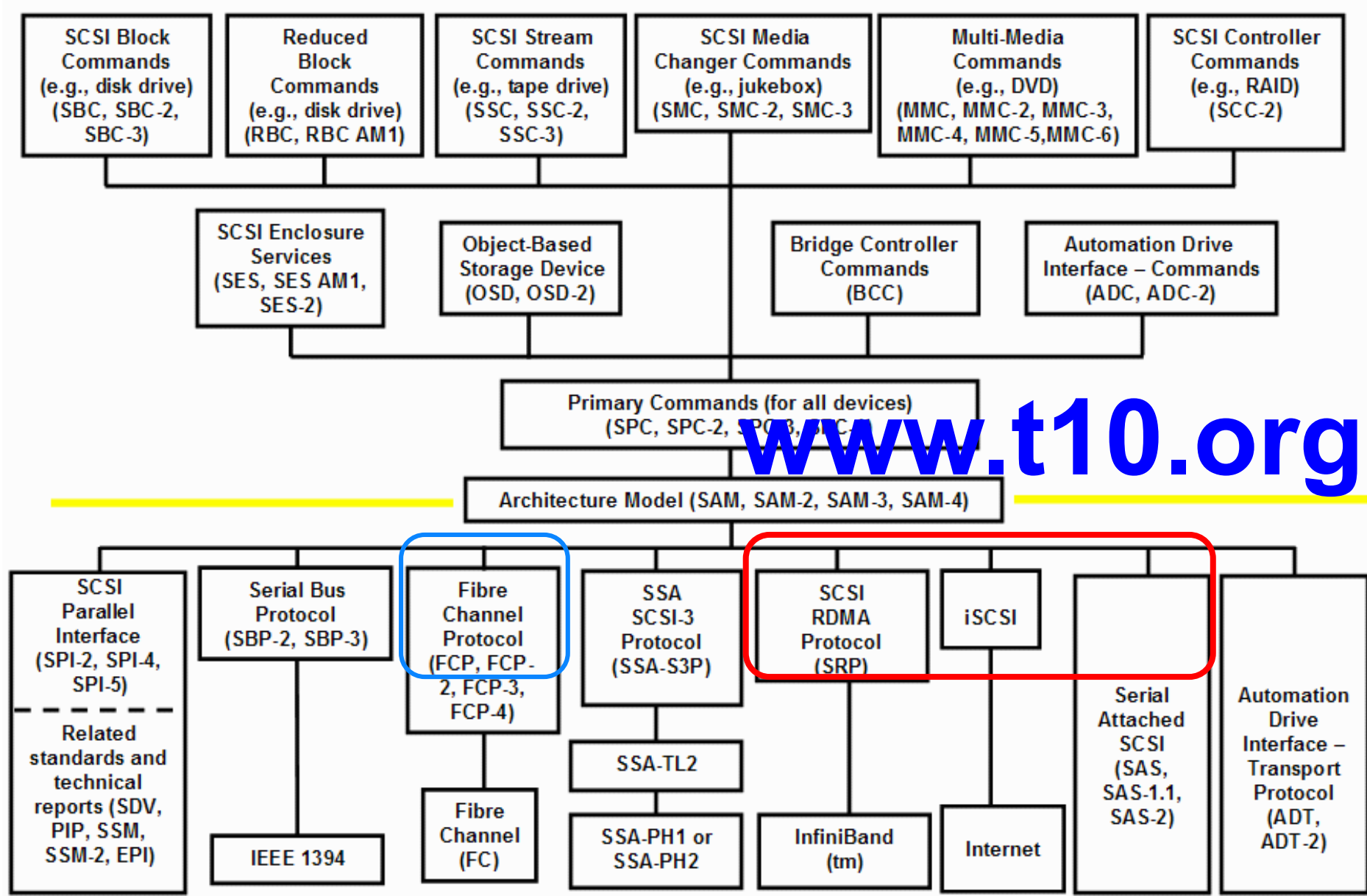


SAS连接

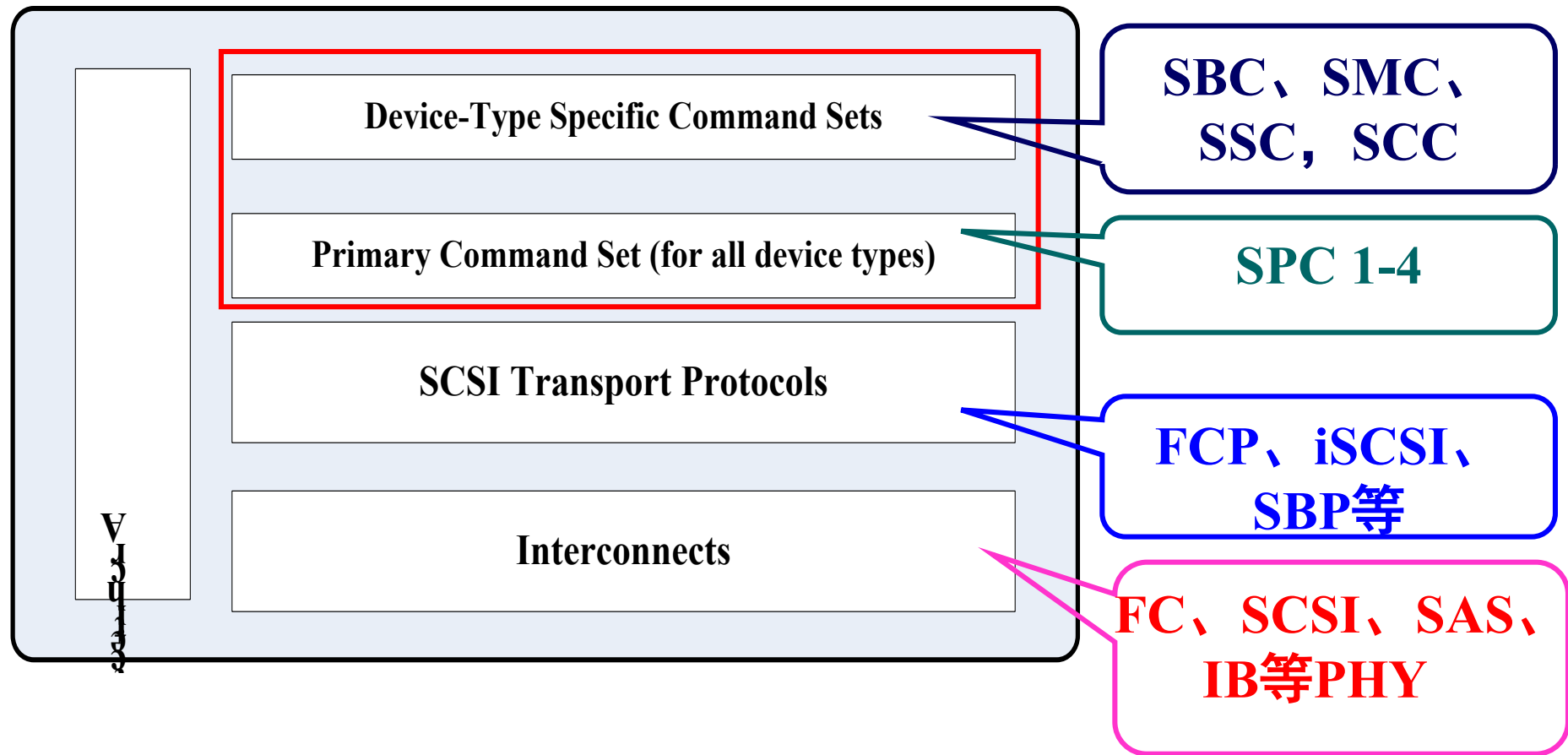
● 连接实例



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 $128 \times 128 = 16384$ 个端口。当然，内部互联至少要占用若干个端口。



