

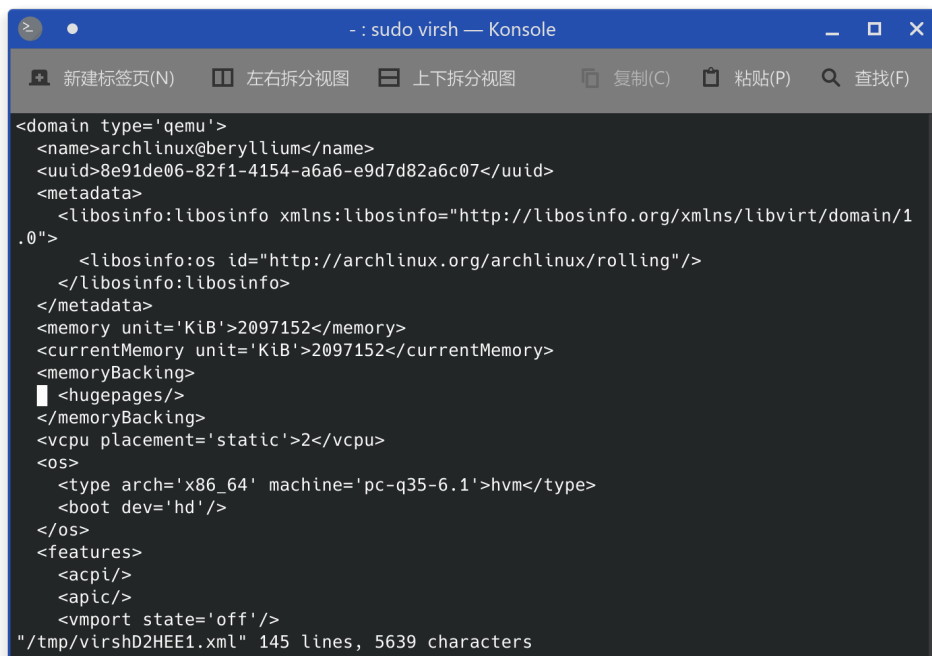
要使虚拟机使用宿主机的Hugepage，只需要在编辑虚拟机对应的xml文件。我们在宿主机中输入以下命令进行编辑。

```
1 | sudo virsh edit archlinux@beryllium
```

写入以下几行：

```
1 | <memoryBacking>
2 | <hugepages/>
3 | </memoryBacking>
```

编辑后的文件如下图所示：



要使虚拟机不使用宿主机的Hugepage，则将这几行删去即可。

### 3 在虚拟机中开启Hugepage并安装测试所用软件

(4) In both VMs allocate and use hugepages or not.

方法与宿主机中完全相同。在虚拟机中启动了512个大小为2kB的Hugepage。

```
> cat /proc/meminfo | grep Huge
AnonHugePages:          0 kB
ShmemHugePages:         0 kB
FileHugePages:          0 kB
HugePages_Total:        0
HugePages_Free:          0
HugePages_Rsud:          0
HugePages_Surp:          0
Hugepagesize:           2048 kB
Hugetlb:                 0 kB

512
> echo 512 | sudo tee /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages
512
> cat /proc/meminfo | grep Huge
AnonHugePages:          0 kB
ShmemHugePages:         0 kB
FileHugePages:          0 kB
HugePages_Total:        512
HugePages_Free:          512
HugePages_Rsud:          0
HugePages_Surp:          0
Hugepagesize:           2048 kB
Hugetlb:                1048576 kB
```

### 安装sysbench:

```
1 | sudo pacman -S sysbench
```

## 4 进行内存测试

(5) Run memory intensive benchmark (e.g. sysbench memory test, in-memory database) on two VMs and record the performance.

