西 安 邮 电 大 学

（计算机学院）

课内实验报告

实验名称：基于 mmap()虚存区的性能有优化

专业名称：计算机科学与技术

班　　级：计科 1801

学生姓名：刘泽钦

学号（8 位）：05189041

指导教师：陈莉君

实验日期：2020 年 5 月 6 日

一. 实验目的及实验环境

1.实验目的

通过对 mmap()系统调用的使用，深入理 mmap 在进程虚拟地址空间做了什么，当缺页中断发生时在物理空间做了什么，当物理内存不够时会发生什么。通过在 proc 目录下查看进程的虚存区，了解进程的虚存区到底是什么？

2.实验环境

（1）硬件

- CPU： Inter® Core™ i7-8550 CPU @1.80GHz 1.99GHz
- 内存： 8.00G
- 显示器： Honor MagicBook
- 硬盘空间： 256G

（2）软件

- 虚拟机名称及版本： Hower Deepin : deepin-15.11-amd64
- 操作系统名称及版本： Ubuntu
- 编译器： gcc

二. 实验内容

1、实验前准备工作

在实验楼阅读《实验 17　基于 mmap()虚存区的性能优化》的资料，了解 mmap()的使用方法，并掌握 mmap 虚存区映射的原理，能够从理论上搞明白 mmap 映射过程与文件系统 read()和 write()的差异，并以此为入口，深入理解虚拟内存管理的请页机制，地址换换地址，页面置换等等。

2、实验要求：

参看[linux 内存映射 mmap 原理分析]
(https://mp.weixin.qq.com/s/o8gjzQzfX032WV1OYfNeww)中的代码，对代码进行改造：

1) 把调用 mmap()和调用 read(),write()拆分成两个程序，形成两个进程 PA 和 PB。

2)输入不同大小的数据进行测试，并比较其时间差异。（建议通过命令行参数带入不同数据的大小，如果觉得有难度，就在程序中修改大小，数据的大小可以为 1b, 100b, 512b, 1kb, 2kb, 4kb, 8kb, 10kb, 1Mb, 1Gb 等）

每次写入大小 | mmap 耗时 | write 耗时
每次写入大小 | mmap 耗时 | read 耗时

3)让两个进程在后台运行，在 proc 目录下相应的进程号下查看其 maps 文件

（也可以通过 objdump 命令查看），例如：

cat /proc/3789/maps

并比较这两个进程的虚存区有何不同？

4) 比较把数据写到缓存中和不写到缓存性能有何差异？参考[Linux 性能优化实战（倪朋飞）---内存]
(https://blog.csdn.net/u012319493/article/details/89856186)

5) 从虚拟内存管理的原理上分析不同数据大小对 mmap()和 write()、read()的差异。

6) 给出对不同数据大小的优化建议，什么时候用 mmap(),什么时候用 write()、read()

7) 选做（挑战）

参考 [深入剖析 mmap 原理 - 从三个关键问题说起]
(https://www.jianshu.com/p/eece39beee20)

场景 A：物理内存+swap space: 16G，映射文件 30G，使用一个进程进行 mmap，成功后映射后持续写入数据

场景 B：物理内存+swap space: 16G，映射文件 15G，使用两个进程进行 mmap，成功后映射后持续写入数据

写代码进行测试，看看发生什么？

3、提问并回答

在云班课的讨论区提出至少两个问题，并给予回答，或同组内，两个同学为一组，一个提问，一个回答。

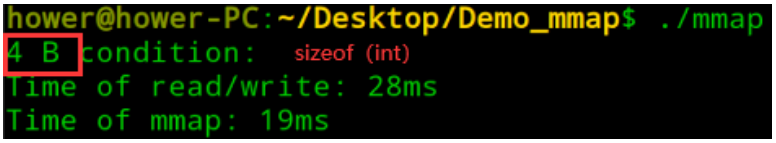
三．方案设计

给出完成上述要求的方案设计

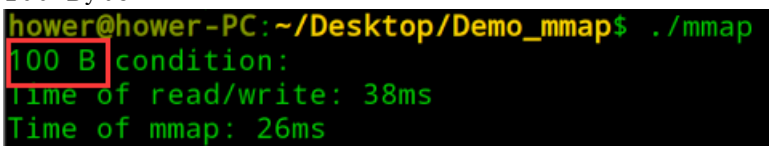
四．测试数据、运行结果以及调试过程截图和说明

拆分程序前进行时间性能对比：

1 个 int 型（4 Byte）



100 Byte



512 Byte

```
hower@hower-PC:~/Desktop/Demo_mmap$ ./mmap
512 B condition:
Time of read/write: 39ms
Time of mmap: 25ms
```

256 个 int 型（1 KB）

```
hower@hower-PC:~/Desktop/Demo_mmap$ ./mmap
1024 B condition:       1K
Time of read/write: 30ms
Time of mmap: 20ms
```

512 个 int 型（2 KB）

```
hower@hower-PC:~/Desktop/Demo_mmap$ ./mmap
2048 B condition:  2KB
Time of read/write: 38ms
Time of mmap: 27ms
```

