

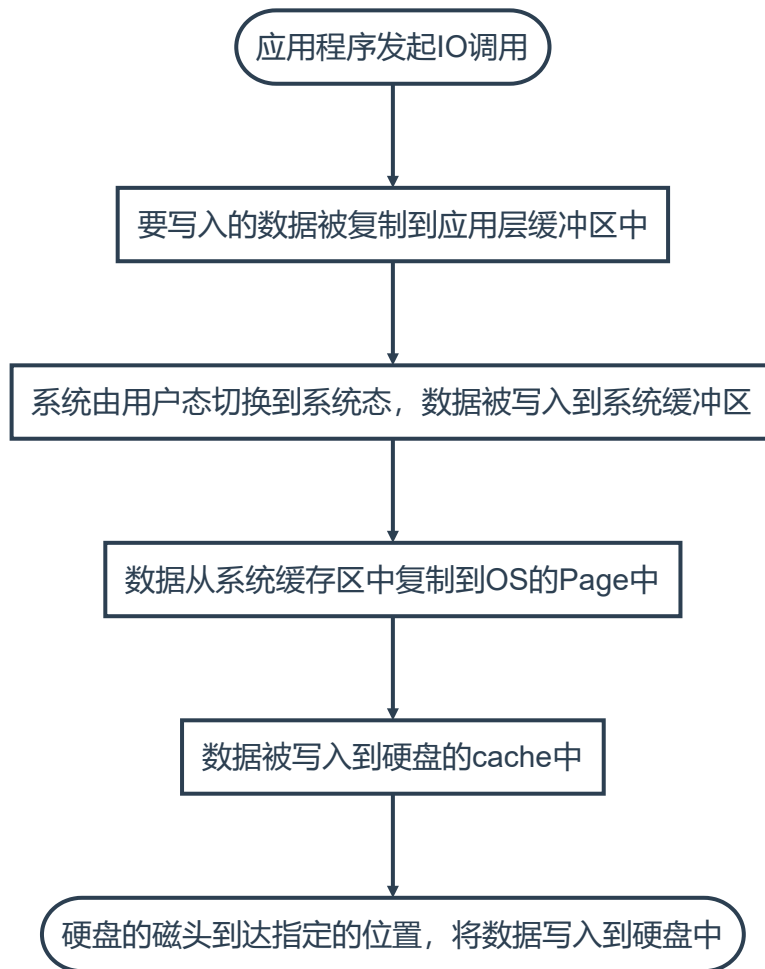
论文阅读与前期工作总结

姓名：郭佳俊，郭章旭，海明皓，黄一飞

**学号：17343034，17343036，17343037，
17343049**

前期工作

使用示意图展示普通文件IO方式(fwrite等)的流程，即进程与系统内核，磁盘之间的数据交换如何进行？为什么写入完成后要调用fsync？



调用 `fsync` 可以保证文件的修改时间也被更新。`fsync`系统调用可以精确的强制每次写入都被更新到磁盘中，以确保数据存入磁盘。

简述文件映射的方式如何操作文件。与普通IO区别？为什么写入完成后要调用`msync`？文件内容什么时候被载入内存？

(使用什么函数，函数的工作流程)

文件映射操作文件：

首先要通过`CreateFile()`函数来创建或打开一个文件内核对象，这个对象标识了磁盘上将要用作内存映射文件的文件。在用`CreateFile()`将文件映像位置通告给操作系统后，只指定了映像文件的路径，映像的长度还没有指定。为了指定文件映射对象需要多大的物理存储空间还需要通过`CreateFileMapping()`函数来创建一个文件映射内核对象以告诉系统文件的尺寸以及访问文件的方式。在创建了文件映射对象后，还必须为文件数据保留一个地址空间区域，并把文件数据作为映射到该区域的物理存储器进行提交。由`MapViewOfFile()`函数负责通过系统的管理而将文件映射对象的全部或部分映射到进程地址空间。此时，对内存映射文件的使用和处理同通常加载到内存中的文件数据的处理方式基本一样，在完成了对内存映射文件的使用时，还要通过一系列的操作完成对其的清除和使用过资源的释放。这部分相对比较简单，可以通过

`UnmapViewOfFile()`完成从进程的地址空间撤消文件数据的映像、通过`CloseHandle()`关闭前面创建的文件映射对象和文件对象。

与普通IO的区别：

最大的区别在于把文件内容映射成内存块，不需要分配内存去容纳你要读的东西，避免一次又一次的read。文件映射是在需要的时候才把文件读入内存，而普通IO是直接把整个文件读入内存。

为什么写入完成后调用msync：

进程在映射空间的对共享内容的改变并不直接写回到磁盘文件中，往往在调用munmap()后才执行该操作。可以通过调用msync()函数来实现磁盘文件内容与共享内存区中的内容一致,即同步操作。

