



登临 HammingTM V2

NPI 工具集使用说明

DL-DG/SW-062A-05

2025-01-06

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
05	新增dIJPEG和dIVID的编解码工具。
04	新增tuFlops工具简介以及使用方法。
03	工具版本升级。
02	增加suFlops工具简介以及使用方法。
01	第一次发布。

目录

目录

1 简介

2 NPI 使用说明

2.1 power_v2

2.1.1 简介

2.1.2 使用方法

2.2 bandwidthTest

2.2.1 简介

2.2.2 使用方法

2.3 p2pBandwidthLatencyTest

2.3.1 简介

2.3.2 使用方法

2.4 p2p_bandwidth

2.4.1 简介

2.4.2 使用方法

2.5 compat.py

2.5.1 简介

2.5.2 测试准备

确认板卡信息

确认服务器状态

2.5.3 SDK版本

2.5.4 使用方法

分析运行结果

2.6 suFLOPS

2.6.1 简介

2.6.2 使用说明

2.6.3 使用方法

2.7 tuFlops

2.7.1 简介

2.7.2 使用方法

2.8 dljpeg_decode

2.8.1 简介

2.8.2 使用方法

2.9 dljpeg_encode

2.9.1 简介

2.9.2 使用方法

2.10 dlvid_decode

2.10.1 简介

2.10.2 使用方法

2.11 dlvid_encode

2.11.1 简介

2.11.2 使用方法

1 简介

NPI工具集是登临研发的、在新产品导入过程中用来评估硬件环境正确性及硬件关键指标的一组测试工具。NPI工具位于 `sdk/tools` 目录，具体由以下工具组成：

- **power_v2**：用来评估登临GPU的最大功耗，可以测量单卡或多卡的最大功耗。
- **bandwidthTest**：该工具是NVIDIA标准的本地（Local）带宽测试程序。能够测量设备到设备的复制带宽、主机到设备的可分页和页锁定内存的复制带宽，以及设备到主机的可分页和页锁定内存的复制带宽。
- **p2pBandwidthLatencyTest**：该工具是NVIDIA标准的点对点（P2P）带宽测试程序。它可以测试多个GPU在使用P2P和不使用P2P时的带宽和延迟。
- **p2p_bandwidth**：该工具在NVIDIA标准的点对点（P2P）带宽测试程序的基础上进行了一些裁剪和定制，目的是更方便地进行多GPU硬件调试和问题定位。该工具不但能够测试P2P带宽，还能验证任意两个GPU之间数据传输的正确性。
- **compat.py**：该工具是一个检测新服务器与登临GPU兼容性的脚本程序。它会收集主机信息、收集每个GPU的硬件信息、收集各种带宽（H2D、D2H、D2D、P2P）数据并生成报告。
- **suFlops**：该工具可以用来评估登临GPU SU的峰值算力，它会收集浮点和定点的峰值算力。
- **tuFlops**：该工具可以用来评估登临GPU TU的峰值算力。
- **dljpeg_decode**：JPEG图像解码性能测试工具。
- **dljpeg_encode**：JPEG图像编码性能测试工具。
- **dlvid_decode**：H264、H265视频解码性能测试工具。
- **dlvid_encode**：H264、H265视频编码性能测试工具。

说明：

每个NPI工具的使用方法，请参考下文章节 [3 NPI使用说明](#)，或参考 `sdk/tools` 下README.md文件。

2 NPI 使用说明

2.1 power_v2

2.1.1 简介

登临power_v2工具用来测量单卡或多卡的最大功耗。

该工具随登临Hamming SDK一起发布，位于 `sdk/tools/power_v2`。

测量多卡功耗时，需要同时启动多个程序并传递不同的gpu_id。当功耗测量程序启动后，可以通过登临提供的dlsmi工具观察功耗变化，如下图所示：

Processes				Cluster-Memory-Usage				GPU-Memory
GPU	GI	PID	TASK Process name	0	1	2	3	Usage
0		10210	7 ./power_v2	0	0	0	0	0
Mon Nov 27 13:48:11 2023								
DL-SMI 11.0				Driver version 0.83.1				
GPU Product-Type				Bus-Id	Cluster-Memory-Usage			
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	0	1	2	3 GPU-Util
								MIG M.
0				0000:01:00.0				100%
0	75C	P4	153W / 150W	465 / 32768	116	116	116	116 Disabled

Power_v2工具在运行过程中每隔30秒会打印如下信息：

- 平均动态功耗
- 芯片温度
- TU利用率（TU：Tensor Unit）
- TE利用率（TE：Tensor Engine，TU内部的主要运算模块）
- CU利用率（CU：Compute Unit）

power_v2程序默认配置为CU和TU同时运行。CU command 和 TU command 分别被推入两个并行的command queues 队列，同时触发，同时运行。

Case配置参数让CU和TU各自运行每轮大致相同的时间，客户可通过命令行配置 `--work_loop=<n>` 来控制跑多少轮，这个命令行选项会相应地配置CU和TU并行运行大致等同的时间。最终两个队列都结束后，测试的核心运算部分全部完成，此时程序打印 CU/TU command 执行完毕的信息和核心运算部分所用的时长。如下图所示：

[illegible]