1024 个 int 型（4 KB）

```
hower@hower-PC:~/Desktop/Demo_mmap$ ./mmap
4096 B condition:  4KB
Time of read/write: 1616ms
Time of mmap: 73ms
```

2048 个 int 型（8 KB）

```
hower@hower-PC:~/Desktop/Demo_mmap$ ./mmap
8192 B condition:  8KB
Time of read/write: 53ms
Time of mmap: 36ms
```

2560 个 int 型（10 KB）

```
hower@hower-PC:~/Desktop/Demo_mmap$ ./mmap
10240 B condition:   10KB
Time of read/write: 60ms
Time of mmap: 58ms
```

262144 个 int 型（1 MB）

```
hower@hower-PC:~/Desktop/Demo_mmap$ ./mmap
1048576 B condition:       1M
Time of read/write: 5008ms
Time of mmap: 1749ms
```

268435456 个 int 型（1 GB）

```
hower@hower-PC:~/Desktop/Demo_mmap$ ./mmap
1073741824 B condition: 1G
Time of read/write: -298862ms
Time of mmap: 162978ms
```

read 和 write 的时间已经超过了 32 位补码表示的最大值，发生了溢出，足可见 read 和 write 的时间性能在大容量操作情况下的低下

拆分后观察进程的 maps

mmap：

```
hower@hower-PC:~/Desktop/Demo_mmap$ ./mmap
1073741824 B condition:
^Z
[1]+  已停止               ./mmap
hower@hower-PC:~/Desktop/Demo_mmap$ pidof mmap
7591
hower@hower-PC:~/Desktop/Demo_mmap$ cat /proc/7591/maps
55dab1f62000-55dab1f63000 r-xp 00000000 08:01 262347                     /home/hower/Desktop/Demo_mmap/mmap
55dab2162000-55dab2163000 r--p 00000000 08:01 262347                     /home/hower/Desktop/Demo_mmap/mmap
55dab2163000-55dab2164000 rw-p 00001000 08:01 262347                     /home/hower/Desktop/Demo_mmap/mmap
55dab36d7000-55dab36f8000 rw-p 00000000 00:00 0                          [heap]
7fb13368a000-7fb17368a000 rw-s 00000000 08:01 262324                     /home/hower/Desktop/Demo_mmap/a.txt
7fb17368a000-7fb1b368b000 rw-p 00000000 00:00 0
7fb1b368b000-7fb1b3820000 r-xp 00000000 08:01 792481                     /usr/lib/x86_64-linux-gnu/libc-2.24.so
7fb1b3820000-7fb1b3a20000 ---p 00195000 08:01 792481                     /usr/lib/x86_64-linux-gnu/libc-2.24.so
7fb1b3a20000-7fb1b3a24000 r--p 00195000 08:01 792481                     /usr/lib/x86_64-linux-gnu/libc-2.24.so
7fb1b3a24000-7fb1b3a26000 rw-p 00199000 08:01 792481                     /usr/lib/x86_64-linux-gnu/libc-2.24.so
7fb1b3a26000-7fb1b3a2a000 rw-p 00000000 00:00 0
7fb1b3a2a000-7fb1b3a4d000 r-xp 00000000 08:01 792029                     /usr/lib/x86_64-linux-gnu/ld-2.24.so
7fb1b3c2c000-7fb1b3c2e000 rw-p 00000000 00:00 0
7fb1b3c4a000-7fb1b3c4d000 rw-p 00000000 00:00 0
7fb1b3c4d000-7fb1b3c4e000 r--p 00023000 08:01 792029                     /usr/lib/x86_64-linux-gnu/ld-2.24.so
7fb1b3c4e000-7fb1b3c4f000 rw-p 00024000 08:01 792029                     /usr/lib/x86_64-linux-gnu/ld-2.24.so
7fb1b3c4f000-7fb1b3c50000 rw-p 00000000 00:00 0
7ffe7a361000-7ffe7a383000 rw-p 00000000 00:00 0                          [stack]
7ffe7a3bb000-7ffe7a3be000 r--p 00000000 00:00 0                          [vvar]
7ffe7a3be000-7ffe7a3c0000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0                  [vsyscall]
```