SAM-3结构

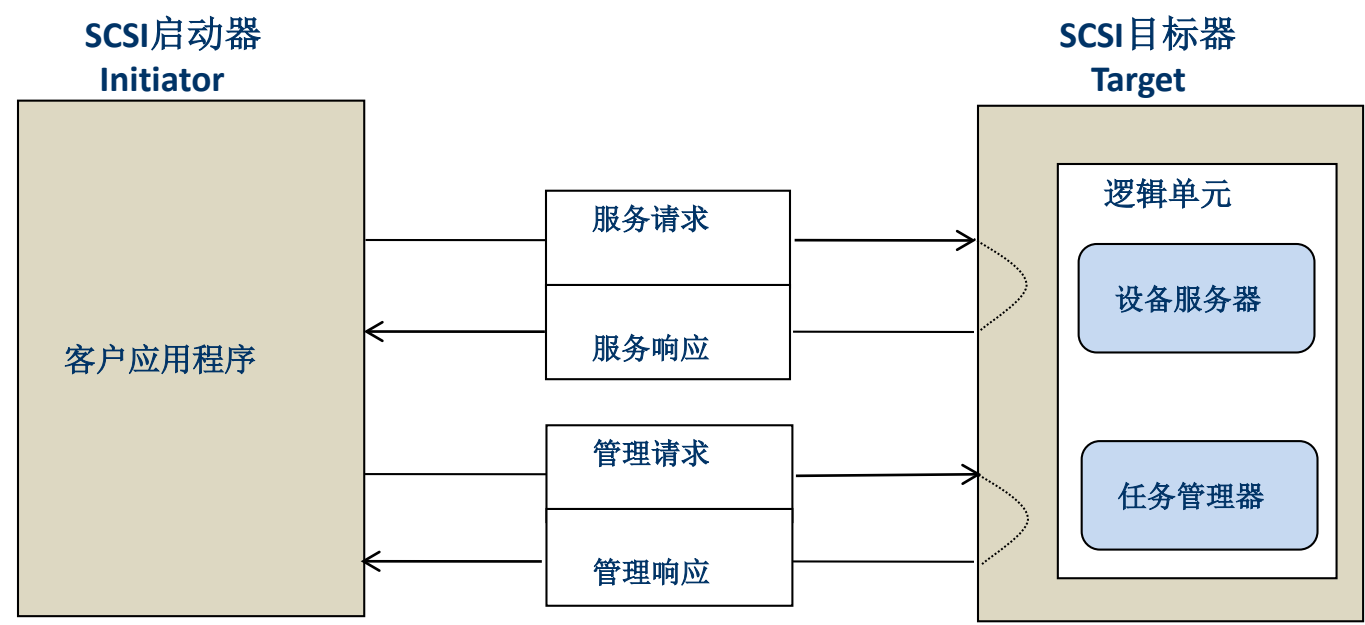


SCSI在系统中的地位

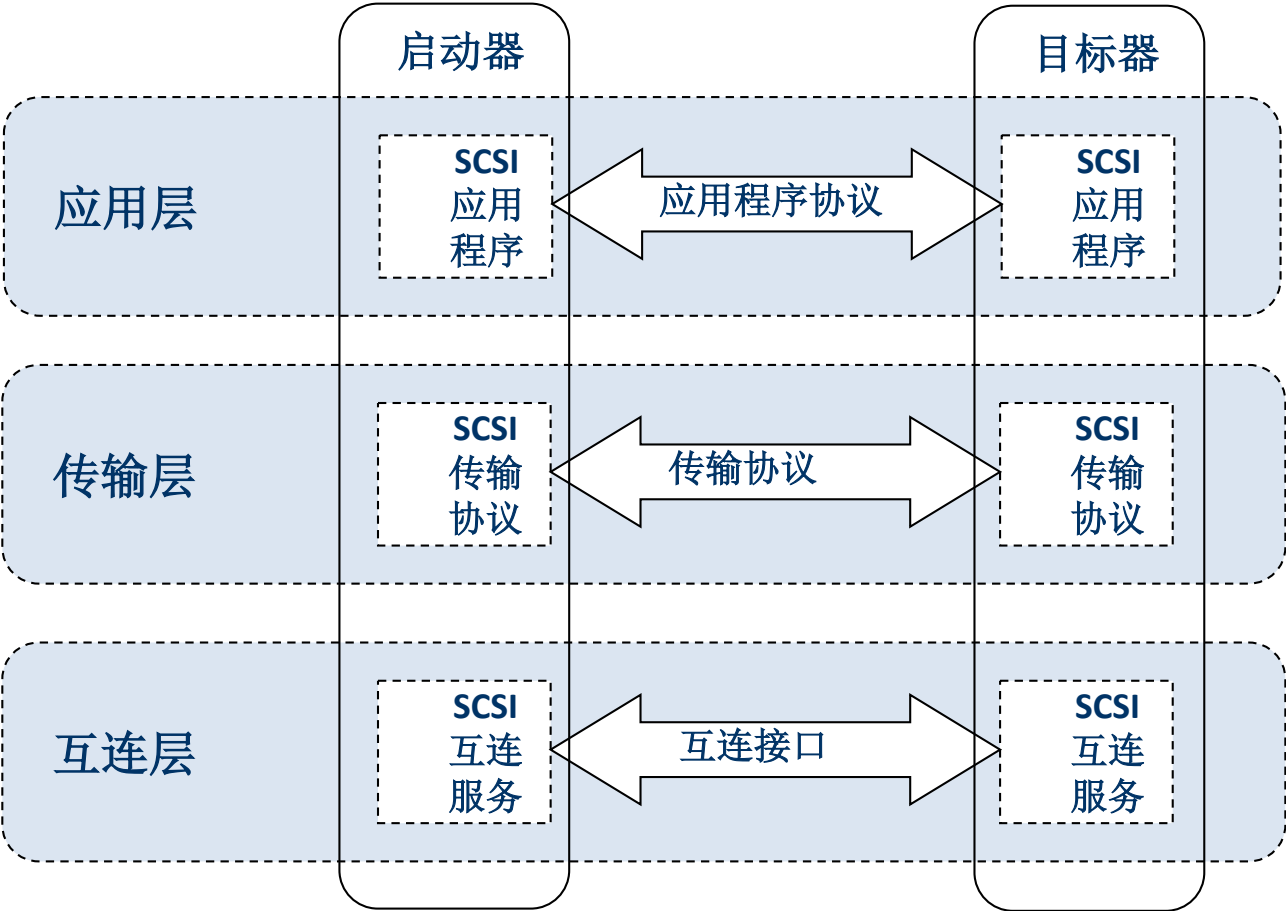


SCSI通信服务模型

- “客户-服务器”模型

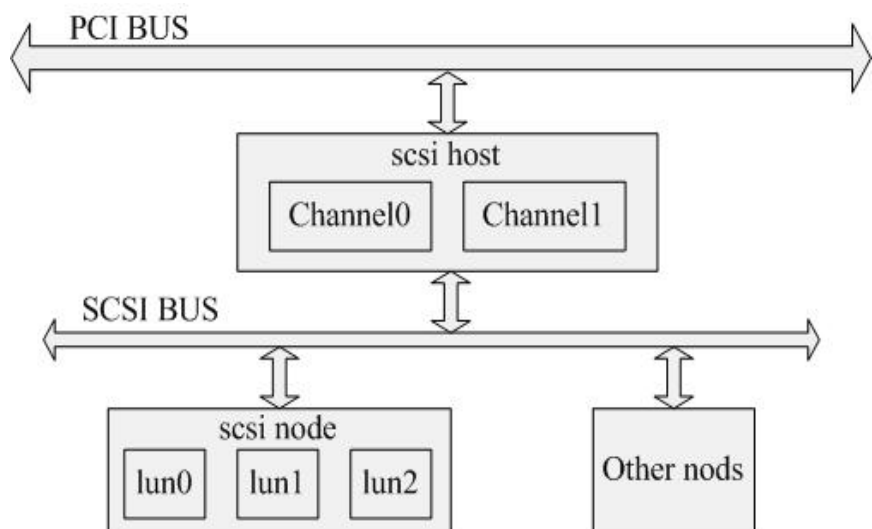


SCSI通信模型



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								
4	TRANSFER LENGTH (if required) PARAMETER LIST LENGTH (if required) ALLOCATION LENGTH (if required)							
5	CONTROL							

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								
6	(LSB)							
7	miscellaneous CDB information							
8	(MSB)							
9	TRANSFER LENGTH (if required)							
	PARAMETER LIST LENGTH (if required)							
	ALLOCATION LENGTH (if required)							
	(LSB)							
	CONTROL							

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								
6								
7	(LSB)							
8	(MSB)							
9	TRANSFER LENGTH (if required)							
10	PARAMETER LIST LENGTH (if required)							
11	ALLOCATION LENGTH (if required)							
12	(LSB)							
13	miscellaneous CDB information							
14	CONTROL							

- 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)		LOGICAL BLOCK ADDRESS						
3									
4									
5									
6									
7									
8									
9									(LSB)
10	(MSB)	TRANSFER LENGTH (if required) PARAMETER LIST LENGTH (if required) ALLOCATION LENGTH (if required)							
11									
12									
13								(LSB)	
14	miscellaneous CDB information								
15	CONTROL								

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

命令码

Bit	7	6	5	4	3	2	1	0
	GROUP CODE			COMMAND CODE				

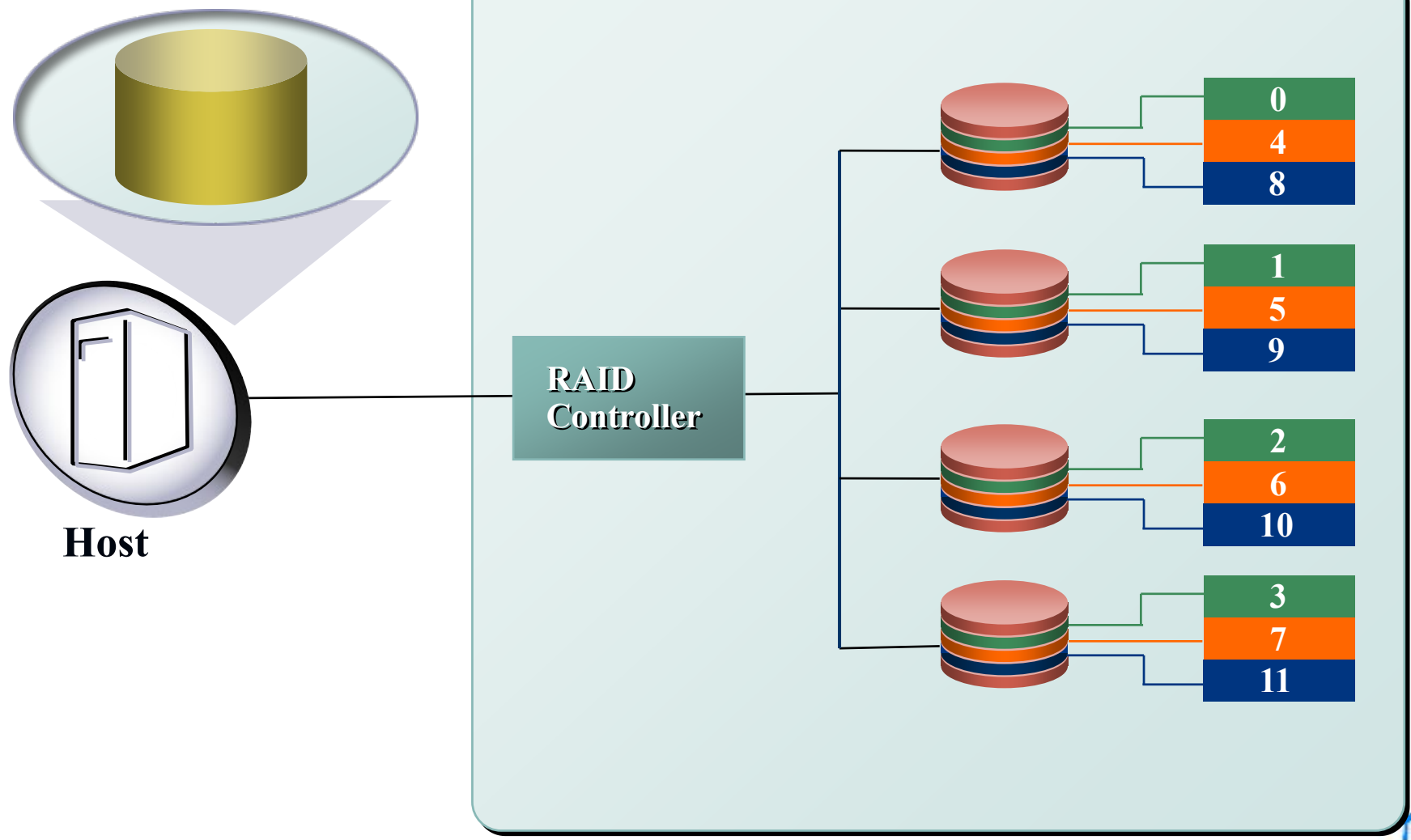
0x

A

B

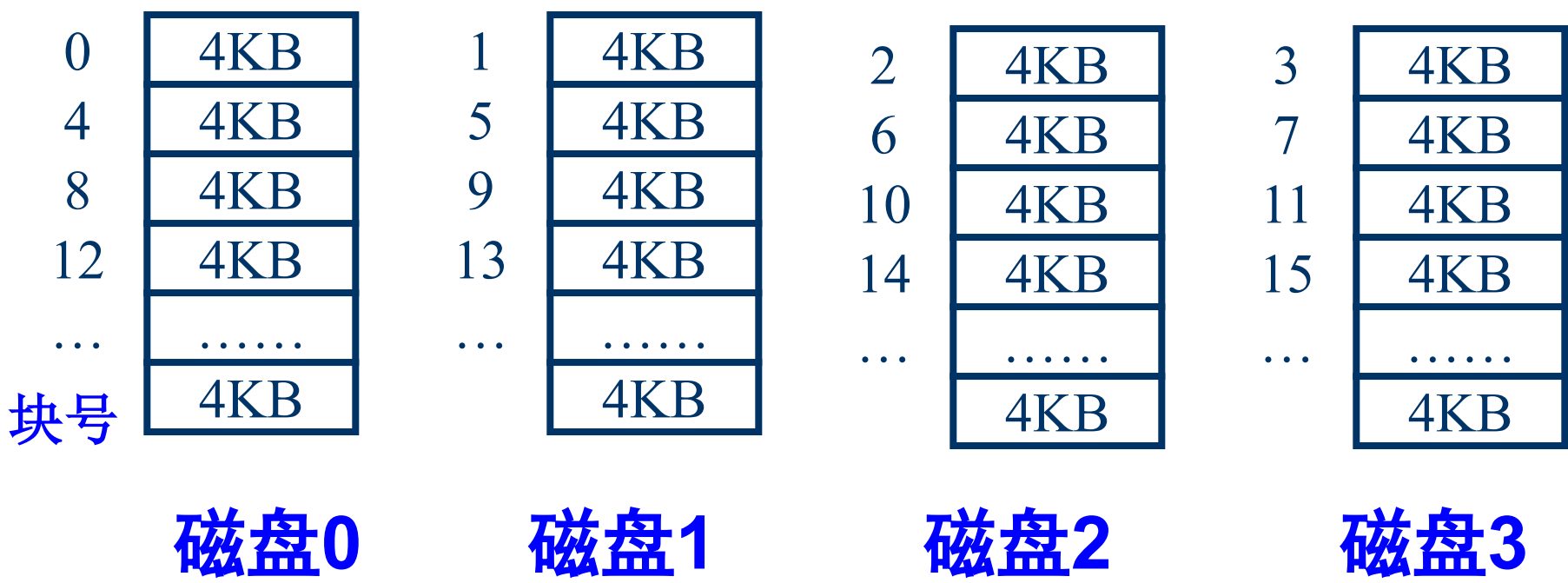
Group code	A	Meaning	example
000	0, 1	6字节命令	0x08, 0x12
001	2, 3	10字节命令	0x25, 0x2a,0x35
010	4, 5	10字节命令	0x55, 0x5a
011	6, 7	reserved	
100	8, 9	16字节命令	0x88, 0x8a,x09e
101	A, B	12字节命令	0xa8, 0xaa
110	C, D	Vendor specific	
111	E, F	Vendor specific	

