@[TOC](Mit6.S081-实验8-locks)

# 一、实验准备

在本实验，你将在 重构代码以提升并行 上获得经验。

多核机器低并行的常见症状是high lock contention。

提高并行，通常包括改变数据结构和锁策略来减少争用。

你将为xv6 memory allocator和block cache这么做。

在写代码之前，确保阅读xv6 book的以下部分：

Chapter6："Locking"和对应代码

Section3.5："Code：Physical memory allocator"

Section8.1到8.3："Overview","Buffer cache layer","Code:Buffer cache"

切换到lock分支

```bash

git fetch、git checkout lock、make clean

```

# 二、Memory allocator

## 1，实验要求

程序user/kalloctest压测xv6的memory allocator：

3个进程grow、shrink它们的地址空间，导致一些调用：kalloc和kfree。

kalloc和kfree获取kmem.lock。

kalloctest打印（”#fetch-and-add”）acquire中的循环迭代编号，由于尝试获取另外一个核已经持有的锁，kem lock和少量其它锁。

acquire循环迭代编号是一个lock contention的粗略测量。

kalloctest的输出看起来类似之前完成的实验。

```bash

$ kalloctest

start test1

test1 results:

--- lock kmem/bcache stats

lock: kmem: #fetch-and-add 83375 #acquire() 433015

lock: bcache: #fetch-and-add 0 #acquire() 1260

--- top 5 contended locks:

lock: kmem: #fetch-and-add 83375 #acquire() 433015

lock: proc: #fetch-and-add 23737 #acquire() 130718

lock: virtio\_disk: #fetch-and-add 11159 #acquire() 114

lock: proc: #fetch-and-add 5937 #acquire() 130786

lock: proc: #fetch-and-add 4080 #acquire() 130786

tot= 83375

test1 FAIL

```

对于每个锁，acquire持有：为了该锁，acquire的调用次数；

acquire循环中，尝试获取但没能设置锁的次数。

kalloctest调用一个system call，让kernel打印那些对于kmem、bcache lock计数，（这是本实验的关注点），也打印5个争用最强烈的锁。

如果存在锁争用，acquire循环迭代中的计数将很多大。

system call返回 为了获取kmem和bcache lock的循环迭代总数。

对于本实验，你必须使用一个专用unloaded多核机器。

如果你使用一个正在做其他事的机器，kalloctest打印的计数将是无用的。

你可以使用专用Athena workstation，或你的笔记本电脑，但不要使用dialup机器。

kalloctest中锁争用的根本原因是kalloc()只有一个free list，受单lock保护。

为了移除锁争用，你将不得不重新设计memory allocator来避免一个单独lock和list。

基本想法是：每个CPU持有一个free list，每个list有其自己的锁。

不同CPU上的分配和释放可以并行执行，因为每个CPU将在不同的list上操作。

主要挑战将是处理这样的场景：

某个CPU上的free list空了，但另外CPU list有free memory。

在这种情况下，cpu必须steal另外CPU free list的一部分。

stealing可能引发锁争用，但不常见。

你的工作是实现每个CPU freelist，当CPU free list为空时stealing。

你必须让所有锁名以kmem开头。

你应该为每把锁调用initlock，传入一个以kmem开头的名字。

执行kalloctest来看是否你的实现已经减少了锁争用。

为了检查仍可以分配所有内存，执行usertests sbrkmuch。

你的输出将和下面显示的相似，减少了kmem lock争用总数，尽管具体数字不同。

确保usertests中所有测试通过。make grade应该表明kalloctests通过。

```bash

$ kalloctest

start test1

test1 results:

--- lock kmem/bcache stats

lock: kmem: #fetch-and-add 0 #acquire() 42843

lock: kmem: #fetch-and-add 0 #acquire() 198674

lock: kmem: #fetch-and-add 0 #acquire() 191534

lock: bcache: #fetch-and-add 0 #acquire() 1242

--- top 5 contended locks:

lock: proc: #fetch-and-add 43861 #acquire() 117281

lock: virtio\_disk: #fetch-and-add 5347 #acquire() 114

lock: proc: #fetch-and-add 4856 #acquire() 117312

lock: proc: #fetch-and-add 4168 #acquire() 117316

lock: proc: #fetch-and-add 2797 #acquire() 117266

tot= 0

test1 OK

start test2

total free number of pages: 32499 (out of 32768)

.....

test2 OK

$ usertests sbrkmuch

usertests starting

test sbrkmuch: OK

ALL TESTS PASSED

$ usertests

...

ALL TESTS PASSED

```

## 2，一些提示

你可以使用kernel/param.h中常量NCPU。

让freerange把所有空闲内存给正在执行freerange的CPU。

函数cpuid返回当前cpu核编号，但仅在中断关闭时可以调用它并使用它的结果。

你应该使用push\_off()和pop\_off()来关闭、开启中断。

看kernel/sprintf.c中的snprintf函数，为string formatting找灵感。

可以将所有锁命名为kmem。

## 3，具体实现

1）修改kernel/kalloc.c，修改原有结构体声明kmem，定义结构体数组kmemArray[NCPU]。

![在这里插入图片描述](https://img-blog.csdnimg.cn/20210310191225419.png)

2）修改kernel/kalloc.c，修改kinit方法，对NCPU个kmem结构体初始化。

![在这里插入图片描述](https://img-blog.csdnimg.cn/20210310200739329.png)

3）修改kernel/kalloc.c，修改kfree方法，获取当前cpuid，释放对应freelist。

![在这里插入图片描述](https://img-blog.csdnimg.cn/20210310191512104.png?x-oss-process=image/watermark,type\_ZmFuZ3poZW5naGVpdGk,shadow\_10,text\_aHR0cHM6Ly9ibG9nLmNzZG4ubmV0L3UwMTM1Nzc5OTY=,size\_16,color\_FFFFFF,t\_70)

4）修改kernel/kalloc.c，修改kalloc方法，当前CPU freelist为空时，从其他CPU freelist处获取。

![在这里插入图片描述](https://img-blog.csdnimg.cn/20210310201105628.png?x-oss-process=image/watermark,type\_ZmFuZ3poZW5naGVpdGk,shadow\_10,text\_aHR0cHM6Ly9ibG9nLmNzZG4ubmV0L3UwMTM1Nzc5OTY=,size\_16,color\_FFFFFF,t\_70)

## 4，测试效果

![在这里插入图片描述](https://img-blog.csdnimg.cn/20210310201549756.png?x-oss-process=image/watermark,type\_ZmFuZ3poZW5naGVpdGk,shadow\_10,text\_aHR0cHM6Ly9ibG9nLmNzZG4ubmV0L3UwMTM1Nzc5OTY=,size\_16,color\_FFFFFF,t\_70)

![在这里插入图片描述](https://img-blog.csdnimg.cn/20210310202457796.png)

# 三、Buffer cache