中山大學 软件工程学院

SUN YAT-SEN UNIVERSITY SCHOOL OF SOFTWARE ENGINEERING

1924-2024 中山大學 世纪华诞 100th ANNIVERSARY SUN YAT-SEN UNIVERSITY

物理内存管理I

SSE202/204: 操作系统原理

苏玉鑫

suyx35@mail.sysu.edu.cn

助教：龙玉丹 单诗雯 毛晨希 沈志轩 郑灿峰 胡伟峰

版权信息

- 部分内容来自：上海交通大学并行与分布式系统研究所操作系统课件
  - https://ipads.se.sjtu.edu.cn/courses/os/
- 其它参考资料：
  - 清华大学操作系统公开课
    - https://open.163.com/newview/movie/courseintro?newurl=ME1NSA351
    - 介绍标准内容，适合考研
  - 南京大学计算机软件研究所
    - http://jyywiki.cn/OS/2025/
    - https://space.bilibili.com/202224425/channel/collectiondetail?sid=192498
    - 比较有趣

管理物理内存资源

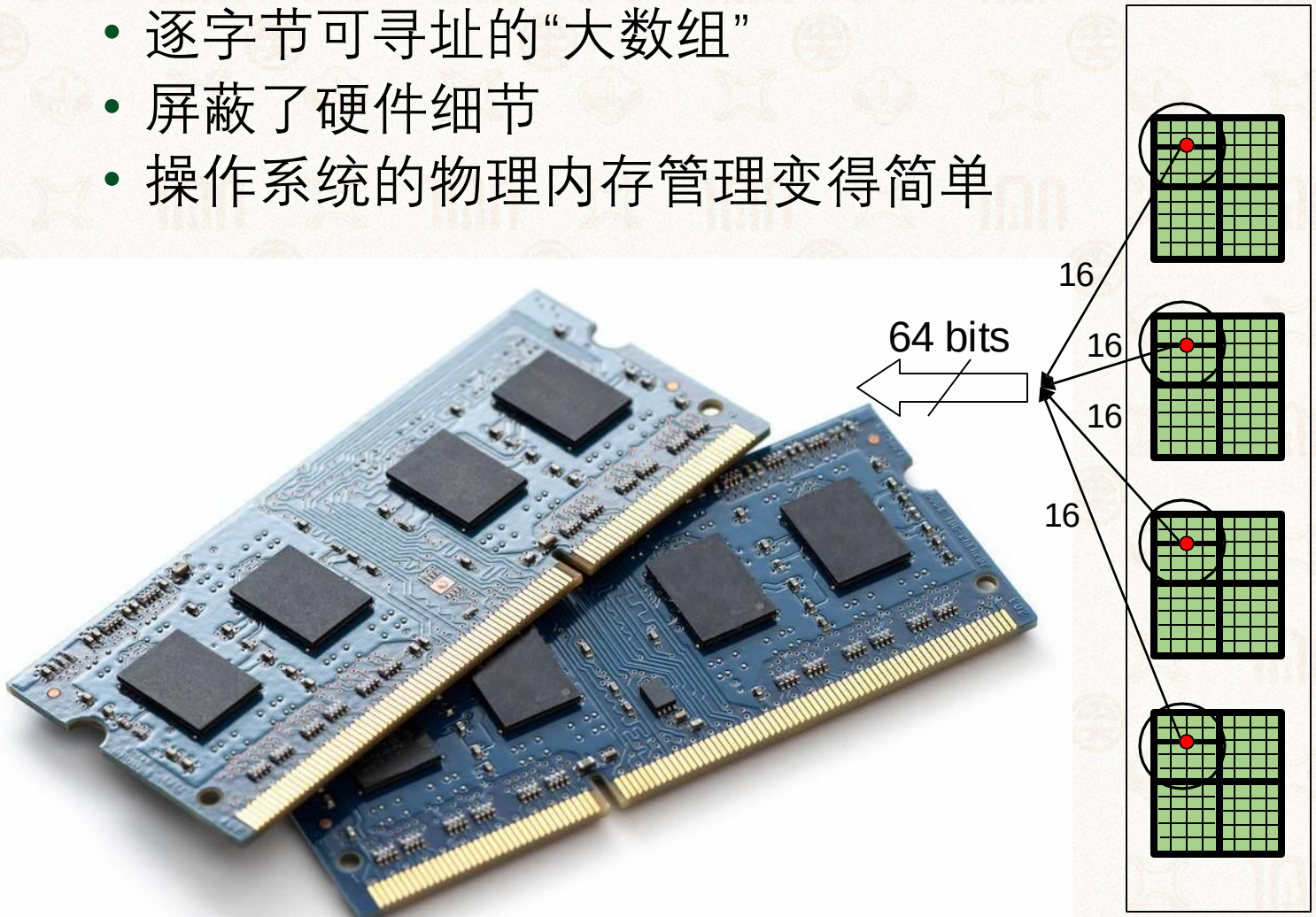
- 目标与评价维度
- 基于位图的连续物理页分配
- 伙伴系统
- SLAB分配器
- Linux内核内存管理架构

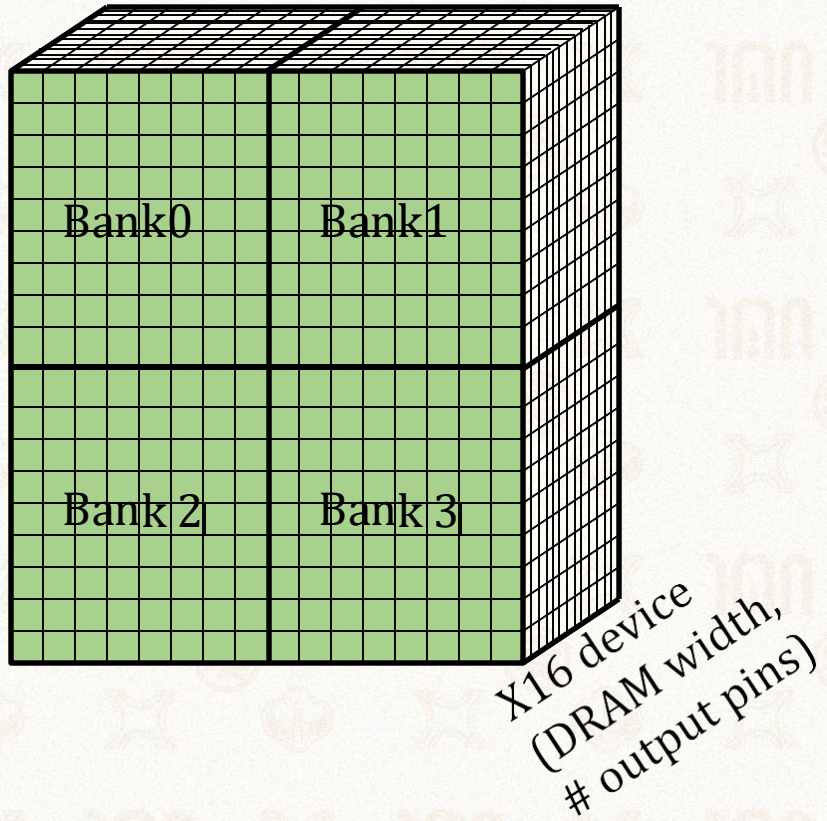
管理物理内存资源

- 目标与评价维度
- 基于位图的连续物理页分配
- 伙伴系统
- SLAB分配器
- Linux内核内存管理架构

内存控制器(Memory Controller)

- 为操作系统提供了易用的物理内存抽象
  - 逐字节可寻址的“大数组”
  - 屏蔽了硬件细节
  - 操作系统的物理内存管理变得简单

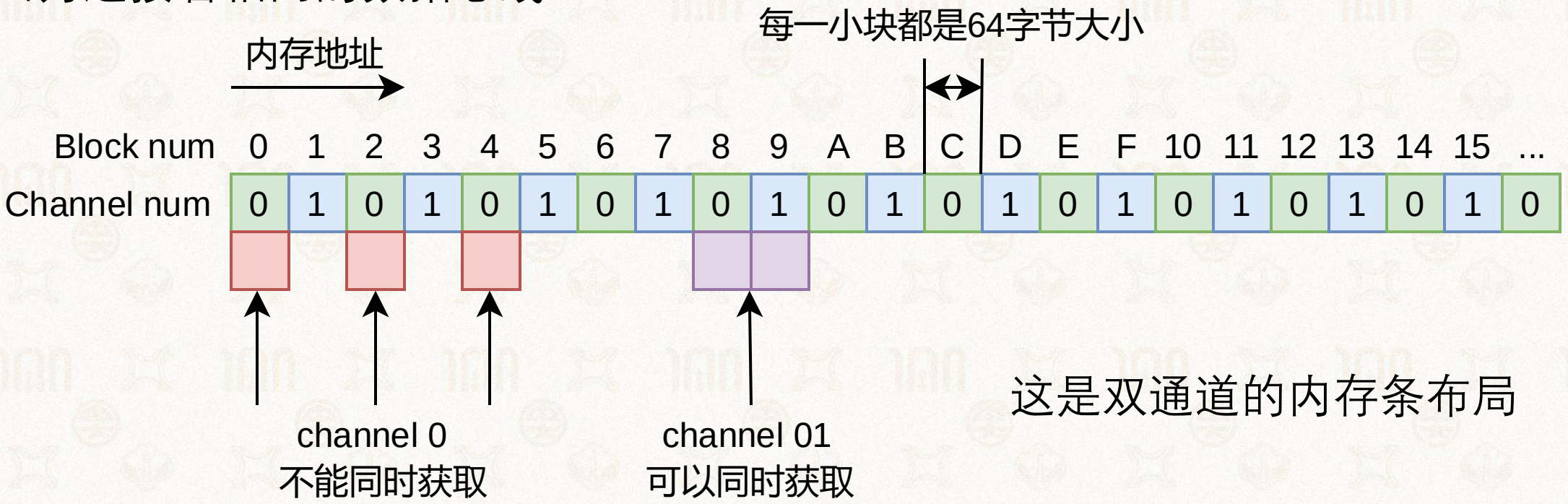




举例：内存多通道

- 细节不能忽视：内存条是分多通道的
  - 不同通道的空间可同时访问
  - 同一个通道(Channel)内的数据不能被同时获取
  - 因为连接着相同的数据总线

使用数组时，单一元素最好不要超过128字节， 为什么?

