DC1000 加速卡测试指导手册

发布版本:　A1

发布日期:　2024-01-26

变更记录

版本	发布日期	变更描述
A0	2024/1/12	首次发布
A1	2024/01/26	增加 DC1000 功耗和渲染算力测试

目 录

一 产品概述

DC1000 GPU 卡是单宽半高半长标准 PCIe 尺寸的加速卡，搭载了最新研发的 SoC。

DC1000 搭载的高性能图形处理芯片，具备超高的渲染性能、超高密度和超低延时的视频处理性能。DC1000 适用于高性能手机、云游戏和云桌面等应用场景，集成高性能渲染核心；针对国内外视频编解码标准，集成了视频核心和解码核心，能实现超高密度、超低延时、超高质量的视频编码能力和超高密度的视频解码能力。DC1000 集成了自主研发支持图像分类、超高分别率、图像增强、自然语言处理等深度学习算法的加速计算架构。DC1000 支持基于 SR-IOV 硬件虚拟化技术，适用于云桌面场景，在虚拟化场景中能提供更好的性能以及更高的安全性。

DC1000 GPU 卡热设计最大功耗可达 70W，采用被动散热方式，在低功耗模式下提供超高图形处理性能、视频处理能力和深度学习 AI 推理能力，广泛用于 Android 云游戏、云手机、云桌面、视频会议、视频转码等应用场景。

二 测试环境部署

2.1 测试环境要求

推荐使用如下操作系统进行适配测试。

Platform	OS	Kernel
aarch64	Ubuntu 20.04	5.4.153-5.4.153-ampere

测试前应确认获取到的测试软件包是否完整、版本是否一致、操作系统以及硬件是否符合测试要求。请参考下表进行确认。

- 软件和工具包

序号	名称	描述
1	va-pci-*_ aarch64.rpm	用于 ARM 服务器 CentOS 系统的板卡驱动
2	va-pci-*_aarch64.deb	用于 ARM 服务器 Ubuntu 系统的板卡驱动
3	docker_*.tar	测试使用的 docker 镜像包
4	2012img	AI 图片数据集

- Docker 版本

序号	名称	版本要求
1	Docker engine	20.10.8 或以上

测试前应确认和记录硬件配置和操作系统环境信息，在测试中遇到问题反馈测试 Issue 时，请提供如下表所示的信息，以便我们更好的复现与定位问题。

序号	名称	描述
1	测试硬件配置信息	服务器型号/CPU 型号/内存/硬盘/外插卡/BIOS 版本/BMC 版本等
2	操作系统环境信息	OS 版本/内核版本/docker 版本/板卡驱动版本
3	问题发生前的操作步骤	详细记录
4	相关测试问题的 log	收集 log

2.2 系统检查

- **OS 版本**

打开 shell 终端，执行 cat /proc/version 命令查询 Ubuntu 系统版本。

```
root@root:~# cat /proc/version
Linux version 5.4.153-5.4.153-ampere
```

- **Kernel**

执行 uname -r 命令查看当前 Ubuntu 系统内核版本信息。

```
root@root:~# uname -r
5.4.153-5.4.153-ampere
```

- **Docker Engine**

执行 docker --version 查看 docker 版本信息。

```
root@root:~# docker --version
Docker version 24.0.2, build cb74dfc
```

执行 docker --version 命令打印出 Docker version 24.0.2 说明系统已安装好 docker。如果未打印出信息则表示该测试环境未安装 docker，需要安装 Docker。Docker 安装方法请参考 2.5 章节的 docker 安装。

2.3 安装 DKMS(Dynamic Kernel Module System)

在安装 PCIe 驱动前，先检查系统是否安装了 DKMS，如果已经安装，请忽略本节的内容，如果未安装请参考下面的安装方法进行在线安装。

在线安装 DKMS (Ubuntu)

```
sudo apt-get update
sudo apt-get -y install dkms
```

2.4 PCIe 驱动安装

将 Ubuntu 系统测试相关的软件包拷贝到服务器系统中，如果测试环境是首次安装 PCIe 驱动，可以直接进行 PCIe 驱动安装。安装命令参考如下：

```
[root@localhost ~]# sudo dpkg -i va-pci-**********_aarch64.deb
```

如果系统已安装了 PCIe 驱动，需先卸载旧驱动，再安装新驱动。在卸载驱动之前，应当先停止通过 DC1000 板卡创建的 docker 容器，具体步骤如下：

1) 查询当前系统安装的驱动信息，执行如下命令；

```
[root@localhost ~]# sudo dpkg -l | grep -i va-pci
```

2) 卸载 PCIe 驱动，执行如下命令；

```
[root@localhost ~]# sudo dpkg -r va-pci-****-dkms
```

3) 执行 reboot 命令重启系统；

```
[root@localhost ~]# reboot
```

4) 系统重启完成，安装 PCIe 驱动，执行如下命令；

```
[root@localhost ~]# sudo dpkg -i va-pci-**********_aarch64.deb
```

