

# Hugepage

工程实践与科技创新III-D 虚拟化与云计算 EI313

李子龙 518070910095

2021 年 11 月 15 日

## 目录

1	要求	1
2	配置大内存页	1
3	创建对照虚拟机	2
4	运行测试	3
5	解释	4

## 1 要求

- (1) Prepare 2MB or 1GB hugepages on your host server. Present your hugepage configure (e.g. `/proc/meminfo`).
- (2) Create a QEMU KVM virtual machine using hugepages on the host.
- (3) Create another QEMU KVM VM without hugepages.
- (4) In both VMs allocate and use hugepages or not.
- (5) Run memory intensive benchmark (e.g. sysbench memory test) on two VMs and record the performance.
- (6) Compare the result and try to give some explanation.

Note: If the OS supports transparent huge page, disable it when you do the tests.

## 2 配置大内存页

向 `/etc/sysctl.conf` 中输入 `vm.nr_hugepages = 100`, 然后重启系统, 如图 1 所示。然后显示大内存页的配置信息如图 2 所示: 显示有 2MB 大内存页 100 个。

```
logcreative@ubuntu: ~  
#  
# Log Martian Packets  
#net.ipv4.conf.all.log_martians = 1  
#  
#####  
# Magic system request key  
# 0:disable, 1:enable all  
# Debian kernels have this set to 0 (disable the key)  
# See https://www.kernel.org/doc/Documentation/sysrq.txt  
# for what other values do  
#kernel.sysrq=1  
#  
#####  
# Protected links  
#  
# Protects against creating or following links under certain conditions  
# Debian kernels have both set to 1 (restricted)  
# See https://www.kernel.org/doc/Documentation/sysctl/fs.txt  
#fs.protected_hardlinks=0  
#fs.protected_symlinks=0  
#  
vm.nr_hugepages=100  
79,19 底端
```

图 1: 配置大页

```
logcreative@ubuntu: ~  
#  
logcreative@ubuntu:~$ grep Huge /proc/meminfo  
AnonHugePages:          0 kB  
ShmemHugePages:         0 kB  
FileHugePages:          0 kB  
HugePages_Total:       100  
HugePages_Free:        100  
HugePages_Rsvd:         0  
HugePages_Surp:         0  
Hugepagesize:           2048 kB  
Hugetlb:               204800 kB  
logcreative@ubuntu:~$
```

图 2: 显示配置

接着按照提示，先关闭透明大页。Transparent HugePages是在运行时动态分配内存的，而标准的HugePages是在系统启动时预先分配内存，并在系统运行时不再改变。因为Transparent HugePages是在运行时动态分配内存的，所以会带来在运行时内存分配延误。<sup>[1]</sup>

Listing 1: [disableTHP.sh](#)

```
1 sudo -s  
2 echo never > /sys/kernel/mm/transparent_hugepage/enabled  
3 cat /sys/kernel/mm/transparent_hugepage/enabled
```

```
root@ubuntu: ~  
#  
root@ubuntu:~# echo never > /sys/kernel/mm/transparent_hugepage/enabled  
root@ubuntu:~#  
root@ubuntu:~# cat /sys/kernel/mm/transparent_hugepage/enabled  
always madvise [never]  
root@ubuntu:~#  
root@ubuntu:~#
```

图 3: 关闭透明大页

### 3 创建对照虚拟机

在 `virt-manager` 中克隆虚拟机，并在其中一个虚拟机中按照上述的方法配置大内存页。两种虚拟机都使用了 QEMU KVM。

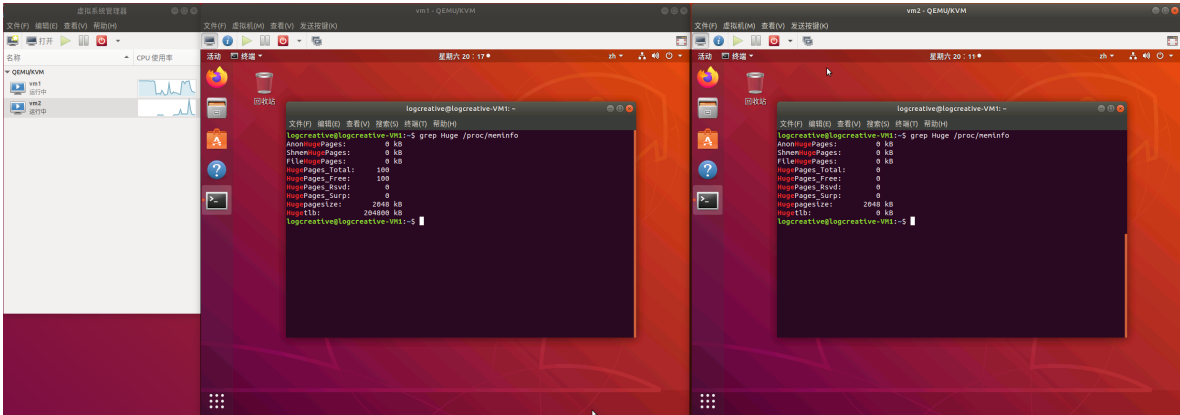


图 4: 两种虚拟机

#### 4 运行测试

为了保证公平，测试仅运行于一个虚拟机开启的情况下。测试使用 **sysbench**。

Listing 2: [benchmark.sh](#)

```
1 sysbench memory --memory-block-size=2M --memory-total-size=2G run
```