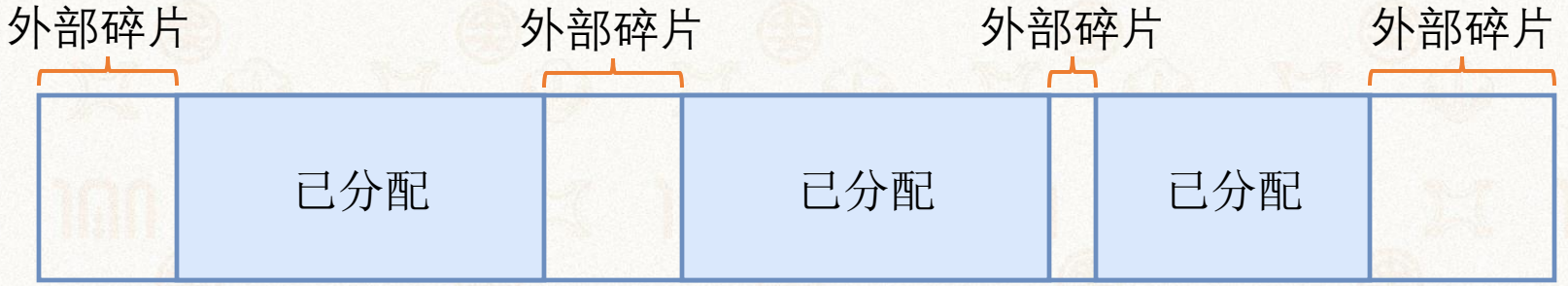


这是双通道的内存条布局

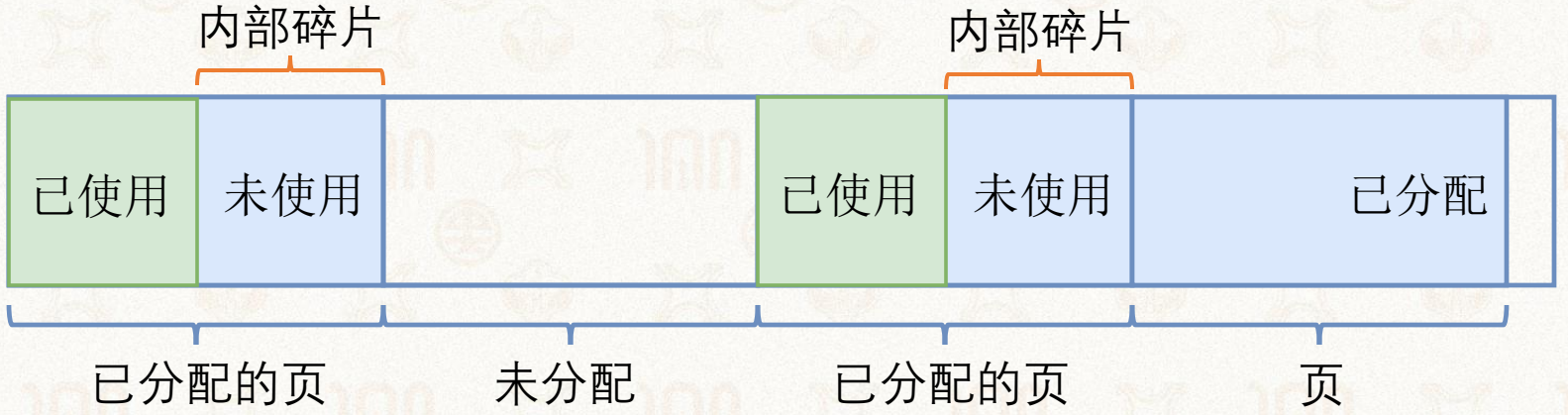
p.s. 家用电脑的内存条一般都是双通道

物理内存管理中的碎片问题

- 外部碎片（空闲的但不连续，无法被使用）

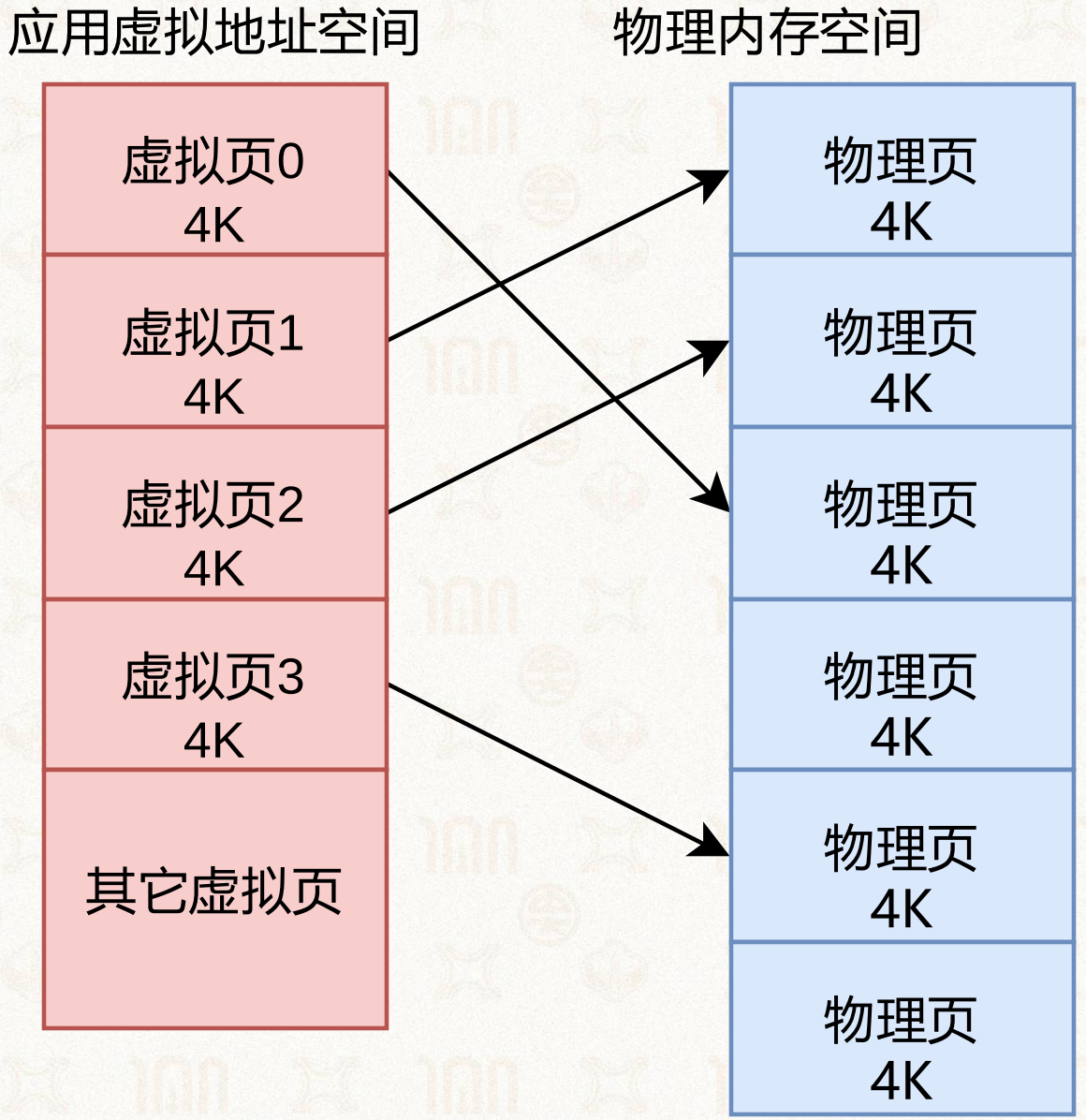


- 内部碎片（分配大小大于实际需要）



物理内存管理的评价指标

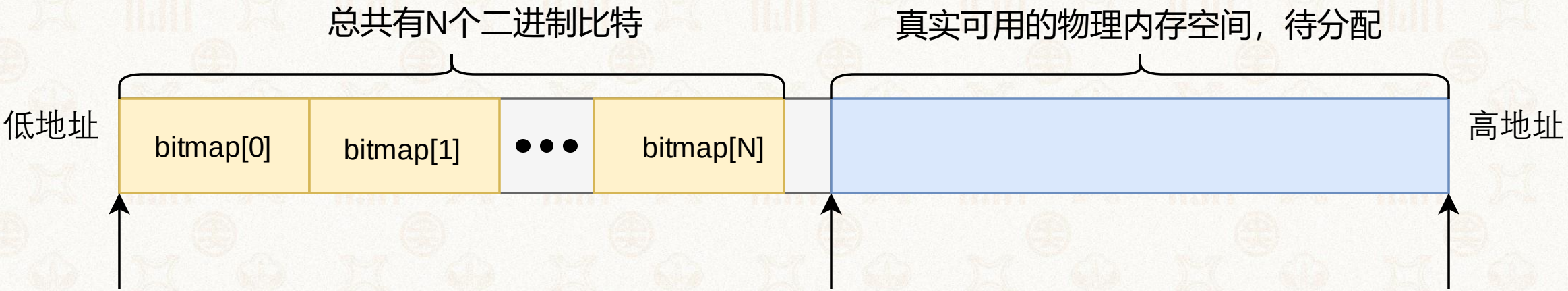
- 内存资源利用率
  - 外部碎片和内部碎片
- 物理内存分配
  - 产生映射需求时，要找到合适的物理页
- 分配速度
  - 复杂的算法可以更好地解决碎片问题
  - 但是内存分配操作的性能同样重要
- Tradeoff?