命令分解举例 (RAID 0)

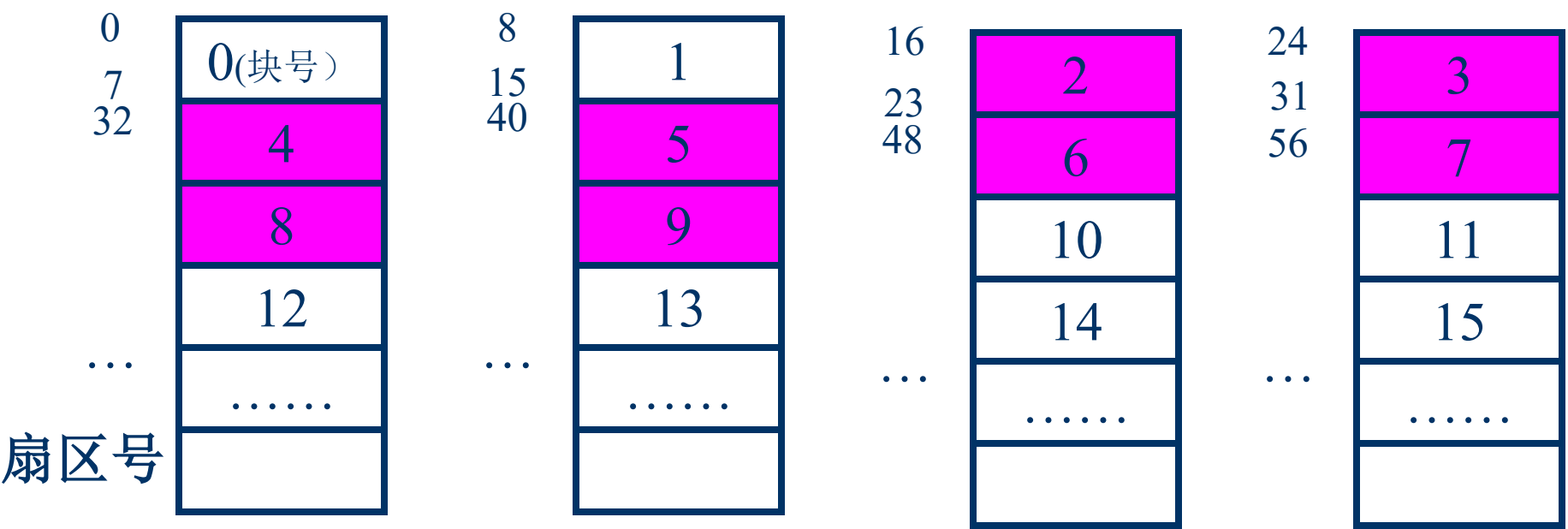


命令分解举例 (RAID 0)

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：



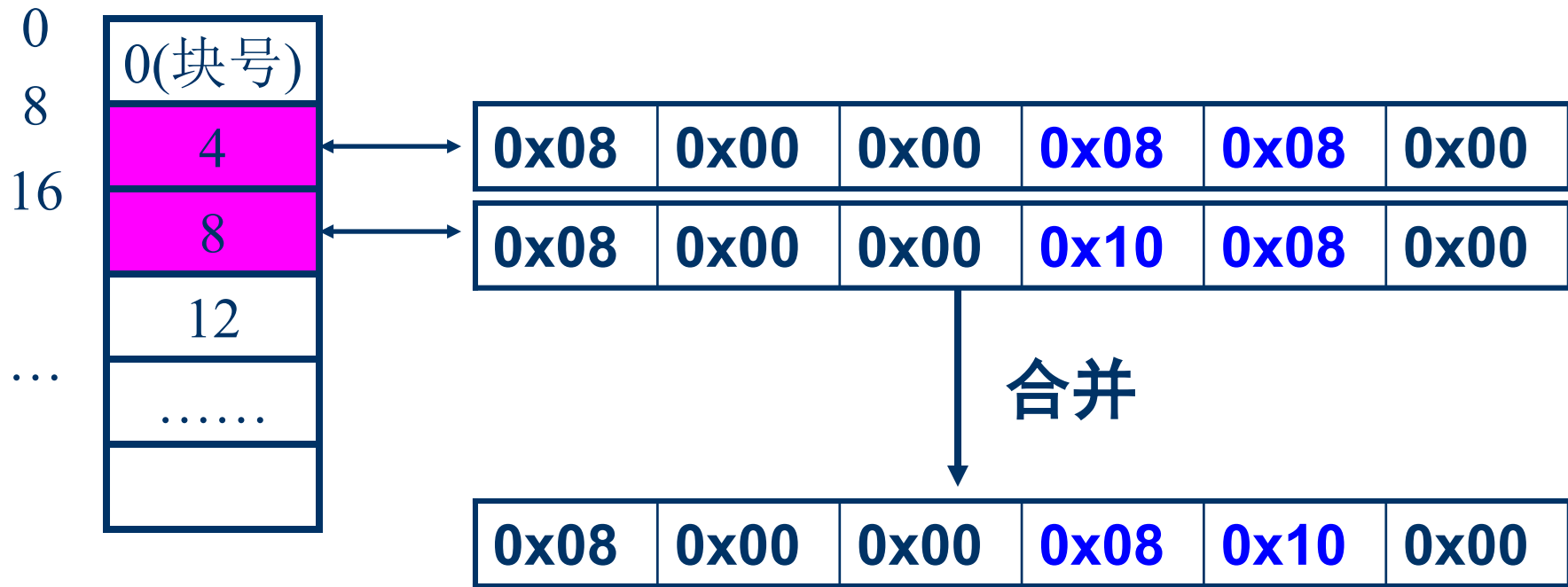
设主机请求读32KB的数据，起始地址为16（单位：扇区），SCSI命令为：(0x08, 0x00, 0x00, 0x10, 0x40, 0x00)