这里我使用sysbench进行测试，首先编写测试脚本如下：

archlinux@beryllium - QEMU/KVM

文件(F) 虚拟机(M) 查看(V) 发送按键(K)

#! /bin/bash

MEMORY\_TOTAL\_SIZE=2048G

read -p "memory-hugetlb? [on/off] " MEMORY\_HUGETLB

read -p "memory-block-size= " MEMORY\_BLOCK\_SIZE

sudo sysbench memory --memory-block-size=\$MEMORY\_BLOCK\_SIZE --memory-total-size=\$MEMORY\_TOTAL\_SIZE --memory-hugetlb=\$MEMORY\_HUGE

TLB --memory-access-mode=rdn run

其中,

- `--memory-block-size` 选项代表了测试内存块的大小，这里我将其设置为测试时输入。
- `--memory-total-size` 选项代表了传输数据的总大小，这里我将其设置为 `2048G`。
- `--memory-hugetlb` 选项决定了测试时虚拟机是否启用Hugepage，这里我将其设置为测试时输入。
- `--memory-access-mode` 选项代表了测试时的存取方式，这里我将其设置为 `rnd`，代表随机读写。

接下来开始测试。由于默认的内存页大小为4kB，而Hugepage大小为2048kB，测试内存块的大小应该设置得大一些，从而凸显出二者的差别。经过测试，我发现当测试内存块的大小设置为刚好2MB时，效果较好。

#### 4.1 宿主机不启用Hugepage，虚拟机不启用Hugepage

<pre>&gt; ./Programming/memoryTest.sh memory-hugetlb? [on/off] off memory-block-size= 2M sysbench 1.0.20 (using system LuaJIT 2.0.5)  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: 2048KiB total size: 2097152MiB operation: write scope: global  Initializing worker threads...  Threads started!  Total operations: 808 ( 80.68 per second) 1616.00 MiB transferred (161.35 MiB/sec)  General statistics: total time: 10.0085s total number of events: 808  Latency (ms): min: 9.93 avg: 12.34 max: 30.12 95th percentile: 16.12 sun: 9969.34  Threads fairness: events (avg/stddev): 808.0000/0.00 execution time (avg/stddev): 9.9693/0.00</pre>	<pre>&gt; ./Programming/memoryTest.sh memory-hugetlb? [on/off] off memory-block-size= 2M sysbench 1.0.20 (using system LuaJIT 2.0.5)  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: 2048KiB total size: 2097152MiB operation: write scope: global  Initializing worker threads...  Threads started!  Total operations: 774 ( 77.32 per second) 1548.00 MiB transferred (154.63 MiB/sec)  General statistics: total time: 10.0050s total number of events: 774  Latency (ms): min: 9.85 avg: 12.88 max: 65.06 95th percentile: 16.41 sun: 9969.12  Threads fairness: events (avg/stddev): 774.0000/0.00 execution time (avg/stddev): 9.9691/0.00</pre>	<pre>&gt; ./Programming/memoryTest.sh memory-hugetlb? [on/off] off memory-block-size= 2M sysbench 1.0.20 (using system LuaJIT 2.0.5)  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: 2048KiB total size: 2097152MiB operation: write scope: global  Initializing worker threads...  Threads started!  Total operations: 742 ( 74.10 per second) 1484.00 MiB transferred (148.21 MiB/sec)  General statistics: total time: 10.0074s total number of events: 742  Latency (ms): min: 9.90 avg: 13.45 max: 79.22 95th percentile: 15.55 sun: 9977.32  Threads fairness: events (avg/stddev): 742.0000/0.00 execution time (avg/stddev): 9.9773/0.00</pre>
---	---	---