管理物理内存资源

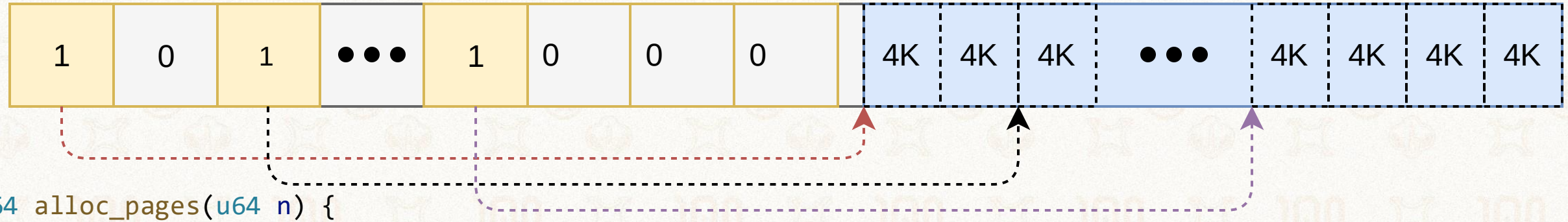
- 目标与评价维度
- 基于位图的连续物理页分配
- 伙伴系统
- SLAB分配器
- Linux内核内存管理架构

基于位图的物理内存池结构



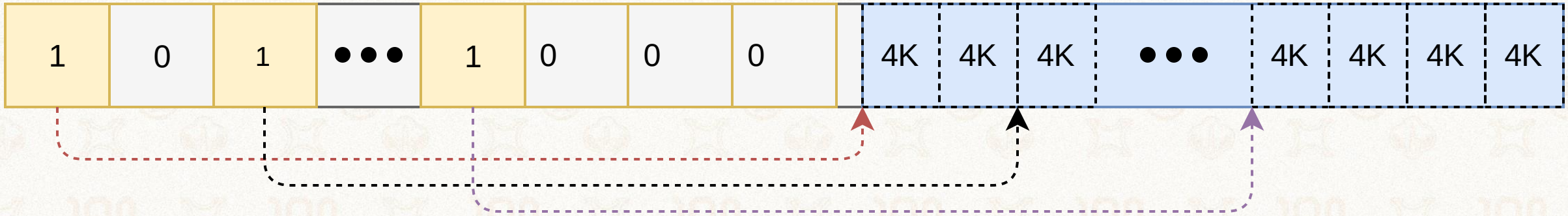

```
// 总共有 N 个 4K 物理页，位图中的每一个对应一个页
bit bitmap[N];

void init_allocator(void) {
    int i;
    for (i = 0; i < N; ++i)
        bitmap[i] = 0;
}
```

分配 n 个连续的物理页




```
u64 alloc_pages(u64 n) {
    int i, j, find;
    for (i = 0; i < N; ++i) {
        find = 1;
        for (j = 0; j < n; ++j) {// 从第 i 个物理页开始判断连续 n 个页是否空闲
            if (bitmap[i+j] != 0) {
                find = 0
                break;
            }
        }
        if (find) { // 将找到的连续 n 个物理页标记为已分配
            for (j = i; j < i+n; ++j)
                bitmap[j] = 1;
            return FREE_MEM_START + i * 4K; // 返回第 i 个物理页的起始地址
        }
    }
    return -1; // 分配失败
}
```

释放 n 个连续的物理页



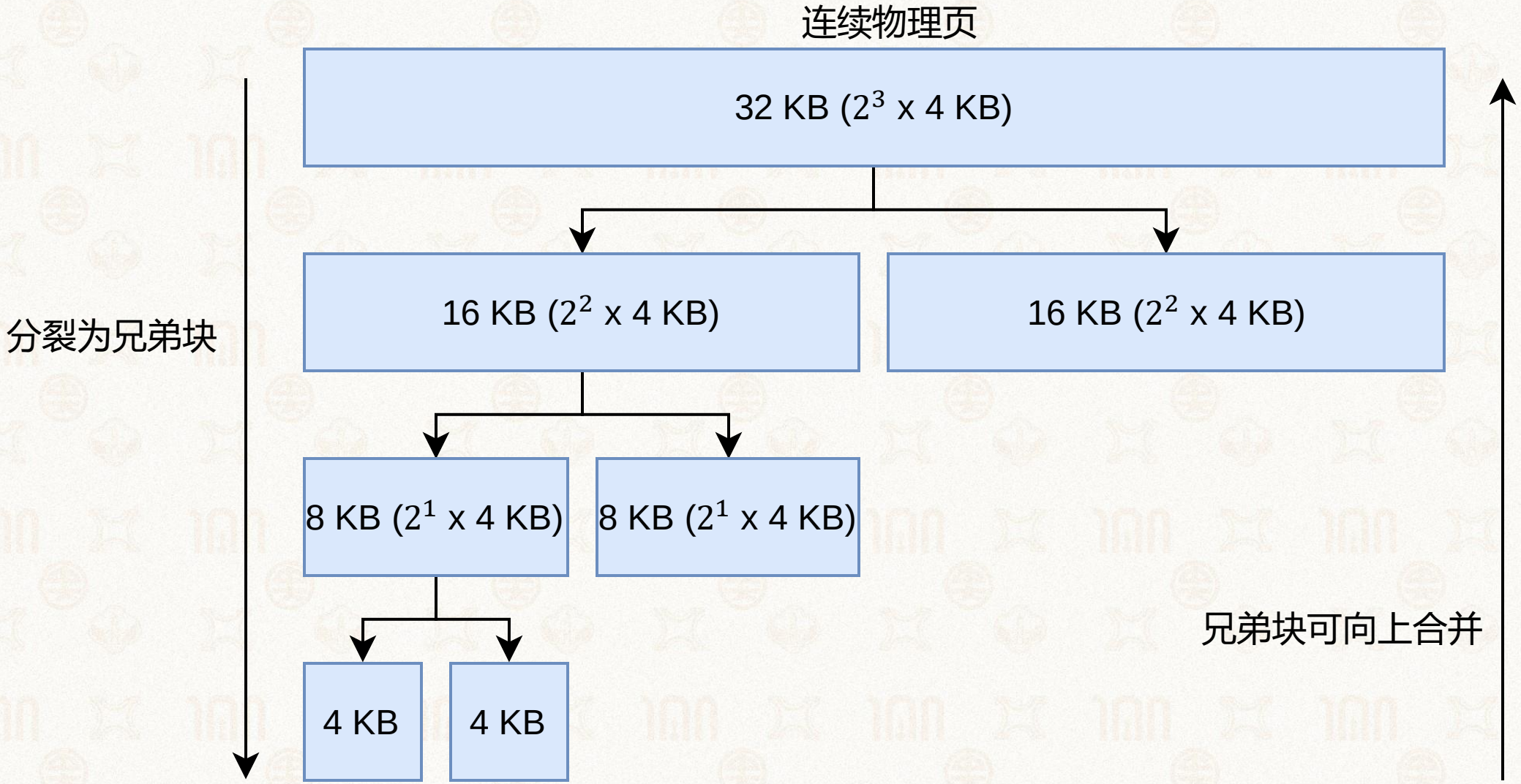

```
void free_pages(u64 addr, u64 n) {
    int page_idx;
    int i;

    // 计算待释放的起始页索引
    page_idx = (addr - FREE_MEM_START) / 4K;

    for (i = 0; i < n; ++i) {
        bitmap[page_idx+i] = 0;
    }
}
```

管理物理内存资源

- 目标与评价维度
- 基于位图的连续物理页分配
- 伙伴系统
- SLAB分配器
- Linux内核内存管理架构

伙伴系统(buddy system)

- 以“块”为单位，分配连续的物理内存页



伙伴系统(buddy system)

- 程序需要15KB内存，如何处理?

连续物理页

32 KB (2^3 x 4 KB)

伙伴系统(buddy system)

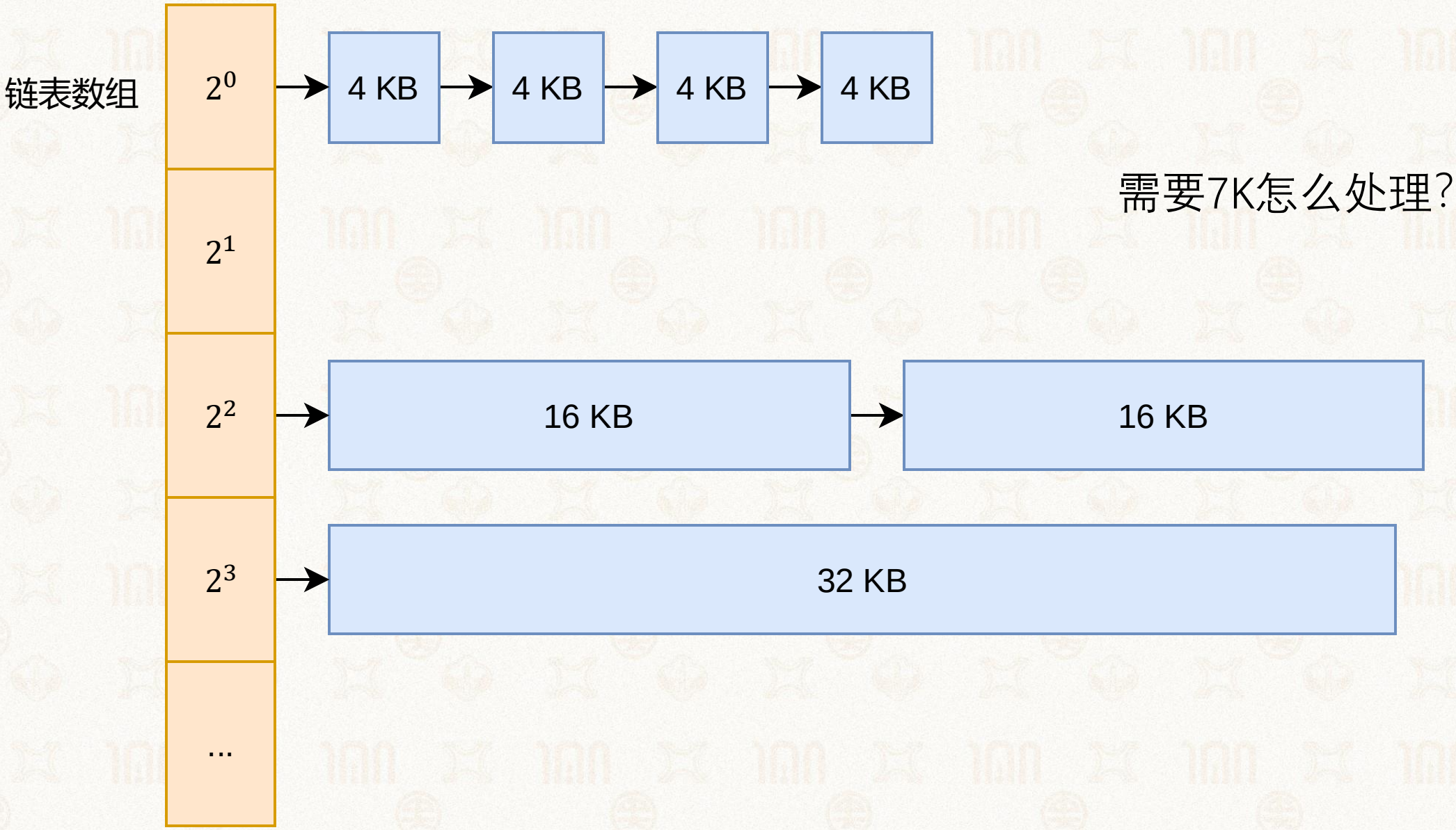
- 15 < 16，只需要16K大小的块，将32KB进行拆分

伙伴系统(buddy system)

