

# 工科创III-D Lab4

## 一：为host server准备巨页

要对于kernel启动的命令行进行设置

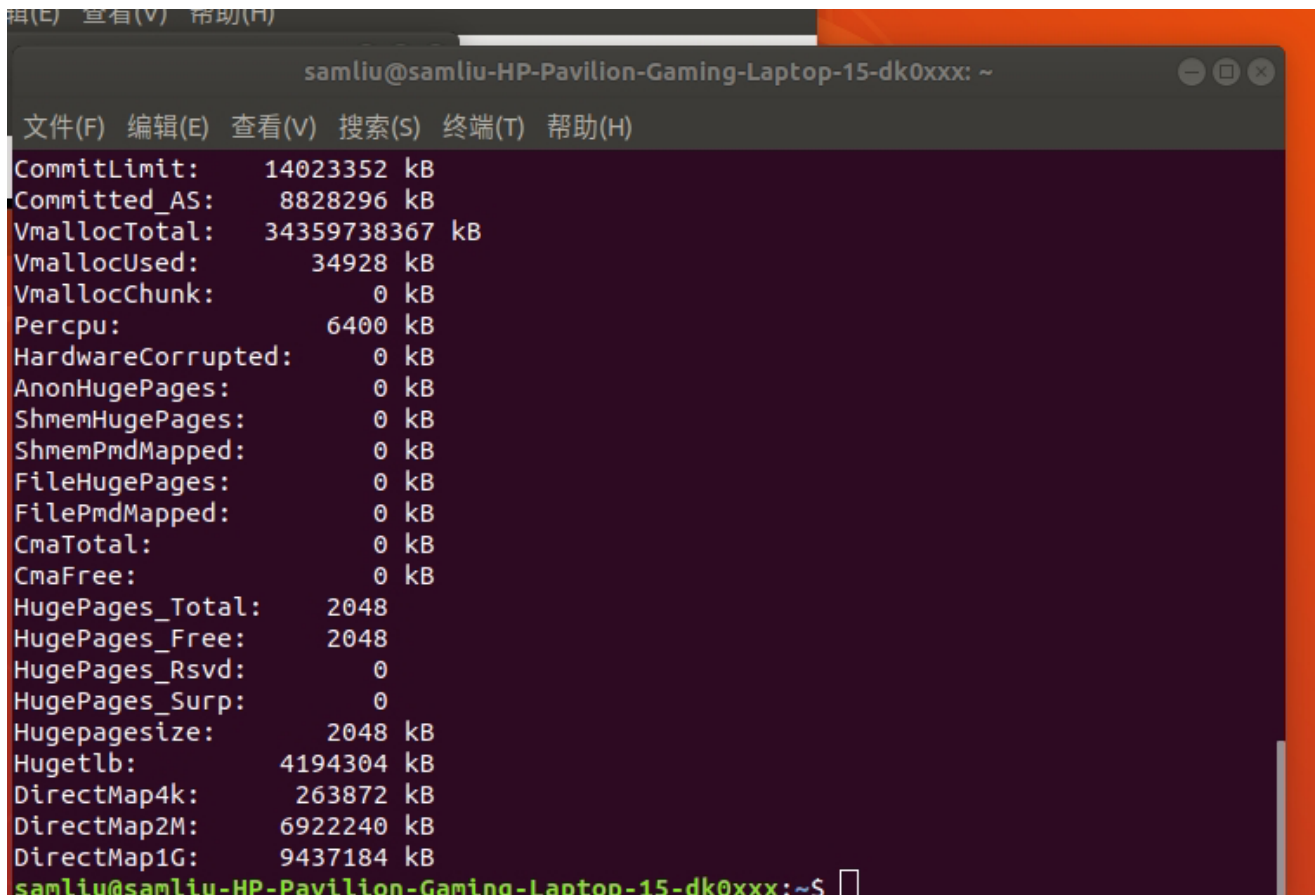
输入 `sudo vim /etc/default/grub`编辑相应文档，对于文件中的GRUB\_CMDLINE\_LINUX\_DEFAULT字段进行修改，新增配置。

```
GRUB_CMDLINE_LINUX_DEFAULT="...略... transparent_hugepage=never default_hugepagesz=2M
hugepagesz=2M hugepages=2048"
```