2.1.2 使用方法

```
./power_v2 [gpu_id] [work_loop]
```

1. 安装sdk, 并 `source sdk/env.sh`
2. `cd sdk/tools`
3. `./power_v2 --gpu_id=0 --work_loop=1`
4. 在新的命令行窗口, 执行`dlsmi`命令, 观察功耗变化。

其中参数含义如下：

- `--gpu_id` : 该参数表示被测量功耗卡的index, 此index就是设备详细信息里面标注的序号, 必须由用户指定。
- `--work_loop` : 该参数表示功耗测量执行次数, 必须由用户指定; 执行次数增加, 则功耗测量持续时间也会随着增加。

work loop参数和测量持续时间按下列公式计算：

- $\text{work_loop} = \text{用户期望功耗持续时间 (秒)} / \text{单次功耗测量时间 (秒)}$
- 单次功耗测量时间 (秒) = `--work loop=1` 时程序执行时长

比如：希望功耗测量持续30分钟，可以这样设置work loop：

1. 把work_loop设置为1，执行一次功耗测量，记录单次功耗测量时间（38秒）。
2. 需要功耗测量持续30分钟， $\text{work_loop} = (30 \times 60) / 38 = 47.36$ ，向上取整 $\text{work_loop} = 48$ 。
3. 执行 `./power_v2 --work_loop=48`，功耗测量持续时间就会达到30分钟左右。

2.2 bandwidthTest

2.2.1 简介

登临bandwidthTest工具是基于CUDA的带宽测试程序。用来测量以下带宽：

- 设备到设备 (Device to Device) 的数据传输带宽。
- 主机到设备 (Host to Device) 的可分页和页锁定内存的数据传输带宽。
- 设备到主机 (Device to Host) 的可分页和页锁定内存的数据传输带宽。

其中主机到设备和设备到主机的数据传输带宽主要用来衡量PCIE的实测带宽，设备到设备的数据传输带宽用来衡量DDR的实测带宽。

2.2.2 使用方法

./bandwidthTest [OPTION]

参数含义如下：

```
Options:
--help  Display this help menu
--csv   Print results as a CSV
--device=[device] Specify the device to be used
        all - compute cumulative bandwidth on all the devices
        0,1,2,...,n - Specify any particular device to be used
--memory=[MEMMODE] Specify which memory mode to use
        pageable - pageable memory
        pinned - non-pageable system memory
--mode=[MODE] Specify the mode to use
        quick - performs a quick measurement
        range - measures a user-specified range of values
        shmoo - performs an intense shmoo of a large range of values
--htod   Measure host to device transfers
--dtoh   Measure device to host transfers
--dtod   Measure device to device transfers
--dtod_sm Measure device to device transfers using kernel mode for testing
--wc     Allocate pinned memory as write-combined
--cputiming Force CPU-based timing always Range mode options
--start=[SIZE] Starting transfer size in bytes
--end=[SIZE] Ending transfer size in bytes
--increment=[SIZE] Increment size in bytes
```

```
[CUDA Bandwidth Test] - Starting...
Running on...

Device 0: Goldwasser II XL512
Quick Mode

Host to Device Bandwidth, 1 Device(s)
PINNED Memory Transfers
Transfer Size (Bytes)      Bandwidth(GB/s)
32000000                  27.8

Device to Host Bandwidth, 1 Device(s)
PINNED Memory Transfers
Transfer Size (Bytes)      Bandwidth(GB/s)
32000000                  26.7

Device to Device Bandwidth, 1 Device(s)
PINNED Memory Transfers
Transfer Size (Bytes)      Bandwidth(GB/s)
32000000                  142.6

Result = PASS
```


2.3 p2pBandwidthLatencyTest

2.3.1 简介

登临p2pBandwidthLatencyTest工具主要用于测试多卡间点对点（Peer-to-Peer，P2P）的带宽和延迟。

2.3.2 使用方法

`./p2pBandwidthLatencyTest [OPTION]`

参数含义如下：

Options:

--help Display this help menu
 --p2p_read Use P2P reads for data transfers between GPU pairs and show corresponding results; Default used is P2P write operation.
 --sm_copy Use SM initiated p2p transfers instead of Copy Engine; Default using Copy Engine.
 --numElems=<NUM_OF_INT_ELEMS> Number of integer elements to be used in p2p copy; Default 64M.

```
[P2P (Peer-to-Peer) GPU Bandwidth Latency Test]
p2p_mechanism=[CE], p2p_method=[P2P_WRITE], numElems=67108864
Device: 0, Goldwasser II ML512, pciBusID: 1, pciDeviceID: 0, pciDomainID: 0
Device: 1, Goldwasser II ML512, pciBusID: 5, pciDeviceID: 0, pciDomainID: 0
Device=0 CAN Access Peer Device=1
Device=1 CAN Access Peer Device=0

***NOTE: In case a device doesn't have P2P access to other one, it falls back to normal memcpy procedure.
So you can see lesser Bandwidth (GB/s) and unstable Latency (us) in those cases.

P2P Connectivity Matrix
  D\D    0    1
  0      1    1
  1      1    1
Unidirectional P2P-Enabled Bandwidth (P2P Writes) Matrix (GB/s)
  D\D    0    1
  0 144.71  1.02
  1  0.77 143.26
Bidirectional P2P-Enabled Bandwidth Matrix (GB/s)
  D\D    0    1
  0 139.14  1.54
  1  1.54 141.50
P2P-Enabled Latency (P2P Writes) Matrix (us)
  GPU    0    1
  0    2.62  2.28
  1    7.45  7.03
```