2.5 Docker 安装

在 Ubuntu 下如何进行 docker 安装，可参考 docker 官方网站提供的方法进行安装；在 Ubuntu 下安装 docker 的方法请查看官方链接：

Install Docker Engine on Ubuntu | Docker Docs

2.6 加载 docker 镜像包

使用 docker load 命令加载 docker 镜像包，加载 docker 包命令如下。

```
[root@localhost ~]# docker load -i docker_xxx.tar
```

Docker 镜像加载需要一些时间，加载时间的快慢取决于 Docker 镜像包的大小，Docker 镜像包越大加载时间就越长。

2.7 查询 docker image ID

Docker 镜像加载成功后，执行 docker images 命令查询 docker 镜像 ID 信息。

```
[root@localhost ~]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED
<none>        <none>    ee2aa9efb0ac   2 days ago
```

2.8 创建业务容器

下面介绍使用特权模式创建业务容器，参考如下命令创建 docker 容器。

```
[root@localhost ~]# docker run --privileged=true -it -v
${host_dataset_path}:/opt/va/vaststream/release/samples/datasets ${IMAGE ID}
/bin/bash
```

说明：

- 根据实际 docker images id 来替换${IMAGE ID};
- “-v” 指示将 host 端特定目录下的文件映射到指定容器内的特定目录下，应根据实际情况替换 host 端文件路径${host_dataset_path}。

2.9 查询 docker 容器

查询当前系统中所有 container 的信息，执行如下命令。

```
[root@localhost ~]# docker ps -a
CONTAINER ID   IMAGE          COMMAND       CREATED      STATUS                      PORTS     NAMES
460532c8dc82   ee2aa9efb0ac   "/bin/bash"   6 days ago   Exited (0) 35 minutes ago
```

2.10 启动 docker 容器

启动容器的前提是已经成功创建容器，启动已创建的 CONTAINER , 可参考如下命令。

```
[root@localhost ~]# docker start ${CONTAINER ID}
root@${containerID}:/#
```

执行 docker start ${CONTAINER ID}后成功打印${CONTAINER ID} 说明容器启动成功，可以通过 docker ps -a 命令查询容器状态为 UP。

注意：

在实际环境下，请根据实际 CONTAINER ID 替换${CONTAINER ID}。

2.11 运行 docker 容器

在 2.9 章节中已经通过执行 docker start {CONTAINER ID} 启动了容器，运行并进入容器执行如下命令。

```
[root@localhost ~]# docker exec -it ${CONTAINER ID} /bin/bash
root@${containerID}:/#
```

当系统提示符由原先的 hostname 变为 CONTAINER ID 时，说明已进入 docker 容器环境。

注意：

在实际环境下，请根据实际 CONTAINER ID 替换命令中的${CONTAINER ID}。

2.12 退出容器

退出 docker 系统环境，执行 exit 命令即可退出容器。

```
root@460532c8dc82:/opt/va/vaststream/samples/transcode# exit
exit
root@vastai:~#
```

三 测试指南

3.1 测试环境

测试环境主要包含硬件环境和软件环境，硬件环境要求主要包含了 CPU 架构和 Memory 大小。

3.1.1 硬件环境

建议硬件测试环境要求如下:

CPU 架构	BIOS	CPU	Mem	Mem Channel
aarch64	NA	CPU	> 64GB	> 4
x86_64	NA	NA	> 64GB	> 4

3.1.2 软件环境

在测试开始前应确认并记录 OS 版本、kernel 版本、docker 版本以及板卡型号等信息。在反馈测试 issue 时，需要提供这些信息，以便我们更好的复现与定位问题。