其中，`default_hugepagesz` 是指定系统默认HugePage大小，`hugepagesz` 为设置每个页的大小，根据实验要求本人设置为2M，`hugepages` 指定的是系统一共生成多少Hugepages，本人分配了2048个，相当于在host server中有4G的RAM是以巨页的形式存储。而`transparent_hugepage=never`也阻止了透明巨页的使用。

修改完后，输入`sudo update-grub`，再重启查询。

输入 `cat /proc/meminfo`查询RAM状态，发现巨页已经被设置好了。



```
编辑(E) 查看(V) 帮助(H)
samlu@samlu-HP-Pavilion-Gaming-Laptop-15-dk0xxx: ~
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
CommitLimit:      14023352 kB
Committed_AS:      8828296 kB
VmallocTotal:      34359738367 kB
VmallocUsed:        34928 kB
VmallocChunk:       0 kB
Percpu:            6400 kB
HardwareCorrupted:  0 kB
AnonHugePages:      0 kB
ShmemHugePages:     0 kB
ShmemPmdMapped:     0 kB
FileHugePages:      0 kB
FilePmdMapped:      0 kB
CmaTotal:           0 kB
CmaFree:            0 kB
HugePages_Total:    2048
HugePages_Free:     2048
HugePages_Rsvd:      0
HugePages_Surp:      0
Hugepagesize:       2048 kB
Hugetlb:            4194304 kB
DirectMap4k:        263872 kB
DirectMap2M:        6922240 kB
DirectMap1G:        9437184 kB
samlu@samlu-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~$
```

## 二：创建使用和没使用巨页的QEMU KVM虚拟机

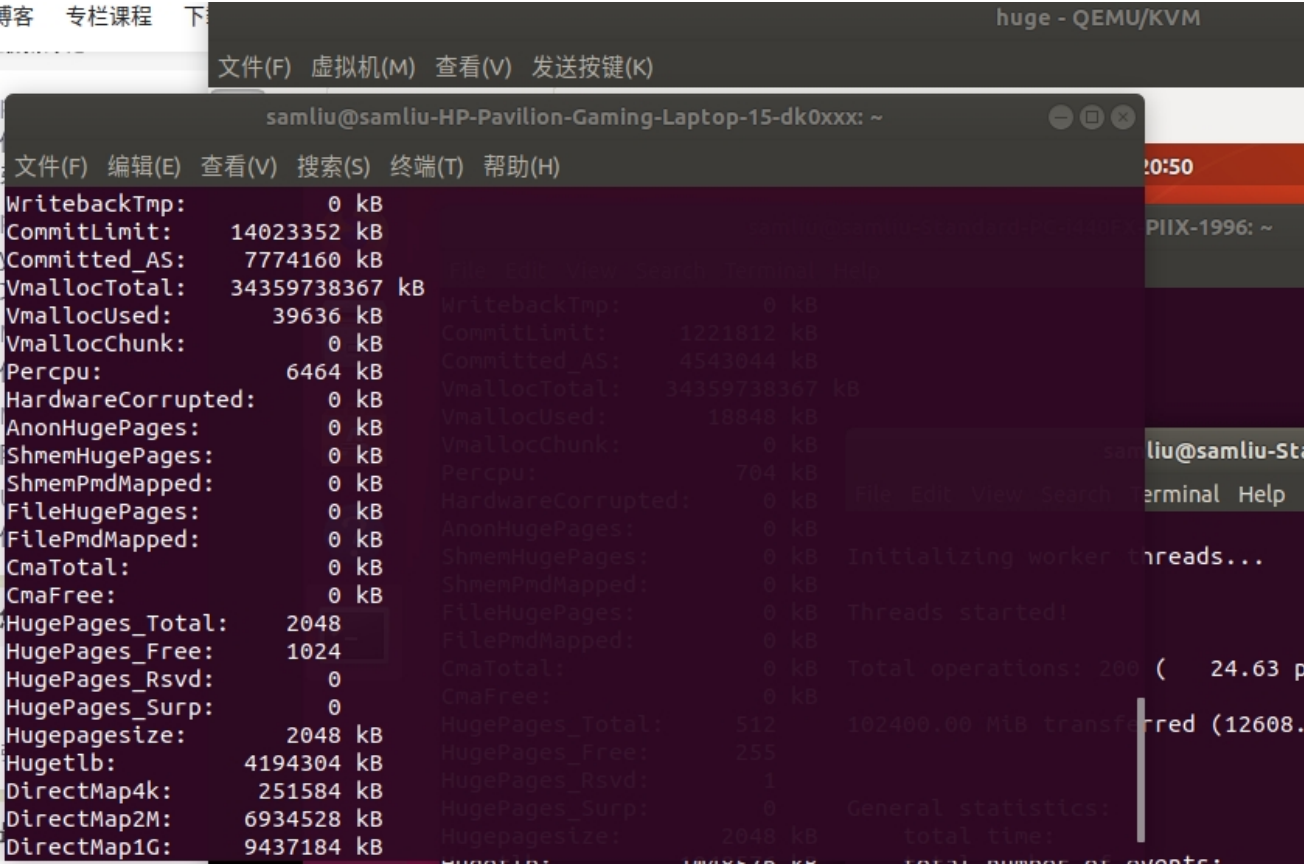
使用virt-manager图形化界面创建虚拟机，过程在之前的几次lab中都已经反复使用此处就不再赘述。