2.4 p2p_bandwidth

2.4.1 简介

登临p2p_bandwidth工具扩展了点点对点（P2P）带宽测试程序（p2pBandwidthLatencyTest），目的是更方便地进行硬件调试和测试。p2p_bandwidth工具不但能够测试P2P带宽，还能验证任意两个GPU之间数据传输的正确性。

2.4.2 使用方法

`./p2p_bandwidth [OPTION]`

参数含义如下：

```
Options:
--help | -h <optional> print help message;
--mode <must> The running mode of the test, default: 0;
    0: means dma_mode, use Copy Engine;
    1: means kernel_mode, use SM/CE Engine;
    2: means dma_mode with data correctness check;
    3: means kernel_mode with data correctness check;
--gpu0 <must> gpu0 id, default: 0;
--gpu1 <must> gpu1 id, default: 1;
--msize_mb <optional> data transfer unit size(MB), default: 128(MB);
--repeat0 <optional> data transfer loop, default: 32;
```

```
=> [Start...] P2P (Peer-to-Peer) GPU Bandwidth Test!
Usage: ./p2p_bandwidth --mode=0 --gpu0=0 --gpu1=1 --msize_mb=32 --repeat0=32
=> TestConfig: gpu0=0, gpu1=1, msize_mb=128(MB), repeat0=32, repeat1=100000; mode=dma;
=> Total number of Device=2
-- Device: [0], Name: Goldwasser II XL512 (UUID: 61b0e16d-8c3a-c011-26bd-618ed9624810); pciBusID: 1, pciDeviceID: 0, pciDomainID:0
-- Device: [1], Name: Goldwasser II XL512 (UUID: 7b39168a-6495-ddf4-ff94-eee0854d42e0); pciBusID: 5, pciDeviceID: 0, pciDomainID:0
=> Test Used Device ID: [0, 1]; Number of devices installed in the system: 2
=> [P2P] Bandwidth Test Between Device[0](Goldwasser II XL512) --> Device[1](Goldwasser II XL512)
=> Device[0] (CAN) Access Peer Device[1]
=> Device[1] (CAN) Access Peer Device[0]
-- Total Transferred Data(write-peer): 4294967296 (byte), [4.29GB]; CostTime: 4194.45 (ms)
-- Total Transferred Data(read-peer): 4294967296 (byte), [4.29GB]; CostTime: 683.49 (ms)
-- Total Transferred Data(readwrite-peer): 8589934592 (byte), [8.59GB]; CostTime: 4643.25 (ms)

=====
*** [Report] BandWidth(write-peer)    Between GPU[0] and GPU[1] IS:   1.02 (GB/s)
*** [Report] BandWidth(read-peer)    Between GPU[0] and GPU[1] IS:   6.28 (GB/s)
*** [Report] BandWidth(readwrite-peer) Between GPU[0] and GPU[1] IS:  1.85 (GB/s)
=====

=> [End...] P2P (Peer-to-Peer) GPU Bandwidth Test Pass!
```

2.5 compat.py

2.5.1 简介

登临compat.py是一个检测新服务器与登临GPU兼容性的脚本程序。它会收集主机信息、收集每个GPU的硬件信息、收集各种带宽（H2D、D2H、D2D、P2P）数据并生成报告。

2.5.2 测试准备

确认板卡信息

Step1：确认固件FW版本。

Step2：确认板卡Bar2 size。板卡Bar2 size需设置为GPU板卡显存容量（如板卡显存64GB，那么Bar2需设置为64GB）。

确认服务器状态

- 关闭ACS功能

BIOS中关闭ACS功能：与服务器厂商确认BIOS中关闭ACS的方法。以intel平台为例，ACS BIOS中关闭方式如下：

1. 重启操作系统开机时按del进入BIOS关闭ACS功能，不关VT-d只关闭ACS功能，具体路径：**Path:**
Advanced -> Chipset Configuration -> North Bridge -> I/O Configuration -> Intel VT for Directed I/O (VT-d) -> ACS Control -> Enable / Disable
2. 通过系统命令 `lspci -vvv | grep -i ACSctl` 查询ACS是否全部关闭完全，如输出结果为：SrcValid+ 表示该功能未关闭，SrcValid- 表示已关闭。
3. 若ACS功能仍未关闭，与服务器厂商确认通过命令行关闭ACS的方式。

- 确认服务器PCIE拓扑

建议PCIE拓扑为balance模式，即GPU卡平均分配在两个CPU下可参考以下方式进行检查，使用 `dlsmi topo -m` 命令查看：

```
dlml: User library version (1.14.4) different but compatible with Kernel driver version (1.14.1)
  GPU0  GPU1  GPU2  GPU3  GPU4  GPU5  GPU6  GPU7  CPU Affinity  NUMA Affinity
GPU0  X    PIX  NODE  NODE  SYS  SYS  SYS  SYS  0-7,16-23  0
GPU1  PIX  X    NODE  NODE  SYS  SYS  SYS  SYS  0-7,16-23  0
GPU2  NODE  NODE  X    PIX  SYS  SYS  SYS  SYS  0-7,16-23  0
GPU3  NODE  NODE  PIX  X    SYS  SYS  SYS  SYS  0-7,16-23  0
GPU4  SYS  SYS  SYS  SYS  X    NODE  NODE  NODE  8-15,24-31  1
GPU5  SYS  SYS  SYS  SYS  NODE  X    PIX  PIX  8-15,24-31  1
GPU6  SYS  SYS  SYS  SYS  NODE  PIX  X    PIX  8-15,24-31  1
GPU7  SYS  SYS  SYS  SYS  NODE  PIX  PIX  X    8-15,24-31  1

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
PIX  = 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 # DLinks
```

