



登临 Hamming V2

GPU P2P 使用说明

DL-DG/SW-052A-03

2024-12-06

Copyright©苏州登临科技有限公司，2019 - 2025，版权所有。

未经苏州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



和其它苏州登临科技有限公司的其它登临科技的图标为苏州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、服务和特性由苏州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。

苏州登临科技有限公司

苏州工业园区扬富路11号南岸新地一期商务楼栋5号1101室，江苏，中国

<http://www.denglin.ai>

email : support@denglin.ai

更新历史

版本	更新描述
03	新增章节 4 RDMA设置 。 增加对SDK加载ko文件的相关描述。
02	增加NPI测试包的相关内容。
01	第一次发布。

目录

目录

1 简介

2 主机兼容性设置验证

3 V2 GPU设置验证

4 RDMA设置

5 带宽测试

6 兼容性评估脚本

登临科技保密材料

1 简介

V2 GPU支持基于PCIe Gen5的Peer-2-Peer通信，两个V2 GPU之间可以通过系统的PCIe fabric tree来完成直接通信，从而避免Host CPU参与引入的开销。

登临GPU间P2P既支持DMA操作，也支持cu kernel之间的互操作，应用程序的编程接口与标准CUDA API基本兼容，同时也推荐一般情况下多卡互联的操作基于登临发布的NCCL库来完成，无需通过基于CUDA API和kernel来操作。

登临发布的NCCL库适配了登临V2的GPU架构，能够更好的兼容V2的架构，发挥V2 GPU P2P通信的性能优势。

2 主机兼容性设置验证

因为V2 GPU的P2P通信是基于主机系统的PCIe fabric结构，所以在使用V2 P2P的功能前，需要对V2 GPU进行一些特定的设置，同时对所在主机需要进行特定的兼容性验证，确保两者可以配合工作。

1. 插卡方式

总体原则：多张V2 GPU尽量插在一个PCIe-Switch下面，如果系统的V2 GPU数量小于等于4张时，插在同一个RC下面；如果V2 GPU数量为8张时，每个RC下面插4张V2 GPU。

由于目前仅支持64G的设备，每张卡都必须是64G的设备。

在同一个PCIe-Switch下的GPU卡，NUMA节点也须相同。可以通过 `dlsmi topo -m` 命令来查看4张卡是否在同一个NUMA节点（8张卡则是4张卡在A NUMA节点，4张卡在B NUMA节点），如下图所示。若不同，则需要联系服务器厂商或者登临技术支持协助重新接线和插卡。

```
(base) (sdk) dlsmi topo -m
GPU0 GPU1 GPU2 GPU3 GPU4 GPU5 GPU6 GPU7 CPU Affinity
GPU0 X PIX PXB PXB SYS SYS SYS SYS 24-31,88-95
GPU1 PIX X PXB PXB SYS SYS SYS SYS 24-31,88-95
GPU2 PXB PXB X PXB SYS SYS SYS SYS 24-31,88-95
GPU3 PXB PXB PXB X SYS SYS SYS SYS 24-31,88-95
GPU4 SYS SYS SYS SYS X PIX PXB PXB 56-63,120-127
GPU5 SYS SYS SYS SYS PIX X PXB PXB 56-63,120-127
GPU6 SYS SYS SYS SYS PXB PXB X PXB 56-63,120-127
GPU7 SYS SYS SYS SYS PXB PXB PXB X 56-63,120-127

Legend:
X = Self
SYS = Connection traversing PCIe as well as the SMP interconnect between NUMA nodes (e.g., QPI/UPI)
NODE = Connection traversing PCIe as well as the interconnect between PCIe Host Bridges within a NUMA node
PHB = Connection traversing PCIe as well as a PCIe Host Bridge (typically the CPU)
PXB = Connection traversing multiple PCIe switches (without traversing the PCIe Host Bridge)
PIX = Connection traversing a single PCIe switch
DL# = Connection traversing a bonded set of # DDLINKS
```

NUMA Affinity
3
3
3
3
7
7
7
7

2. 关闭PCIe ACS 特性

ACS是access control services简称，该特性可以强制Peer-2-Peer的通信上传到PCIe root complex进行访问校验，以防止不可靠的访问。

