SimpleKernel优化现有内存管理代码的实现

一、项目设计方案

1.现有内存管理方案

- 先通过开源引导程序grub获取目前硬件的可用内存信息。
- 将可用内存划分为物理页，并计算总数。
- 在该内存中设置内核栈，并将其和内核代码一起映射到内核页表中。
- 使用页目录项-页表项-页表三级管理机制对物理内存进行管理。
- 使用first-fit算法和slab算法对用户可用页进行管理，包括堆和用户栈等。

2.改进方案

①对物理内存进行分区

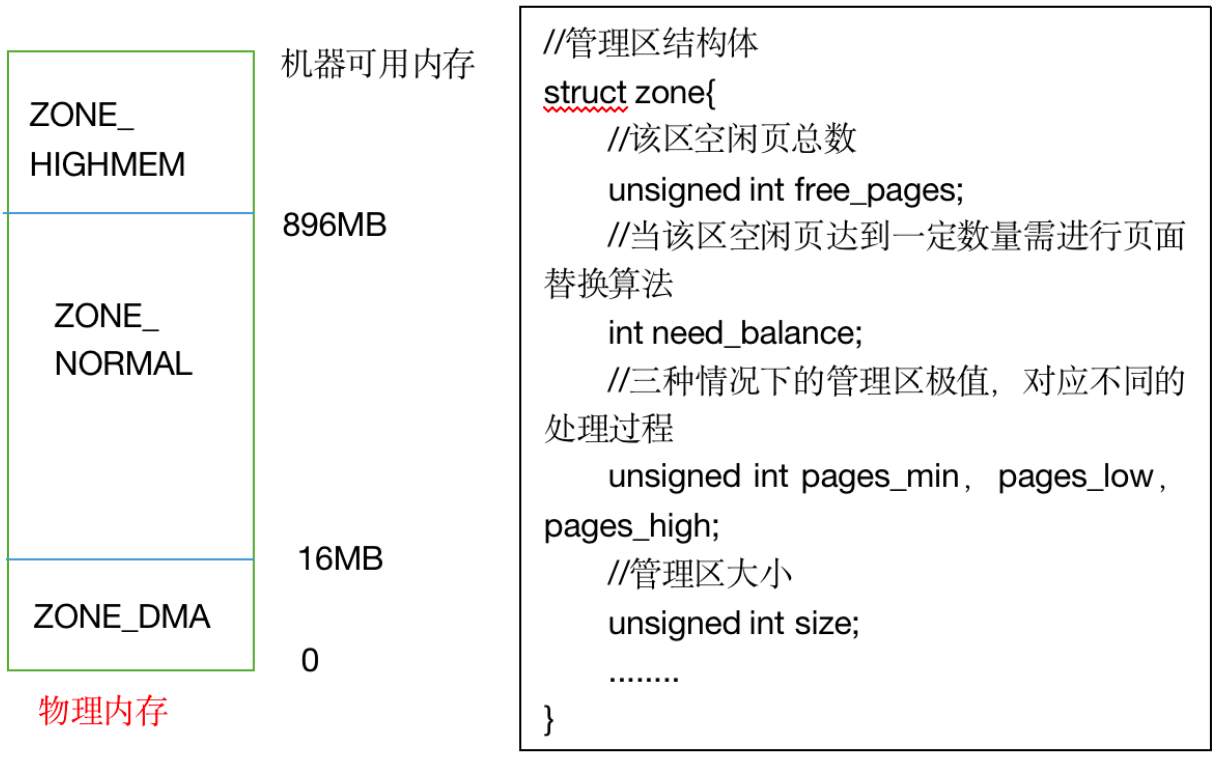

图1 物理内存划分为3个管理区

如图1所示，将机器可用内存分为3个管理区，其中0-16MB为ZONE_DMA区域，某些ISA设备可能需要用到它；16MB-896MB为ZONE_NORMAL区域，该区域内存由内核直接映射到线性地址空间的较高部分；896MB-可用内存末尾为ZONE_HIGHMEM区域，该部分为系统预留的内存，不能被内核直接映射。