如上图所示，GPU0~3在NUMA0（也就是物理CPU0）下，GPU4~7在NUMA1（也就是物理CPU1）；GPU0~3与4~7的连接方式为SYS，即通过CPU互连。

2.5.3 SDK版本

须使用日期Tag为202404102135及之后的Hamming V2 SDK版本。

2.5.4 使用方法

compat.py 脚本使用示例如下：

```
cd $SDK_DIR/bin/                # SDK_DIR为当前使用的SDK根目录
cp $SDK_DIR/bin/dl-modprobe /usr/bin/
cd $SDK_DIR/
source env.sh
cd $SDK_DIR/tools/CompatTools
python3 compat.py
```

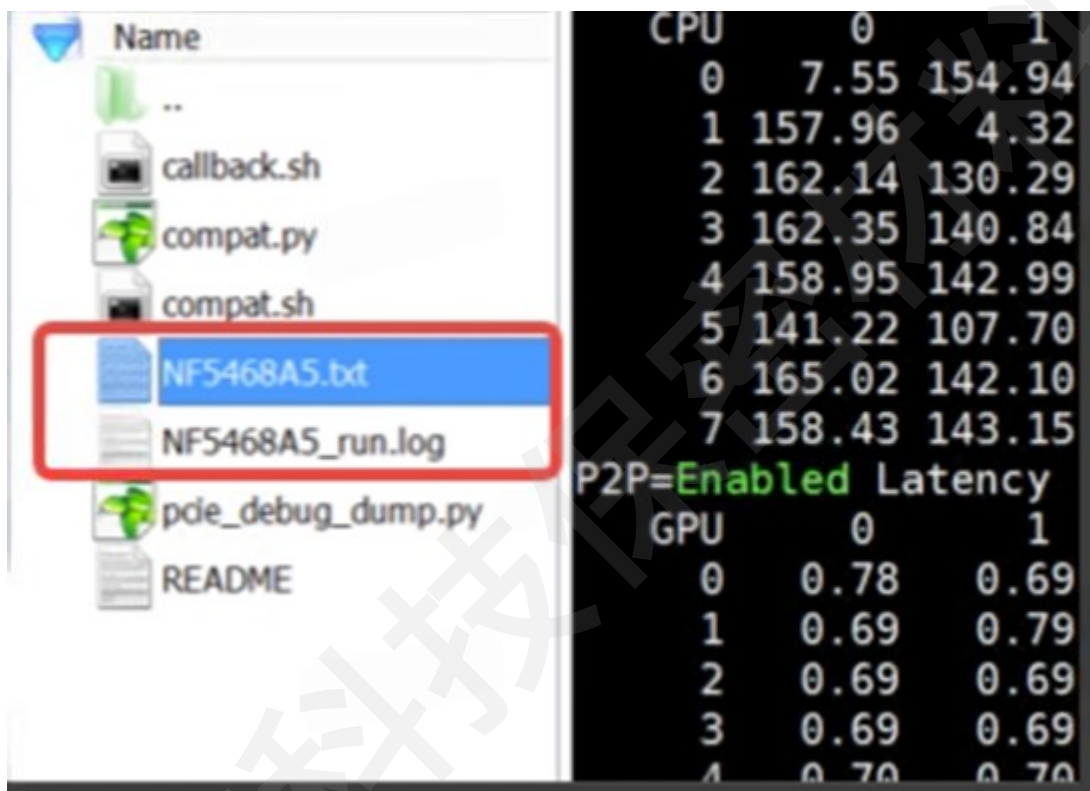
说明：

- 关于compat的具体运行说明，参考CompatTools目录下的README文件。
- compat需要在Root账户下运行，非Root账户参考README文件。
- 若远端P2P不通，可以加一个分组参数，命令如下：`python3 compat.py --ko_args max_p2p_group=1`。

分析运行结果

运行结果位于 `$SDK_DIR/tools/CompatTools/`，结果文件名称为“服务器型号.txt”与“服务器型号.log”。

以NF5468A5服务器为例：



结果数据主要关注“p2p_bw”与“pcie_bw”，该数据位于“.txt”文件。文件内容参考如下：

pcie_bw：


```
'pcie_bw': {'mem_on_node_0': {0: {'d2d': '122.436',
                                   'd2h': '38.1863',
                                   'filld': '29.6277',
                                   'fillh': '22.7017',
                                   'h2d': '30.7846'},
                                1: {'d2d': '122.404',
                                   'd2h': '37.9558',
                                   'filld': '28.0024',
                                   'fillh': '23.6115',
                                   'h2d': '30.5087'},
                                2: {'d2d': '122.223',
                                   'd2h': '38.9652',
                                   'filld': '28.2089',
                                   'fillh': '23.6403',
                                   'h2d': '30.7815'},
                                3: {'d2d': '122.735',
                                   'd2h': '37.9759',
                                   'filld': '28.2095',
                                   'fillh': '23.6767',
                                   'h2d': '30.6668'},
                                4: {'d2d': '122.229',
                                   'd2h': '38.9523',
                                   'filld': '28.2089',
                                   'fillh': '23.6403',
                                   'h2d': '30.7815'}}
```