三次测试，平均写入速率为154.73MiB/s。

## 4.2 宿主机不启用Hugepage，虚拟机启用Hugepage

<pre>&gt; ./Programming/memoryTest.sh memory-hugetlb? [on/off] on memory-block-size= 2M sysbench 1.0.20 (using system LuaJIT 2.0.5)  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: 2048KiB total size: 2097152MiB operation: write scope: global  Initializing worker threads...  Threads started!  Total operations: 917 ( 91.60 per second) 1834.00 MiB transferred (183.21 MiB/sec)  General statistics: total time: 10.0045s total number of events: 917  Latency (ms): min: 9.84 avg: 10.80 max: 31.14 95th percentile: 13.22 sun: 9981.50  Threads fairness: events (avg/stddev): 917.0000/0.00 execution time (avg/stddev): 9.9815/0.00</pre>	<pre>&gt; ./Programming/memoryTest.sh memory-hugetlb? [on/off] on memory-block-size= 2M sysbench 1.0.20 (using system LuaJIT 2.0.5)  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: 2048KiB total size: 2097152MiB operation: write scope: global  Initializing worker threads...  Threads started!  Total operations: 842 ( 84.04 per second) 1604.00 MiB transferred (160.00 MiB/sec)  General statistics: total time: 10.0131s total number of events: 842  Latency (ms): min: 9.87 avg: 11.87 max: 26.36 95th percentile: 14.73 sun: 9991.91  Threads fairness: events (avg/stddev): 842.0000/0.00 execution time (avg/stddev): 9.9919/0.00</pre>	<pre>&gt; ./Programming/memoryTest.sh memory-hugetlb? [on/off] on memory-block-size= 2M sysbench 1.0.20 (using system LuaJIT 2.0.5)  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: 2048KiB total size: 2097152MiB operation: write scope: global  Initializing worker threads...  Threads started!  Total operations: 847 ( 84.56 per second) 1694.00 MiB transferred (169.13 MiB/sec)  General statistics: total time: 10.0104s total number of events: 847  Latency (ms): min: 9.86 avg: 11.79 max: 28.68 95th percentile: 15.00 sun: 9988.04  Threads fairness: events (avg/stddev): 847.0000/0.00 execution time (avg/stddev): 9.9880/0.00</pre>
--	--	--

三次测试，平均写入速率为173.47MiB/s。

## 4.3 宿主机启用Hugepage，虚拟机不启用Hugepage

<pre>&gt; ./Programming/memoryTest.sh memory-hugetlb? [on/off] off memory-block-size= 2M sysbench 1.0.20 (using system LuaJIT 2.0.5)  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: 2048KiB total size: 2097152MiB operation: write scope: global  Initializing worker threads...  Threads started!  Total operations: 828 ( 82.73 per second) 1656.00 MiB transferred (165.46 MiB/sec)  General statistics: total time: 10.0027s total number of events: 828  Latency (ms): min: 9.82 avg: 12.05 max: 28.90 95th percentile: 15.55 sun: 9978.05  Threads fairness: events (avg/stddev): 828.0000/0.00 execution time (avg/stddev): 9.9781/0.00</pre>	<pre>&gt; ./Programming/memoryTest.sh memory-hugetlb? [on/off] off memory-block-size= 2M sysbench 1.0.20 (using system LuaJIT 2.0.5)  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: 2048KiB total size: 2097152MiB operation: write scope: global  Initializing worker threads...  Threads started!  Total operations: 800 ( 79.90 per second) 1600.00 MiB transferred (159.81 MiB/sec)  General statistics: total time: 10.0062s total number of events: 800  Latency (ms): min: 9.75 avg: 12.48 max: 57.53 95th percentile: 16.71 sun: 9987.21  Threads fairness: events (avg/stddev): 800.0000/0.00 execution time (avg/stddev): 9.9872/0.00</pre>	<pre>&gt; ./Programming/memoryTest.sh memory-hugetlb? [on/off] off memory-block-size= 2M sysbench 1.0.20 (using system LuaJIT 2.0.5)  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: 2048KiB total size: 2097152MiB operation: write scope: global  Initializing worker threads...  Threads started!  Total operations: 846 ( 84.49 per second) 1632.00 MiB transferred (168.99 MiB/sec)  General statistics: total time: 10.0054s total number of events: 846  Latency (ms): min: 9.93 avg: 11.81 max: 30.07 95th percentile: 15.27 sun: 9988.17  Threads fairness: events (avg/stddev): 846.0000/0.00 execution time (avg/stddev): 9.9802/0.00</pre>
---	---	---

