Lab5 Copy-on-write fork

2023.11.21

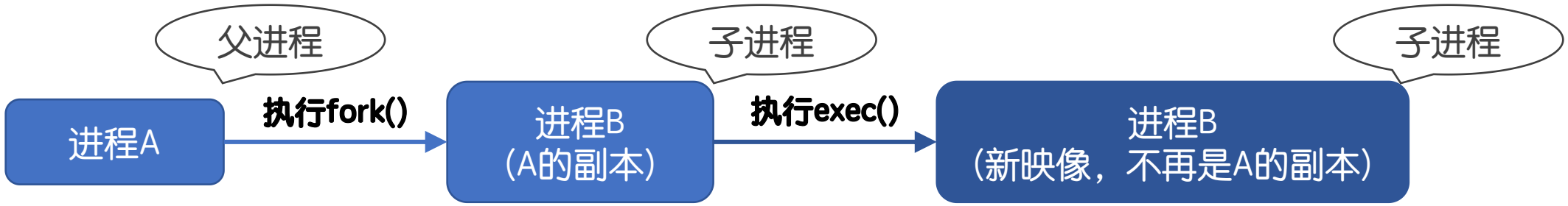
目录

- fork() 系统调用
  - 传统fork()+exec()调用
  - Copy-on-write fork()

- 实验
  - 实验配置及步骤
    - Tips
    - 结果示例
  - 提交方式

传统fork()+exec()调用

- xv6对一个进程调用fork(*kernel/proc.c*)，会创造一个跟当前进程「父进程」完全相同的「子进程」（除pid外）；
- 传统做法下，fork 会将父进程的整个内存布局都拷贝到子进程空间中，父、子进程之间的数据段和堆栈是相互独立的。
- **exec()系统调用**：
  - fork后常会使用exec装载新的程序覆盖当前进程内存空间中的映像，从而执行不同任务
  - exec系列函数在执行时会直接替换掉当前进程的地址空间



传统调用方式的问题

- 传统做法
  - 通过uvmcopy，为子进程分配物理内存，然后拷贝到子进程的空间中。

```
// Copy user memory from parent to child.
if(uvmcopy(p->pagetable, np->pagetable, p->sz) < 0){
  freeproc(np);
  release(&np->lock);
  return -1;
}
np->sz = p->sz;
```

对当前进程的所有内存进行拷贝。

- **问题**：某些情况下对整份内存的拷贝是非必须的
  1. 若父进程较大，则复制时间会很长；
  2. 子进程执行exec()后复制内存会被丢弃；
  3. 但父子进程均可能进行内存写入，需要创建内存副本。

⇨ 优化策略 **Copy-on-write**

Copy-on-write

- 流程：
  1. Cow fork()时，为子进程只创建一个页表，页表中的PTE条目指向父进程的实际内存页；
  2. Cow fork() 将父进程和子进程的所有条目全部设置为 read-only；
  3. 当父进程或子进程中任意一个进程试图对页面执行写操作时，会触发缺页中断；
  4. 为故障进程分配一页物理内存，将原始页面复制到新页面，并修改故障进程中的相关 PTE 以引用新页面，这次的 PTE 标记为可写。

实验配置

- 环境配置
  - commit lab4的代码

    git commit -am lab4
  - 切换到traps分支

    git checkout cow

    make clean
- 实验目标
  - 实现 copy-on write
  - 如果成功通过 cowtest和 usertests则实现成功

实验结果示例

- **cowtest**

```
xv6 kernel is booting

hart 1 starting
hart 2 starting
init: starting sh
$ cowtest
simple: ok
simple: ok
three: ok
three: ok
three: ok
file: ok
ALL COW TESTS PASSED
```

- **usertests**

```
$ usertests
usertests starting
test MAXVAplus: OK
test manywrites: OK
test execout: OK
test copyin: OK
test copyout: OK
test copyinstr1: OK
test copyinstr2: OK
test copyinstr3: OK
test rwsbrk: OK
test truncate1: OK
test truncate2: OK
```

⋮

```
test mem: OK
test pipe1: OK
test killstatus: OK
test preempt: kill... wait... OK
test exitwait: OK
test rmdot: OK
test fourteen: OK
test bigfile: OK
test dirfile: OK
test iref: OK
test forktest: OK
test bigdir: OK
ALL TESTS PASSED
```

实验步骤

1. 修改 uvmcopy()：
   - 将父节点的物理页映射到子节点，而非分配新页。
   - 清除子进程和父进程 PTE 中的 PTE_W。
2. 修改 usertrap()：
   - 识别页面故障。当 COW 页面发生页面故障时，使用 kalloc() 分配一个新页面，将旧页面复制到新页面，并将新页面安装到设置了 PTE_W 的 PTE 中。
3. 引用计数：
   - 确保每个物理页在最后一次 PTE 引用消失时被释放，而不是在此之前。可以为每个物理页保留一个 「引用计数」（引用该页的用户页表的数量）。
   - 在 kalloc() 分配页面时，将页面的引用计数设为 1。只有当页面的引用计数为零时，kfree() 才会将该页面放回空闲列表。
4. 修改 copyout()：
   - 以在遇到 COW 页面时，使用与用户态 page fault相同的方案

Tips

1. 可以使用 RISC-V PTE 中的 RSW（为软件保留）位记录每个 PTE 是否为 COW 映射，进而处理 COW page fault。
2. 在 kernel/riscv.h 的末尾有一些有用的宏和页表标志定义。
3. 如果发生 COW 页故障，且没有可用内存，则应杀死进程。

实验提交

- 提交到邮箱**fduos2023lab25@163.com**：
  - 命名为：学号-姓名-授课教师-lab5.pdf，报告内容包含：
    1. 概述题标清练习题编号；
    2. 实验题标清练习题编号，说明实验思路，实验过程（必须包括但不限于实验代码、系统调用流程等）实验效果截图；
    3. 实验过程中遇到的问题及解决方案；
- 截止日期：2023年12月4日23时
- 注意：请各位同学独立完成实验，参考代码需注明