- 将16KB(4个页)一起分配给程序，留另一半16KB伙伴保持空闲

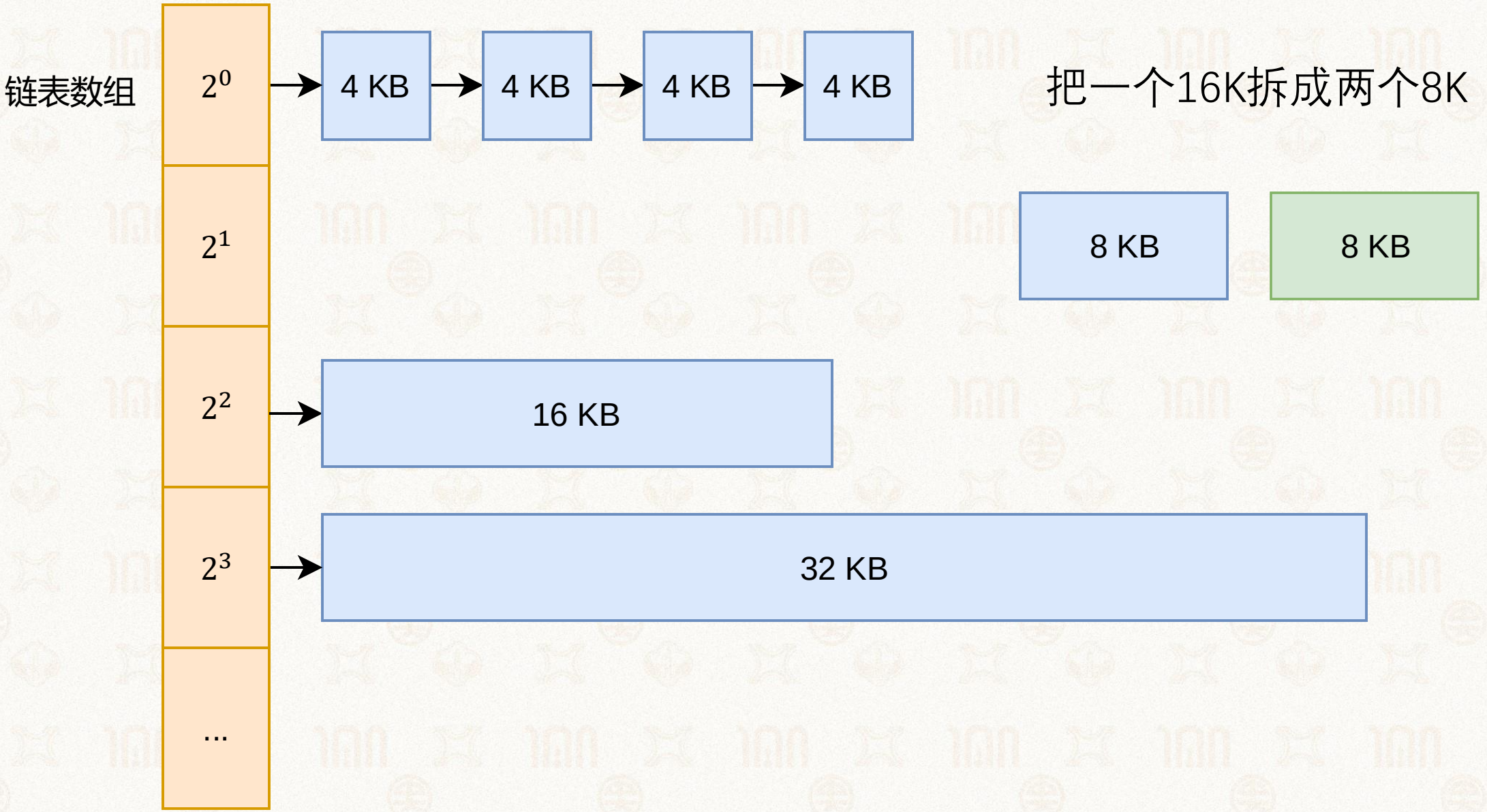
- 空闲块怎么管理?

伙伴系统：用链表数组管理空闲块

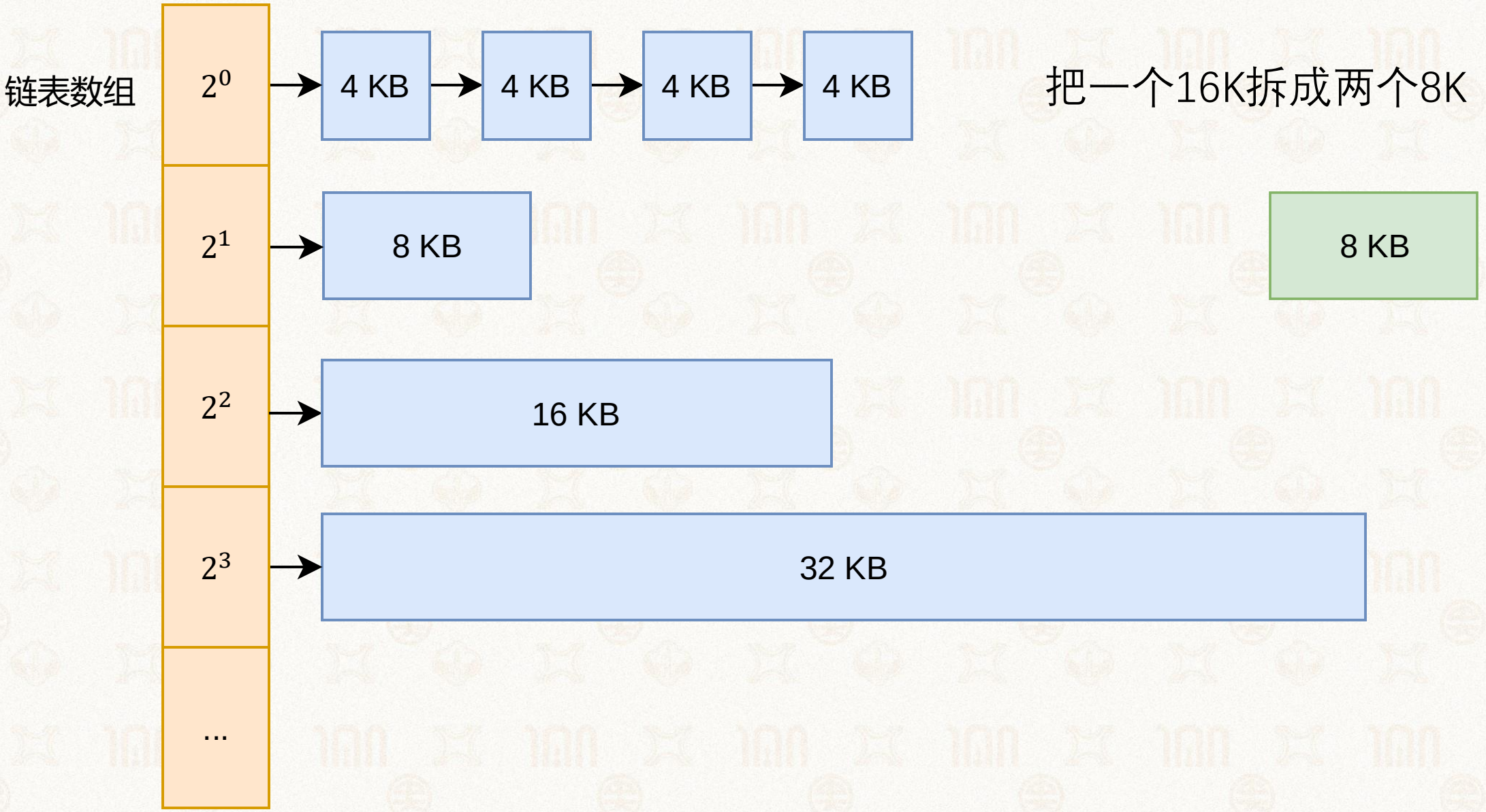


需要7K怎么处理?