文件内容什么时候被载入内存：

文件映射只有在需要的时候才把文件读入内存。

参考Intel的NVM模拟教程模拟NVM环境，用fio等工具测试模拟NVM的性能并与磁盘对比（关键步骤结果截图）。

（推荐Ubuntu 18.04LTS下配置，跳过内核配置，编译和安装步骤）

图1为配置成功后的显示：

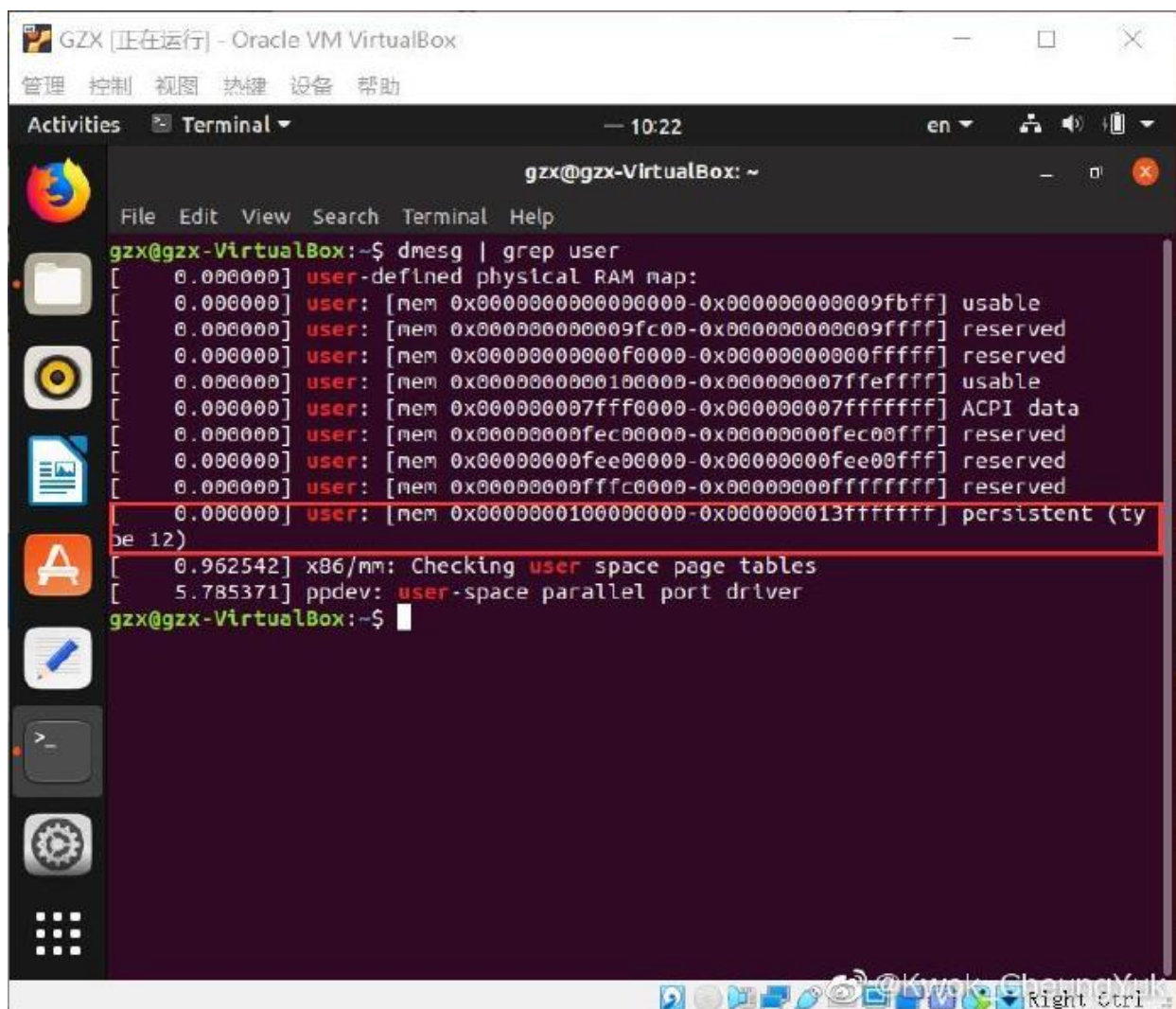


图1：持久性内存区突出显示为 (type 12)。

图2为持久化内存块的显示：

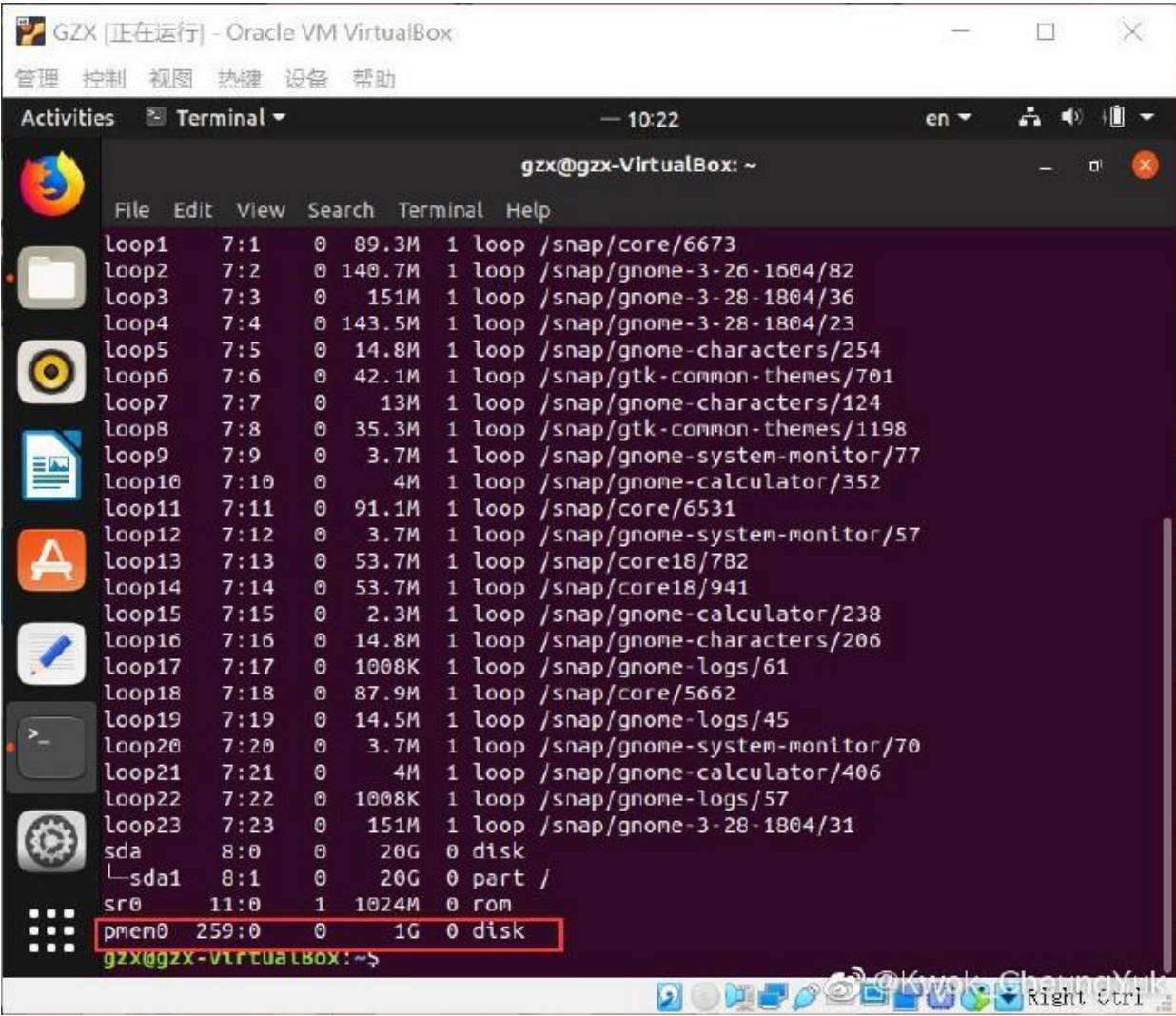


图2：持久性内存块。

图3、4、5分别为使用fio工具测试NVM性能

随机读指令：

```
fio -filename=/mnt/pmemdir/test_randread -direct=1 -iodepth 1 -thread -rw=randread -
ioengine=psync -bs=16k -size=512MB -numjobs=10 -runtime=60 -group_reporting -name=mytest
```



```

root@gzx-VirtualBox: /home/gzx

File Edit View Search Terminal Help

mytest: (groupid=0, jobs=10): err= 0: pid=2471: Mon Apr 22 13:05:06 2019
read: IOPS=239k, BW=3732MiB/s (3913MB/s)(5120MiB/1372msec)
  clat (nsec): min=1883, max=105002k, avg=35959.93, stdev=1053435.88
  lat (nsec): min=1906, max=105002k, avg=36116.35, stdev=1054905.58
  clat percentiles (usec):
    | 1.00th=[ 3], 5.00th=[ 3], 10.00th=[ 3], 20.00th=[ 3],
    | 30.00th=[ 3], 40.00th=[ 3], 50.00th=[ 3], 60.00th=[ 3],
    | 70.00th=[ 3], 80.00th=[ 3], 90.00th=[ 4], 95.00th=[ 6],
    | 99.00th=[ 26], 99.50th=[ 38], 99.90th=[11338], 99.95th=[32375],
    | 99.99th=[43779]