3.1.3 服务器 firmware 设置

3.1.3.1 BMC 设置

因为 BMC 没有适配，所以需要调整风扇转速为 100%，以免板卡温度过高；适配后，风扇转速可以设置为 default，由系统的散热策略进行自动调节。

3.1.3.2 BIOS/UEFI 设置

在进行业务测试前，为了板卡的性能，需要设定 BIOS 下性能模式为高性能模式（High-performance），非高性能模式可能会导致性能测试值下降。

3.1.4 DC1000 加速卡

DC1000 加速卡安装到服务器中，进入系统安装对应版本的驱动，如图-1，执行 vasmi list 命令可查看系统已识别到的加速卡列表，其中 AIC 栏为设备的 index 编号，测试中可通过该 index 编号来指定对应的设备。

```
root@vastai:/home/lianjie/DC1000/tools-3.0.1-20240112-74bc36c-linux-aarch64-fusion/tools# ./vasmi list
Smi version:3.0.1
  AIC        Name  DevId   VF/PF     PCIE_Bus_Id    pkgid:devfn:pfn:vfn  AI_Node       Video_Node     Render_Node
               SN                                                                      Gfx_Card_Node  Nulldisp_Node
=============================================================================================================
    0      DC1000      0   PF**      0000:01:00.0    0:0:0:-              vacc0       va_video0      N/A
     ECA23BV00162                                                                    N/A            N/A
                    -------------------------------------------------------------------------------------------
                        1   PF        0000:01:00.1    0:1:1:-              vacc0       va_video0      N/A
                                                                                     N/A            N/A
                    -------------------------------------------------------------------------------------------
                        2   PF        0000:01:00.2    0:2:2:-              vacc0       va_video0      N/A
                                                                                     N/A            N/A
                    -------------------------------------------------------------------------------------------
                        3   PF        0000:01:00.3    0:3:3:-              vacc0       va_video0      N/A
                                                                                     N/A            N/A
=============================================================================================================
    1      DC1000      4   PF**      0001:01:00.0    0:0:0:-              vacc0       va_video0      N/A
     ECA23BV00115                                                                    N/A            N/A
                    -------------------------------------------------------------------------------------------
                        5   PF        0001:01:00.1    0:1:1:-              vacc0       va_video0      N/A
                                                                                     N/A            N/A
                    -------------------------------------------------------------------------------------------
                        6   PF        0001:01:00.2    0:2:2:-              vacc0       va_video0      N/A
                                                                                     N/A            N/A
                    -------------------------------------------------------------------------------------------
                        7   PF        0001:01:00.3    0:3:3:-              vacc0       va_video0      N/A
                                                                                     N/A            N/A
=============================================================================================================
```

图-1

3.2 测试

3.2.1 H264 转码测试

进入容器测试环境，通 cd 命令切换到/opt/va/vaststream/samples/transcode 目录下；执行

h264.sh 脚本开始 48 路 1080P h264 转码测试，详细测试方法可参考如下测试步骤。

```
root@06c11156be7d:/# cd /opt/va/vaststream/samples/transcode/
root@06c11156be7d:/opt/va/vaststream/samples/transcode# ./h264.sh 1 all
2023-10-20 09:34:02 --- Start h264 Testing
       test loops = 1
installed devices = 0,1,2
     test devices = 0,1,2
       test nodes =
va_video0,va_video1,va_video2,va_video3,va_video4,va_video5,va_video6,va_video7,va_video8,va_vi
deo9,va_video10,va_video11

dev 0: nodes= va_video0 va_video1 va_video2 va_video3, channels/die = 12
channels=12
dev 1: nodes= va_video4 va_video5 va_video6 va_video7, channels/die = 12
channels=12
dev 2: nodes= va_video8 va_video9 va_video10 va_video11, channels/die = 12
channels=12
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from
'/opt/va/vaststream/ParkScene_1920x1080_30fps_h264_4M.mp4':
  Metadata:
    major_brand     : isom
    minor_version   : 512
    compatible_brands: isomiso2avc1mp41
    encoder         : Lavf59.27.100
  Duration: 00:00:33.33, start: 0.000000, bitrate: 4294 kb/s
    Stream #0:0(und): Video: h264 (High) (avc1 / 0x31637661), yuv420p, 1920x1080, 4291 kb/s, 30
fps, 30 tbr, 15360 tbn, 60 tbc (default)
    Metadata:
      handler_name    : VideoHandler
      encoder         : Lavc59.37.100 h264_vastapi
......
H264Trans round 1 test has finished

Result Directory: test_h264_2023-10-20_09-34-02
================================================================
Test Result Summary:
loop set: 1
H264Trans round 1 test : PASS = 144
================================================================
test devices : 0,1,2
 total loops : 1
  FAIL loops : 0
```

命令说明：

h264.sh <parm1> <parm2>

parm1： 该参数控制测试循环次数，若不提供默认为 1。可根据测试需要定义该参数的值。

parm2：该参数设置测试加速卡，若测试系统中所有的加速卡，可以定义该参数为 all；若只测试第一和第三张卡；可设置该参数为 device index id，device index id 卡之间用逗号隔开。示例：“./h264.sh 10 1,3”对 device index id 为 1 和 3 的卡进行 10 个循环的 h264 转码测试。