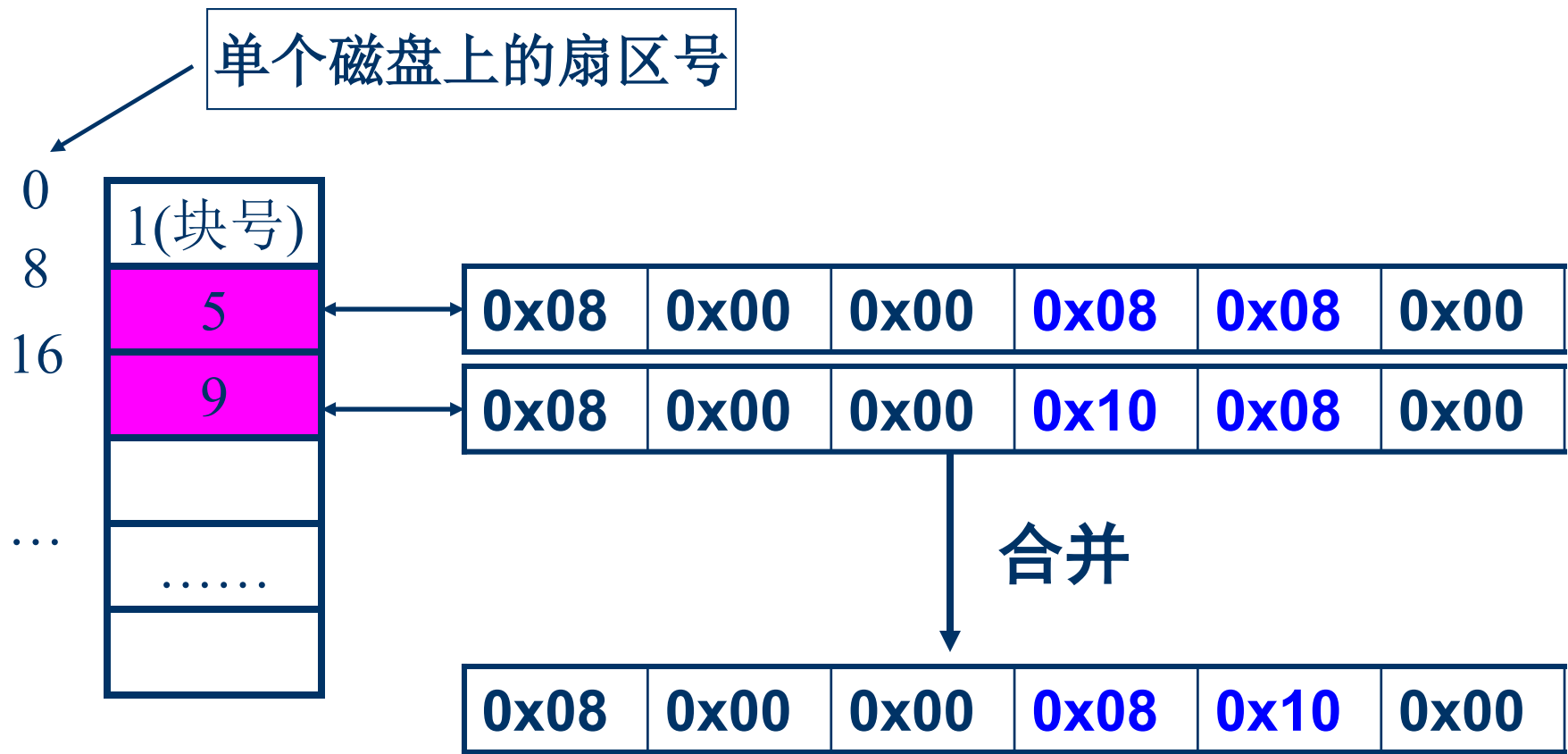
要读的数据在磁盘上的分布

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。

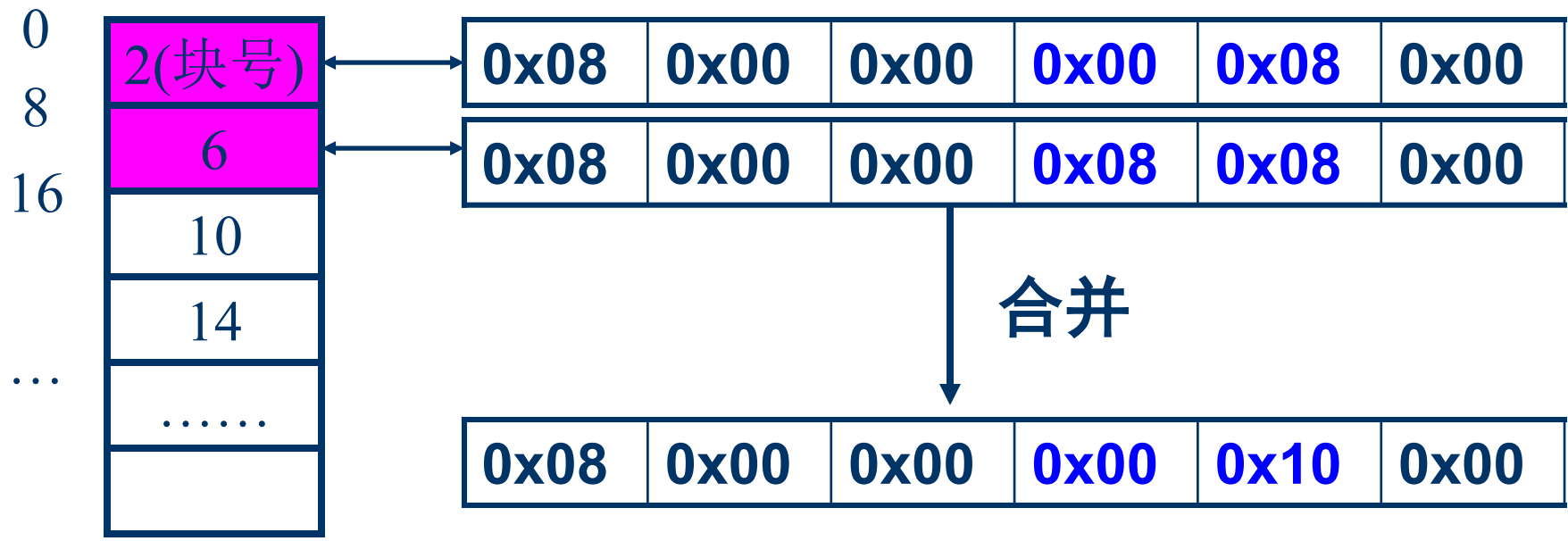
对应单个磁盘上的子命令：
磁盘1：



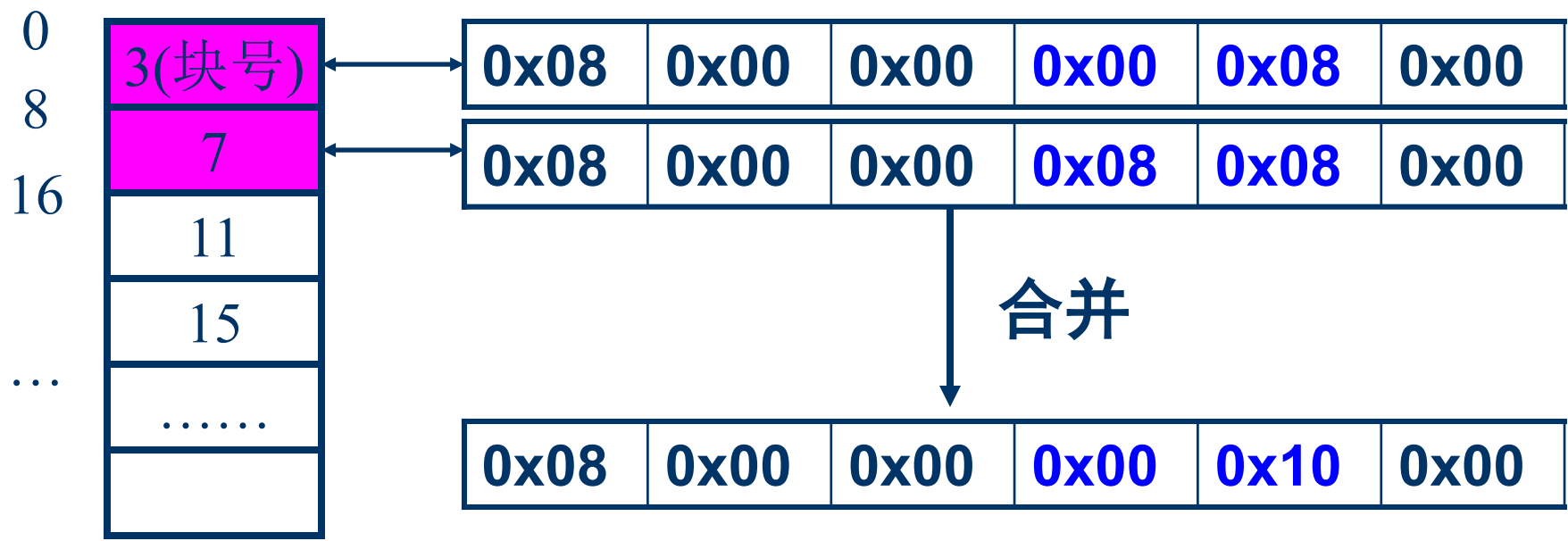
磁盘2



磁盘3



磁盘4



Next

时间开销问题

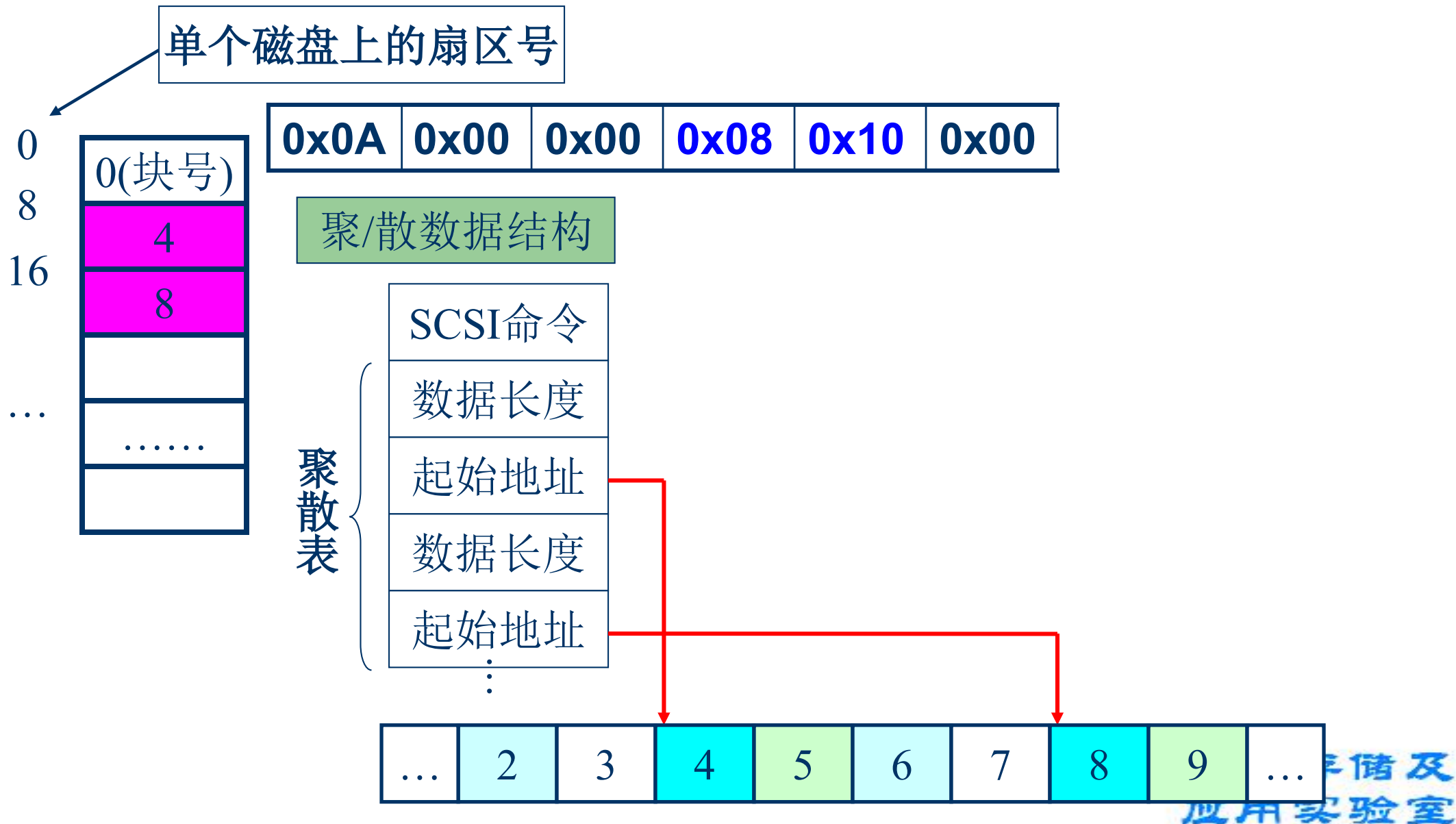
减少磁盘I/O操作的次数极大地提高阵列的性能，所以采用了I/O合并技术，I/O合并引起数据在内存中的移动，这种移动开销相对于磁盘I/O开销要小的多，但性能的影响还是相当大的（近25%）。如何消除这种不必要的开销？

聚/散技术

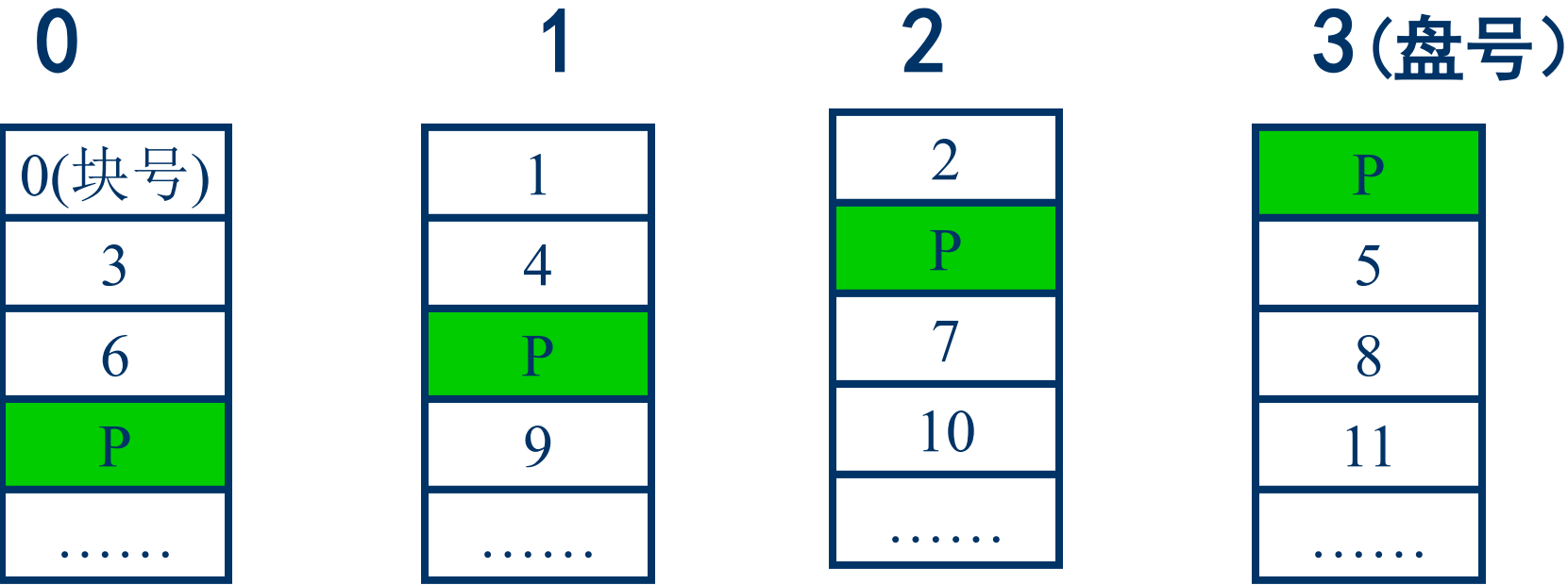
消除数据移动带来的额外开销，聚散技术的基本思想是将散落在内存的各块数据聚集起来传输。