②在ZONE_NALMAL中增加一段缓冲区，用于文件和进程间的传输。

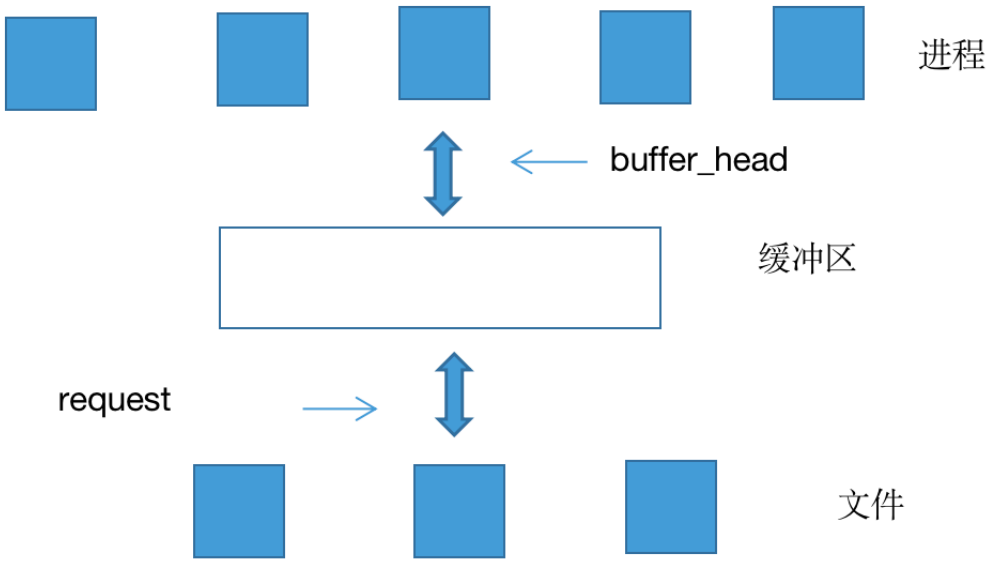

图2 在内存中增加一段缓冲区

缓冲区的设计思想就是进程访问内存的速度快，而文件是放在存储设备上的，其访问内存的速度很慢。进程在访问一个文件时，都需要将其读入内存中，而在关闭文件时，也需将其更改信息放入存储设备，缓冲区作为内存的一部分，只增加了内存到缓冲区的复制过程，但因为缓冲区具有共享机制,当其他进程访问相同文件时，只需从缓冲区读取，而无需访问存储设备，运行效率能提高2个量级。此外，一个缓冲块和一个页大小相等，十分方便管理。

缓冲区的管理主要由下面2个结构体实现，如图3所示。

```
//缓冲块结构体
struct buffer_head{
        //块数据
        char *data;
        //该块是否被改写过
        unsigned char dirt;
        //该块被多少进程占用
        unsigned int count;
        //该块是否被锁
        unsigned char lock;
        //等待使用缓冲块的进程链表
        struct task_struct *wait;
        //缓冲块链表
        struct buffer_head *next;
        ........
}
```

```
//请求项结构体
struct buffer_head{
        //存储设备,可能是硬盘或软盘
        int dev;
        //文件的读或写
        int cmd;
        //文件对应的扇区号
        unsigned int sector;
        //该请求项需要使用的缓冲块
        struct buffer_head *bh;
        //请求项链表
        struct request *next;
        ........
}
```

图3 缓冲区管理结构体

③页面置换机制的实现

物理内存是有限的，这就导致一些物理页并不能长时间的停留在内存中，需要先放入硬盘等存储设备进行一个缓冲。

1. 并不是所有物理页都可以放入硬盘中，只有映射到用户空间且被用户程序直接访问的页面能够被放入硬盘，而内核使用的页面则不能被置换出去。
2. 目前内存中物理页的使用情况放在一个数组中，所以需要定义两个链表，一个用于管理已被放入硬盘的物理页，一个用于管理未被放入硬盘的物理页。
3. 当发生缺页中断或者可用物理页达到阈值时，使用最优页面置换算法或者先进显出页面置换算法对物理页进行管理，清理出部分物理页用于相关的操作。

④共享内存和写时复制机制的实现

共享内存主要是为了将来不同进程之间能够通过共享内存实现数据共享，这也可看作进程间通信的一种方法，而写时复制则是通过父子进程的页面共享实现的，当fork时父子进程对应的虚拟页映射到同一个物理页中，主要通过以下结构体实现，如图4所示。

```
//共享内存结构体
struct vma_struct{
    //对应页表信息
    struct mm_struct *vm_mm;
    //共享该内存的进程链表
    struct task_struct *share;
    //该内存被多少进程占用
    unsigned int count;
    ........
}
```