Pass 标准：

如果 FAIL loops 的结果为 0，则说明测试结果 Pass。

3.2.2 H265 转码测试

在容器测试环境下，通过 cd 命令切换到/opt/va/vaststream/samples/transcode 目录下，执行 hevc.sh 脚本开始 48 路 1080P 转码测试，详细测试方法可参考如下测试步骤。

```
root@06c11156be7d:/# cd /opt/va/vaststream/samples/transcode/
root@06c11156be7d:/opt/va/vaststream/samples/transcode# ./hevc.sh 1 all
2023-10-20 10:29:06 --- Start hevc Testing
test loops = 1
installed devices = 0,1,2
test devices = 0,1,2
test nodes =
va_video0,va_video1,va_video2,va_video3,va_video4,va_video5,va_video6,va_video7,va_video8,va_video9,va_video10,va_video11
dev 0: nodes= va_video0 va_video1 va_video2 va_video3, channels/die = 12
channels=12
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from
'/opt/va/vaststream/ParkScene_1920x1080_30fps_h265_4M.mp4':
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from
'/opt/va/vaststream/ParkScene_1920x1080_30fps_h265_4M.mp4':
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from
'/opt/va/vaststream/ParkScene_1920x1080_30fps_h265_4M.mp4':
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from
'/opt/va/vaststream/ParkScene_1920x1080_30fps_h265_4M.mp4':
Metadata:
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from
'/opt/va/vaststream/ParkScene_1920x1080_30fps_h265_4M.mp4':
Metadata:
……
HevcTrans round 1 test has finished

Result Directory: test_hevc_2023-10-20_10-29-06
================================================================
Test Result Summary:
loop set: 1
HevcTrans round 1 test : PASS = 144
================================================================
test devices : 0,1,2
 total loops : 1
  FAIL loops : 0
```

命令说明：

hevc.sh <parm1> <parm2>

parm1： 该参数控制测试循环次数，若不提供默认为 1。可根据测试需要定义该参数的值。

Parm2： 该参数设置测试加速卡，若测试系统中所有的加速卡，可以定义该参数为 all；若只测试第一和第三张卡；可设置该参数为 device index id，id 之间用逗号隔开。示例：”./hevc.sh 10 1,3” 对 device index id 为 1 和 3 的卡进行 10 个循环的 hevc 转码测试。

Pass 标准：

如果 FAIL loops 的结果为 0，则说明测试结果 Pass。

3.2.3 H264/H265 转码压力测试

在容器测试环境下，通过 cd 命令切换到/opt/va/vaststream/samples/transcode 目录下，执行 stress.sh 脚本开始 48 路 1080P H264/H265 转码压力测试，详细测试方法可参考如下测试步骤。

```
root@06c11156be7d:/# cd /opt/va/vaststream/samples/transcode/
root@06c11156be7d:/opt/va/vaststream/samples/transcode# ./stress.sh 1 all
2023-10-20 10:37:30 --- Start press Testing
       test loops = 1
installed devices = 0,1,2
     test devices = 0,1,2
       test nodes =
va_video0,va_video1,va_video2,va_video3,va_video4,va_video5,va_video6,va_video7,va_video8,v
a_video9,va_video10,va_video11
dev 0: nodes= va_video0 va_video1 va_video2 va_video3, channels/die = 12
channels=12
dev 1: nodes= va_video4 va_video5 va_video6 va_video7, channels/die = 12
channels=12
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from
'/opt/va/vaststream/ParkScene_1920x1080_30fps_h264_4M.mp4':
  Metadata:
    major_brand     : isom
    minor_version   : 512
    compatible_brands: isomiso2avc1mp41
    encoder         : Lavf59.27.100
  Duration: 00:00:33.33, start: 0.000000, bitrate: 4294 kb/s
    Stream #0:0(und): Video: h264 (High) (avc1 / 0x31637661), yuv420p, 1920x1080, 4291 kb/s,
30 fps, 30 tbr, 15360 tbn, 60 tbc (default)
    Metadata:
      handler_name    : VideoHandler
      encoder         : Lavc59.37.100 h264_vastapi
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from
'/opt/va/vaststream/ParkScene_1920x1080_30fps_h264_4M.mp4':
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from
'/opt/va/vaststream/ParkScene_1920x1080_30fps_h264_4M.mp4':
  Metadata:
  Metadata:
......
HevcTrans round 1 test has finished

Result Directory: test_press_2023-10-20_10-37-30
================================================================
Test Result Summary:
loop set: 1
H264Trans round 1 test : PASS = 144
HevcTrans round 1 test : PASS = 144
================================================================
test devices : 0,1,2
 total loops : 1
```

```
FAIL loops : 0
```

命令说明：

stress.sh <parm1>

parm1： 该参数控制测试循环次数，若不提供默认为 1。可根据测试需要定义该参数的值。

Parm2： 该参数设置测试加速卡，若不提供默认为 all(容器环境下所有的卡)。若只测试第一和第三张卡；可设置该参数为 device index id，id 之间用逗号隔开。示例："./stress.sh 10 1,3" 对 device index id 为 1 和 3 的卡进行 10 个循环的转码压力测试。