  bw ( KiB/s): min=331465, max=455121, per=10.06%, avg=384522.90, stdev=32672.
65, samples=20
  iops       : min=20716, max=28445, avg=24032.50, stdev=2042.21, samples=20
  lat (usec)  : 2=0.03%, 4=93.58%, 10=2.47%, 20=1.93%, 50=1.05%
  lat (usec)  : 100=0.15%, 250=0.05%, 500=0.01%, 750=0.01%, 1000=0.01%
  lat (msec)  : 2=0.01%, 4=0.01%, 10=0.01%, 20=0.02%, 50=0.08%
  lat (msec)  : 100=0.01%, 250=0.01%
  cpu         : usr=1.86%, sys=7.98%, ctx=502, majf=0, minf=36
  IO depths   : 1=100.0%, 2=0.0%, 4=0.0%, 8=0.0%, 16=0.0%, 32=0.0%, >=64=0.0%
    submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
    complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
    issued rwts: total=327680,0,0,0 short=0,0,0,0 dropped=0,0,0,0
    latency   : target=0, window=0, percentile=100.00%, depth=1

```

图3: NVM随机读测试结果

混合随机写指令:

fio -filename=/mnt/pmemdir/test_randread -direct=1 -iodepth 1 -thread -rw=randwrite -
ioengine=psync -bs=16k -size=512MB -numjobs=10 -runtime=60 -group_reporting -name=mytest

```

root@gzx-VirtualBox: /home/gzx

File Edit View Search Terminal Help

00m:00s]
mytest: (groupid=0, jobs=10): err= 0: pid=2437: Mon Apr 22 13:03:47 2019
write: IOPS=164k, BW=2560MiB/s (2684MB/s)(5120MiB/2000msec)
