



登临Hamming™

虚拟机虚拟化使用手册

DL-DG/SW-027A-03

2023-10-17

Copyright©上海登临科技有限公司，2019 - 2023，版权所有。

未经上海登临科技有限公司事先书面同意，不得以任何形式或方式复制或传播本文件的任何部分。

商标和许可



和其它上海登临科技有限公司的其它登临科技的图标为上海登临科技有限公司的商标。

本手册中提及的所有其他商标均为其各自所有者的财产。

通知

所购买的产品、服务和特性由上海登临科技有限公司与客户签订的合同规定。本文件中描述的所有或部分产品、服务和特性可能不在采购范围或使用范围内。除非合同中另有规定，本文件中的所有声明、信息和建议均按“原样”提供，无任何明示或暗示的保证或陈述。本手册中的信息如有更改，恕不另行通知。本文件在编制过程中已尽一切努力确保内容的准确性，本文件中的所有声明、信息和建议不构成任何明示或暗示的保证。

上海登临科技有限公司

上海浦东新区春晓路439号10幢，上海，中国

<http://www.denglin.ai>

email : support@denglin.ai

变更历史

版本	描述
03	更新文档名称；新增Device Hang注意事项；对于mdev的描述更新为vgpu。
02	新增CentOS下Grub打开IOMMU的方法。
01	初始版本。

章节目录

简介

直通模式

共享模式

直通模式的使用

环境与版本要求

使用说明

宿主机环境准备

配置直通

共享模式的使用

环境与版本要求

使用说明

宿主机环境准备

管理虚拟设备

虚拟设备的类型

查看可创建虚拟设备的信息

创建虚拟设备

查看已创建虚拟设备信息

删除虚拟设备

QEMU中使用虚拟设备

宿主机卸载虚拟机虚拟化环境

注意事项

设备隔离性说明

获取虚拟设备的状态

使用示例

简介

为满足用户在虚拟机场景下对登临GPGPU设备的使用，登临系统软件虚拟机虚拟化提供两种模式：直通（Pass-Through）模式和共享模式。

在直通模式下，可以将一个登临GPGPU分配给某个虚拟机独占使用。

在共享模式下，可以为一个物理登临GPGPU设备创建多个虚拟设备，且这些虚拟设备可分别供给多个虚拟机同时使用。

直通模式

直通模式依赖于Linux VFIO框架，一个物理设备只能被一个虚拟机独占使用，无法共享。在Guest操作系统内，被直通的设备会表现为一个PCIe设备，使用方式和在宿主机上使用物理设备一样。

共享模式

共享模式的实现依赖于MIG（Multi-Instance GPU）技术，不同虚拟设备之间采用硬件隔离而非分时复用的方式共享。

对于一个拥有4个cluster的登临GPGPU，最多可以创建4个虚拟设备供4个虚拟机同时使用，每个虚拟设备拥有单个cluster的算力和设备内存空间。虚拟设备本身只能被某个虚拟机独占使用，无法再次共享。

在Guest操作系统内，虚拟设备也是一个PCIe设备。因此，可以正常安装和使用登临Hamming™ SDK。

直通模式的使用

环境与版本要求

- Host OS：支持Linux VFIO框架的Linux操作系统。
- Guest OS：支持安装登临Hamming SDK的Linux操作系统。
- QEMU：版本2.5及以上。

使用说明

宿主机环境准备

1. 在宿主机BIOS设置中打开CPU的虚拟化支持，如Intel的VT-d或AMD的IOMMU。
2. 确认在Linux引导时IOMMU功能被打开。先查看宿主机 `/sys/kernel/iommu_groups` 目录中是否有IOMMU group文件夹。IOMMU group文件夹示例如下：

```
(sdk) root@open_pc_094:/sys/kernel/iommu_groups# ls -al
total 0
drwxr-xr-x 18 root root 0 12月 21 15:03 ./
drwxr-xr-x 14 root root 0 12月 21 15:03 ../
drwxr-xr-x  3 root root 0 12月 21 15:03 0/
drwxr-xr-x  3 root root 0 12月 21 15:03 1/
drwxr-xr-x  3 root root 0 12月 21 15:03 2/
drwxr-xr-x  3 root root 0 12月 21 15:03 3/
drwxr-xr-x  3 root root 0 12月 21 15:03 4/
```