对应磁盘1上的写命令聚散格式：

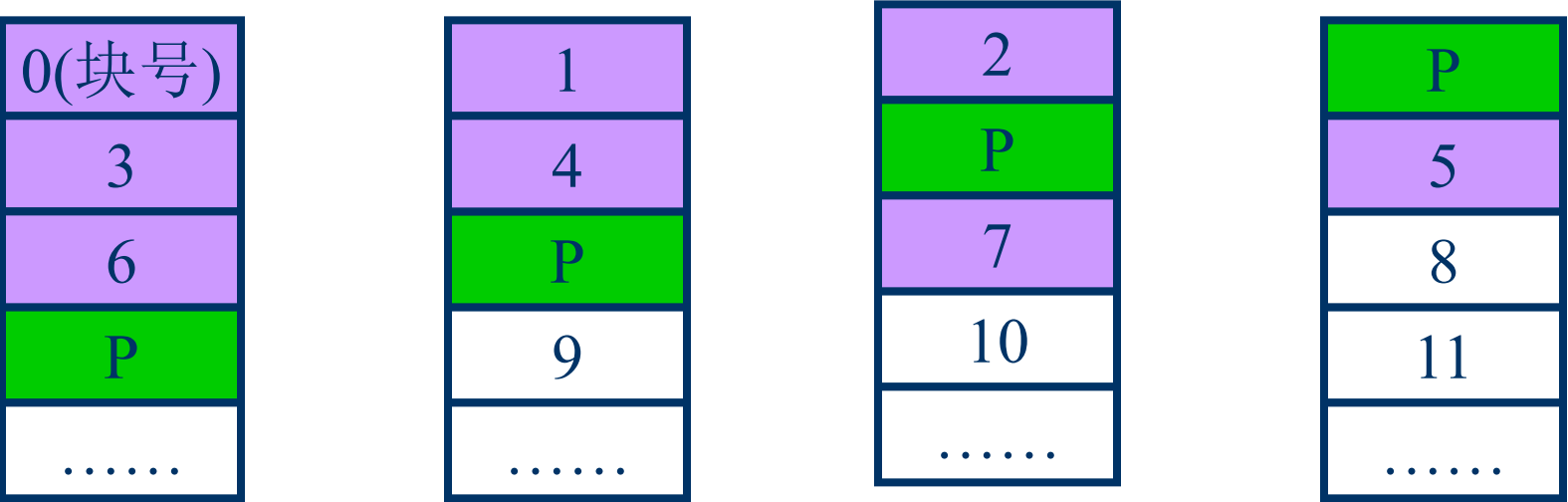


RAID5命令分解过程：



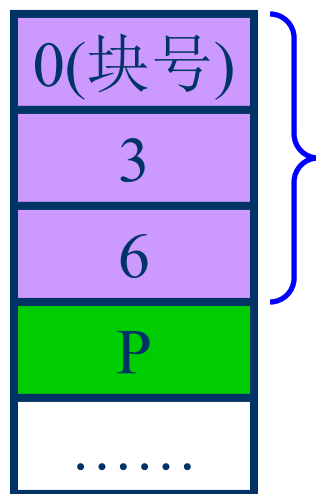
阵列参数：左不对称，4块盘，分块大小为2K

读命令： 0x08, 数据长度16k, 起始地址为0



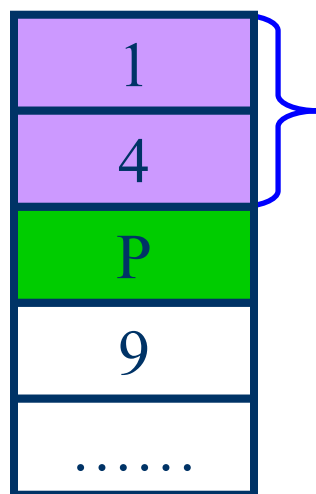
则要读的有效数据为图中的0~7块

0号盘上的子命令：



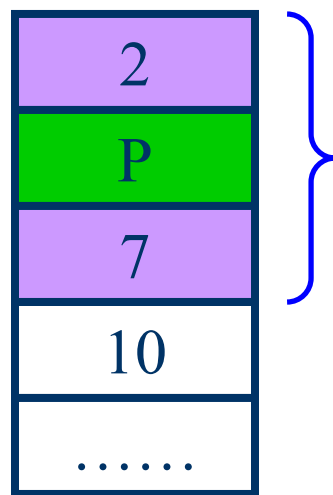
顺序读出0，3，6块，可以合并为一个读命令，数据长度为6k

1号盘上的子命令:



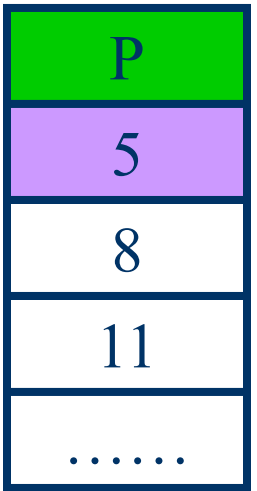
顺序读出1, 4块, 可以合并为一个读命令, 数据长度为4k

2号盘上的子命令：



读2号块和7号有效数据块，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

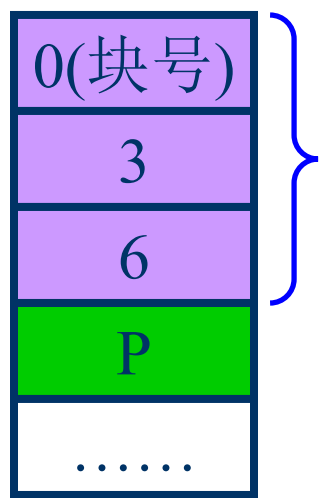
RAID5 块大小2KB

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	
	

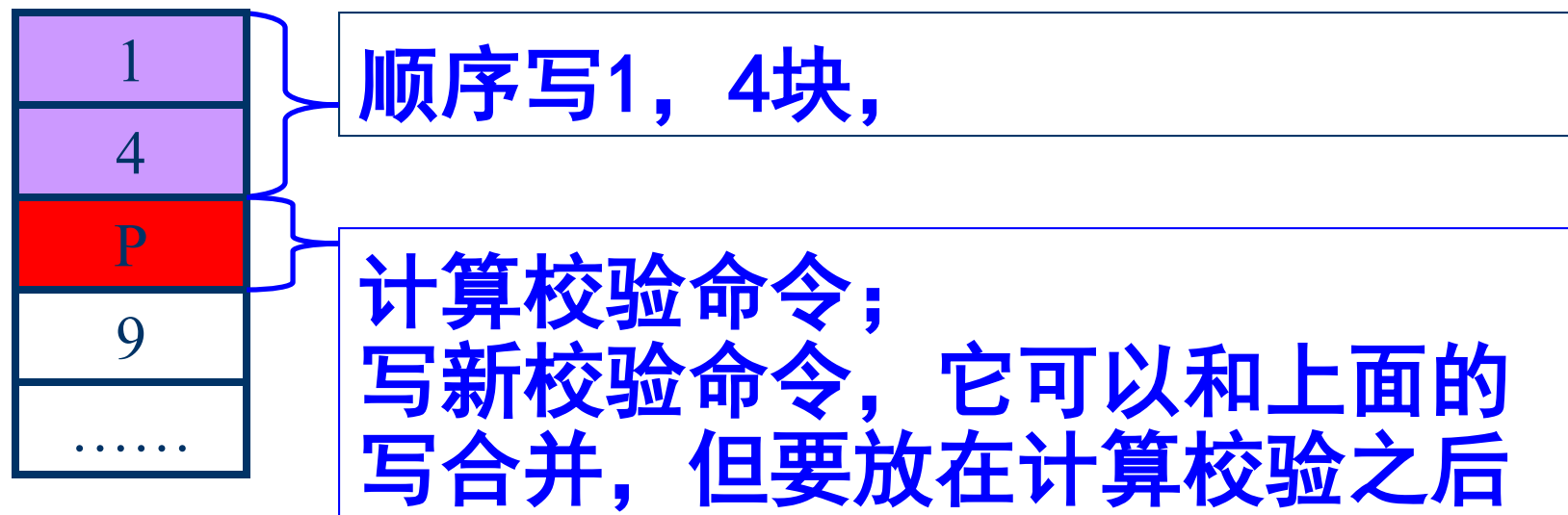
则要写的有效数据为图中的0~7块

0号盘上的子命令：



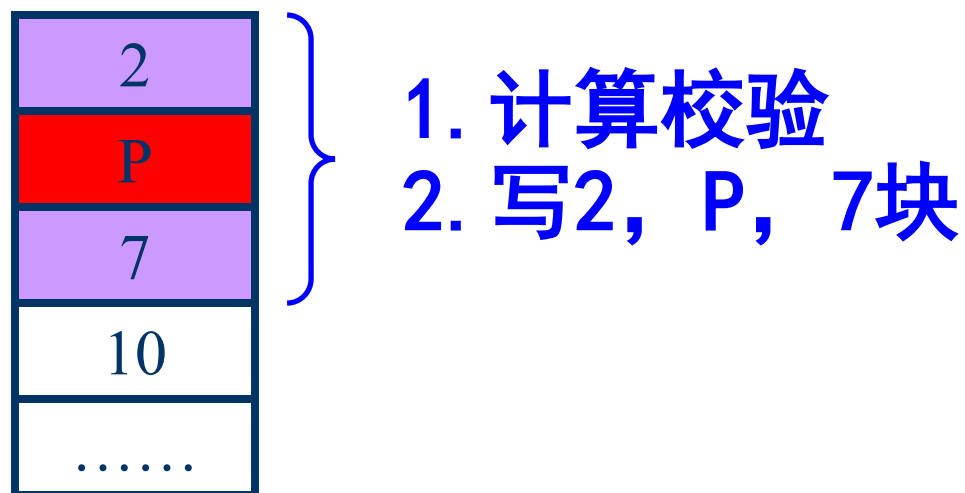
顺序写出0， 3， 6块， 可以合并为一个读命令， 数据长度为6k

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：

P
5
8
11
.....