三次测试，平均写入速率为164.75MiB/s。

## 4.4 宿主机启用Hugepage，虚拟机启用Hugepage

<pre>&gt; ./Programming/memoryTest.sh memory-hugetlb? [on/off] on memory-block-size= 2M sysbench 1.0.20 (using system LuaJIT 2.0.5)  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: 2048KiB total size: 2097152MiB operation: write scope: global  Initializing worker threads...  Threads started!  Total operations: 925 ( 92.33 per second) 1850.00 MiB transferred (184.67 MiB/sec)  General statistics: total time: 10.0115s total number of events: 925  Latency (ms): min: 9.63 avg: 10.89 max: 21.56 95th percentile: 13.22 sum: 9993.76  Threads fairness: events (avg/stddev): 925.0000/0.00 execution time (avg/stddev): 9.9930/0.00</pre>	<pre>&gt; ./Programming/memoryTest.sh memory-hugetlb? [on/off] on memory-block-size= 2M sysbench 1.0.20 (using system LuaJIT 2.0.5)  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: 2048KiB total size: 2097152MiB operation: write scope: global  Initializing worker threads...  Threads started!  Total operations: 817 ( 81.56 per second) 1634.00 MiB transferred (163.11 MiB/sec)  General statistics: total time: 10.0109s total number of events: 817  Latency (ms): min: 9.51 avg: 12.23 max: 30.97 95th percentile: 15.27 sum: 9980.64  Threads fairness: events (avg/stddev): 817.0000/0.00 execution time (avg/stddev): 9.9806/0.00</pre>	<pre>&gt; ./Programming/memoryTest.sh memory-hugetlb? [on/off] on memory-block-size= 2M sysbench 1.0.20 (using system LuaJIT 2.0.5)  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: 2048KiB total size: 2097152MiB operation: write scope: global  Initializing worker threads...  Threads started!  Total operations: 925 ( 92.37 per second) 1850.00 MiB transferred (184.73 MiB/sec)  General statistics: total time: 10.0087s total number of events: 925  Latency (ms): min: 9.81 avg: 10.79 max: 23.14 95th percentile: 13.22 sum: 9979.01  Threads fairness: events (avg/stddev): 925.0000/0.00 execution time (avg/stddev): 9.9790/0.00</pre>
--	--	--

三次测试，平均写入速率为177.50MiB/s。

## 5 结论与分析

(6) Compare the result and try to give some explanation.

### 5.1 测试结果

测试结果如下：

	虚拟机不启用Hugepage	虚拟机启用Hugepage
宿主机不启用Hugepage	154.73MiB/s	173.47MiB/s
宿主机启用Hugepage	164.75MiB/s	177.50MiB/s

可以看出，无论是宿主机还是虚拟机，启用Hugepage都会对内存的随机写入性能带来一定提升。当二者均启用Hugepage时，其性能要比均不启用时高出14.7%。

### 5.2 分析

经过查阅资料<sup>3 4 5</sup>，我认为Hugepage给内存性能带来提升，主要有以下几个原因。

- 使用Hugepage可以减少内存中的页表层级。这不仅可以降低页表的内存占用，也能降低从虚拟内存到物理内存转换的性能损耗。
- Hugepage能降低TLB（Translation lookaside buffer）的压力，在相同的内存大小情况下使得需要管理的虚拟地址数量变少，因此TLB可以包含更多的地址空间，从而带来更高的缓存命中率，CPU 有更高的几率可以直接在 TLB中获取对应的物理地址。
- 使用Hugepage可以减少获取大内存的次数，使用 HugePages 每次可以获取 2MB 的内存，是 4KB 的默认页效率的 512 倍。

## 6 致谢

感谢我的同学秦健行、卿云帆、陈浩南、刘梓睿和蒋圩溟在本次实验中对我的帮助！在与他们的交流和讨论中我解决了许多困难，并学到了很多。

2. KVM - Using Hugepages - Community Help Wiki ↩ ↪
3. 为什么 HugePages 可以提升数据库性能 - 面向信仰编程 ↩
4. Linux 中的“大内存页”（hugepage）是个什么? - 知乎 ↩
5. Linux HugePage 特性乐沙弥的世界-CSDN博客hugepage ↩