read&write：

```
hower@hower-PC:~/Desktop/Demo_mmap$ ./rewr
1073741824 B condition:
^Z

[2]+  已停止               ./rewr
hower@hower-PC:~/Desktop/Demo_mmap$
hower@hower-PC:~/Desktop/Demo_mmap$ pidof rewr
7659
hower@hower-PC:~/Desktop/Demo_mmap$ cat /proc/7659/maps
5633ced25000-5633ced26000 r-xp 00000000 08:01 262348                     /home/hower/Desktop/Demo_mmap/rewr
5633cef25000-5633cef26000 r--p 00000000 08:01 262348                     /home/hower/Desktop/Demo_mmap/rewr
5633cef26000-5633cef27000 rw-p 00001000 08:01 262348                     /home/hower/Desktop/Demo_mmap/rewr
5633d0f0b000-5633d0f2c000 rw-p 00000000 00:00 0                          [heap]
7f9943cd1000-7f9983cd2000 rw-p 00000000 00:00 0
7f9983cd2000-7f9983e67000 r-xp 00000000 08:01 792481                     /usr/lib/x86_64-linux-gnu/libc-2.24.so
7f9983e67000-7f9984067000 ---p 00195000 08:01 792481                     /usr/lib/x86_64-linux-gnu/libc-2.24.so
7f9984067000-7f998406b000 r--p 00195000 08:01 792481                     /usr/lib/x86_64-linux-gnu/libc-2.24.so
7f998406b000-7f998406d000 rw-p 00199000 08:01 792481                     /usr/lib/x86_64-linux-gnu/libc-2.24.so
7f998406d000-7f9984071000 rw-p 00000000 00:00 0
7f9984071000-7f9984094000 r-xp 00000000 08:01 792029                     /usr/lib/x86_64-linux-gnu/ld-2.24.so
7f9984273000-7f9984275000 rw-p 00000000 00:00 0
7f9984291000-7f9984294000 rw-p 00000000 00:00 0
7f9984294000-7f9984295000 r--p 00023000 08:01 792029                     /usr/lib/x86_64-linux-gnu/ld-2.24.so
7f9984295000-7f9984296000 rw-p 00024000 08:01 792029                     /usr/lib/x86_64-linux-gnu/ld-2.24.so
7f9984296000-7f9984297000 rw-p 00000000 00:00 0
7fff779d4000-7fff779f6000 rw-p 00000000 00:00 0                          [stack]
7fff779f7000-7fff779fa000 r--p 00000000 00:00 0                          [vvar]
7fff779fa000-7fff779fc000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0                  [vsyscall]
```

区别：mmap 进程的 maps 中有对文件 a.txt 的映射，而 read&write 进程中没有，这说明了 mmap 的机理是写时复制，而 read&write 是确切操作到文件的。

数据写入缓存和不写入缓存的区别：原理上，数据写入缓存就让操作结果在处理器与缓存上传输，不写入缓存就是让操作结果直接在内存与处理器之间传输，效率相对较低，而数据先写入缓存可以却只在必要时候才与内存进行依次数据传输，所以缓存是利用局部性原理以及更多的操作高性能硬件的优势节约时间的。

五．总结

1. 实验过程中遇到的问题及解决办法；

   遇到的问题：拆分原本代码时候出现段错误

   解决办法：段错误实际上是使用了操作系统未分配的虚拟地址造成的，代码上的错误就是代码中 free 的指针变量没有放对位置，free 是与 malloc 相匹配的，而 mmap 申请的虚拟映射空间需要用 munmap 释放，因此将 free 代码提前即可。

2. 对设计及调试过程的心得体会。

   本次实验让我了解到 mmap 采用的是一种写时复制技术，其时间效率高

的原因在于 mmap 并不是每一次都真正操作磁盘，节省的时间是 CPU 与磁盘进行数据交换的时间。

六．附录：源代码

mmap:

```
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<sys/time.h>
#include<fcntl.h>
#include<sys/mman.h>

#define MAX 268435456

int main()
{
int i=0;
int count=0, fd=0;
struct timeval tv1, tv2;
int *array;// = (int *)malloc( sizeof(int)*MAX );
printf("%d B condition:\n",MAX*4);

/*mmap*/

array=(int *)malloc(sizeof(int)*MAX);
gettimeofday( &tv1, NULL );
fd = open( "a.txt", O_RDWR );
array = mmap( NULL, sizeof(int)*MAX, PROT_READ|PROT_WRITE, MAP_SHARED, fd, 0 );
for( i=0; i<MAX; ++i )
      ++array[ i ];
munmap( array, sizeof(int)*MAX );
pause();
msync( array, sizeof(int)*MAX, MS_SYNC );
close( fd );
gettimeofday( &tv2, NULL );
```

```
printf( "Time of mmap: %dms\n", tv2.tv_usec-tv1.tv_usec );
return 0;
}
```

read&write:

```
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<sys/time.h>
#include<fcntl.h>
#include<sys/mman.h>

#define MAX 268435456

int main()
{
int i=0;
int count=0, fd=0;
struct timeval tv1, tv2;
int *array = (int *)malloc( sizeof(int)*MAX );
printf("%d B condition:\n",MAX*4);

/*read*/

gettimeofday( &tv1, NULL );
fd = open( "a.txt", O_RDWR );
if( sizeof(int)*MAX != read( fd, (void *)array, sizeof(int)*MAX ) )
{
        printf( "Reading data failed...\n" );
        return -1;
}
for( i=0; i<MAX; ++i )
       ++array[ i ];
if( sizeof(int)*MAX != write( fd, (void *)array, sizeof(int)*MAX ) )
```

```
{
        printf( "Writing data failed...\n" );
        return -1;
}
pause();
free( array );
close( fd );
gettimeofday( &tv2, NULL );
printf( "Time of read/write: %dms\n", tv2.tv_usec-tv1.tv_usec );

return 0;
}
```