- 2. 计算校验命令
- 3. 写数据和校验（合并为1命令）
- 1. 读旧的数据命令

写命令： 0x0a, 数据长度38k, 起始地址为0

	0		1		2		3		4		5		P	满条
	6		7		8		9		10		P		11	满条
	12		13		14		15		P		16		17	满条
	18		19		20		P		21		22		23	小写
	

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

- $P_{old} = D1 \oplus D2 \oplus D3 \oplus \dots Dn$

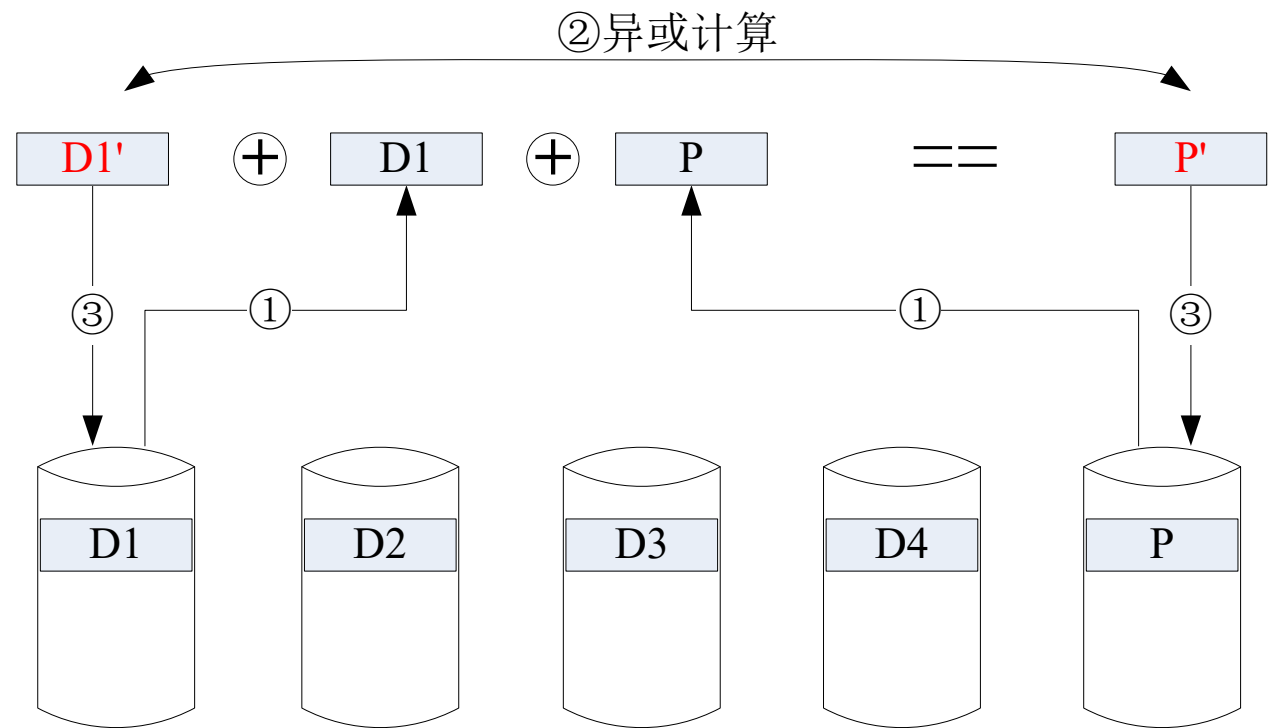
$$P = D1 \oplus D1' \oplus P_{old}$$

$$P = \text{D1} \oplus D1' \oplus \text{D1} \oplus D2 \oplus D3 \oplus \dots Dn$$

$$P = D1' \oplus D2 \oplus D3 \oplus \dots Dn = P_{new}$$

RAID5 Write Hole Problem

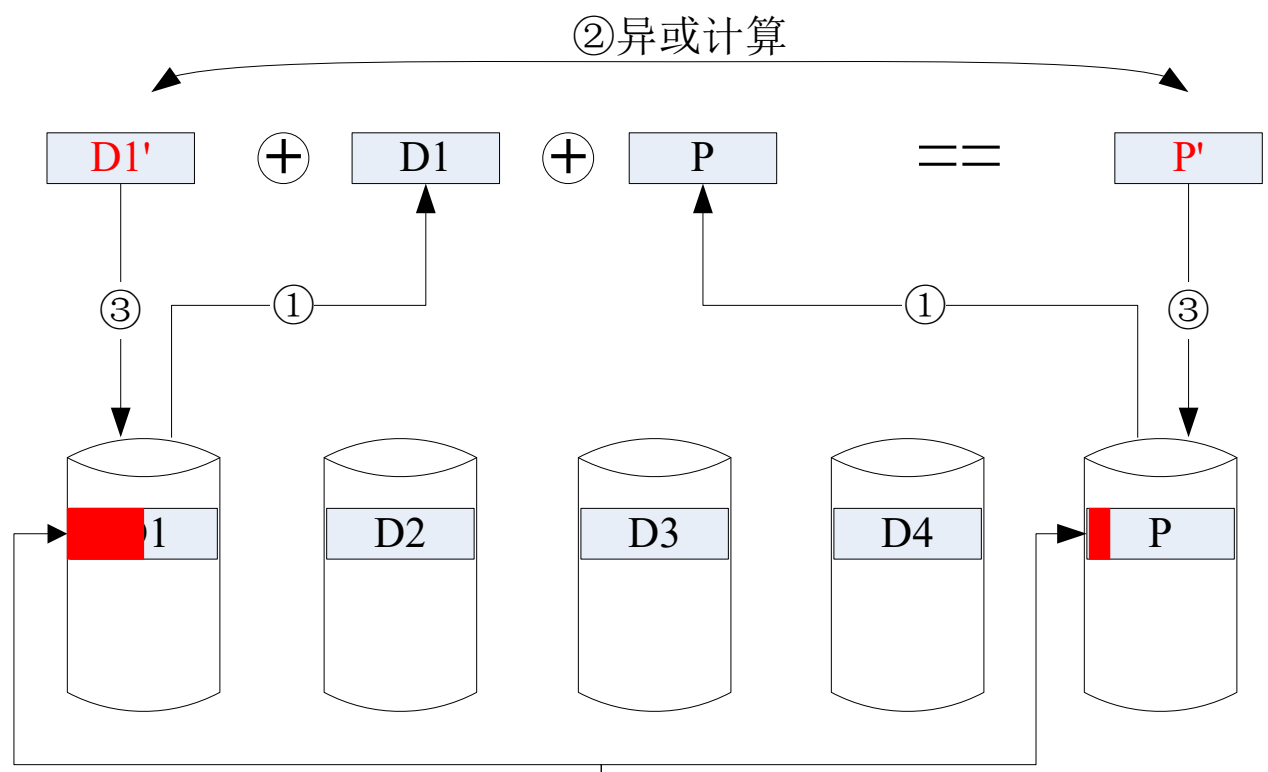
- 奇偶校验盘阵列（RAID5/6）中校验数据丢失（污染）问题（Parity Pollution）



- 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

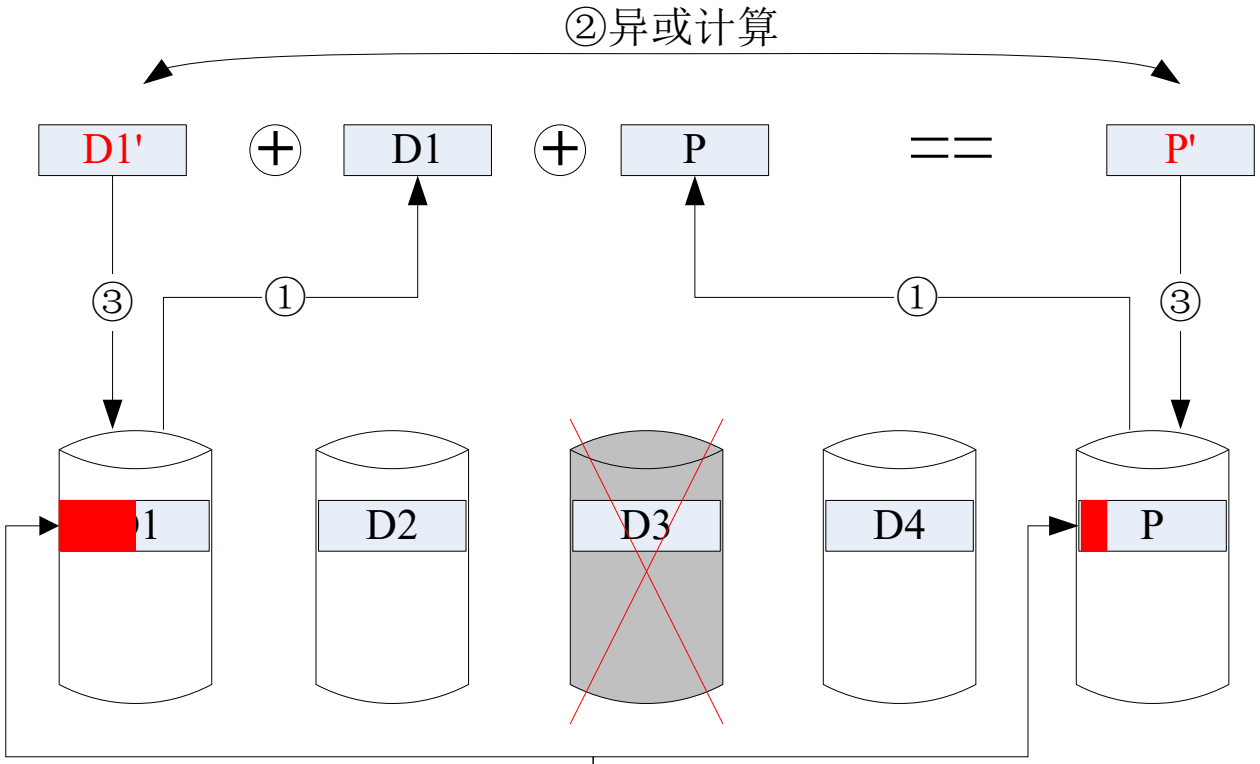


⑤：系统重启后，再次写入D1'，此时只有RCW (Read-Reconstruct-Write)机制才能恢复数据，否则会导致条带不一致现象，进而会丢失数据（再次发生磁盘故障）；