伙伴系统：用链表数组管理空闲块



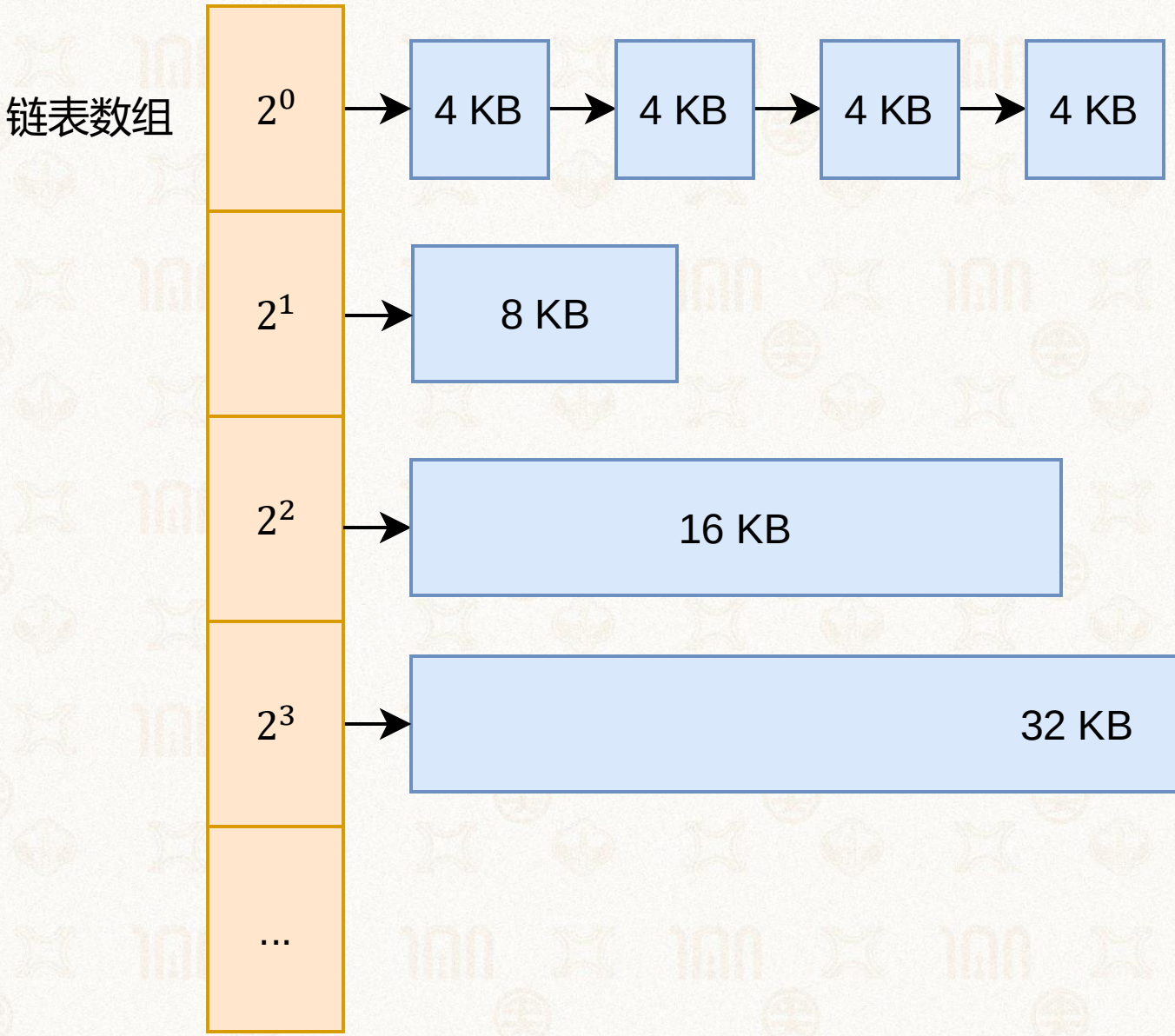
伙伴系统：用链表数组管理空闲块



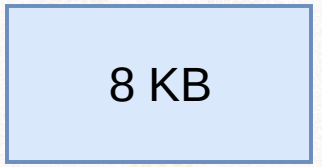
把一个16K拆成两个8K

8 KB

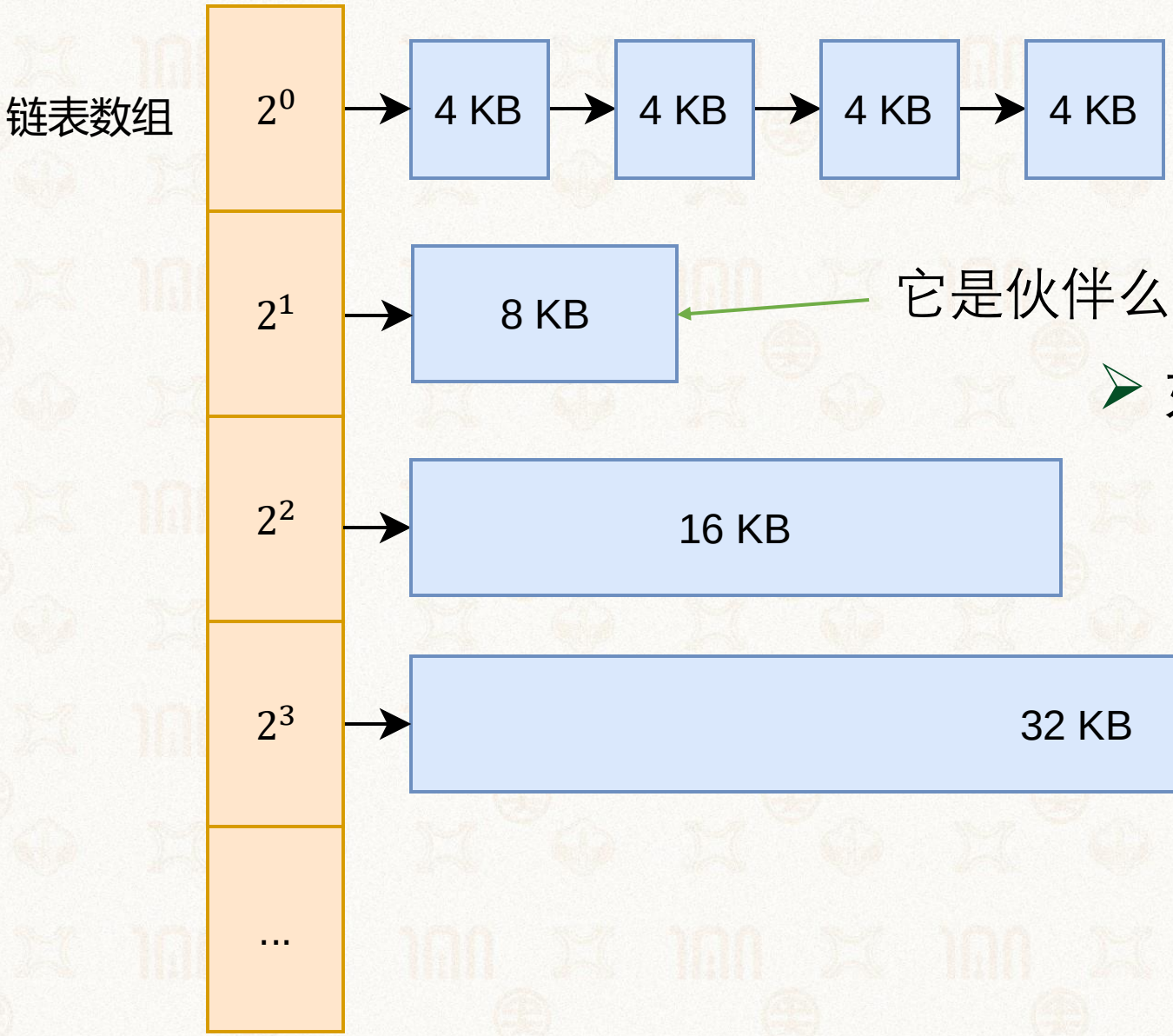
伙伴系统：用链表数组管理空闲块



反过来，现在释放了一个8KB的块，怎么管理？



伙伴系统：用链表数组管理空闲块



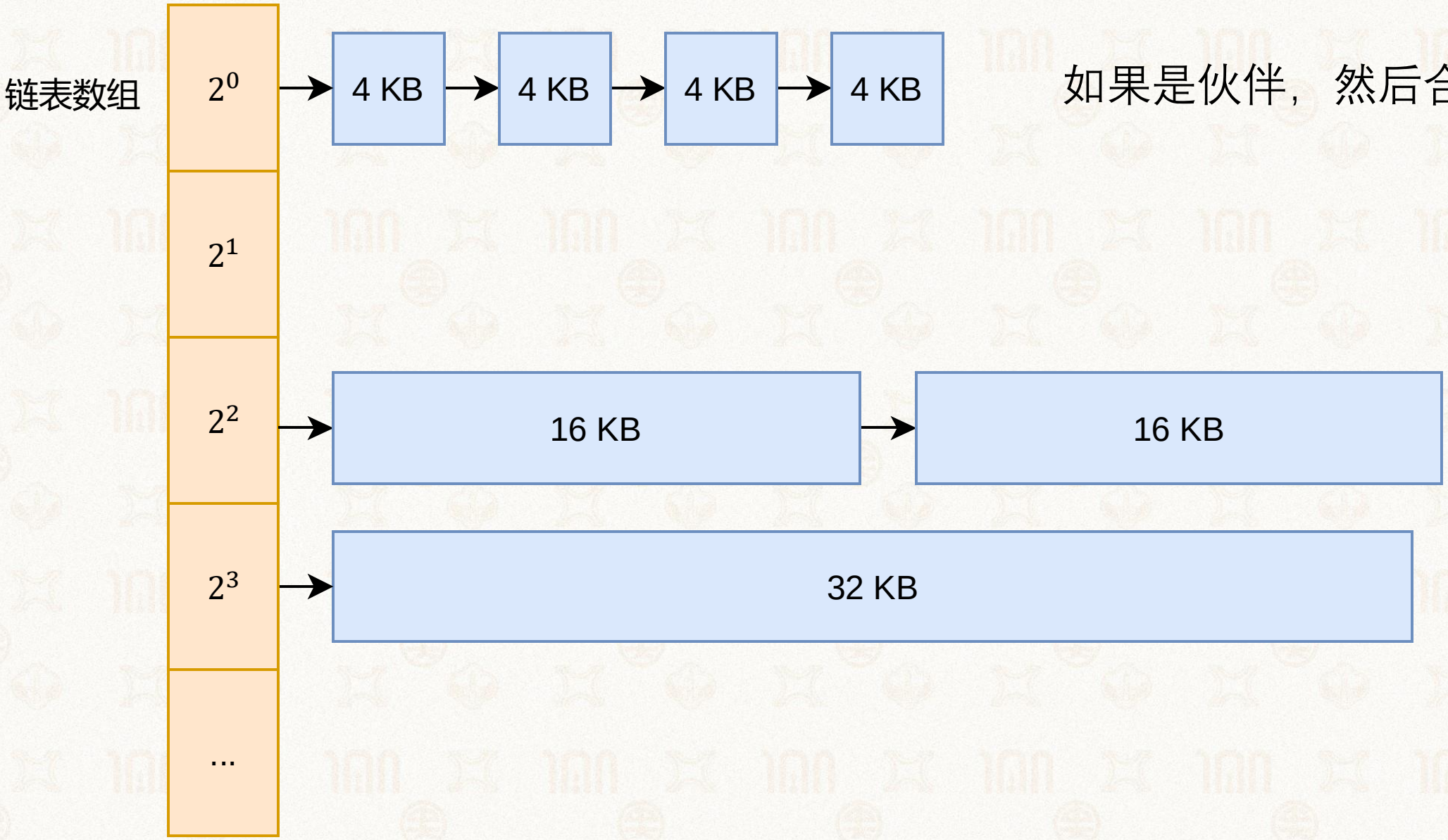
需要找到另一个8K伙伴，然后合并

8 KB

它是伙伴么？

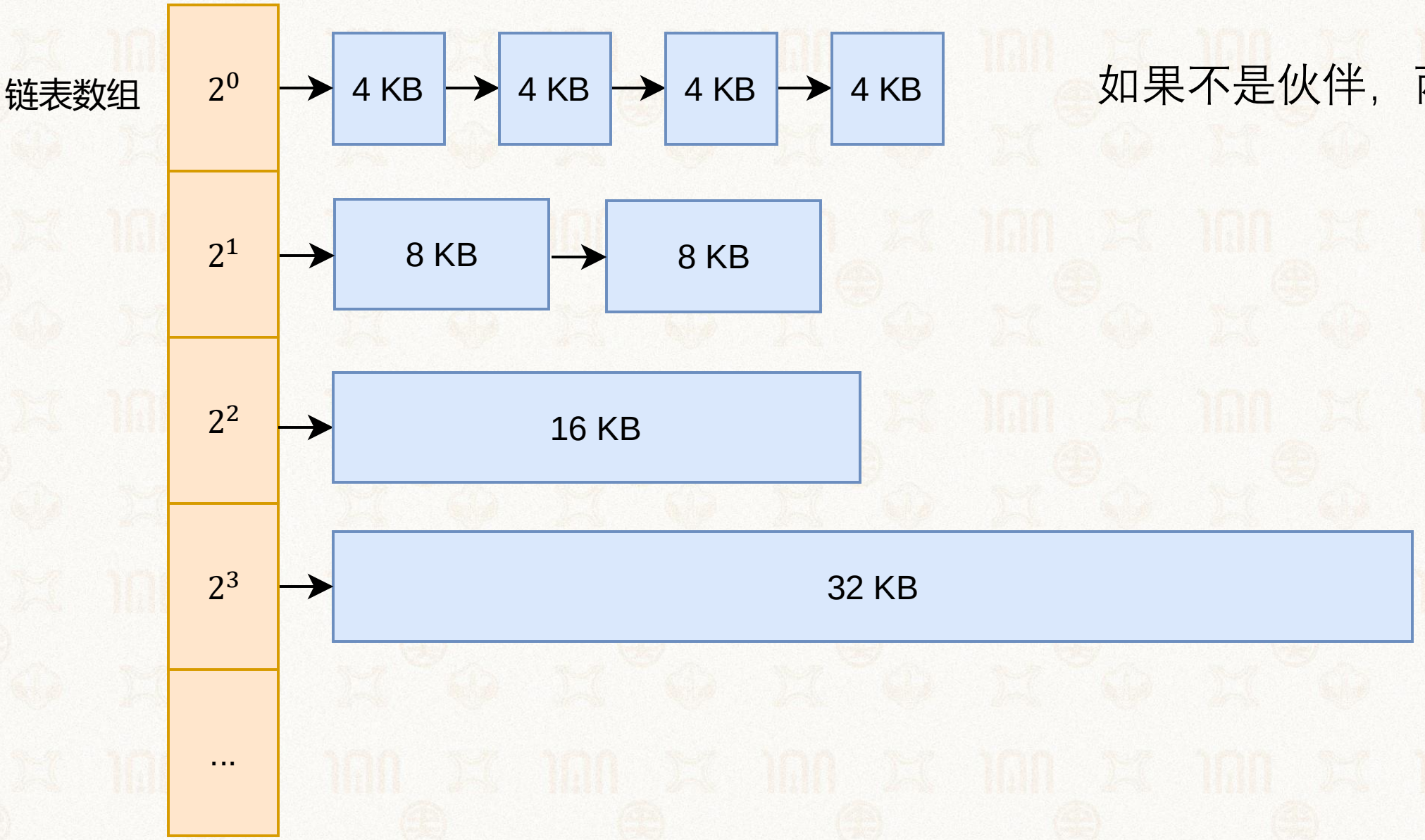
- 如何判定伙伴？
  - 两个块地址只有一位不同：
  - … 1010 0000 0000 0000
  - … 1000 0000 0000 0000

伙伴系统：用链表数组管理空闲块



如果是伙伴，然后合并成为16K

伙伴系统：用链表数组管理空闲块



如果不是伙伴，两个8K级联

伙伴系统的巧妙之处

- 高效地找到伙伴块
  - 互为伙伴的两个块的物理地址仅有一位不同
  - … 1010 0000 0000 0000
  - … 1000 0000 0000 0000

  - 一个是0，另一个是1