  clat (usec): min=2, max=47552, avg=56.56, stdev=1027.09
  lat (usec): min=2, max=47552, avg=56.86, stdev=1032.32
  clat percentiles (usec):
    | 1.00th=[ 3], 5.00th=[ 3], 10.00th=[ 3], 20.00th=[ 3],
    | 30.00th=[ 3], 40.00th=[ 3], 50.00th=[ 4], 60.00th=[ 4],
    | 70.00th=[ 4], 80.00th=[ 4], 90.00th=[ 8], 95.00th=[ 11],
    | 99.00th=[ 36], 99.50th=[ 93], 99.90th=[20579], 99.95th=[24511],
    | 99.99th=[30540]
  bw ( KiB/s): min=104768, max=358861, per=9.23%, avg=242059.90, stdev=85735.1
7, samples=30
  iops       : min= 6548, max=22428, avg=15128.40, stdev=5358.19, samples=30
  lat (usec)  : 4=81.05%, 10=13.55%, 20=2.50%, 50=2.20%, 100=0.21%
  lat (usec)  : 250=0.07%, 500=0.06%, 750=0.01%, 1000=0.01%
  lat (msec)  : 2=0.01%, 4=0.03%, 10=0.09%, 20=0.10%, 50=0.12%
  cpu         : usr=1.53%, sys=8.33%, ctx=2088, majf=0, minf=0
  IO depths   : 1=100.0%, 2=0.0%, 4=0.0%, 8=0.0%, 16=0.0%, 32=0.0%, >=64=0.0%
    submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
    complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
    issued rwts: total=0,327680,0,0 short=0,0,0,0 dropped=0,0,0,0
    latency   : target=0, window=0, percentile=100.00%, depth=1