- 如果IOMMU group文件夹存在，表示IOMMU已经在Linux引导时被打开。
- 否则，表示Linux引导时IOMMU功能未被打开，需要通过编辑Linux内核引导参数来开启IOMMU功能：
 - 对于Ubuntu16.04操作系统，参考操作方法如下：

```
#vim编辑文件，需要root权限
vim /etc/default/grub
# Intel CPU为例，在GRUB_CMDLINE_LINUX_DEFAULT的末尾，添加intel_iommu=on
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash intel_iommu=on"
#保存/etc/default/grub并退出，然后更新grub
update-grub
```

- 对于CentOS 7操作系统，参考操作方法如下：

```
#vim编辑文件，需要root权限
vim /etc/default/grub
# Intel CPU为例，在GRUB_CMDLINE_LINUX_DEFAULT的末尾，添加intel_iommu=on
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash intel_iommu=on"
#保存/etc/default/grub并退出，然后更新grub
grub2-mkconfig -o /boot/efi/EFI/centos/grub.cfg
```

更新之后重启宿主机，再次确认宿主机中 `/sys/kernel/iommu_groups` 目录下IOMMU group文件夹是否存在。

3. 加载vfio模块以及vfio-pci模块，操作方法如下：

```
sudo modprobe vfio
sudo modprobe vfio-pci
```

此时 `lsmod | grep vfio` 可以看到以下内容：

```
(sdk) root@open_pc_094:/sys/kernel/iommu_groups# lsmod | grep vfio
vfio_pci                45056  0
vfio_virqfd             16384  1 vfio_pci
vfio_iommu_type1        24576  0
vfio                    28672  2 vfio_iommu_type1,vfio_pci
irqbypass               16384  2 vfio_pci,kvm
```

配置直通

1. 将需要直通的登临GPGPU从Denglin的driver解绑，然后绑定到VFIO的驱动。

```
# 显示登临GPGPU的DBDF信息
lspci -D
# 将登临GPGPU从denglin kernel driver解绑（以DBDF为0000:01:00.0为例）
echo 0000:01:00.0 > /sys/bus/pci/devices/0000\:01\:00.0/driver/unbind
# 将登临GPGPU与VFIO的驱动绑定
echo 1e27 0002 > /sys/bus/pci/drivers/vfio-pci/new_id
# 此时，在/dev/vfio目录下，可以看到新设备
```

2. 将需要直通的设备通过参数传递给QEMU，命令如下：

```
# DLI-BDF是直通设备在主机PCIe总线上的位置，如01:00.0，可使用命令 "lspci | grep 1e27" 进行查询。
-device vfio-pci,host=<DLI-BDF>
```

3. 启动虚拟机。

此时在Guest操作系统中执行 `lspci` 命令，便可以看到直通的登临GPGPU设备。在Guest操作系统中安装登临Hamming SDK，就可以像在Host中一样使用登临GPGPU。

4. 当在虚拟机中使用完毕后，关闭虚拟机。使用如下命令将设备从VFIO驱动解绑，并重新绑定到Host的denglin kernel driver，之后可以在Host上正常使用该设备。

```
# 仍以设备0000:01:00.0为例
echo 0000:01:00.0 > /sys/bus/pci/devices/0000\:01\:00.0/driver/unbind
echo 0000:01:00.0 > /sys/bus/pci/drivers/denglin/bind
```

共享模式的使用

环境与版本要求

- Host OS：支持安装登临Hamming SDK，同时支持Linux VFIO-MDev框架的Linux操作系统（kernel version >= 4.15）。
- Guest OS：支持安装登临Hamming SDK的Linux操作系统。
- QEMU：版本2.6及以上。
- 登临Hamming SDK：版本0.2.11及以上，Host与Guest的登临Hamming SDK版本需一致（登临Hamming SDK的版本请查看安装主路径的version.txt文件）。

使用说明

宿主机环境准备

1. 按照直通的要求准备宿主机环境。
2. 加载VFIO-MDev内核模块。VFIO-MDev模块是Linux内核自带模块，可以通过 `modprobe` 命令加载：

```
sudo modprobe vfio-mdev
```

3. 宿主机上安装支持虚拟化的登临Hamming SDK，安装登临Hamming SDK的步骤请参考《登临Hamming SDK 安装指南》。
4. 安装与Hamming SDK版本一致的登临虚拟机虚拟化软件安装包：
 1. 解压安装包：

```
tar -xf <package>.tar
```