Pass 标准：

如果 FAIL loops 的结果为 0，则说明测试结果 Pass。

3.2.4 Decode 测试

在容器系统下，通过 cd 命令切换到/opt/va/vaststream/samples/transcode 目录下，执行 decode.sh 脚本开始 48 路 1080P H264/H265 解码测试，详细测试方法可参考如下测试步骤。

```
root@06c11156be7d:/opt/soft# cd /opt/va/vaststream/samples/transcode/
root@06c11156be7d:/opt/va/vaststream/samples/transcode# ./decode.sh 1 all
2023-10-20 10:48:12 --- Start decode Testing
      test loops = 1
installed devices = 0,1,2
vast_devices_comma= 0,1,2
vast_devcnt=3
    test devices = 0,1,2
      test nodes =
va_video0,va_video1,va_video2,va_video3,va_video4,va_video5,va_video6,va_video7,va_video8,va_vi
deo9,va_video10,va_video11
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from
'/opt/va/vaststream/ParkScene_1920x1080_30fps_1000.mp4':
  Metadata:
    major_brand     : isom
    minor_version   : 512
    compatible_brands: isomiso2avc1mp41
    encoder         : Lavf59.27.100
  Duration: 00:00:33.33, start: 0.000000, bitrate: 6451 kb/s
    Stream #0:0Input #0, mov,mp4,m4a,3gp,3g2,mj2, from
'/opt/va/vaststream/ParkScene_1920x1080_30fps_1000.mp4':
(und)  Metadata:
: Video: h264 (High) (avc1 / 0x31637661), yuv420p, 1920x1080, 6448 kb/s    major_brand     : ,
isom30 fps,
30 tbr,     minor_version   : 15360 tbn, 51260 tbc
 (default)    compatible_brands:
isomiso2avc1mp41    Metadata:
……
Decode round 1 test has finished

Result Directory: test_decode_2023-10-20_10-48-12
================================================================
Test Result Summary:
```

```
loop set: 1
Decode round 1 test : PASS = 144
================================================================
test devices : 0,1,2
 total loops : 1
  FAIL loops : 0
```

命令说明：

decode.sh <parm1> <parm2>

parm1： 该参数控制测试循环次数，若不提供默认为 1。可根据测试需要定义该参数的值。

parm2： 该参数设置测试卡，若测试系统中所有的卡，可以定义该参数为 all；若只想测试第一张和第三张卡；可设置该参数为 device index id，id 之间用逗号隔开。示例：”./decode.sh 10 1,3“ 对 device index id 为 1 和 3 的加速卡进行 10 个循环的解码测试。

Pass 标准：

如果 FAIL loops 的结果为 0，则说明测试结果 Pass。

3.2.5 Resnet18 性能测试

进入容器测试环境下，通过 cd 命令切换到/opt/va/vaststream/samples/resnet18 目录下，然后执行“performance.sh”脚本进行 resnet18 的性能测试。

```
root@460532c8dc82:/opt/va/vaststream/samples/resnet18# ./performance.sh 1
Start Resnet18 Precision Testing
[20230810 06:56:59.990][INFO ] ===== [start] =====
[20230810 06:56:59.990][INFO ] Built: Thu 03 Aug 2023 03:30:28
[20230810 06:56:59.990][INFO ] ===== SDK_VERSION_10 =====
[20230810 06:56:59.990][INFO ] ------------------------------------ run stream config -----
[20230810 06:56:59.990][INFO ] die_id0[0], die_id1[-1], batchsize[1], buffer_size[11], iterations[1],
pool_num[8], vdsp_balance_mode[0]
[20230810 06:56:59.990][INFO ] sit[0], round[1], regModelAsync[1]
[20230810 06:56:59.990][INFO ] result_path: acc_2023-08-10_06-56-58/Resnet18_0.txt
[20230810 06:56:59.990][INFO ] jfile_name: resnet18-int8-yuv.json
[20230810 06:56:59.990][INFO ] ------------------------------------------------------------
[20230810 06:56:59.990][INFO ] Initialize for SDK 1.0
[43]: Initializing VACM logger with default config.
[20230810 06:56:59.993][INFO ] Start thread-pool with pool_num 1
[20230810 06:56:59.994][INFO ] Start thread-pool normal
[20230810 06:56:59.994][INFO ] ===== Run for Round[1/1] =====
[20230810 06:56:59.994][INFO ] [Streams count] 1
[20230810 06:56:59.994][INFO ] ------------------------------stream_config----------------
[20230810 06:56:59.994][INFO ] begin work thread[0]: 45
[20230810 06:56:59.994][INFO ] path: resnet18-int8-yuv.json, die: 0, threads: 1
[20230810 06:56:59.994][INFO ] ------------------------------------------------------------
[20230820 12:36:03.562][INFO ] ProcessOutputData[49000]
[20230820 12:36:04.175][INFO ] Iteration[0] Done. input_id[50000], runstream_count[50000]
[20230820 12:36:04.175][INFO ] ProcessOutputData[50000]
[20230820 12:36:04.175][INFO ] ProcessOutputData thread exit!!
[20230820 12:36:04.176][INFO ] [Stream Thread][0][resnet18] ended, tid = 85398
[20230820 12:36:04.263][INFO ] [GC][tid = 85398] resnet18 -> resnet18-int8-yuv.json
[20230820 12:36:04.281][INFO ] StreamBase::ReleaseStream
```

```
[20230820 12:36:04.288][INFO ] end work thread[0]: 85398
[20230820 12:36:04.292][INFO ] Total time consume [31641] ms
[20230820 12:36:04.292][INFO ] ===== [stop] =====
[20231023 11:46:19.987][INFO ] ===== [stop] =====
/opt/va/syseng/syseng_funs.sh: line 594:   450 Killed
${vaprofiler_path} -f -d ${test_devices_comma} > ${round_test_log}
Resnet18: Device 0, Round 1 is PASS. The fps is 1624.39286
2023-10-23 11:46:22 --- Round 1 finished.

Result Directory: /performance/test_performance_2023-10-23_11-45-16
================================================================
Test Result Summary:
Resnet18: Device 0, Round 1 is PASS. The fps is 1624.39286
================================================================
test devices : 0
 total loops : 1
  FAIL loops : 0
```