该特性与V2 GPU P2P访问的需求冲突，为了保证两个V2 GPU之间能够直接通信而不被RC干预或者阻挡，必须确保ACS特性处于关闭状态。

执行以下命令检查机器的ACS是否关闭。

```
sudo lspci -vvv | grep -i acsctl
```

- 如果没有打印输出，表明机器没有ACS相关特性。

- 如果出现如下打印，需要确保每一项都是“-”；若出现“+”，请在BIOS设置关闭ACS。若BIOS不能关闭ACS，请联系服务器厂商或者登临技术支持工程师。

```
ACSCtl: SrcValid- TransBlk- ReqRedir- CmpltRedir- UpstreamFwd- EgressCtrl-
DirectTrans-
```

3. 关闭IOMMU

在ACS关闭的情况下，所有的设备同属一个IOMMU group，IOMMU设备隔离安全性受到极大限制。同时，开启IOMMU会降低设备访问内存性能，所以建议关闭IOMMU或设置IOMMU为Passthrough模式。

可以在机器启动时进入BIOS设置进行修改，修改方式根据主板菜单而定。在x86机器上，系统启动时会在dmesg里打印相关信息，若开启了IOMMU，会有类似如下打印。

```
dmesg | grep IOMMU
[ 0.317371] pci 0000:00:00.2: AMD-Vi: IOMMU performance counters supported
[ 0.319891] pci 0000:00:00.2: AMD-Vi: Found IOMMU cap 0x40
[ 0.650389] perf/amd_iommu: Detected AMD IOMMU #0 (2 banks, 4 counters/bank)
```

3 V2 GPU设置验证

- 为了支持 V2 GPU P2P功能，请确保V2 GPU的硬件Firmware版本在V1.0.2以上，并且需要确保V2 GPU的BAR2的size和V2 GPU的DDR size大小一致。BAR2 size可通过以下步骤查看。

- 查看所有V2 GPU的 BDF，如下图所示。

```
lspci | grep 1e27
```

```
04:00.0 Co-processor: Device 1e27:0003
05:00.0 Co-processor: Device 1e27:0003
43:00.0 Co-processor: Device 1e27:0003
44:00.0 Co-processor: Device 1e27:0003
```

- 根据BDF依次查看所有V2 GPU的BAR2 size，如下图所示。

```
sudo lspci -s 04:00.0 -vv
```

```
04:00.0 Co-processor: Device 1e27:0003
Subsystem: Device 1e27:0209
Control: I/O- Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx+
Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx-
Latency: 0, Cache Line Size: 64 bytes
Interrupt: pin A routed to IRQ 200
Region 0: Memory at f4000000 (32-bit, non-prefetchable) [size=16M]
Region 1: Memory at f5000000 (32-bit, non-prefetchable) [size=1M]
Region 2: Memory at 4f00000000 (64-bit, prefetchable) [size=64G]
Region 4: Memory at 5000000000 (64-bit, prefetchable) [size=256M]
```

3. 如果V2 GPU的BAR2的size和V2 GPU的DDR size不同，需要通过V2 SDK中的dlsmi命令修改BAR，如下所示，然后重启机器。

```
sudo dlsmi -pbm 1
```

4. 如果升级失败，则需要检查板卡显存是否均为64G，固件Firmware版本是否相同且大于等于V1.0.2。

2. 执行以下命令加载驱动，并启动Hamming SDK环境。

- tag为202411020057及之后版本的Hamming SDK加载驱动，执行如下命令：

```
sudo insmod <sdk_install_path>/kernel/denglin.ko
source <sdk_install_path>/env.sh
```

- tag为202411020057之前版本的Hamming SDK加载驱动时，需要添加参数，示例如下：

```
sudo insmod <sdk_install_path>/kernel/denglin.ko host_su_access=1
p2p_cross_nodes=0
source <sdk_install_path>/env.sh
```

其中 `<sdk_install_path>` 为Hamming SDK安装路径，`host_su_access=1` 保证在kernel中可以直接访问host memory，`p2p_cross_nodes=0` 保证GPU可以按照host bridge进行分组，从而可以确保8卡设备kernel中可以直接访问host memory，这是NCCL库所需要的重要特性。

1. Debian系操作系统（例如ubuntu）可以在 `/etc/modprobe.d/` 目录下创建 `denglin.conf` 配置文件，并添加如下内容，使得操作系统重启后也能使用相同参数自动加载。

```
options denglin host_su_access=1
options denglin p2p_cross_nodes=0
```

2. 执行以下命令来检查驱动是否指定了对应参数。

```
cat /sys/module/denglin/parameters/host_su_access
cat /sys/module/denglin/parameters/p2p_cross_nodes
```

- 若输出 `host_su_access=1`，`p2p_cross_nodes=0`，说明参数已指定。
- 否则，说明驱动加载时没有指定参数（机器重启了或者加载驱动时参数字符错误等），需要执行以下命令重新加载驱动：

```
sudo rmmod denglin && sudo insmod <sdk_install_path>/kernel/denglin.ko
host_su_access=1 p2p_cross_nodes=0
```