2. 安装解压出来的 `.run` 文件：

```
# 安装过程需要sudo权限
# 参数install_path为安装路径，参数denglin_hamming_sdk_dir为登临Hamming SDK在本机的安装路径
sudo ./<denglin_hamming-vgpu-xxx>.run --target <install_path>
<denglin_hamming_sdk_dir>
```

3. 正常安装完成后，终端会有 `Installation successful` 的提示。可以用 `lsmod` 命令验证安装是否成功：

```
# 至少会显示以下5个模块
lsmod | grep denglin_vgpu
denglin_vgpu_deps      16384  0
denglin_vgpu           126976  1
denglin                344064  1 denglin_vgpu
mdev                   20480  2 denglin_vgpu,vfio_mdev
vfio                   28672  13 denglin_vgpu,...
```

至此，宿主机环境准备完毕，可以开始创建和使用虚拟设备。

管理虚拟设备

虚拟设备的类型

根据一个虚拟设备所拥有的cluster数量的不同，一共有3种虚拟设备类型。一个拥有4个cluster的登临GPGPU，每种虚拟设备类型和描述，以及最多可创建的数量如下表所示：

虚拟设备类型	描述	最多可创建虚拟设备的数量
denglin-MIG-1C	拥有1个cluster算力和设备内存的虚拟设备	4个
denglin-MIG-2C	拥有2个cluster算力和设备内存的虚拟设备	2个
denglin-MIG-4C	拥有4个cluster算力和设备内存的虚拟设备	1个

查看可创建虚拟设备的信息

在查看某个登临GPGPU卡可创建的虚拟设备信息之前，需要先打开对应卡的MIG模式，命令如下：

```
# 打开主机上第k张卡的MIG模式
dlsmi -i <k> -mig 1
```

- 通过 sys 文件系统查看待创建的虚拟设备信息：

```
cat /sys/bus/pci/devices/<DLI-DBDF>/mdev_supported_types/denglin-MIG-
<1|2|4>C/next_placement
```

其中参数 DLI-DBDF 是需要进行虚拟设备创建的登临GPGPU在宿主机PCIe槽上的位置，可使用 `lspci -D | grep 1e27` 进行查询；参数 denglin-MIG-<1|2|4>C 是要创建的设备类型。

上述命令的输出格式为 `start_cluster_index:cluster_num`，例如 0:1 表示接下来创建的这个虚拟设备，会占用从编号0开始的1个cluster。如果空闲的cluster数目不够，输出会为空，此时无法创建该类型的虚拟设备。

- 通过 sys 文件系统查看可创建的虚拟设备数量：

```
cat /sys/bus/pci/devices/<DLI-DBDF>/mdev_supported_types/denglin-MIG-
<1|2|4>C/available_instances
```

其中参数 DLI-DBDF 和 denglin-MIG-<1|2|4>C 的含义和上述一样。

创建虚拟设备

在为登临GPGPU创建具体的虚拟化设备之前，也需要先打开对应卡的MIG模式。

在打开MIG模式后，可以通过如下命令创建虚拟设备：

```
# 先生成一个UUID（如linux系统自带的uuidgen命令）
uuidgen
# 使用刚才创建的UUID创建虚拟设备
echo <UUID> > /sys/bus/pci/devices/<DLI-DBDF>/mdev_supported_types/denglin-MIG-
<1|2|4>C/create
```

参数 `DLI-DBDF` 和 `denglin-MIG-<1|2|4>C` 的含义和前面一样。一个UUID只能用来创建一个设备，要创建其它的虚拟设备，需要新的UUID。

查看已创建虚拟设备信息

在创建了虚拟设备后，可以在目录 `/sys/bus/mdev/devices` 下查看所有已创建的设备。该目录中包含所有以UUID为名称的子目录。每个子目录表示一个虚拟设备。

若要查看某个虚拟设备的信息，可以使用以下命令：

```
# 查看虚拟设备信息，参数UUID为创建该虚拟设备时使用的UUID
cat /sys/bus/mdev/<UUID>/mdev/mdev_info
```

该命令会展示3个信息：

- parent DLI UUID：当前虚拟设备所属的物理登临GPGPU的UUID；
- vfunc ID：当前虚拟设备在物理登临GPGPU中所属的MIG instance ID；
- cluster mask：以16进制掩码表示当前虚拟设备所使用的cluster(s)：
 - `0x1` 表示第1个cluster
 - `0x2` 表示第2个cluster
 - `0x4` 表示第3个cluster
 - `0x8` 表示第4个cluster