命令说明：

performance.sh <parm1>

<parm1>： 控制测试循环次数，默认为 1，用户可自定义循环测试次数。

Pass 标准：

Resnet18 性能帧率值大于 1500。FAIL loops 为 0 说明测试结果 Pass。

备注：

Resnet18 性能测试暂不支持多卡测试。

第一轮测试花费的时间会较长是正常现象，原因是需要从磁盘加载数据集到内存。

3.2.6 Resnet18 精度测试

进入容器测试环境下，通过 cd 命令切换到/opt/va/vaststream/samples/resnet18 目录下，然后执行"precision.sh"进行 Resnet18 的精度测试。

```
root@cf4ff0e48785:/opt/va/vaststream/samples/resnet18# ./precision.sh 1
Start Resnet18 Precision Testing
[20230820 12:45:12.758][INFO ] ===== [start] =====
[20230820 12:45:12.758][INFO ] Built: Thu 03 Aug 2023 03:30:28
[20230820 12:45:12.758][INFO ] ===== SDK_VERSION_10 =====
[20230820 12:45:12.758][INFO ] ---------------------------------- run stream config -----
[20230820 12:45:12.758][INFO ] die_id0[0], die_id1[-1], batchsize[1], buffer_size[11], iterations[1],
pool_num[8], vdsp_balance_mode[0]
[20230820 12:45:12.758][INFO ] sit[0], round[1], regModelAsync[1]
[20230820 12:45:12.758][INFO ] result_path: ./Precision_2023-08-20_12-45-
12/Resnet18_precision_1.txt
[20230820 12:45:12.758][INFO ] jfile_name: resnet18-int8-yuv.json
[20230820 12:45:12.758][INFO ] ------------------------------------------------------------
[20230820 12:45:12.758][INFO ] Initialize for SDK 1.0
[97107]: Initializing VACM logger with default config.
[20230820 12:45:12.758][INFO ] Start thread-pool with pool_num 1
[20230820 12:45:12.758][INFO ] Start thread-pool normal
```

```
[20230820 12:45:12.758][INFO ] ===== Run for Round[1/1] =====
[20230820 12:45:12.758][INFO ] [Streams count] 1
[20230820 12:45:12.758][INFO ] ------------------------------stream_config---------------
[20230820 12:45:12.758][INFO ] path: resnet18-int8-yuv.json, die: 0, threads: 1
[20230820 12:45:12.758][INFO ] ------------------------------------------------------------
[20230820 12:45:12.758][INFO ] [Threads count] 1
[20230820 12:45:12.758][INFO ] begin work thread[0]: 97109
[20230820 12:45:12.759][INFO ] [Stream Thread][0][resnet18] started, tid = 97109
[20230820 12:45:12.815][INFO ] files count: 50000
.....
[20231023 11:44:10.206][INFO ] Iteration[0] Done. input_id[50000], runstream_count[50000]
[20231023 11:44:10.206][INFO ] ProcessOutputData[50000]
[20231023 11:44:10.206][INFO ] ProcessOutputData thread exit!!
[20231023 11:44:10.206][INFO ] [Stream Thread][0][resnet18] ended, tid = 407
[20231023 11:44:10.267][INFO ] [GC][tid = 407] resnet18 -> resnet18-int8-yuv.json
[20231023 11:44:10.270][INFO ] StreamBase::ReleaseStream
[20231023 11:44:10.274][INFO ] end work thread[0]: 407
[20231023 11:44:10.274][INFO ] Total time consume [153059] ms
[20231023 11:44:10.274][INFO ] ===== [stop] =====
Test Result: ./precision/Precision_2023-10-23_11-41-36

----------------------------------------------------------------
Resnet18 Precision Loop 1 times Test Result Summary:
  TOP1_PREC: 68.2938
  TOP5_PREC: 88.2282
Test Result:  PASS
----------------------------------------------------------------
```