④系统断电，数据部分写入，且两个盘中写入的数据长度不一定相等（校验不一致）

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？？？



④系统断电，数据部分写入，且两个盘中写入的数据长度不一定相等（校验不一致）

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： $0 \oplus 1 \oplus 0 = 1$

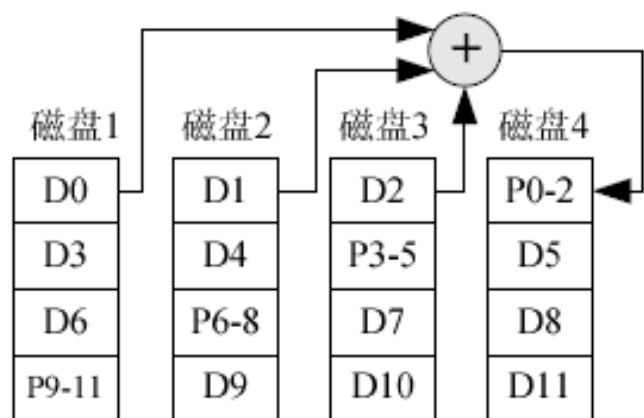
若刚写入的盘出现了故障，需要数据恢复：

恢复的数据 = $0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 1 = 1$

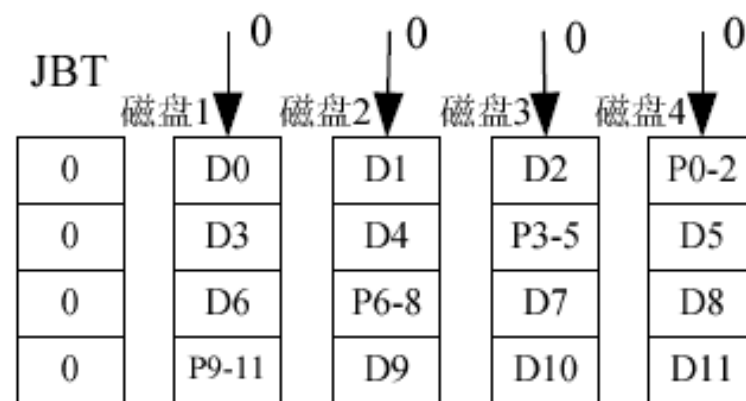
依据错误的校验信息进行重建得出的数据也是不正确的，
所以需要**初始化**过程

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

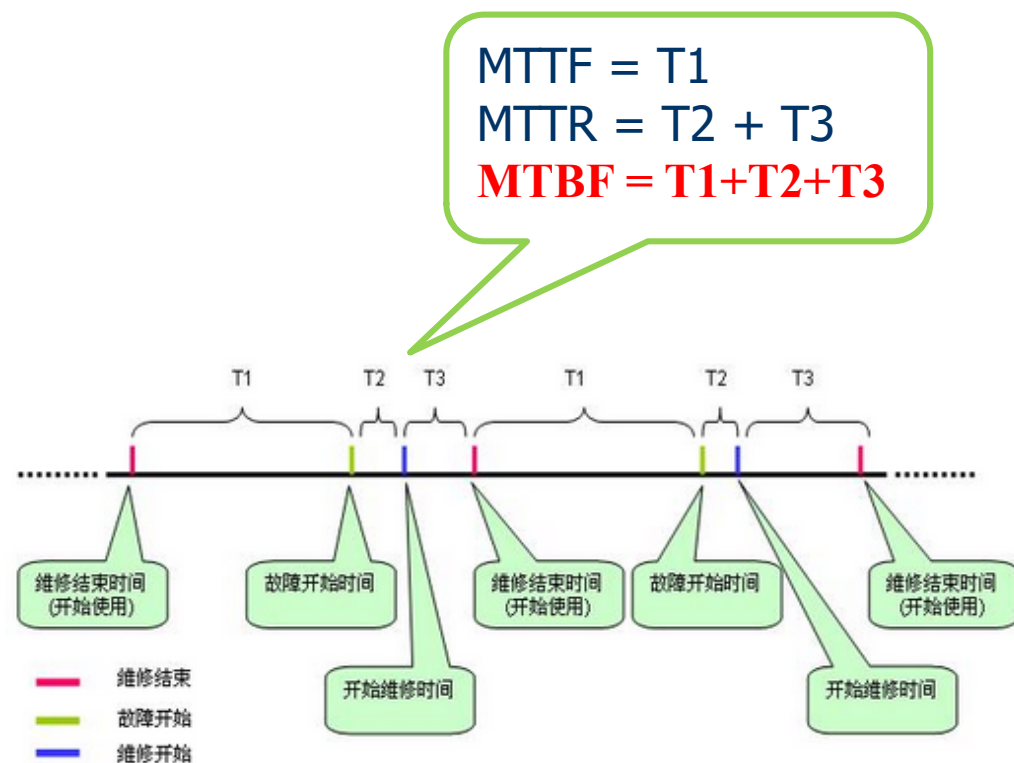
RAID高可靠、高可用技术

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：

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

N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

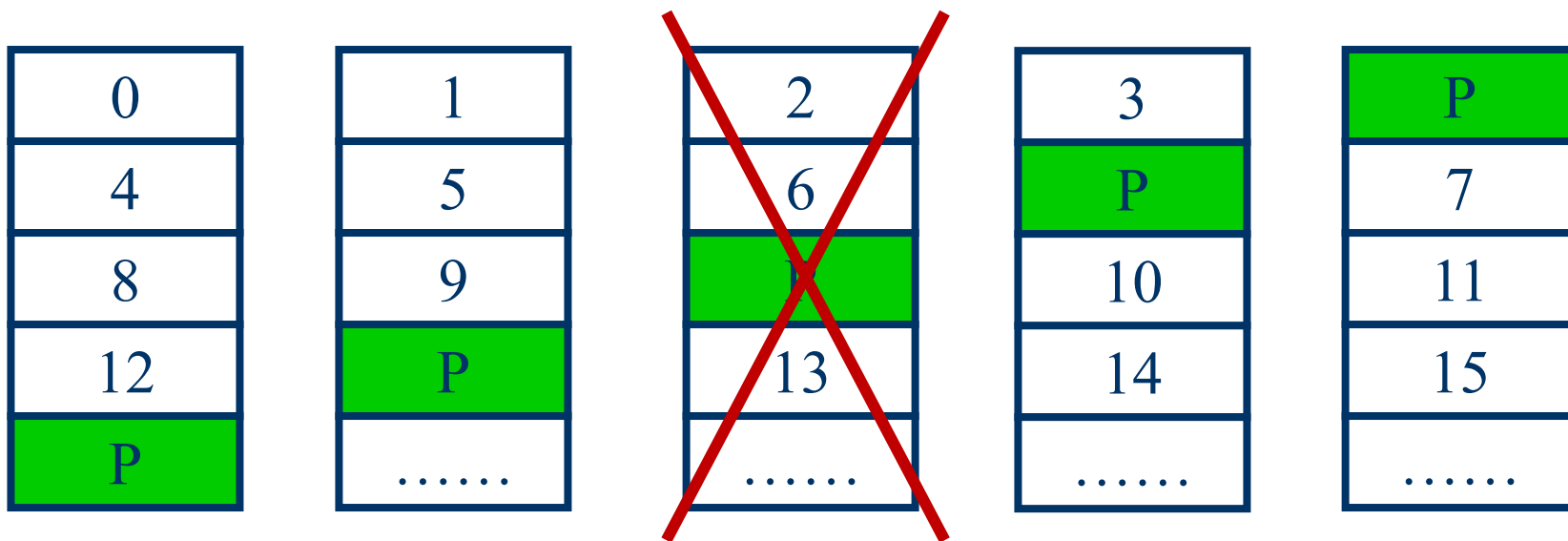
基于热度的多线程重建调度优化算法PRO

——FAST' 07

基于I/O负载重定向的重建优化算法WorkOut

——FAST' 09

RAID5



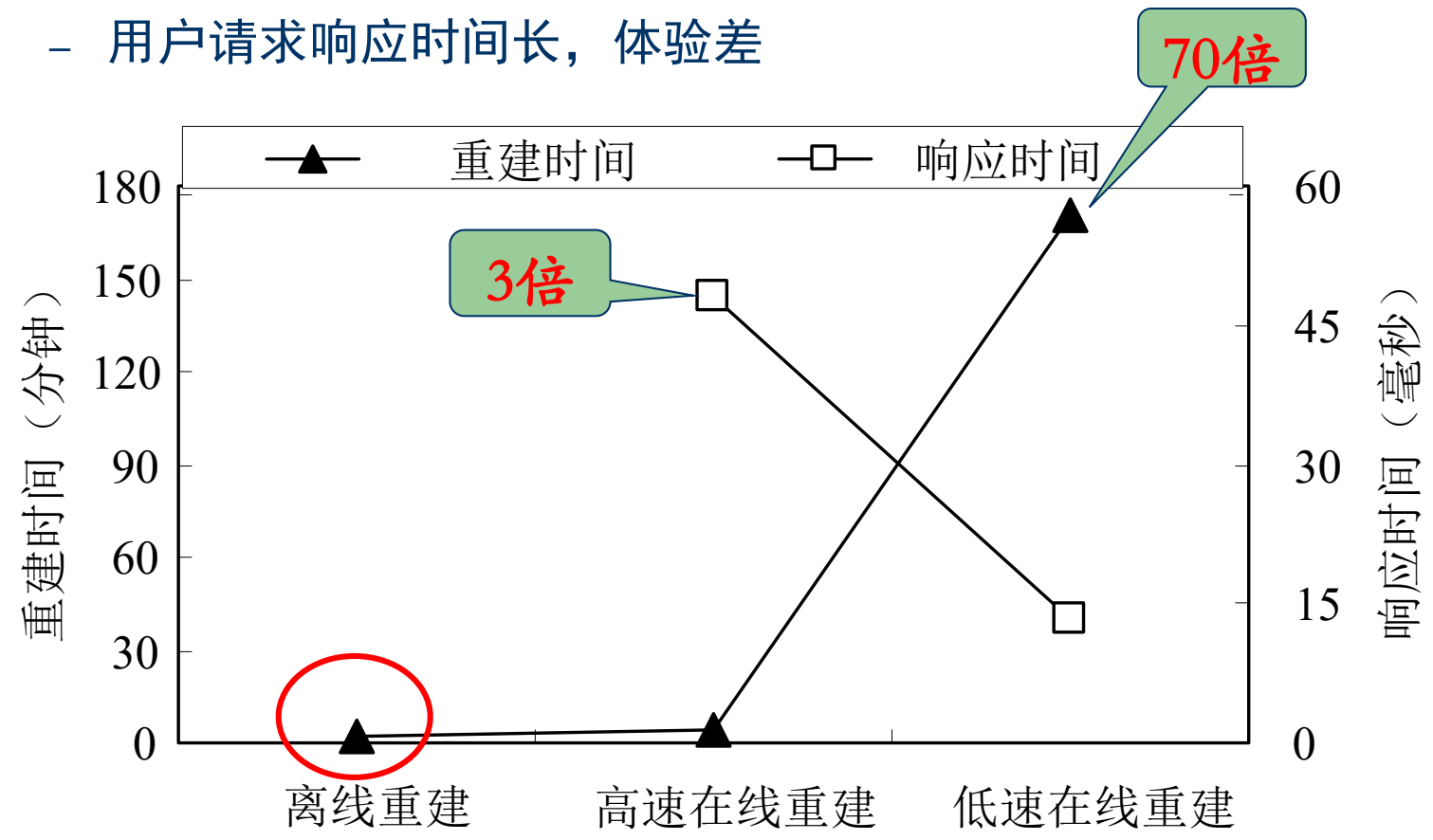
➤ 重建IO

➤ 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）