3. 执行以下命令查看不同device之间的P2P状态。

```
sudo dlsmi topo -p2p rw
```

- 状态为“OK”表示P2P功能是正常的。


```

root@inspur1:/LocalRun/sdk/kernel# dlsmi topo -p2p rw
      GPU0  GPU1  GPU2  GPU3  GPU4  GPU5  GPU6  GPU7
GPU0  X      OK      OK      OK      GNS   GNS   GNS   GNS
GPU1  OK      X      OK      OK      GNS   GNS   GNS   GNS
GPU2  OK      OK      X      OK      GNS   GNS   GNS   GNS
GPU3  OK      OK      OK      X      GNS   GNS   GNS   GNS
GPU4  GNS     GNS     GNS     GNS     X      OK      OK      OK
GPU5  GNS     GNS     GNS     GNS     OK      X      OK      OK
GPU6  GNS     GNS     GNS     GNS     OK      OK      X      OK
GPU7  GNS     GNS     GNS     GNS     OK      OK      OK      X

Legend:
X      = Self
OK     = Status OK
GNS    = Chipset not supported
GNS    = GPU not supported
TNS    = Topology not supported
DIS    = Disabled by regkey
NS     = Not supported
U      = Unknown

```

- “GNS”表示P2P不支持。不同的GPU处在不同的host bridge下会导致P2P不通。

4 RDMA设置

为了正常使用V2 GPU RDMA 功能，我们还需要加载 `denglin-peermem.ko`，命令如下：

```
sudo insmod <sdk_install_path>/kernel/denglin-peermem.ko
```

由于硬件限制，需要设置RINC的 `MaxPayload` 为128B，`MaxReadReq` 为128B。设置完成后按照如下步骤进行V2 GPU RDMA功能的验证。

1. 下载perftest（<https://github.com/denglin-github/perftest>），然后编译（确保主机环境已经安装好SDK和IB相关的环境）。
2. 利用`ib_send`、`ib_write`（可利用IB环境安装自带的，也可利用perftest编译的bin）进行host主机的连接验证，保证RNIC是能够互相通信的。
3. 利用perftest编译的`ib_send`、`ib_write`进行V2 GPU RDMA的验证。

perftest编译和使用方法可参考 <https://github.com/denglin-github/perftest> 的README文件中“4.GPU Drrrect usage”内容。

保证V2 GPU RDMA 设置正确后，测试NCCL多机时，如果需要使用到RNIC，需要添加环境变量`NCCL_IB_HCA`去指定使用的RDMA interfaces，比如 `export NCCL_IB_HCA = m1x5_0:1,m1x5_1:1` 代表使用 `m1x5_0` 的port1 和 `m1x5_1` 的port1。

5 带宽测试

完成前面的配置后，需要下载NPI包并解压，进一步通过 `<npi_install_path>/tools/p2p_bandwidth` 进行P2P功能验证和P2P带宽性能测试。

说明：

NPI包请联系登临技术支持获取。

1. 通过以下命令检查不同GPU间的P2P功能是否正确。

```
./p2p_bandwidth --mode=2 --gpu0=<a> --gpu1=<b>
./p2p_bandwidth --mode=3 --gpu0=<a> --gpu1=<b>
```

其中 a 和 b 为不同GPU的编号。

2. 通过以下命令测试不同device间通过DMA进行P2P的带宽。

```
./p2p_bandwidth --mode=0 --gpu0=<a> --gpu1=<b>
```

3. 通过以下命令测试不同GPU间通过kernel进行P2P的带宽。

```
./p2p_bandwidth --mode=1 --gpu0=0 --gpu1=1
```

6 兼容性评估脚本

登临提供了一键式抓取系统兼容性数据的脚本，该脚本存放在 `<npi_install_path>/tools/CompatTools`。

脚本使用方法如下：

```
python3 compat.py --ko_args="host_su_access=1 p2p_cross_nodes=0" # root用户
sudo ./compat.sh $SDK_DIR "--ko_args='host_su_access=1 p2p_cross_nodes=0'" # 非root用户
```

脚本执行完后，会生成两个文件，分别以log和txt后缀结尾；log文件是脚本运行的完整日志，txt文件是输出的兼容性报告。

关于脚本更详细的使用说明请参考 `<npi_install_path>/tools/CompatTools` 目录下的README文件。

另外，脚本执行后会卸载驱动ko，需要按照之前参数重新insmod。