测试结果分别如图 5 和图 6 所示。

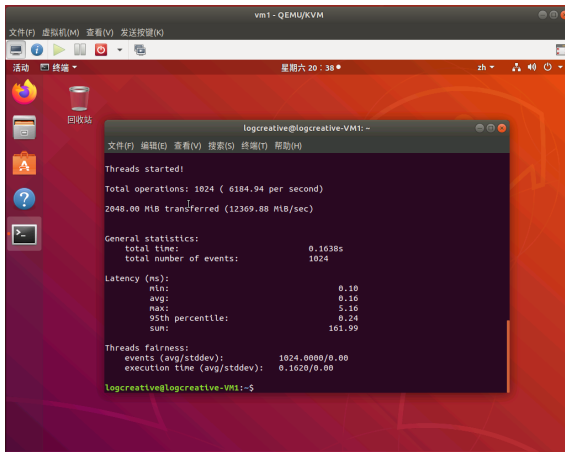


图 5: 含有大内存页

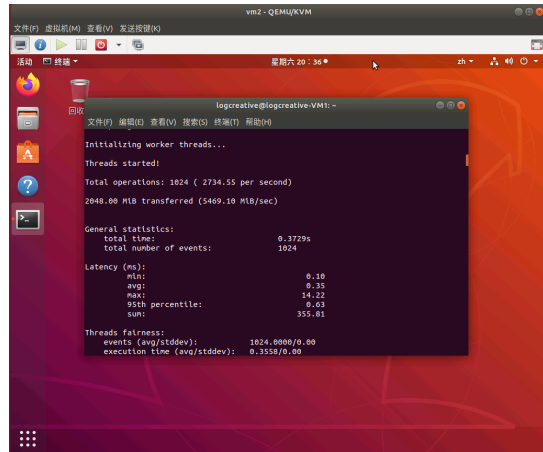


图 6: 不含有大内存页

## 5 解释

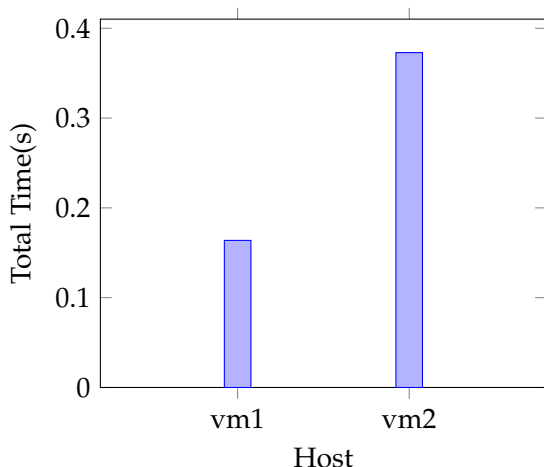


图 7: 总时间比较

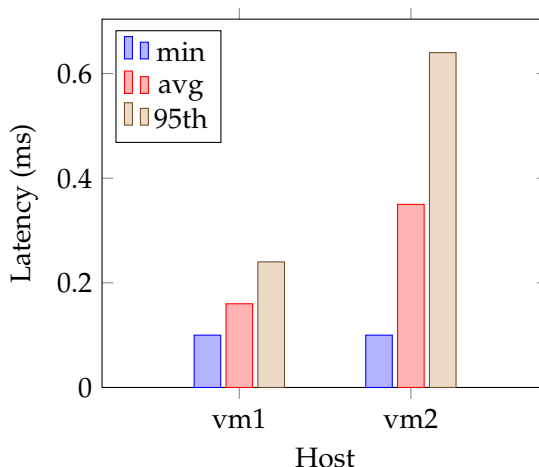


图 8: 延迟比较

由图 7 可见，对于 2G 内存（2MB 块大小）写入测试上，开启了大内存页的 VM1 确实更占优势（降低了约 56%），且在图 8 的延迟比较上也占上风（总延迟降低了 54%）。

在虚拟内存管理中，内核维护一个将虚拟内存地址映射到物理地址的表，对于每个页面操作，内核都需要加载相关的映射。如果你的内存页很小，那么你需要加载的页就会很多，导致内核会加载更多的映射表。而这会降低性能。使用“大内存页”，意味着所需要的页变少了。从而大大减少由内核加载的映射表的数量。这提高了内核级别的性能最终有利于应用程序的性能。从而减少访问的开销。<sup>[2]</sup>

而使用的大内存页都是 2M 的，并排布了 100 个，至少可以有效减少一部分内存页的访问开销，从而减少访问时间和延迟。

## 参考文献

- [1] 海东潮. Linux的Transparent Hugepage与关闭方法[EB/OL]. 2018. <https://www.cnblogs.com/DataArt/p/9975281.html>.
- [2] LAVHATE S. Linux 中的“大内存页”（hugepage）是个什么？[EB/OL]. 2018. <https://linux.cn/article-9450-1.html>.