p2p_bw : DMA MODE

```
1 DMA MODE:[P2P (Peer-to-Peer) GPU Bandwidth Latency Test]
2 p2p_mechanism=[CE], p2p_method=[P2P_READ], numElems=6/108864
3 Device: 0, Goldwasser II XL512, pciBusID: 1b, pciDeviceID: 0, pciDomainID: 0
4 Device: 1, Goldwasser II XL512, pciBusID: 1c, pciDeviceID: 0, pciDomainID: 0
```

```
5 Unidirectional P2P=Enabled Bandwidth (P2P Writes) Matrix (GB/s)
6   D\D   0   1   2   3   4   5   6   7
7   0 132.74 35.54 35.54 35.54 18.05 19.18 19.24 19.23
8   1 35.50 133.84 35.49 35.49 18.14 19.19 19.22 19.24
9   2 35.49 35.52 133.76 35.49 18.40 19.23 19.25 19.23
0   3 35.53 35.53 35.53 134.36 18.35 19.23 19.24 19.25
1   4 19.04 19.01 19.04 19.04 133.31 35.52 35.51 35.51
2   5 18.90 19.04 19.01 19.10 35.49 133.11 35.48 35.52
3   6 19.01 19.08 19.10 19.12 35.54 35.54 133.94 35.53
4   7 19.07 19.10 19.13 19.14 35.51 35.50 35.51 134.34
5 Unidirectional P2P=Enabled Bandwidth (P2P Reads) Matrix (GB/s),
6   D\D   0   1   2   3   4   5   6   7
7   0 133.93 36.10 36.11 36.10 18.90 19.59 19.61 19.61
8   1 35.95 133.10 36.07 36.09 18.91 19.61 19.63 19.59
9   2 35.97 36.06 134.32 36.22 18.81 19.59 19.63 19.67
0   3 36.01 36.21 36.30 132.65 19.06 19.58 19.54 19.56
1   4 19.80 19.80 19.81 19.85 133.63 36.04 36.04 35.95
2   5 19.75 19.79 19.77 19.80 36.04 132.14 36.21 36.04
3   6 19.77 19.80 19.85 19.82 36.36 36.24 133.03 36.28
4   7 19.76 19.81 19.79 19.80 35.97 35.91 36.28 133.31
```

p2p_bw : KERNEL MODE

```
NOTE: The CUDA Samples are not meant for performance measurements.
KERNEL MODE:...[P2P (Peer-to-Peer) GPU Bandwidth Latency Test]
p2p_mechanism=[SM], p2p_method=[P2P_READ], numElems=67108864
```

2.6 suFLOPS

2.6.1 简介

登临suFLOPS工具可以用来评估登临GPU SU的峰值算力，它会收集浮点和定点的峰值算力。

2.6.2 使用说明

在命令行中输入./suFLOPS 命令，工具会以不同的计算强度扫描GPU的算力峰值，算力会随着计算强度的增大而增大，直到硬件达到最大算力，此时得到的就是峰值算力。默认情况下会输出FP16的最大算力值的情况，如下图所示。

```
----- CSV data -----
Experiment ID, Half precision ops,,,,
Compute iters, Hflops/byte, ex.time, GFLOPS, GB/sec
1024, 1024.500, 35.31, 3894.32, 3.80
```