命令说明：

precision.sh <parm1>

<parm1>： 控制测试循环次数，默认为 1，用户可自定义循环测试次数。

Pass 标准：

Resnet18 精度：Top1_Prec > 68.30 ± 1 Top5_Prec > 88.30 ± 1；测试完成打印测试结果，查看 Test Result: PASS 。

备注：

Resnet18 精度测试暂不支持多卡测试。

3.2.7 Resnet18 性能压力测试

参考 3.7 节的 Resnet18 性能测试步骤，通过修改控制脚本的循环次数来达到长时间压力测试的目的。详细测试步骤参考如下。

```
root@cf4ff0e48785:/opt/va/vaststream/samples/resnet18# ./performance.sh 10
Start Resnet18 Precision Testing
[20230820 12:45:12.758][INFO ] ===== [start] =====
[20230820 12:45:12.758][INFO ] Built: Thu 03 Aug 2023 03:30:28
[20230820 12:45:12.758][INFO ] ===== SDK_VERSION_10 =====
[20230820 12:45:12.758][INFO ] ---------------------------------- run stream config -----
[20230820 12:45:12.758][INFO ] die_id0[0], die_id1[-1], batchsize[1], buffer_size[11], iterations[1],
```

```
pool_num[8], vdsp_balance_mode[0]
[20230820 12:45:12.758][INFO ] sit[0], round[1], regModelAsync[1]
[20230820 12:45:12.758][INFO ] result_path: ./Precision_2023-08-20_12-45-12/Resnet18_precision_1.txt
[20230820 12:45:12.758][INFO ] jfile_name: resnet18-int8-yuv.json
[20230820 12:45:12.758][INFO ] ------------------------------------------------------------
[20230820 12:45:12.758][INFO ] Initialize for SDK 1.0
[97107]: Initializing VACM logger with default config.
[20230820 12:45:12.758][INFO ] Start thread-pool with pool_num 1
[20230820 12:45:12.758][INFO ] Start thread-pool normal
[20230820 12:45:12.758][INFO ] ===== Run for Round[1/1] =====
[20230820 12:45:12.758][INFO ] [Streams count] 1
[20230820 12:45:12.758][INFO ] ----------------------------stream_config---------------
[20230820 12:45:12.758][INFO ] path: resnet18-int8-yuv.json, die: 0, threads: 1
[20230820 12:45:12.758][INFO ] ------------------------------------------------------------
[20230820 12:45:12.758][INFO ] [Threads count] 1
[20230820 12:45:12.758][INFO ] begin work thread[0]: 97109
[20230820 12:45:12.759][INFO ] [Stream Thread][0][resnet18] started, tid = 97109
[20230820 12:45:12.815][INFO ] files count: 50000
.....
Resnet18: Device 0, Round 10 is PASS. The fps is 1623.85714
2023-10-24 07:01:21 --- Round 10 finished.

Result Directory: /performance/test_performance_2023-10-24_06-50-27
============================================================
Test Result Summary:
Resnet18: Device 0, Round 10 is PASS. The fps is 1624.28571
Resnet18: Device 0, Round 10 is PASS. The fps is 1625.00000
Resnet18: Device 0, Round 10 is PASS. The fps is 1625.41818
Resnet18: Device 0, Round 10 is PASS. The fps is 1626.12727
Resnet18: Device 0, Round 10 is PASS. The fps is 1626.21818
Resnet18: Device 0, Round 10 is PASS. The fps is 1624.30357
Resnet18: Device 0, Round 10 is PASS. The fps is 1624.00000
Resnet18: Device 0, Round 10 is PASS. The fps is 1623.76786
Resnet18: Device 0, Round 10 is PASS. The fps is 1623.69643
Resnet18: Device 0, Round 10 is PASS. The fps is 1623.85714
============================================================
test devices : 0
 total loops : 10
  FAIL loops : 0
```