在创建完成后，在host server的终端中输入：virsh edit guestname指的是希望使用host server中巨页的虚拟机的名称，我设置为“huge”。此命令是为了修改虚拟机中的配置文件。

打开配置文件后，在合适的位置添加以下内容：

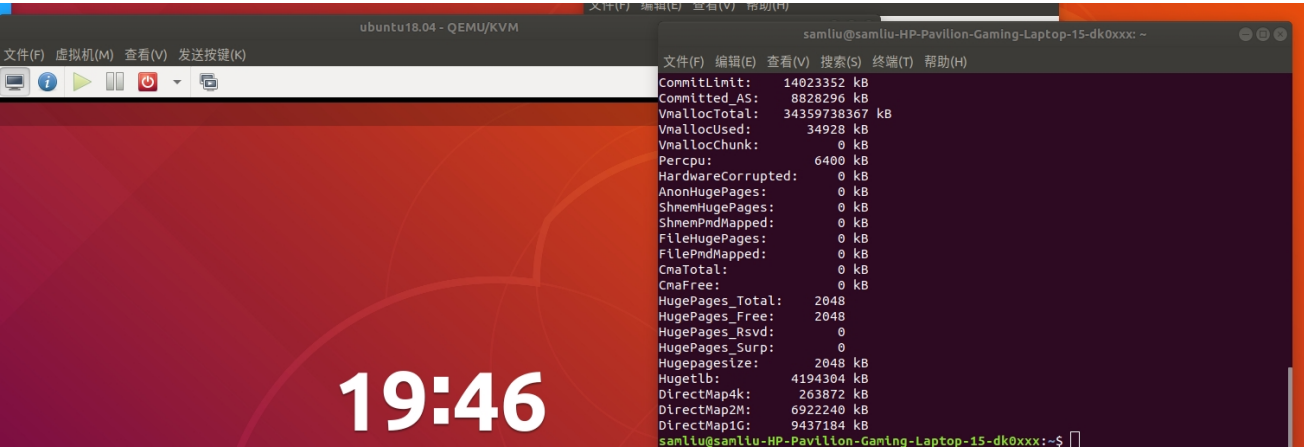
```
<memoryBacking>
  <hugepages/>
</memoryBacking>
```

保存后退出，打开虚拟机查看结果。



由于创建的虚拟机RAM内存为2G，因此在打开此虚拟机后，有1024个2M的巨页被分配给了“huge”虚拟机。

而没有被修改xml文件的虚拟机，在打开之后host server的巨页剩余数量仍然是2048，如下图所示：



因此我此时已经创建好了一个使用host server的巨页的虚拟机和一个没有使用巨页的虚拟机。

### 三：巨页原理分析

在理解巨页的使用之前，需要先理解页表是如何在内存访问中发挥作用的。

在操作系统中，默认一个页的大小为4K，以本机为例，如果要用4K的页来表示16G的内存大小，就需要很多页表项，那么就需要一个很大的页表来存储并映射到这些页表项上，但是小的页优势在于可以减少内部碎片。