当需要输出全部的计算和精度的情况，需要加上 --all 参数，即执行命令 ./suFLOPS --all，工具最后会打印完整的计算结果，如下图所示。

```
----- CSV data -----
Experiment ID, Single Precision ops,,,, Half precision ops,,,, Integer operations,,,,
Compute iters, Fflops/byte, ex.time, GFLOPS, GB/sec, Hflops/byte, ex.time, GFLOPS, GB/sec, Iops/byte, ex.time, GLOPS, GB/sec
0, 0.250, 0.99, 33.74, 134.95, 0.500, 1.00, 66.78, 133.55, 0.250, 0.97, 34.55, 138.20
1, 0.750, 1.00, 100.76, 134.34, 1.500, 1.01, 198.54, 132.36, 0.750, 0.95, 106.18, 141.57
2, 1.250, 1.00, 167.66, 134.13, 2.500, 1.03, 327.00, 130.80, 1.250, 0.98, 170.37, 136.30
3, 1.750, 1.00, 234.09, 133.76, 3.500, 0.95, 492.20, 140.63, 1.750, 0.97, 243.11, 138.92
4, 2.250, 0.98, 308.96, 137.32, 4.500, 0.97, 622.05, 138.23, 2.250, 0.97, 310.20, 137.87
5, 2.750, 0.96, 384.81, 139.93, 5.500, 0.96, 767.80, 139.60, 2.750, 0.98, 378.10, 137.49
6, 3.250, 0.96, 455.27, 140.08, 6.500, 0.99, 884.86, 136.13, 3.250, 0.98, 445.12, 136.96
7, 3.750, 1.04, 483.40, 128.91, 7.500, 0.94, 1073.88, 143.18, 3.750, 0.97, 518.19, 138.18
8, 4.250, 0.98, 581.49, 136.82, 8.500, 0.96, 1192.99, 140.35, 4.250, 0.97, 587.11, 138.14
9, 4.750, 0.97, 659.21, 138.78, 9.500, 0.97, 1308.70, 137.76, 4.750, 0.99, 646.31, 136.07
10, 5.250, 0.99, 708.33, 134.92, 10.500, 0.96, 1470.19, 140.02, 5.250, 0.96, 730.57, 139.16
11, 5.750, 1.00, 768.35, 133.63, 11.500, 1.01, 1532.18, 133.23, 5.750, 1.01, 765.90, 133.20
12, 6.250, 0.98, 852.79, 136.45, 12.500, 1.00, 1682.04, 134.56, 6.250, 0.97, 861.40, 137.82
13, 6.750, 0.97, 937.66, 138.91, 13.500, 1.00, 1811.53, 134.19, 6.750, 1.00, 904.19, 133.95
14, 7.250, 0.99, 985.48, 135.93, 14.500, 0.98, 1990.35, 137.27, 7.250, 1.07, 913.11, 125.95
15, 7.750, 1.04, 1000.26, 129.07, 15.500, 0.98, 2125.03, 137.10, 7.750, 1.03, 1010.38, 130.37
16, 8.250, 1.03, 1076.19, 130.45, 16.500, 0.99, 2246.47, 136.15, 8.250, 1.02, 1088.26, 131.91
17, 8.750, 1.00, 1180.28, 134.89, 17.500, 1.01, 2334.31, 133.39, 8.750, 0.99, 1184.95, 135.42
18, 9.250, 1.00, 1242.64, 134.34, 18.500, 1.00, 2473.80, 133.72, 9.250, 0.97, 1281.64, 138.56
20, 10.250, 1.03, 1335.47, 130.29, 20.500, 1.02, 2697.44, 131.58, 10.250, 1.02, 1352.96, 132.00
22, 11.250, 0.90, 1676.00, 148.98, 22.500, 0.90, 3363.53, 149.49, 11.250, 0.90, 1683.55, 149.65
24, 12.250, 0.97, 1698.06, 138.62, 24.500, 0.97, 3388.56, 138.31, 12.250, 0.96, 1706.39, 139.30
28, 14.250, 1.10, 1731.91, 121.54, 28.500, 1.11, 3449.50, 121.03, 14.250, 1.10, 1740.95, 122.17
32, 16.250, 1.24, 1757.26, 108.14, 32.500, 1.25, 3488.41, 107.34, 16.250, 1.24, 1765.84, 108.67
40, 20.250, 1.51, 1797.43, 88.76, 40.500, 1.51, 3590.43, 88.65, 20.250, 1.51, 1799.57, 88.87
48, 24.250, 1.78, 1826.89, 75.34, 48.500, 1.78, 3649.34, 75.24, 24.250, 1.78, 1832.40, 75.56
56, 28.250, 2.05, 1848.24, 65.42, 56.500, 2.06, 3682.02, 65.17, 28.250, 2.04, 1855.08, 65.67
64, 32.250, 2.34, 1852.10, 57.43, 64.500, 2.34, 3696.96, 57.32, 32.250, 2.32, 1869.41, 57.97
80, 40.250, 2.87, 1882.66, 46.77, 80.500, 2.89, 3740.11, 46.46, 40.250, 2.86, 1887.59, 46.90
96, 48.250, 3.41, 1899.12, 39.36, 96.500, 3.43, 3780.04, 39.17, 48.250, 3.41, 1900.43, 39.39
128, 64.250, 4.49, 1920.22, 29.89, 128.500, 4.51, 3827.55, 29.79, 64.250, 4.49, 1922.23, 29.92
192, 96.250, 6.90, 1871.50, 19.44, 192.500, 6.75, 3827.48, 19.88, 96.250, 6.71, 1925.13, 20.00
256, 128.250, 9.08, 1896.15, 14.78, 256.500, 8.93, 3854.28, 15.03, 128.250, 8.91, 1932.48, 15.07
512, 256.250, 18.00, 1910.30, 7.45, 512.500, 17.66, 3895.31, 7.60, 256.250, 17.59, 1954.89, 7.63
1024, 512.250, 35.93, 1913.51, 3.74, 1024.500, 35.23, 3902.92, 3.81, 512.250, 34.92, 1968.78, 3.84
```

分析测试结果：

请观察程序输出的表格，标红的一列就是不同计算强度下的实测算力值。此列最大的算力值就是硬件的实测最大算力。通常这个实测值达到理论峰值算力的90%以上，就说明硬件的峰值算力符合硬件设计标准。

2.6.3 使用方法

./suFLOPS [OPTION]

