Lab5 Lock

21307110014

本次实验目标是通过减少锁竞争以提高系统性能。

1 内存分配器

1.1 实现

- In kernel/kalloc.c

1. 把共享的内存链表`kmem`改为每个CPU独立的链表`kem[NCPU]`
2. 在`kinit`中初始化所有`kmem[i]`的锁
3. 在`kfree`和`kalloc`中需要用`cpuid()`得到当前cpu的索引，从而找到对应的内存链表
4. 在`kalloc`中还需要实现当该cpu的freelist已满时从其他cpu的freelist那里拿一个run过来的机制

1.2 运行结果

```
$ kalloctest
start test1
test1 results:
--- lock kmem/bcache stats
lock: kmem: #fetch-and-add 0 #acquire() 67096
lock: kmem: #fetch-and-add 0 #acquire() 182106
lock: kmem: #fetch-and-add 0 #acquire() 183896
lock: bcache: #fetch-and-add 0 #acquire() 1376
--- top 5 contended locks:
lock: proc: #fetch-and-add 48671 #acquire() 150248
lock: proc: #fetch-and-add 48162 #acquire() 150245
lock: proc: #fetch-and-add 19410 #acquire() 150308
lock: proc: #fetch-and-add 10797 #acquire() 150311
lock: virtio_disk: #fetch-and-add 10167 #acquire() 114
tot= 0
test1 OK
start test2
total free number of pages: 32499 (out of 32768)
.....
test2 OK
```

2 磁盘缓存

2.1 实现

- In kernel/buf.h

1. 在buf结构体中新加一个time变量，用于LRU replacement

- In kernel/bio.c

2. 把`bcache`从链表改为哈希表（多头链表，且改为单向），每个头有自己的锁。哈希函数即用trivial的取模
3. `binit`中初始化每个头的锁，然后把buf里的块均匀分配到每个头下面

```
void binit(void)
{
  struct buf *b;

  for (int i = 0; i < NBUCKETS; i++) {
    initlock(&bcache.lock[i], "bcache");
  }

  // append bufs to buckets
  for (b = bcache.buf; b < bcache.buf+NBUF-1; b++) {
    int i = (b - bcache.buf) % NBUCKETS;
    if (!bcache.head[i].next) bcache.head[i].next = b;
    else b->next = bcache.head[i].next, bcache.head[i].next = b;
    initsleeplock(&b->lock, "buffer");
  }
}
```

4. 修改`bget`逻辑。判断所查找的块是否在缓存中需要遍历各个头，而如果查找块不在缓存需要寻找一个LRU的unreferenced的块。这时可能需要访问其他头，为避免死锁，需要为不同头之间指定一个偏序关系，这里由`can_lock`定义：如果当前头为`i`，只能访问从`i`到`i+6`（`%13`）的这7个头，可以证明这样满足`can_lock(i, j) == !can_lock(j, i)`。

```
int can_lock(int i, int j) {
  for (int k = 0; k <= NBUCKETS/2; k++) {
    if ((i+k) % NBUCKETS == j) return 1;
  }
  return 0;
}
```

寻找LRU块时先确定目标块所在的头，再遍历头找到目标块更新和返回。

```
  // ...
  // Not cached.
  // Recycle the least recently used (LRU) unused buffer.
  int mni = -1;
  uint mnt = __UINT32_MAX__;
  // find the LRU unused buffer
  for (int j = 0; j < NBUCKETS; j++) {
    if (!can_lock(id, j)) continue;
    if (j != id) acquire(&bcache.lock[j]);
    for (b = bcache.head[j].next; b; b = b->next) {
      if (b->refcnt) continue;
      if (b->time < mnt) {
        mnt = b->time;
        if (~mni && mni != j && holding(&bcache.lock[mni]))
          release(&bcache.lock[mni]);
```

```
        mni = j;
      }
    }
    if (j != id && j != mni && holding(&bcache.lock[j]))
      release(&bcache.lock[j]);
  }
  if (mni == -1)
    panic("bget: no buffers");
  selected = bcache.head[mni].next;
  for (b = &bcache.head[mni]; b->next; b = b->next) {
    if ((b->next)->refcnt == 0 && (b->next)->time == mnt) {
      selected = b->next;
      b->next = b->next->next;
      break;
    }
  }
  if (mni != id && holding(&bcache.lock[mni]))
    release(&bcache.lock[mni]);

  for (b = &bcache.head[id]; b->next; b = b->next);
  b->next = selected;
  selected->next = 0;
  selected->dev = dev;
  selected->blockno = blockno;
  selected->valid = 0;
  selected->refcnt = 1;
  if (holding(&bcache.lock[id]))
    release(&bcache.lock[id]);
  acquiresleep(&selected->lock);
  return selected;
```

2.2 运行结果

```
$ bcachetest
start test0
test0 results:
--- lock kmem/bcache stats
lock: kmem: #fetch-and-add 0 #acquire() 32987
lock: kmem: #fetch-and-add 0 #acquire() 157
lock: kmem: #fetch-and-add 0 #acquire() 44
lock: bcache: #fetch-and-add 0 #acquire() 4159
lock: bcache: #fetch-and-add 0 #acquire() 2150
lock: bcache: #fetch-and-add 0 #acquire() 4301
lock: bcache: #fetch-and-add 0 #acquire() 4346
lock: bcache: #fetch-and-add 0 #acquire() 6353
lock: bcache: #fetch-and-add 0 #acquire() 6342
lock: bcache: #fetch-and-add 0 #acquire() 6625
lock: bcache: #fetch-and-add 0 #acquire() 6649
lock: bcache: #fetch-and-add 0 #acquire() 7225
lock: bcache: #fetch-and-add 0 #acquire() 6221
lock: bcache: #fetch-and-add 0 #acquire() 4164
lock: bcache: #fetch-and-add 0 #acquire() 4164
lock: bcache: #fetch-and-add 0 #acquire() 2161
--- top 5 contended locks:
lock: virtio_disk: #fetch-and-add 135163 #acquire() 1104
lock: proc: #fetch-and-add 120292 #acquire() 71571
lock: proc: #fetch-and-add 72181 #acquire() 71541
lock: proc: #fetch-and-add 37589 #acquire() 71892
lock: proc: #fetch-and-add 16665 #acquire() 71575
tot= 0
test0: OK
start test1
test1 OK
```