```

图4: NVM随机写测试结果

混合随机读写指令：

```
fio -filename=/mnt/pmemdir/test_randread -direct=1 -iodepth 1 -thread -rw=randrw -
ioengine=psync -bs=16k -size=512MB -numjobs=10 -runtime=60 -group_reporting -name=mytest
```

```
root@gzx-VirtualBox: /home/gzx
File Edit View Search Terminal Help

lat (nsec): min=1728, max=85788k, avg=62391.92, stdev=320843.53
clat percentiles (usec):
| 1.00th=[ 3], 5.00th=[ 18], 10.00th=[ 33], 20.00th=[ 41],
| 30.00th=[ 45], 40.00th=[ 49], 50.00th=[ 53], 60.00th=[ 58],
| 70.00th=[ 63], 80.00th=[ 72], 90.00th=[ 91], 95.00th=[ 119],
| 99.00th=[ 178], 99.50th=[ 200], 99.90th=[ 347], 99.95th=[ 2147],
| 99.99th=[ 7046]
bw ( KiB/s): min=120750, max=169568, per=9.98%, avg=158881.63, stdev=9935.75
, samples=30
iops : min= 7546, max=10598, avg=9929.97, stdev=621.18, samples=30
write: IOPS=99.6k, BW=1556MiB/s (1631MB/s)(2561MiB/1646msec)
clat (usec): min=2, max=70054, avg=35.30, stdev=218.90
lat (usec): min=2, max=70054, avg=35.42, stdev=218.90
clat percentiles (usec):
| 1.00th=[ 3], 5.00th=[ 3], 10.00th=[ 3], 20.00th=[ 3],
| 30.00th=[ 3], 40.00th=[ 4], 50.00th=[ 16], 60.00th=[ 47],
| 70.00th=[ 53], 80.00th=[ 60], 90.00th=[ 73], 95.00th=[ 93],
| 99.00th=[ 157], 99.50th=[ 180], 99.90th=[ 247], 99.95th=[ 429],
| 99.99th=[ 6915]
bw ( KiB/s): min=119089, max=170944, per=9.97%, avg=158906.67, stdev=9677.14
, samples=30
iops : min= 7443, max=10684, avg=9931.57, stdev=604.84, samples=30
lat (usec) : 2=0.01%, 4=26.52%, 10=0.44%, 20=1.00%, 50=26.90%
lat (usec) : 100=39.07%, 250=5.92%, 500=0.07%, 750=0.01%, 1000=0.01%
```

图5：NVM混合随机读写结果

图6、7、8分别为使用fio工具测试普通磁盘性能：

随机读指令：

```
fio -filename=/tmp/test_randread -direct=1 -iodepth 1 -thread -rw=randread -ioengine=psync -
bs=16k -size=512MB -numjobs=10 -runtime=60 -group_reporting -name=mytest
```



```

root@gzx-VirtualBox: /home/gzx
File Edit View Search Terminal Help
mytest: (groupid=0, jobs=10): err= 0: pid=2526: Mon Apr 22 13:11:00 2019
read: IOPS=172, BW=2756KiB/s (2822kB/s)(162MiB/60065msec)
clat (usec): min=100, max=1222.0k, avg=58012.49, stdev=68331.63
lat (usec): min=100, max=1222.0k, avg=58012.80, stdev=68331.63
clat percentiles (msec):
| 1.00th=[ 4], 5.00th=[ 13], 10.00th=[ 16], 20.00th=[ 21],
| 30.00th=[ 26], 40.00th=[ 32], 50.00th=[ 40], 60.00th=[ 50],
| 70.00th=[ 63], 80.00th=[ 83], 90.00th=[ 114], 95.00th=[ 153],
| 99.00th=[ 313], 99.50th=[ 518], 99.90th=[ 642], 99.95th=[ 1099],
| 99.99th=[ 1217]
bw ( KiB/s): min= 31, max= 832, per=10.15%, avg=279.68, stdev=99.99, samples=1181
iops      : min= 1, max= 52, avg=17.20, stdev= 6.25, samples=1181
lat (usec) : 250=0.80%, 500=0.04%
lat (msec) : 4=0.31%, 10=2.85%, 20=15.21%, 50=41.49%, 100=25.49%
lat (msec) : 250=12.32%, 500=0.99%, 750=0.41%, 1000=0.01%
cpu        : usr=0.06%, sys=0.03%, ctx=10352, majf=0, minf=36
IO depths  : 1=100.0%, 2=0.0%, 4=0.0%, 8=0.0%, 16=0.0%, 32=0.0%, >=64=0.0%
submit     : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
complete   : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
issued rwts: total=10347,0,0,0 short=0,0,0,0 dropped=0,0,0,0
latency    : target=0, window=0, percentile=100.00%, depth=1

Run status group 0 (all jobs):

```

图6: 磁盘随机读测试结果

随机写指令:

fio -filename=/tmp/test_randread -direct=1 -iodepth 1 -thread -rw=randwrite -ioengine=psync -bs=16k -size=512MB -numjobs=10 -runtime=60 -group_reporting -name=mytest

```

root@gzx-VirtualBox: /home/gzx
File Edit View Search Terminal Help
lat (usec): min=64, max=2067.0k, avg=41208.21, stdev=196132.23
clat percentiles (usec):
| 1.00th=[ 82], 5.00th=[ 90], 10.00th=[ 93],
| 20.00th=[ 101], 30.00th=[ 113], 40.00th=[ 133],
| 50.00th=[ 149], 60.00th=[ 165], 70.00th=[ 186],
| 80.00th=[ 219], 90.00th=[ 322], 95.00th=[ 183501],
| 99.00th=[1216349], 99.50th=[1350566], 99.90th=[1669333],
| 99.95th=[1870660], 99.99th=[2071987]
bw ( KiB/s): min= 31, max= 2528, per=17.35%, avg=673.51, stdev=398.07, samples=695
iops      : min= 1, max= 158, avg=41.94, stdev=24.91, samples=695
lat (usec) : 100=19.10%, 250=66.25%, 500=6.60%, 750=0.46%, 1000=0.25%
lat (msec) : 2=0.83%, 4=0.05%, 10=0.08%, 20=0.05%, 50=0.95%
lat (msec) : 100=0.18%, 250=0.51%, 500=0.89%, 750=1.64%, 1000=0.59%
cpu        : usr=0.01%, sys=0.13%, ctx=30147, majf=0, minf=0
IO depths  : 1=100.0%, 2=0.0%, 4=0.0%, 8=0.0%, 16=0.0%, 32=0.0%, >=64=0.0%
submit     : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
complete   : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
issued rwts: total=0,14645,0,0 short=0,0,0,0 dropped=0,0,0,0
latency    : target=0, window=0, percentile=100.00%, depth=1

Run status group 0 (all jobs):
WRITE: bw=3882KiB/s (3976kB/s), 3882KiB/s-3882KiB/s (3976kB/s-3976kB/s), io=22
9MiB (240MB), run=60354-60354msec

```

图7: 磁盘随机写测试结果

混合随机读写指令：

```
fio -filename=/tmp/test_randread -direct=1 -iodepth 1 -thread -rw=randrw -ioengine=psync -
bs=16k -size=512MB -numjobs=10 -runtime=60 -group_reporting -name=mytest
```

```
root@gzx-VirtualBox: /home/gzx
File Edit View Search Terminal Help
lat (usec): min=401, max=589249, avg=95655.91, stdev=87503.79
clat percentiles (msec):
| 1.00th=[ 17], 5.00th=[ 36], 10.00th=[ 45], 20.00th=[ 55],
| 30.00th=[ 63], 40.00th=[ 69], 50.00th=[ 77], 60.00th=[ 84],
| 70.00th=[ 93], 80.00th=[ 105], 90.00th=[ 128], 95.00th=[ 194],
| 99.00th=[ 510], 99.50th=[ 531], 99.90th=[ 575], 99.95th=[ 575],
| 99.99th=[ 592]
bw ( KiB/s): min= 31, max= 288, per=10.34%, avg=115.99, stdev=52.73, samples=1162
iops : min= 1, max= 18, avg= 7.01, stdev= 3.36, samples=1162
write: IOPS=72, BW=1156KiB/s (1183kB/s)(68.0MiB/60276msec)
clat (usec): min=76, max=578660, avg=45415.20, stdev=67816.95
lat (usec): min=76, max=578660, avg=45415.66, stdev=67817.01
clat percentiles (usec):
| 1.00th=[ 103], 5.00th=[ 127], 10.00th=[ 151], 20.00th=[ 169],
| 30.00th=[ 194], 40.00th=[ 237], 50.00th=[ 40633], 60.00th=[ 55837],
| 70.00th=[ 66847], 80.00th=[ 78119], 90.00th=[ 94897], 95.00th=[114820],
| 99.00th=[476054], 99.50th=[497026], 99.90th=[530580], 99.95th=[557843],
| 99.99th=[574620]
bw ( KiB/s): min= 31, max= 544, per=10.75%, avg=124.20, stdev=77.57, samples=1119
iops : min= 1, max= 34, avg= 7.53, stdev= 4.87, samples=1119
lat (usec) : 100=0.48%, 250=20.40%, 500=2.83%, 750=0.05%, 1000=0.01%
lat (msec) : 2=0.01%, 10=0.15%, 20=0.57%, 50=10.73%, 100=49.99%
```

图8：磁盘混合随机读写结果

通过上述图3、4、5与图6、7、8分析，无论是带宽 (bw)、每秒钟的IO数 (iops)，还是延迟 (lat)，仿真NVM的性能都要比普通磁盘的性能要好。

使用PMDK的libpmem库编写样例程序操作模拟NVM（关键实验结果截图，附上编译命令和简单样例程序）。

（样例程序使用教程的即可，主要工作是编译安装并链接PMDK库）

实验环境：Ubuntu 18.04.2

1.配置PMDK

Installing PMDK from Source:

Install Prerequisites:

```
$ sudo install autoconf automake pkg-config glib2 glib2-devel libfabric
libfabric-devel doxygen graphviz pandoc ncurses
```

Install NDCTL Packages from Source:

Install Prerequisites:

```
$ sudo apt-get install -y git gcc g++ autoconf automake asciidoc asciidoctor
bash-completion libtool pkg-config libglib2.0-0 libglib2.0-dev doxygen graphviz
```



```

pandoc libncurses5 libkmod2 libkmod-dev libudev-dev uuid-dev libjson-c-dev
libkeyutils-dev
$ cd /downloads
$ sudo git clone https://github.com/pmem/ndctl
$ cd ndctl
$ ./autogen.sh
$ ./configure CFLAGS='-g -O2' --prefix=/usr/local --sysconfdir=/etc --
libdir=/usr/local/lib64
$ make
$ sudo make install
$ cd /downloads
$ git clone https://github.com/pmem/pmdk
$ cd pmdk
$ make
$ sudo make install

```

2.运行例程

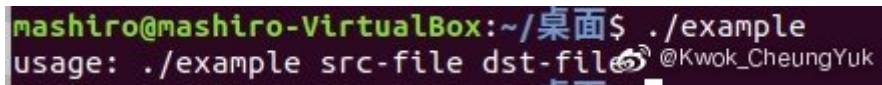
运行<https://github.com/pmem/pmdk/tree/master/src/examples/libpmem>中的fullcopy:

```

$ sudo ldconfig
$ gcc example.c -o example -lpmem
$ ./example

```

运行结果:



```

mashiro@mashiro-VirtualBox:~/桌面$ ./example
usage: ./example src-file dst-file

```

图9: fullcopy运行结果

源代码:

```

/*
 * Copyright 2014-2017, Intel Corporation
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * * Redistributions of source code must retain the above copy
 * right

```

* notice, this list of conditions and the following disclaimer.

*

* * Redistributions in binary form must reproduce the above copyright

* notice, this list of conditions and the following disclaimer in

* the documentation and/or other materials provided with the

* distribution.

*

* * Neither the name of the copyright holder nor the names of its

* contributors may be used to endorse or promote products derived

* from this software without specific prior written permission.

*

* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS

* "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT

* LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR

* A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT

* OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,

* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT

* LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,

* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY


```
    * THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR
    TORT

    * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF T
    HE USE

    * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DA
    MAGE.

    */

/*

* full_copy.c -- show how to use pmem_memcpy_nodrain()

*

* usage: full_copy src-file dst-file

*

* Copies src-file to dst-file in 4k chunks.

*/

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <stdio.h>

#include <errno.h>

#include <stdlib.h>

#ifdef _WIN32

#include <unistd.h>

#else

#include <io.h>

#endif

#endif
```

```
#include <string.h>

#include <libpmem.h>

/* copying 4k at a time to pmem for this example */

#define BUF_LEN 4096

/*
 * do_copy_to_pmem -- copy to pmem, postponing drain step until the end
 */

static void
do_copy_to_pmem(char *pmemaddr, int srcfd, off_t len)
{
    char buf[BUF_LEN];

    int cc;

    /* copy the file, saving the last flush step to the end */

    while ((cc = read(srcfd, buf, BUF_LEN)) > 0) {

        pmem_memcpy_nodrain(pmemaddr, buf, cc);

        pmemaddr += cc;

    }

    if (cc < 0) {

        perror("read");

        exit(1);

    }
}
```



```
/* perform final flush step */

pmem_drain();

}

/*

* do_copy_to_non_pmem -- copy to a non-pmem memory mapped file

*/

static void

do_copy_to_non_pmem(char *addr, int srcfd, off_t len)

{

    char *startaddr = addr;

    char buf[BUF_LEN];

    int cc;

    /* copy the file, saving the last flush step to the end */

    while ((cc = read(srcfd, buf, BUF_LEN)) > 0) {

        memcpy(addr, buf, cc);

        addr += cc;

    }

    if (cc < 0) {

        perror("read");

        exit(1);

    }

    /* flush it */
```

```
    if (pmem_msync(startaddr, len) < 0) {  
  
        perror("pmem_msync");  
  
        exit(1);  
  
    }  
  
}  
  
int  
  
main(int argc, char *argv[])  
  
{  
  
    int srcfd;  
  
    struct stat stbuf;  
  
    char *pmemaddr;  
  
    size_t mapped_len;  
  
    int is_pmem;  
  
  
    if (argc != 3) {  
  
        fprintf(stderr, "usage: %s src-file dst-file\n", argv[0]);  
  
        exit(1);  
  
    }  
  
  
  
    /* open src-file */  
  
    if ((srcfd = open(argv[1], O_RDONLY)) < 0) {  
  
        perror(argv[1]);  
  
        exit(1);  
  
    }  
  