参数含义如下：

```
Options:
--help           Display this help menu
--device=[0-n]   <optional> Specify the device device to be used. default: 0
--all            Print full result about all precision
```

```
----- Device specifications -----
Device: Goldwasser II XL512
CUDA driver version: 11.70
GPU clock rate: 1007 MHz
Memory clock rate: 524 MHz
Memory bus width: 64 bits
WarpSize: 32
L2 cache size: 2048 KB
Total global mem: 65536 MB
ECC enabled: No
Compute Capability: 7.0
Total SPs: 1024 (16 MPs x 64 SPs/MP)
Compute throughput: 2063.60 GFlops (theoretical single precision FMAs)
Memory bandwidth: 16.78 GB/sec
-----
Total GPU memory 68719476736, free 68584071168
Buffer size: 256MB
Trade-off type: compute with global memory (block strided)
Elements per thread: 8
Thread fusion degree: 4
----- CSV data -----
Experiment ID, Single Precision ops,,,,, Half precision ops,,,,, Integer operations,,,,,
Compute iters, Flops/byte, ex.time, GFLOPS, GB/sec, Flops/byte, ex.time, GFLOPS, GB/sec, Iops/byte, ex.time, GIOPS, GB/sec
0, 0.250, 1.84, 18.26, 73.04, 0.500, 1.71, 39.20, 78.40, 0.250, 1.69, 19.87, 79.48
1, 0.750, 1.53, 65.84, 87.79, 1.500, 1.28, 156.71, 104.47, 0.750, 1.25, 80.39, 107.18
2, 1.250, 1.26, 133.41, 106.73, 2.500, 1.27, 263.34, 105.34, 1.250, 1.21, 138.65, 110.92
3, 1.750, 1.21, 193.42, 110.53, 3.500, 1.48, 317.62, 90.75, 1.750, 1.40, 167.83, 95.90
4, 2.250, 1.45, 208.50, 92.67, 4.500, 1.47, 411.12, 91.36, 2.250, 1.45, 208.31, 92.58
5, 2.750, 1.25, 294.73, 107.17, 5.500, 1.24, 593.09, 107.83, 2.750, 1.15, 321.30, 116.84
6, 3.250, 1.25, 350.06, 107.71, 6.500, 1.28, 680.16, 104.64, 3.250, 1.23, 354.28, 109.01
7, 3.750, 1.19, 422.40, 112.64, 7.500, 1.27, 789.87, 105.32, 3.750, 1.29, 389.75, 103.93
8, 4.250, 1.21, 469.52, 110.48, 8.500, 1.13, 1010.18, 118.84, 4.250, 1.14, 501.85, 118.08
9, 4.750, 1.13, 566.15, 119.19, 9.500, 1.13, 1133.08, 119.27, 4.750, 1.16, 547.43, 115.25
96, 48.250, 4.74, 1365.34, 28.30, 96.500, 4.50, 2879.78, 29.84, 48.250, 4.28, 1511.87, 31.33
128, 64.250, 5.50, 1566.99, 24.39, 128.500, 5.20, 3316.00, 25.81, 64.250, 5.22, 1652.46, 25.72
192, 96.250, 7.91, 1632.92, 16.97, 192.500, 7.43, 3478.39, 18.07, 96.250, 7.44, 1736.11, 18.04
256, 128.250, 11.24, 1530.91, 11.94, 256.500, 10.19, 3377.26, 13.17, 128.250, 10.09, 1705.75, 13.30
512, 256.250, 19.52, 1761.66, 6.87, 512.500, 19.18, 3585.93, 7.00, 256.250, 18.24, 1885.72, 7.36
1024, 512.250, 36.52, 1882.62, 3.68, 1024.500, 37.20, 3696.70, 3.61, 512.250, 35.50, 1936.63, 3.78
-----
```

2.7 tuFlops

2.7.1 简介

登临tuFlops工具用来计算登临GPU TU的峰值算力。

2.7.2 使用方法

./tuFlops [OPTION]

参数含义如下：

- -d 指定数据类型
- -n 指定循环次数
- -s 指定是否开启 sparisty
- -D 用于指定设备ID（多卡机器上选择设备，默认值为0）
- 其余参数可以使用缺省值（如 -i -f 参数用于指定 shape，缺省值已设置为适合发挥硬件性能的shape）

命令执行如下图所示：

```
./tuFLOPS -d fp16 -n 10000
./tuFLOPS -d fp16 -n 10000 -s
./tuFLOPS -d fp16 -n 10000 -D 0
dtype: int4/uint4/int8/uint8/fp16/bf16/fp32/tf32
Usage: ./tuFLOPS [OPTIONS]

Options:
  -h, --help                Print this help message and exit
  -i UINT ...               input shape, n,d,h,w,c
  -f UINT ...               filter shape, o,d,h,w,i
  -D UINT                   device_id
  -n UINT                   numRuns
  -s                        sparisty_enable
  -d TEXT                   data type
```

输出结果如下图，会打印数据类型、conv shape、循环次数、运行时最大功率、总计算量、时间戳和TU算力等。

```
=====
Data Type : int8
Input Shape : 1,1,64,64,512
Filter Shape : 512,1,1,1,512
Repeat Counter : 10000
Sparsity : false
Rated Power : 150W
Exec Power : 121W
Total Tops : 21.4748
Start Time : 17:26:27.969702
Start Time : 17:26:28.011616
Total Time : 0.0419134s
TU Perf : 512.36T
=====
```


2.8 dljpeg_decode

2.8.1 简介

dljpeg_decode工具使用登临GPU上的JPEG硬件编解码单元（JPU）对JPEG文件进行解码并测试性能。

登临GPU包含了16个JPU（不同型号可能会有所差异），这些JPU可以并行工作。

JPU可以支持JPEG硬件编码和JPEG硬件解码。

dljpeg_decode工具使用登临Hamming SDK的dlJPEG API进行JPEG解码，更多信息请参考SDK中的dlJPEG API文档。

2.8.2 使用方法

`./dljpeg_decode [option]`

Options :