图4 共享内存结构体

⑤优化后进程申请内存方式如图5所示。

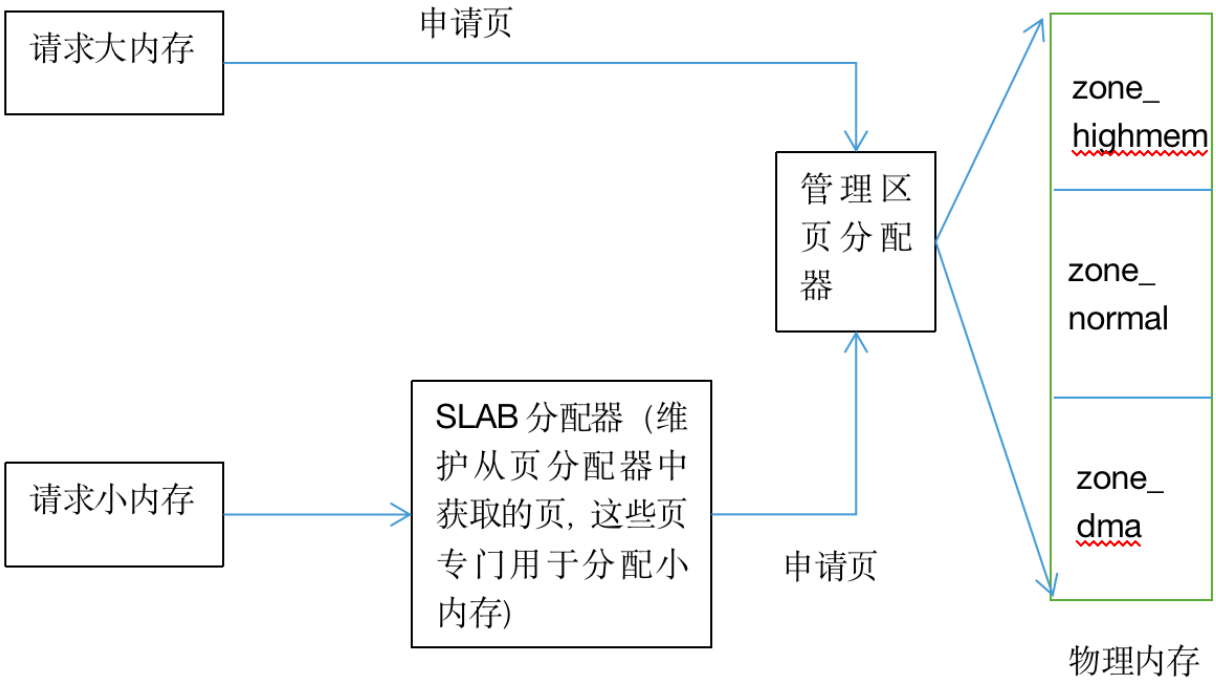

图5 内存申请流程

二、项目时间规划

为了让新手更快速的了解内核的整体内存管理架构，整个项目主要包含4类工作，如下所示。

工作内容	产出
整体内存规划与架构	内存规划文档
功能模块的架构与思考过程	功能模块文档
编码	易阅读的代码（注释和去耦合等）
调试	调试文档以及对应的测试用例

任务规划如下表所示。

月份	核心任务	周	具体任务
2020年7月	1、熟悉当前内存管理方案。 2、增加内存区域隔离机制，为多进程系统的实现奠定基础。 3、增加缓冲区模块。	第一周	整体内存规划与架构。
		第二周	对物理内存分区模块进行架构，并写下思考过程，同时进行编码和调试。
		第三周	对缓冲区模块进行架构，并写下思考过程，同时进行编码及调试。
		第四周	缓冲区编码及调试。
2020年8月	1、增加页面置换机制。 2、增加共享内存和写时复制机制。	第一周	缓冲区编码及调试。
		第二周	对页面置换机制进行架构，并写下思考过程，同时进行编码及调试。
		第三周	页面置换机制编码及调试。
		第四周	对共享内存和写时复制机制进行架构，并写下思考过程，同时进行编码及调。
2020年9月	1、完善测试文档和说明文档。	第一周	共享内存和写时复制机制编码及调试。
		第二周	优化进程申请内存方式，完善内存分配算法并写下思考过程，同时编码及调试。
		第三周	对整体内存规划进行调试，找出bug并改正。
		第四周	编写详细的内存规划文档，让新手更容易入门。

三、申请理由

我一直很想开发一个简单的os，由于各种原因，目前也只完成了引导程序的编写，kernel部分完成了一些简单的内存打印等，申请这个项目一方面是自己的兴趣爱好，另一方面是给自己一个ddl，有压力才会有动力，完成这个项目不仅可以让自己深入了解linux内核，也可以为开源社区做点贡献，从而帮助初学者更好地理解操作系统的内存管理模块。无论这次申请成功与否，未来也会加入到这方面的工作，完善更多的功能模块，比如文件系统，外设驱动的编写等。

四、参考资料

1. 深入理解Linux虚拟内存(Linux的zone机制)。
2. Linux内核设计的艺术(缓冲区设计)。

3. ucore操作系统指导书(页面置换机制和共享内存及写时复制机制)。