为了减少访问页表的时间，在一级快速缓存区中会存储一个TLB页表，当希望访问的数据在TLB页表中时，操作系统可以通过访问一级快速缓存来获取这些信息而不需要访问页表，这样子可以缩短时间。但是一级快速缓存区的大小是有限的，TLB页表中往往只能存储少量的页表项，当访问内容不在TLB中时，便会发生TLB miss，此时操作系统就需要去访问整个页表来寻找所需要的那块内存内容。

对于4K的页而言，它需要 $2^{22}$ 个页表项，那么就会有一个极大的页表，与此同时，能存储在TLB中的页表项所占比例也很低，因此在产生很多TLB miss时，它的访问时间会被大大的延长。

那么这就体现出巨页的作用了。对于一个2M的页而言，它所需要的页表项大大减少，由于TLB的大小是不变的，与此同时存储在TLB中的页表项的比例也得到提高。因此在使用巨页时，TLB hit的概率会提高，并且miss后需要检索的页表大小也会缩减，因此可以减少访问时间。

### 四：分别在两个虚拟机中使用和不使用巨页进行测试与结果分析

在虚拟机上创建巨页的过程与在host server上创建巨页的过程一致。我给虚拟机分配了2G的内存，并申请了512个2M的巨页。

需要注意的是在创建完毕后，需要让巨页真正地被使用起来，需要输入以下命令：

```
sudo apt-get install libhugetlbfs-dev
$ mkdir -p /mnt/hugetlbfs
$ mount -t hugetlbfs none /mnt/hugetlbfs
LD_PRELOAD=libhugetlbfs.so HUGETLB_MORECORE=yes <your app command line>
```

your app command line将会被sysbench的命令所替代。

sysbench memory命令可以加以下参数：

- memory-block-size=SIZE 测试内存块的大小，默认为1K
- memory-total-size=SIZE 数据传输的总大小，默认为100G
- memory-scope=STRING 内存访问的范围，包括全局和本地范围，默认为global
- memory-hugetlb=[on|off] 是否从HugeTLB池分配内存的开关，默认为off
- memory-oper=STRING 内存操作的类型，包括read, write, none，默认为write
- memory-access-mode=STRING 内存访问模式，包括seq,rnd两种模式，默认为seq

为了能够使用到巨页，在使用命令进行测试时，我们需要把内存块的大小调大，大到必须要使用巨页才能够容纳，否则没有任何意义，并且还要使用多个巨页，否则和未使用巨页也没任何区别。

测试样例如下图所示，在使用了host server巨页的虚拟机中：

```
samliu@samliu-Standard-PC-i440FX-PIIX-1996: ~
File Edit View Search Terminal Help
WritebackTmp:      0 kB
CommitLimit:      1221812 kB
Committed_AS:     4526024 kB
VmallocTotal:     34359738367 kB
VmallocUsed:      18800 kB
VmallocChunk:     0 kB
Percpu:           704 kB
HardwareCorrupted: 0 kB
AnonHugePages:    0 kB
ShmemHugePages:   0 kB
ShmemPmdMapped:   0 kB
FileHugePages:    0 kB
FilePmdMapped:    0 kB
CmaTotal:         0 kB
CmaFree:          0 kB
HugePages_Total:  512
HugePages_Free:   255
HugePages_Rsvd:   1
HugePages_Surp:   0
Hugepagesize:     2048 kB
Hugetlb:          1048576 kB
DirectMap4k:      128868 kB
DirectMap2M:      1968128 kB
samliu@samliu-Standard-PC-i440FX-PIIX-1996: ~$ LD_PRELOAD=libhugetlbfs.so HUGETLB_MORECORE=yes sysbench --test=memory --memory-block-size=512M --memory-total-size=100G run
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
sysbench 1.0.11 (using system LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 1
Initializing random number generator from current time

Running memory speed test with the following options:
block size: 524288KiB
total size: 102400MiB
operation: write
scope: global

Initializing worker threads...

Threads started!

Total operations: 200 ( 25.58 per second)

102400.00 MiB transferred (13099.01 MiB/sec)
```

```
Activities Terminal 20:56
samliu@samliu-Standard-PC-i440FX-PIIX-1996: ~
File Edit View Search Terminal Help
WritebackTmp:      0 kB
CommitLimit:      1746100 kB
Committed_AS:     5129160 kB
VmallocTotal:     34359738367 kB
VmallocUsed:      18928 kB
VmallocChunk:     0 kB
Percpu:           704 kB
HardwareCorrupted: 0 kB
AnonHugePages:    0 kB
ShmemHugePages:   0 kB
ShmemPmdMapped:   0 kB
FileHugePages:    0 kB
FilePmdMapped:    0 kB
CmaTotal:         0 kB
CmaFree:          0 kB
HugePages_Total:  0
HugePages_Free:   0
HugePages_Rsvd:   0
HugePages_Surp:   0
Hugepagesize:     2048 kB
Hugetlb:          0 kB
DirectMap4k:      128868 kB
DirectMap2M:      1968128 kB
samliu@samliu-Standard-PC-i440FX-PIIX-1996: ~$ LD_PRELOAD=libhugetlbfs.so HUGETLB_MORECORE=yes sysbench --test=memory --memory-block-size=512M --memory-total-size=100G --memory-access-mode=seq run
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
libhugetlbfs: WARNING: New heap segment map at 0x55ccd3800000 failed: Cannot allocate memory
sysbench 1.0.11 (using system LuaJIT 2.1.0-beta3)

Threads fairness:
events (avg/stddev):      200.0000/0.00
execution time (avg/stddev): 7.5417/0.00

Running the test with following options:
Number of threads: 1
Initializing random number generator from current time

Running memory speed test with the following options:
block size: 524288KiB
total size: 102400MiB
operation: write
scope: global

Initializing worker threads...

Threads started!

Total operations: 200 ( 23.10 per second)

102400.00 MiB transferred (11829.47 MiB/sec)
```