- -i 输入文件名，需要解码的jpeg文件，如果未设置默认为1080p.jpg。
- -j JPU id，选择用来解码的JPU单元，如果未设置默认为0。
- -o 输出文件，解码结果YUV数据输出到该文件，如果未为设置默认为不输出。

示例1：

```
./dljpeg_decode
```

输出如下图：

```
=====
Jpu id      : 0
Jpu count   : 16
Type        : decode
Input       : 1080p.jpg
Image Size  : 1920x1080
Time Elapsed : 0.001978 seconds
Fps         : 505.561188
=====
```

示例2：

```
./dljpeg_decode -i other.jpg -j 3 -o other.yuv
```

2.9 dljpeg_encode

2.9.1 简介

dljpeg_encode工具使用登临GPU上的JPEG硬件编解码单元（JPU）对JPEG文件进行编码并测试性能。

该工具使用登临Hamming SDK的dlJPEG API进行JPEG编码，更多信息请参考SDK中的dlJPEG API文档。

2.9.2 使用方法

`./dljpeg_encode [option]`

Options :

- -i 输入文件，需要编码的YUV文件，如果未设置默认为1080p.yuv。
- -j JPU id，选择用来编码的JPU单元，如果未设置默认为0。
- -f 输入YUV格式，可以为420，422，440，444。如果未设置默认为420，表示YUV420P。
- -h 输入YUV高度，如果未设置默认为1080。
- -w 输入YUV宽度，如果未设置默认为1920。
- -o 输出文件，解码结果jpeg数据输出到该文件，如果未为设置默认为不输出。

示例1：

```
./dljpeg_encode
```

输出如下图：

```
=====
Jpu id      : 0
Jpu count   : 16
Type        : encode
Input       : 1080p.yuv
Image Size  : 1920x1080
Image Format : YUV420
Time Elapsed : 0.001969 seconds
Fps         : 507.872009
=====
```

示例2：

```
./dljpeg_encode -i 1280x720.yuv -w 1280 -h 720 -f 420 -o 720p.jpg
```

2.10 dlvid_decode

2.10.1 简介

dlvid_decode工具使用登临GPU上的视频硬件解码单元（VPU Decoder）对视频码流文件进行解码并测试性能。

登临GPU包含了16个VPU Decoder（不同型号可能会有所差异），这些VPU Decoder可以并行工作。

VPU Decoder可以支持视频硬件解码。

该工具使用登临Hamming SDK的dVID API进行视频解码，更多信息请参考SDK中的dVID API文档。

2.10.2 使用方法

`./dlvid_decode [option]`

Options :

- -i 输入文件名，码流文件名，如果未设置默认为1080p.265。
- -c 码流格式，H264或者H265，如果未设置默认为H265。
- -f 最大FPS限制。如果未设置默认为不限制。
- -s 同时解码数量。工具会开启多线程同时进行多路解码，如果未设置默认为1。
- -v VPU Decoder ID，选择用来解码的VPU decoder单元，如果未设置默认为0。
- -o 输出文件，解码结果YUV数据输出到该文件，如果未为设置默认为不输出。

示例1：

```
./dlvid_decode
```

输出如下图：

```
=====
Decoder id   : 0
Decoder count: 16
Type         : decode
Format       : h265
Input        : 1080p.265
Resolution   : 1920x1080
Frames       : 160
Time Elapsed : 0.182230 seconds
Fps          : 878.011304
=====
```

示例2：

```
./dlvid_decode -i other.264 -c h264 -v 3 -o other.yuv
```

2.11 dlvid_encode

2.11.1 简介

dlvid_encode工具使用登临GPU上的视频硬件编码单元（VPU Encoder）对YUV视频文件进行编码并测试性能。

登临GPU包含了8个VPU Encoder（不同型号可能会有所差异），这些VPU Encoder可以并行工作。

VPU Encoder可以支持视频硬件编码。

dlvid_encode工具使用登临Hamming SDK的dVID API进行视频编码，更多信息请参考SDK中的dVID API文档。

2.11.2 使用方法

`./dlvid_encode [option]`

Options :

- -i 输入文件名，需要编码的YUV文件，如果未设置默认为1080p.yuv。
- -c 编码格式，H264或者H265，如果未设置默认为H265。
- -s 同时编码数量。工具会开启多线程同时进行多路编码，如果未设置默认为1。
- -v VPU Encoder ID，选择用来编码的VPU encoder单元，如果未设置默认为0。
- -o 输出文件，解码结果YUV数据输出到该文件，如果未为设置默认为不输出。
- -h 输入YUV高度，如果未设置默认为1080。
- -w 输入YUV宽度，如果未设置默认为1920。

示例1：

```
./dlvid_encode
```

输出如下图：

```
=====
Jpu id       : 0
Jpu count    : 16
Type         : encode
Input        : 1080p.yuv
Image Size   : 1920x1080
Image Format  : YUV420
Time Elapsed : 0.001969 seconds
Fps          : 507.872009
=====
```

示例2：

```
./dlvid_encode -i 1280x720.yuv -c h265 -w 1280 -h 720 -v 3 -o 720p.265
```