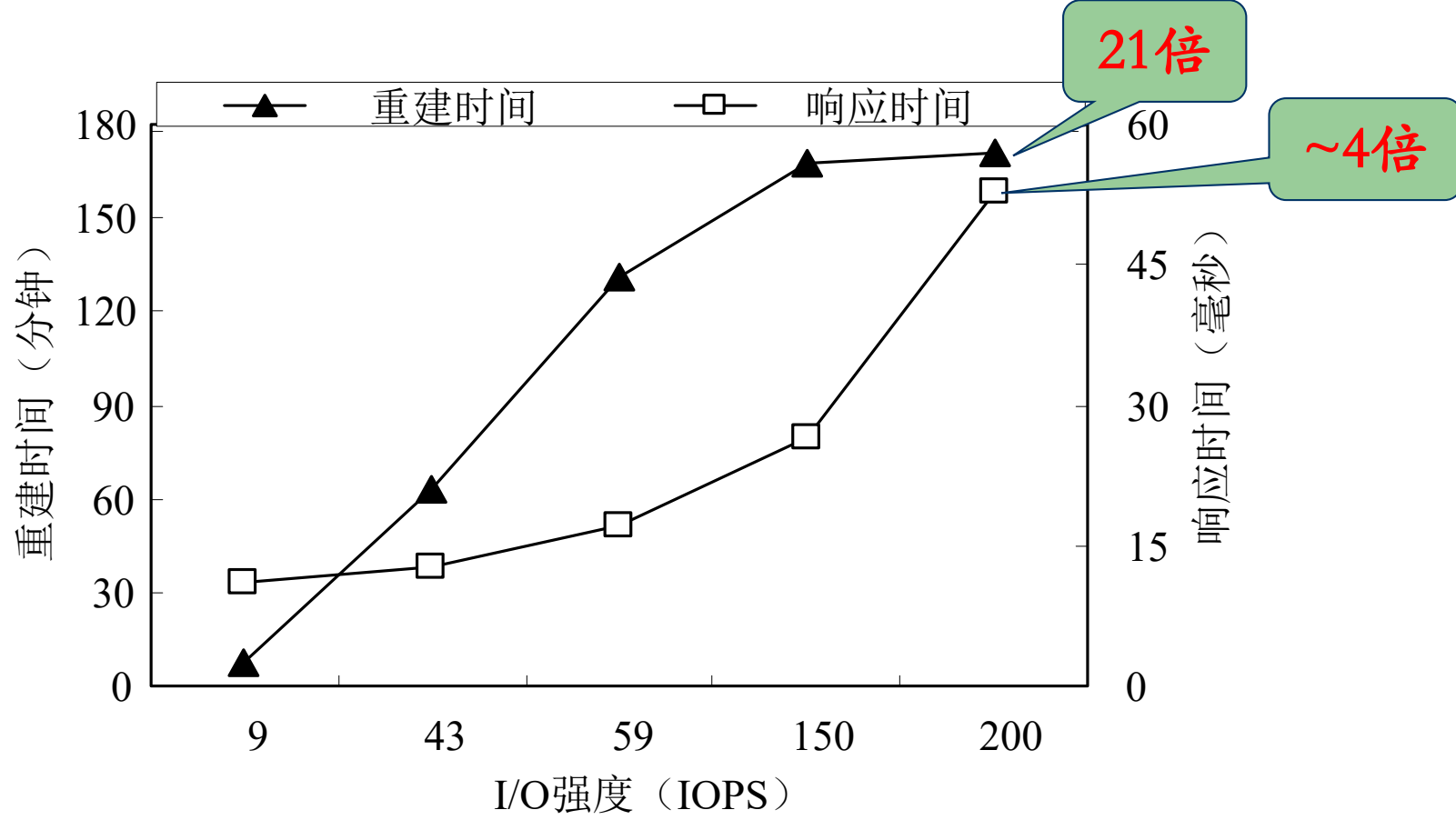
- 重建面临的挑战：重建I/O与用户I/O争用资源
 - 时间长，速度慢
 - 用户请求响应时间长，体验差



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

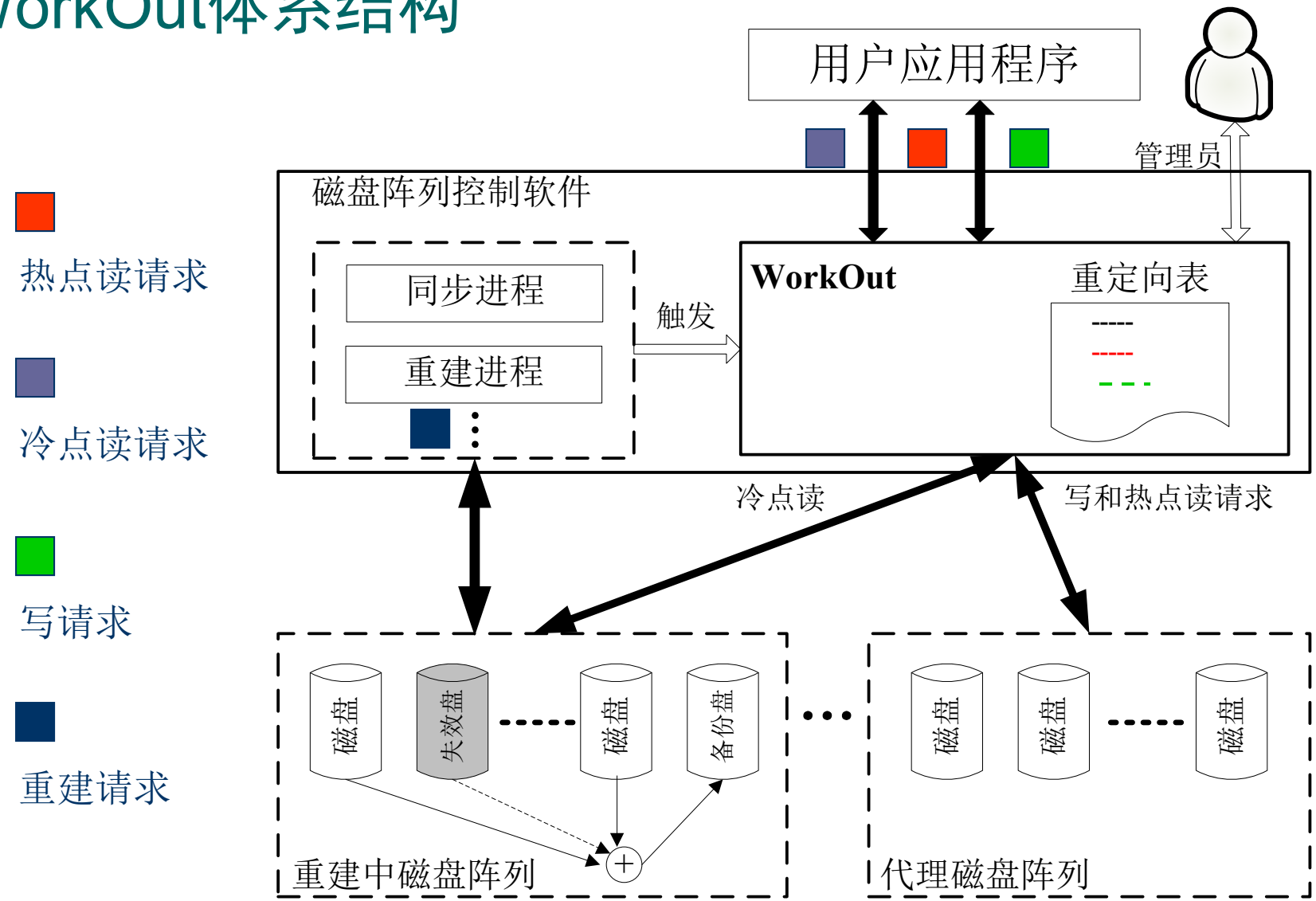
- 通过将重建中磁盘阵列收到的所有写请求和热的读请求重定向到代理磁盘阵列中，从而减少重建中磁盘阵列的用户I/O负载强度，使重建进程得到更多的磁盘资源，进而提高了重建效率

- 目标

- 提高数据中心的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:

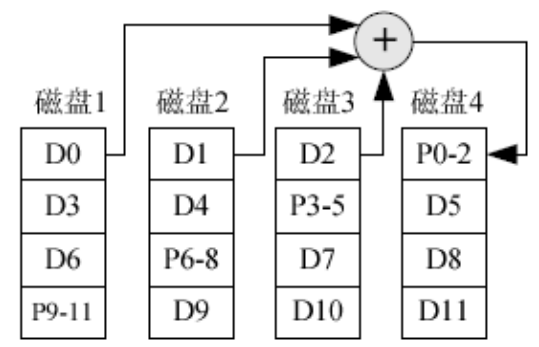
基于日志的重建

- 原理:

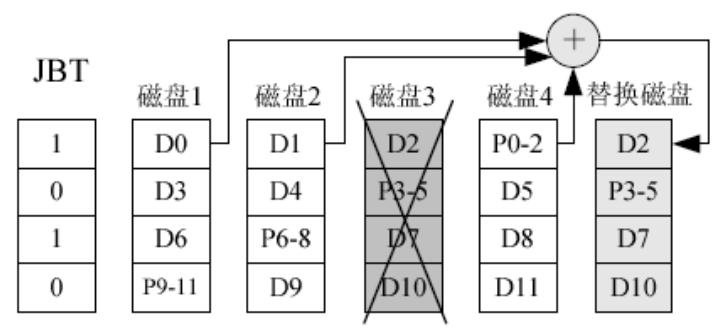
- 初始化全部写0
- 存储空间的空闲达50%，相当一部分空间一次都没修改过
- 记录修改过的条块，重建的时候，对修改的条块重新计算，未修改的部分直接写0

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)

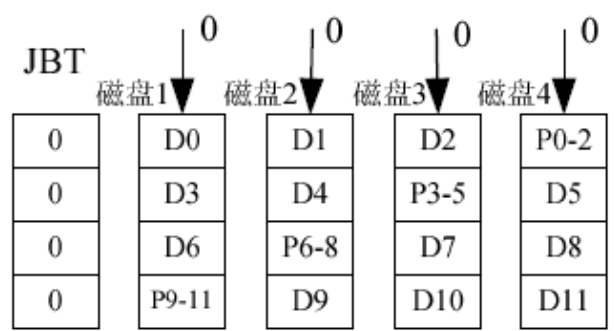
解决办法3：
基于日志的重建



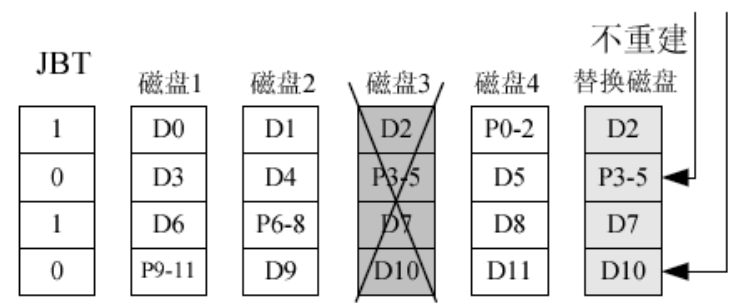
(a) 传统的数据同步过程



(a) 当bitmap=1时的重建过程



(b) 改进的数据同步过程



(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：

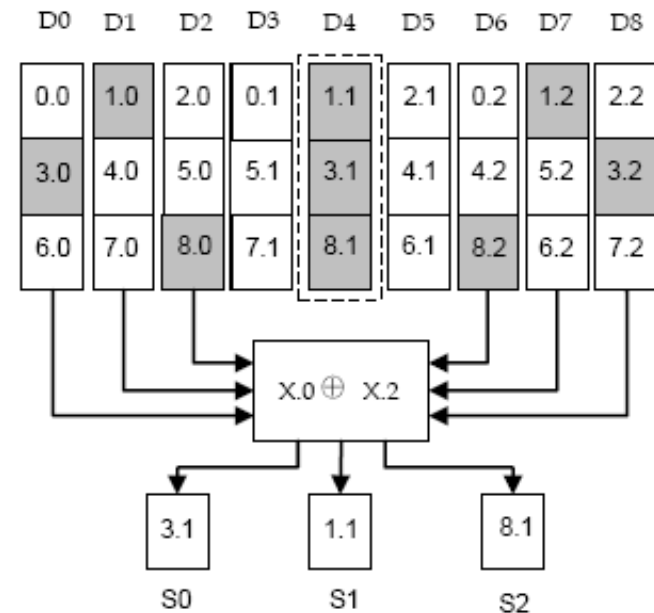
$$1.2\text{TB}/30*60\text{s} = 667\text{MB/s}$$

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.