可以看出，在使用了巨页后，传输速度有了明显的提升，与理论分析一致

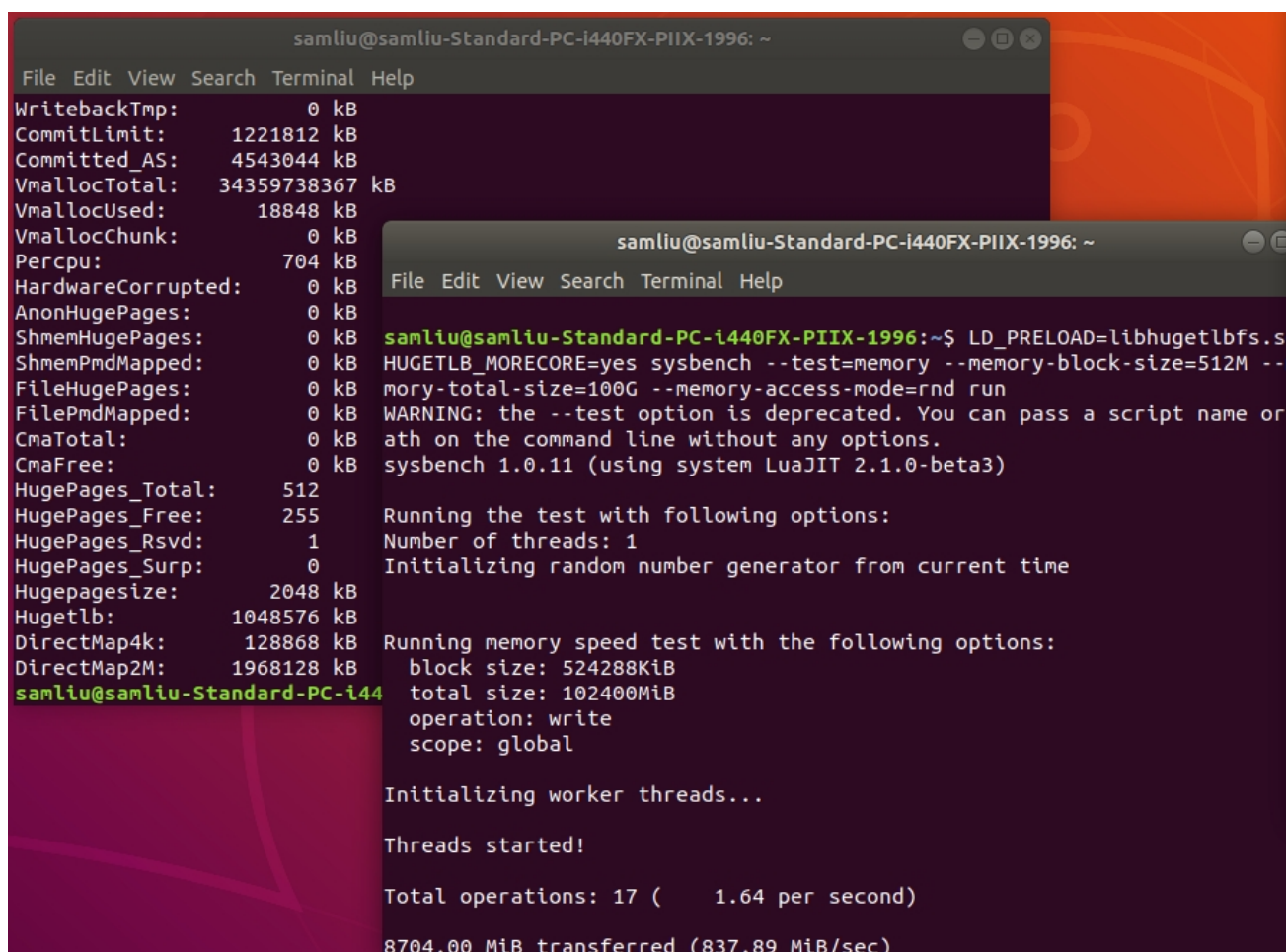
经过多次测量后得到结果表格如下：



机器序号	是否使用host server巨页	是否在内部使用巨页	block-size=512M	block-size=1M
1	√	√	13099M/s	21314M/s
2	√	×	11829M/s	21301M/s
3	×	√	12631M/s	21291M/s
4	×	×	11373M/s	21293M/s

从实验结果中可以看出，总体来说使用了host server巨页的虚拟机会比没有使用的略快一点点。当block-size=512M时，使用了巨页的虚拟机要比没有使用的快了将近10%，而当block-size=1M时差距并不明显。究其原因，当block-size更大时更加容易发生TLB miss，因此更加能够展现出巨页在这方面的优势。

与此同时我还测试了随机访问下的传输速度，结果如下：



```

samlu@samlu-Standard-PC-i440FX-PIIX-1996: ~
File Edit View Search Terminal Help
WritebackTmp:      0 kB
CommitLimit:     1221812 kB
Committed_AS:    4543044 kB
VmallocTotal:    34359738367 kB
VmallocUsed:      18848 kB
VmallocChunk:     0 kB
Percpu:          704 kB
HardwareCorrupted: 0 kB
AnonHugePages:    0 kB
ShmemHugePages:   0 kB
ShmemPmdMapped:   0 kB
FileHugePages:    0 kB
FilePmdMapped:    0 kB
CmaTotal:         0 kB
CmaFree:          0 kB
HugePages_Total:  512
HugePages_Free:   255
HugePages_Rsvd:   1
HugePages_Surp:   0