如果虚拟设备占用多个cluster，那么mask可以是其每个cluster的组合，例如 `0x3` 表示第1个和第2个cluster，`0xc` 表示第3个和第4个cluster等。

删除虚拟设备

删除虚拟设备前要确保虚拟设备没有被使用（如被QEMU打开）。如果不再需要某个已经创建的虚拟设备，则可以通过 `sys` 文件系统进行删除，命令如下：

```
# 删除一个虚拟设备，参数UUID为创建该虚拟设备时使用的UUID
echo 1 > /sys/bus/mdev/devices/<UUID>/remove
```

虚拟设备被移除之后，会将该虚拟设备对硬件资源的所有权归还给物理设备。后续可以继续被主机使用，如被用来创建MIG设备，或其它的虚拟设备。

QEMU中使用虚拟设备

创建了虚拟设备后，就可以将该虚拟设备通过QEMU的启动参数传递给Guest Linux操作系统使用了。QEMU启动参数需要添加的内容如下：

```
# 将一个虚拟设备传递给QEMU，其中UUID为创建该虚拟设备时使用的UUID
-device vfio-pci,syfsdev=/sys/bus/mdev/devices/<UUID>
```

如果想在虚拟机中使用多个虚拟设备，可以使用不同的UUID重复指定上述参数多次。

在Guest中，虚拟设备是一个普通的PCIe设备，可以像在Host上一样，正常安装登临Hamming SDK并使用。Guest OS中的Hamming SDK版本，需要和Host上的一致。

宿主机卸载虚拟机虚拟化环境

1. 移除所有已经创建的虚拟设备，操作方法前面已述。
2. 移除虚拟化内核模块，操作如下：

```
rmmod denglin_vgpu_deps
rmmod denglin_vgpu
```

至此，在主机上卸载虚拟机虚拟化环境完成。

注意事项

设备隔离性说明

1. 登临系统软件虚拟机虚拟化共享机制采用空分（非时分），因此每个虚拟设备拥有自己独享的算力，虚拟设备之间的算力是最大粒度进行隔离的。
2. 可以在一张物理登临GPGPU卡上同时创建虚拟设备和MIG设备，分别供给虚拟机和主机使用；虚拟设备只能供给虚拟机使用，不能供给Docker使用。
3. 不支持虚拟机嵌套使用。在虚拟机内无法再嵌套部署和安装登临虚拟机虚拟化软件。
4. 如果虚拟设备执行任务时导致Device Hang，可能会影响该虚拟设备所属的物理登临GPGPU卡无法正常工作。

获取虚拟设备的状态

1. 登临Hamming SDK附带了 `dlsmi` 命令行工具，可以方便地查看系统内登临GPGPU的设备信息，如MIG模式下的MIG设备、内存使用量以及设备使用率等。虚拟设备并不是MIG设备，不会展示为MIG设备。
2. 在Host上，`dlsmi` 命令可以获取到虚拟机内虚拟设备对应cluster的内存使用量。同时，设备整体使用率也会包含每个虚拟设备的使用率。

但是在Guest OS内，只能获取到该Guest上虚拟设备的信息，无法获取到其它虚拟机的虚拟设备信息，也无法获取到Host上物理设备的内存使用量和设备使用率等。
3. 由于Host和Guest是两套独立的操作系统环境，在Host上，`dlsmi` 命令可以获取到Host上正在使用设备的进程信息，但是无法获取到Guest上正在使用设备的进程信息；同理，每个虚拟机只能获取到自己环境内正在使用设备的进程信息。

使用示例

准备好宿主机环境后，就可以创建并使用虚拟设备了。

操作步骤如下：

1. 打开主机上第1张卡的MIG模式，生成UUID，并创建一个拥有1个cluster资源的虚拟设备。

```
(sdk) root@openpc-13-84:/LocalRun/demo# dlsmi -i 0 -mig 1
Enabled MIG Mode for GPU 00000000:01:00.0
All done.
(sdk) root@openpc-13-84:/LocalRun/demo# uuidgen
725c2752-9aa6-4639-926b-46f60d4ad0a9
(sdk) root@openpc-13-84:/LocalRun/demo# echo 725c2752-9aa6-4639-926b-46f60d4ad0a9 > /sys/bus/pci/devices/0000\:01\:00.0/mdev_supported_types/denglin-MIG-1C/create
```