  - 块的大小决定是哪一位

建立在伙伴系统之上的分配器

- SLAB分配器家族 (Linux)
  - SLAB分配器
  - SLUB分配器
  - SLOB分配器
- 伙伴系统分配的最小单位是一个物理页（4K）
  - 操作系统里面的结构体大小常位几十、几百字节
  - 避免内部碎片

管理物理内存资源

- 目标与评价维度
- 基于位图的连续物理页分配
- 伙伴系统
- SLAB分配器
- Linux内核内存管理架构

SLAB分配器

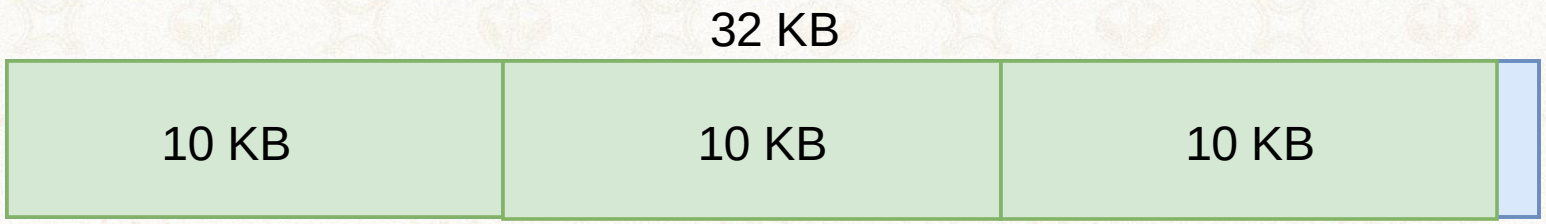
- 目标：快速分配小内存对象

- SLAB分配器历史
  - 上世纪 90 年代，Jeff Bonwick在Solaris 2.4中首创SLAB
  - 07年左右，Christoph Lameter在Linux中提出SLUB
    - SLAB的设计过于复杂
    - 在Linux-2.6.23及之后的版本中，成为默认分配器

- 发展过程中，针对内存稀缺场景又提出了SLOB

SLUB

- 观察
  - 操作系统频繁分配的对象大小相对比较固定
- 基本思想
  - 从伙伴系统获得大块内存
  - 进一步细分成固定大小的小块内存进行管理
  - 块大小通常是 2^n 个字节（一般来说，$3 \leq n < 12$）
  - 可以额外增加特殊大小如198字节从而减小内部碎片