```



```
/* find the size of the src-file */

if (fstat(srcfd, &stbuf) < 0) {

    perror("fstat");

    exit(1);

}


/* create a pmem file and memory map it */

if ((pmemaddr = pmem_map_file(argv[2], stbuf.st_size,

    PMEM_FILE_CREATE|PMEM_FILE_EXCL,

    0666, &mapped_len, &is_pmem)) == NULL) {

    perror("pmem_map_file");

    exit(1);

}


/* determine if range is true pmem, call appropriate copy routine */

if (is_pmem)

    do_copy_to_pmem(pmemaddr, srcfd, stbuf.st_size);

else

    do_copy_to_non_pmem(pmemaddr, srcfd, stbuf.st_size);


close(srcfd);

pmem_unmap(pmemaddr, mapped_len);

exit(0);

}
```

论文阅读

总结一下本文的主要贡献和观点(500字以内)(不能翻译摘要)。

(回答本文工作的动机背景是什么, 做了什么, 有什么技术原理, 解决了什么问题, 其意义是什么)

答: 在当今SCM结构大量被使用的现状下, 本文从B+树的持久性入手, 希望能够在开销最小的情况下将数据持久化, 然而这么做将会导致模型速度降低、延迟增高的问题。针对这个问题, 本文提出了其构建的新型树状结构——FPTree。其中主要的技术原理包括了指纹识别技术、采用选择性持久化的方式、采用选择性并发以及使用了可靠的编程模型。指纹识别技术是为了在查找没有被排序的叶节点时能够提供更优秀的性能和查找效率, 相比于WBTree和NVTree结构能提供更低的查找复杂度。选择性持久化方式是指将基本的数据存放在SCM里面, 而将非基本的数据存放在DRAM里面, 对应FPTree结构就是将叶节点存放在SCM里面而将非叶节点存放在DRAM里面, 这样做的意义是让整个查询过程只在叶节点花费较高代价, 以节约查询成本。选择性并发的意思是对短暂的与持久的两部分采用不同的并发策略。FPTree使用HTM来解决非叶节点的并发性而用锁的方式来解决叶节点的并发性, 选择性并发机制就能够解决HTM与SCM结构所需要的持久化基元不兼容的问题。在SCM编程中, 保持数据的一致性、局部写入、数据恢复以及内存泄露都是可能面临的挑战, 为了解决这些问题本文采用了一种可靠的编程模型, 其中包含了对持久化指针的使用、优化过的防止崩溃的安全体系、持续的分配器和内存泄露预防体系。

SCM硬件有什么特性? 与普通磁盘有什么区别? 普通数据库以页的粒度读写磁盘的方式适合操作SCM吗?

答: 相较于普通硬件, SCM硬件具有低延迟、高读写速度且能保持非易失性, 具有持久化和字节级访问的特性, 因此从理论上是有可能实现将永久的数据结构存放在SCM里面并且同时能够有接近DRAM级的性能。否, SCM和普通数据库有着不同的I/O机制, 所以不能直接用普通数据库的操作方式。

操作SCM为什么要调用CLFLUSH等指令?

(写入后不调用, 发生系统崩溃有什么后果)

答: 首先, 我们先解释一下这些指令的作用: CLFLUSH指令是清除缓冲中的缓存行, 并将它们的内容写入内存里面; 当一条MFENCE指令发出之后, 所有待定的载入和存储内存的操作会执行结束。调用这些指令的意义在于保持SCM里面数据的持久性。假如突然发生了系统崩溃, 所有在缓存里面的数据都会因为没有被写入内存而丢失, 这是我们所不希望看到的, 因此我们需要调用CLFLUSH指令将它们存在内存里面, 内存里面的数据是不会掉电丢失的, 所以就能够保持数据的稳定性。

FPTree的指纹技术有什么重要作用？

答：在SCM里面，没有被排序的叶节点需要一次线性的查找，而这样一次查找是相当花费时间的，所以会大大降低搜索的性能。在指纹技术里面，指纹代表着叶节点关键字里面的单字节的哈希值，当我们扫描指纹的时候，在平均情况下我们可以限制叶节点探测到的关键字的数量至1，从而显著的提升性能。故指纹技术就是为了提升查找叶节点的性能而发明的一项技术。

为了保证指纹技术的数学证明成立，哈希函数应如何选取？

（哈希函数生成的哈希值具有什么特征，能简单对键值取模生成吗？）

答：哈希函数生成的哈希值需要满足一致分布的特点。如果是简单的对键值取模，可能会让整个哈希值的分布很零散，容易出现数据倾斜的情况，不能够很好的满足一致分布的特性。所以为了保证指纹函数的数学证明成立，也就是为了保证公式中求解A事件的概率出现 $1/n$ 的近似算法，可以使用一致性哈希算法来进行实现。

持久化指针的作用是什么？与课上学到的什么类似？

答：当遇到错误或访问失败时，持久化指针仍然能够保持有效性，所以当程序重新启动的时候持久化指针会重新修复数据并且刷新那些不稳定的指针，让它们重新生效。与我们在课堂上学到的非易失性内存类似。