2. 查看生成的虚拟设备信息。

```
(sdk) root@openpc-13-84:/LocalRun/demo# cat /sys/bus/mdev/devices/725c2752-9aa6-4639-926b-46f60d4ad0a9/mdev/mdev_info
parent DLI UUID: 614a22b4-f83b-e954-a151-4386f695c81c
vfunc ID: 0
cluster mask: 0x1
```

3. 将该虚拟设备传递给QEMU使用。

```
(sdk) root@openpc-13-84:/LocalRun/demo# qemu-system-x86_64 -m 16384 -smp 4 -hda /LocalRun/demo/ubuntu1604_1.img -device vfio-pci,syfsdev=/sys/bus/mdev/devices/725c2752-9aa6-4639-926b-46f60d4ad0a9 -enable-kvm -cpu host -net nic -net user,hostfwd=tcp::8888-:22
VNC server running on 127.0.0.1:5900
qemu-system-x86_64: vfio: Cannot reset device 725c2752-9aa6-4639-926b-46f60d4ad0a9, no available reset mechanism.
qemu-system-x86_64: vfio: Cannot reset device 725c2752-9aa6-4639-926b-46f60d4ad0a9, no available reset mechanism.
qemu-system-x86_64: vfio-pci: Cannot read device rom at 725c2752-9aa6-4639-926b-46f60d4ad0a9
Device option ROM contents are probably invalid (check dmesg).
Skip option ROM probe with rombar=0, or load from file with romfile=
```

4. 在虚机中安装登临Hamming SDK，之后执行 `lspci` 命令。

执行成功会显示已存在的登临GPGPU设备的信息。

```
ubuntu1604@ubuntu1604-Standard-PC-i440FX-PIIX-1996:~$ lspci
00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)
00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II]
00:01.1 IDE interface: Intel Corporation 82371SB PIIX3 IDE [Natoma/Triton II]
00:01.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 03)
00:02.0 VGA compatible controller: Device 1234:1111 (rev 02)
00:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 03)
00:04.0 Co-processor: Device 1e27:0002 (rev 01)
```

5. (在虚机中已安装登临Hamming SDK)，执行 `dlsmi` 命令。

执行成功可以看到前面生成的虚拟设备，且该设备只有第一个cluster可用。

```
ubuntu1604@ubuntu1604-Standard-PC-i440FX-PIIX-1996:~$ dlsmi
+-----+-----+
| DL-SMI 11.0 | Driver version 4.0.0 |
+-----+-----+
| GPU Product-Type | Bus-Id | Cluster-Memory-Usage | GPU-Util |
| Fan Temp Perf Pwr:Usage/Cap | Memory-Usage | 0 | 1 | 2 | 3 | MIG M. |
+-----+-----+
| 0 Goldwasser L256 Customized 11103 | 0000:00:04.0 | 104 | 0 | 0 | 0 | Disabled |
| N/A N/A N/A N/A / N/A | 104 / 8192 | | | | | |
+-----+-----+
+-----+-----+
| Processes | Cluster-Memory-Usage |
| GPU TASK | PID Process name | Memory-Usage | 0 | 1 | 2 | 3 |
+-----+-----+
| No running processes found |
+-----+-----+
ubuntu1604@ubuntu1604-Standard-PC-i440FX-PIIX-1996:~$
```

6. 此时虚拟设备可以正常运行CUDA、网络以及图像、视频处理等任务。

```
ubuntu1604@ubuntu1604-Standard-PC-i440FX-PIIX-1996:~/demo/samples$ ./vector_add
[Vector addition of 50000 elements]
Copy input data from the host memory to the dli cuda device
Copy output data from the dli device to the host memory
Test PASSED
ubuntu1604@ubuntu1604-Standard-PC-i440FX-PIIX-1996:~/demo/samples$
```

7. 当虚拟设备不再使用时，关闭QEMU，然后通过 `sys` 文件系统移除该设备。

```
(sdk) root@openpc-13-84:/LocalRun/demo# echo 1 > /sys/bus/mdev/devices/725c2752-9aa6-4639-926b-46f60d4ad0a9/remove
(sdk) root@openpc-13-84:/LocalRun/demo# ls /sys/bus/mdev/devices/
(sdk) root@openpc-13-84:/LocalRun/demo# cat /sys/bus/pci/devices/0000:01:00.0/mdev_supported_types/denglin-MIG-1C/available_instances
4
```