Hugepagesize:     2048 kB
Hugetlb:          1048576 kB
DirectMap4k:      128868 kB
DirectMap2M:      1968128 kB
samlu@samlu-Standard-PC-i440FX-PIIX-1996: ~$ LD_PRELOAD=libhugetlbfs.so
HUGETLB_MORECORE=yes sysbench --test=memory --memory-block-size=512M --m
mory-total-size=100G --memory-access-mode=rnd run
WARNING: the --test option is deprecated. You can pass a script name or
ath on the command line without any options.
sysbench 1.0.11 (using system LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 1
Initializing random number generator from current time

Running memory speed test with the following options:
block size: 524288KiB
total size: 102400MiB
operation: write
scope: global

Initializing worker threads...

Threads started!

Total operations: 17 ( 1.64 per second)

8704.00 MiB transferred (837.89 MiB/sec)

```

可以看到此时的传输速度要远远低于顺序访问，因为在随机访问时会产生比顺序访问时多得多的TLB miss，此时操作系统要不断地去查表，因此TLB的作用就没有发挥出来，因此传输速度会大大降低。

## 总结

本次实验让我学会了如何在机器中申请并使用巨页，并学会了如何给虚拟机分配巨页，还让我对内存和巨页有了更深入的理解，是一次收获颇丰的体验！