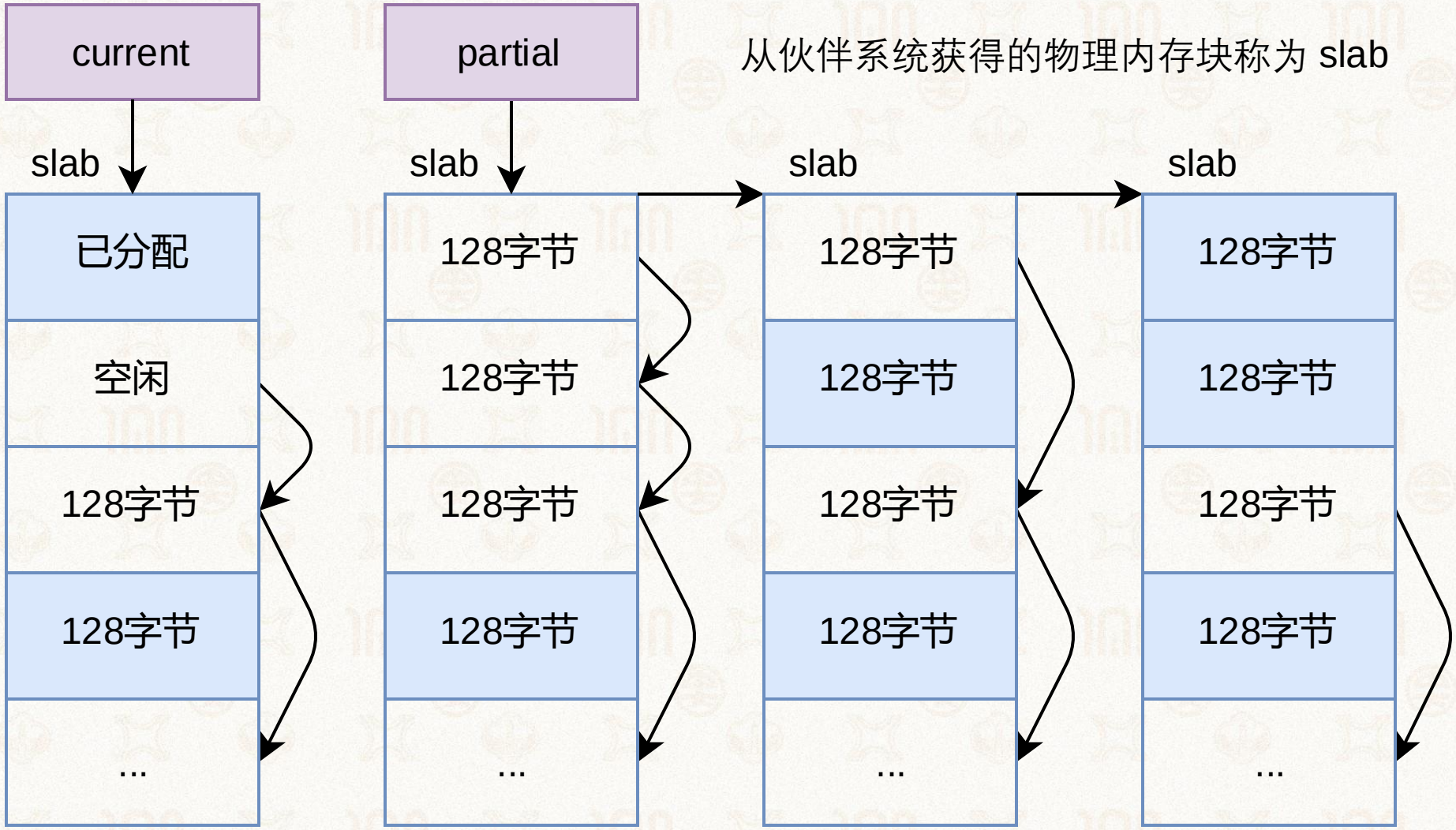
SLUB设计

- 只分配固定大小块
  - 对于每个固定块大小，SLUB 分配器都会使用独立的内存资源池进行分配
  - 采用best fit定位资源池

SLUB数据结构: 128字节的资源池示例

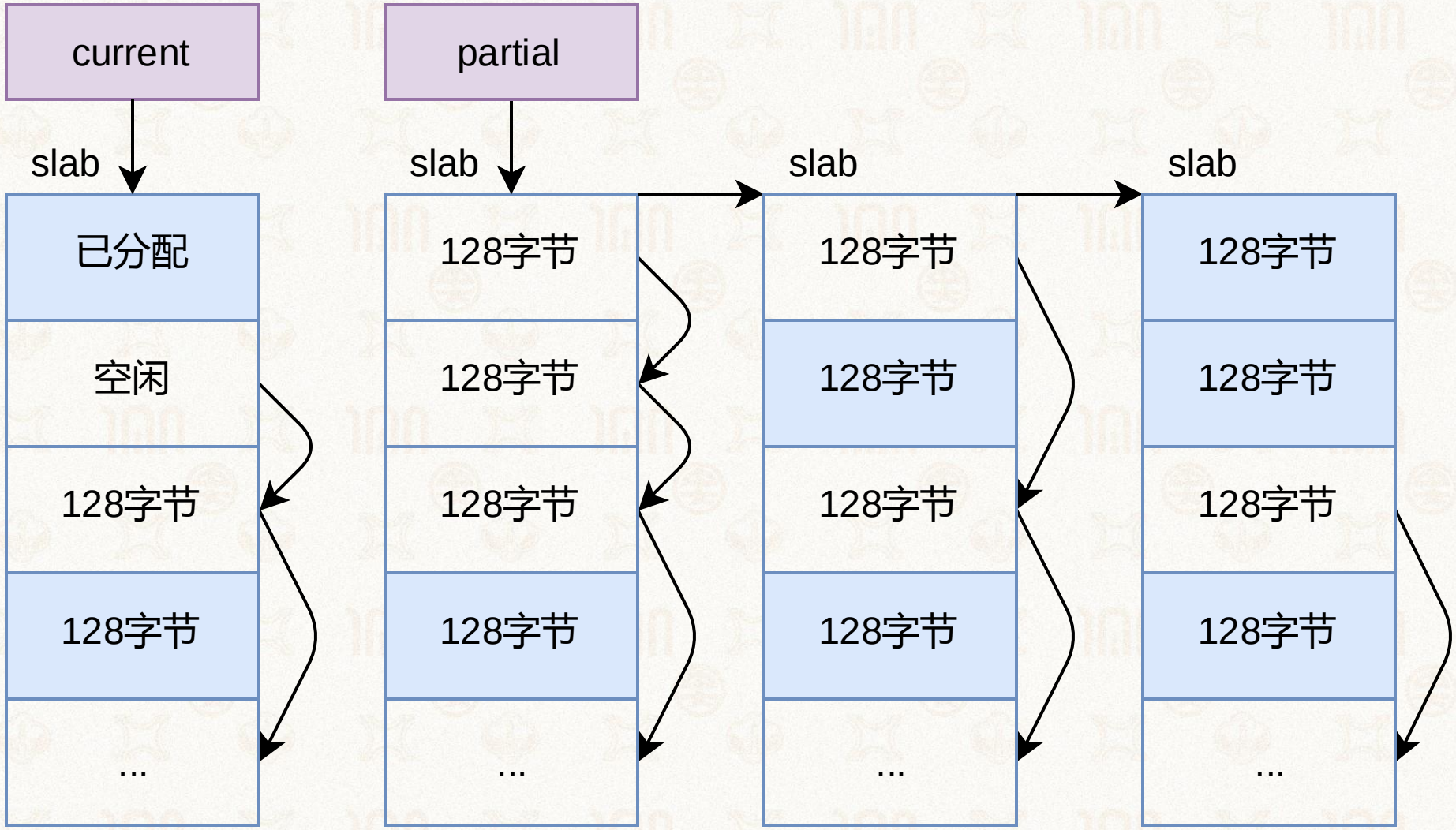
- 三个指针
  - Current仅指向一个slab
  - Partial指向未满slab链表
  - Full指向全满slab链表

slab内部组织为空闲链表



SLUB数据结构: 128字节的资源池示例

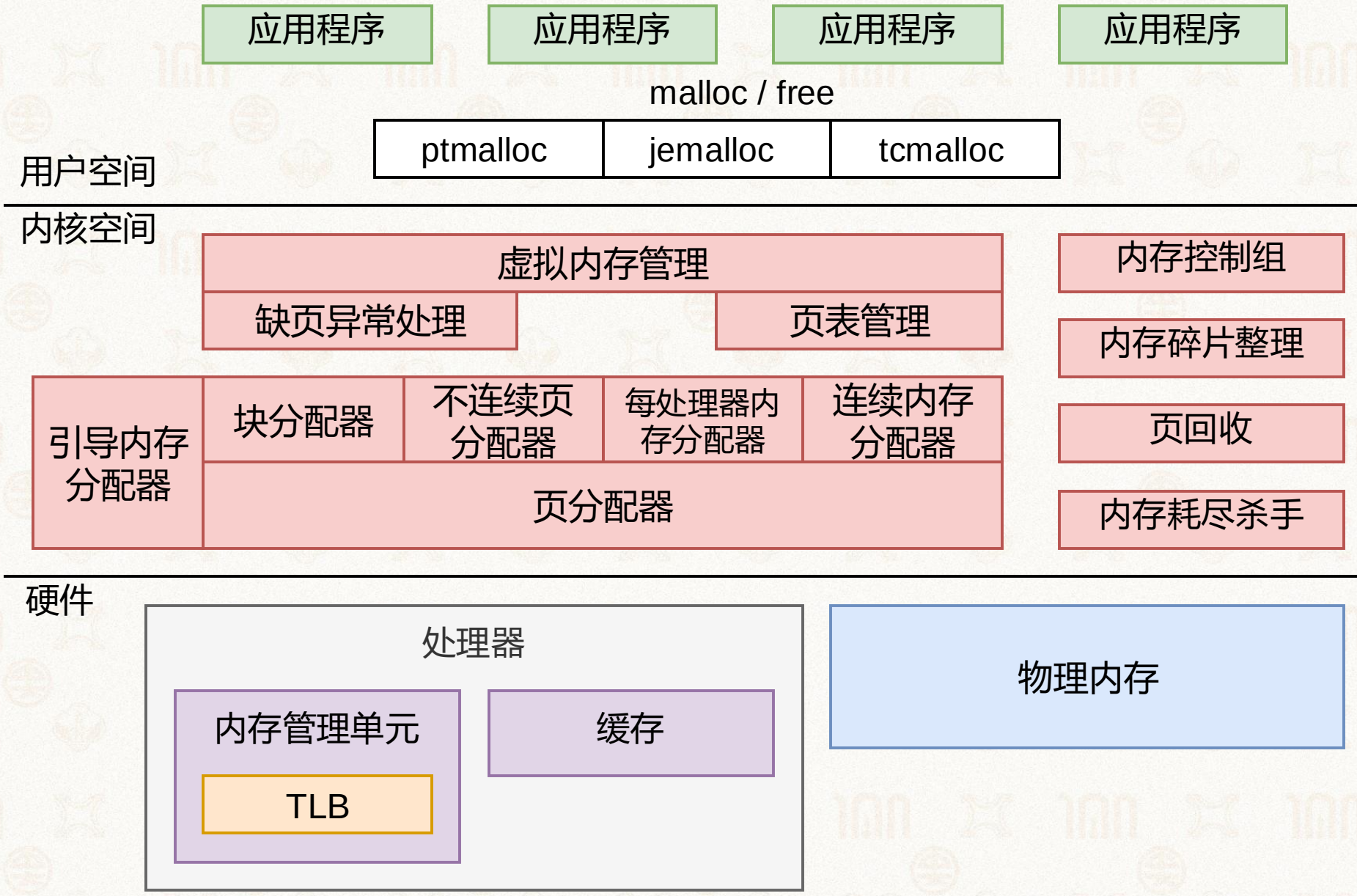
- 分配
  - 使用current slab
  - 若满
    - 从partial指向的slab中取空闲区域
    - 把current指向的slab移到full
- 释放
  - 到对应的slab
  - 移动 full 到 partial
  - 若partial全空则还给伙伴系统



管理物理内存资源

- 目标与评价维度
- 基于位图的连续物理页分配
- 伙伴系统
- SLAB分配器
- Linux内核内存管理架构

Linux内存管理架构



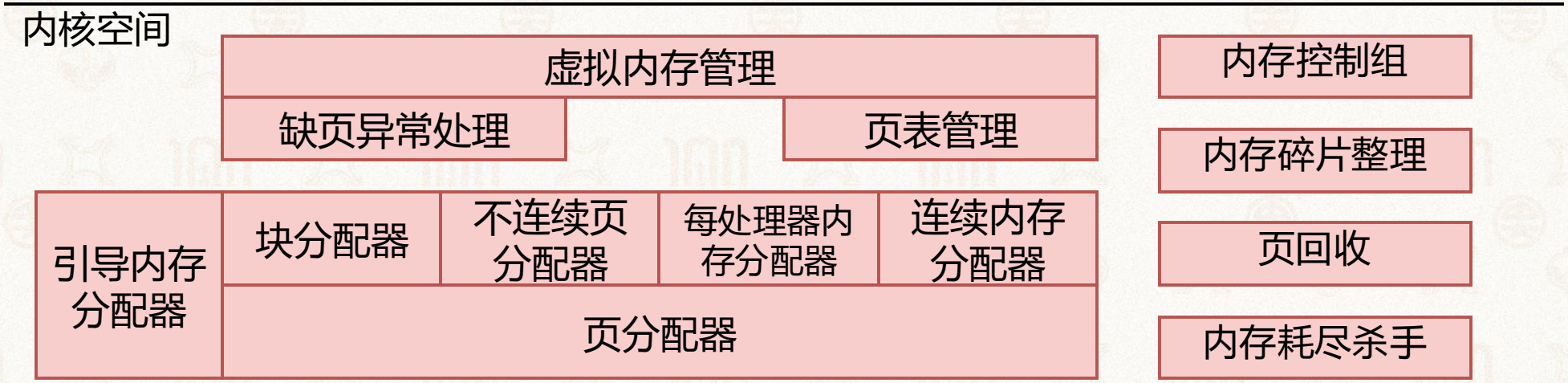
用户空间的使用

- malloc() 和 free() 是glibc库的内存分配器ptmalloc提供的接口
- ptmalloc使用系统调用brk或mmap向内核以页为单位申请内存，然后划分成小内存块分配给应用程序

- 另外两种内存分配器
  - tcmalloc是google提供的内存分配器，
  - jemalloc是FreeBSD的内存分配器

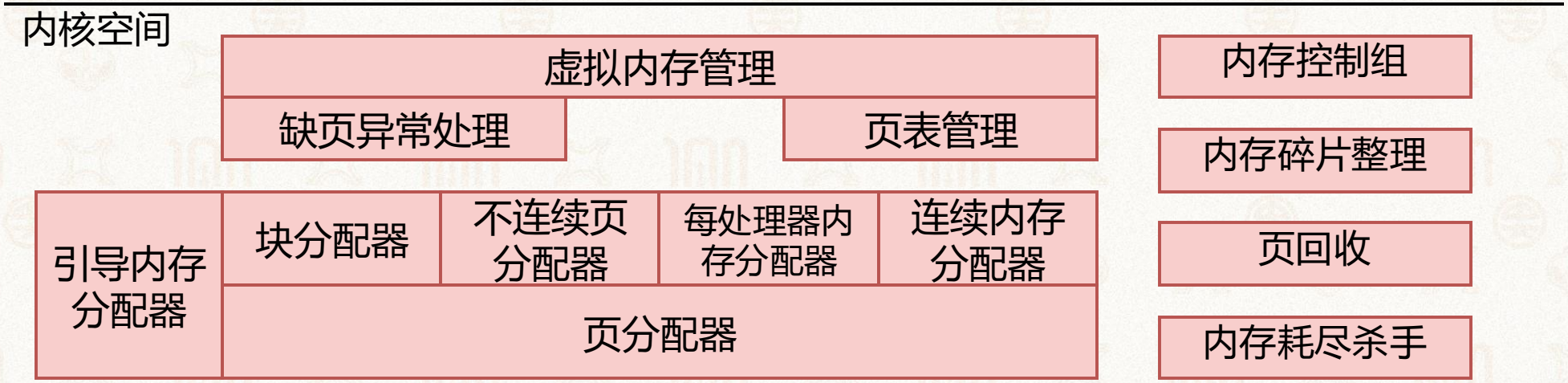
内核空间的基本功能

- 系统调用
  - sys_brk 用来扩大或收缩堆
  - sys_mmap 在内存映射区域分配虚拟页
  - sys_munmap 释放虚拟页
- 延迟分配物理内存策略
  - 第一次访问虚拟页，先触发缺页异常处理
- 页分配器：伙伴分配器
- 内核初始化时，需要引导内存分配器



内核空间的扩展功能

- 连续内存分配器(Contiguous Memory Allocator, CMA)
  - 给驱动程序预留连续空间
  - 如果驱动不用，留给应用程序
- 内存控制组
  - 控制每个应用程序占用的内存资源
- 内存耗尽杀手(Out-of-Memory killer, OOM killer)
  - 如果内存不足，且页回收失败，则把应用程序杀掉



物理页：page

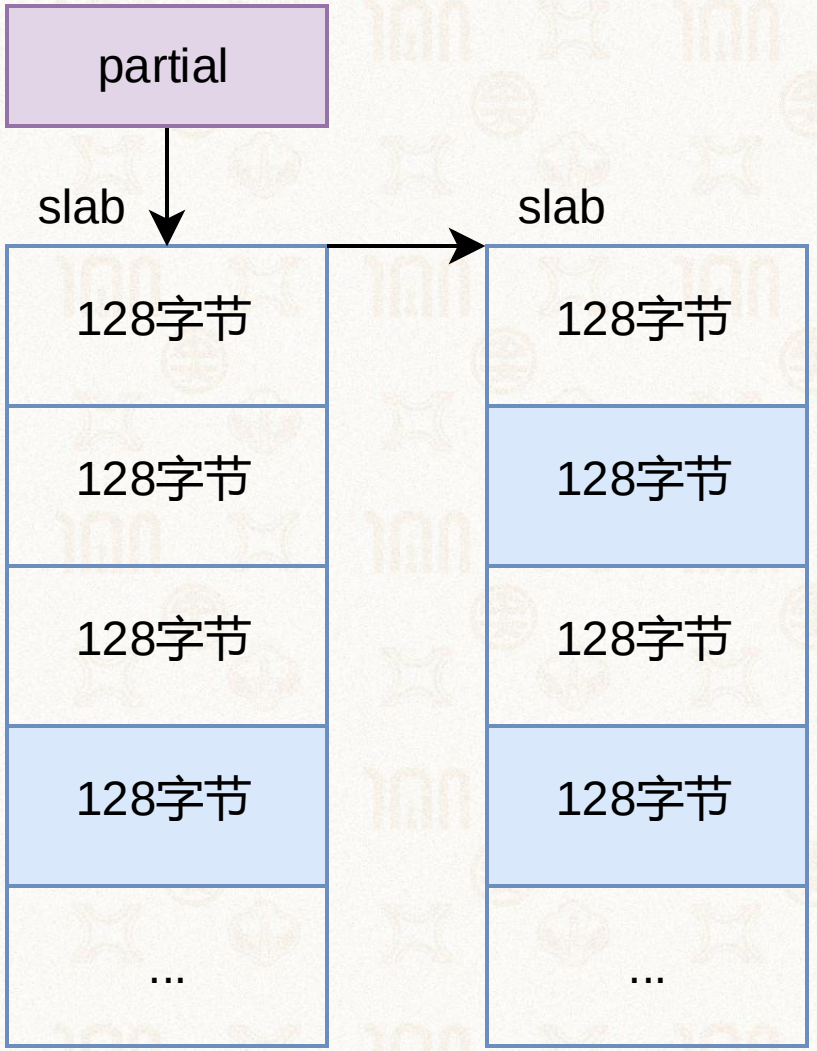
- 见：https://elixir.bootlin.com/linux/v5.16.14/source/include/linux/mm_types.h#L71

- 物理页数量巨大，尽量避免结构体新增成员导致占用过多内存
- 使用联合体，尽量减少结构体page的大小

https://elixir.bootlin.com/linux/v5.16.14/source/include/linux/mmzone.h#L809

Slab缓存

```
struct kmem_cache {
    struct kmem_cache_cpu __percpu *cpu_slab;
    /* Used for retrieving partial slabs, etc. */
    slab_flags_t flags;
    unsigned long min_partial;
    unsigned int size;  /* The size of an object including metadata */
    unsigned int object_size;/* The size of an object without metadata
*/
    struct reciprocal_value reciprocal_size;
    unsigned int offset;    /* Free pointer offset */
#ifdef CONFIG_SLUB_CPU_PARTIAL
    /* Number of per cpu partial objects to keep around */
    unsigned int cpu_partial;
    /* Number of per cpu partial pages to keep around */
    unsigned int cpu_partial_pages;
#endif
    const char *name;   /* Name (only for display!) */
    struct list_head list;  /* List of slab caches */
    unsigned int useroffset;    /* Usercopy region offset */
    unsigned int usersize;      /* Usercopy region size */
    struct kmem_cache_node *node[MAX_NUMNODES];
};
```



https://elixir.bootlin.com/linux/v5.16.14/source/include/linux/slub_def.h#L90

管理物理内存资源

- 目标与评价维度
- 基于位图的连续物理页分配
- 伙伴系统
- SLAB分配器
- Linux内核内存管理架构

谢谢

微信：suyuxin
钉钉：苏玉鑫
B站：https://space.bilibili.com/502854403
软工集市课程专区：https://ssemarket.cn/new/course
匿名提问箱：https://suask.me/ask-teacher/106/苏玉鑫