Pass 标准：

Resnet18 性能帧率值大于 1500 fps。FAIL loops 为 0 说明测试结果为 Pass。

备注：

Resnet18 性能压力测试仅支持单卡测试，暂不支持多卡测试。

第一轮测试花费的时间会较长是正常现象，原因是需要从磁盘加载数据集到内存。

3.2.8 PCIe 带宽测试

在容器测试环境下，通过 cd 命令切换到/opt/va/vaststream/tools/vaqual/bin 目录下，执行如下

命令开始 PCIe 带宽测试。

```
[root@localhost bin]# ./vaqual pcie reboot
[Pcie reboot]
start pcie reboot...
>>> pcie reboot success in dev0<pfn=0>
Pcie reboot success!
[root@localhost bin]# ./vaqual pcie reboot_pciebw
[Reboot pciebw]
Reboot pciebw success, use [pcie bw] cmd to get pcie bandwidth
[root@localhost bin]# ./vaqual pcie bw -d 0
[Pcie bandwidth test]
dev        dev->host       host->dev
0         17.17 GB/s      17.17 GB/s
========================================
dev        dev->host       host->dev
0         17.17 GB/s      17.16 GB/s
========================================
dev        dev->host       host->dev
0         17.16 GB/s      17.17 GB/s
========================================
dev        dev->host       host->dev
0         17.17 GB/s      17.18 GB/s
========================================
```

PCIe 带宽测试每秒钟输出一次带宽测试结果，如果需要退出 PCIe 带宽测试，可通过按 Ctrl+c 组合键中断 PCIe 带宽测试。

命令说明：

vaqual pcie reboot -d 0,1

vaqual pcie reboot_pciebw -d 0,1

vaqual pcie bw -d 0,1

-d 选项用来指定 device index id；进行多卡测试时，device index id 之间用逗号隔开。如果不加 -d 参数，默认是测试所有的加速卡。

Pass 标准:

PCI Express 版本	x8 理论带宽	x8 实测带宽	x16 理论带宽	x16 实测带宽
Gen 4.0	15.754 GB/s	> 8 GB/s	31.508 GB/s	> 15 GB/s

注意：

服务器影响板卡 PCIe 带宽性能的因素有如下几点，可根据实际情况进行分析和调优。

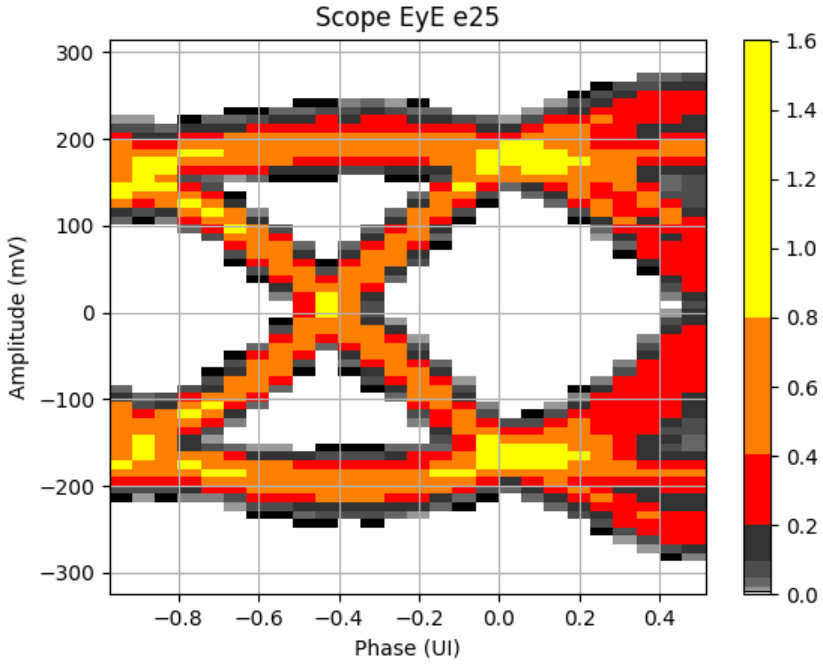
1. 服务器硬件配置，内存的数量、频率、安装方式、CPU 的频率和 PCIe Lanes 等；
2. UEFI 性能配置，主要涉及内存和 CPU 的性能设置，组合设置为高性能模式；
3. 内存 NUMA、PCIe 和 PCIe slot 设置等。

PCIe 带宽测试完成，需要执行 vaqual pcie reboot 进行 PCIe 重启。否则，会影响其他业务功能测试。

3.2.9 PCIe 眼图测试

在容器测试环境下，通过 cd 命令切换到/opt/va/vaststream/tools/vaqual/bin 目录下，参考如下测试步骤开始 PCIe 眼图测试。

```
[root@localhost bin]# ./vaqual pcie reboot
[Pcie reboot]
start pcie reboot...
[root@localhost bin]# ./vaqual pcie eye -d 1
[Pcie eye diagram test]
Dev[1] Gen4 Total Lane: 16
>>> Dev[1] Lane[0] eye scope: done.
>>> Dev[1] Lane[1] eye scope: start.
>>> Dev[1] Lane[1] eye scope: done.
>>> Dev[1] Lane[2] eye scope: start.
>>> Dev[1] Lane[2] eye scope: done.
>>> Dev[1] Lane[3] eye scope: start.
>>> Dev[1] Lane[3] eye scope: done.
>>> Dev[1] Lane[4] eye scope: start.
>>> Dev[1] Lane[4] eye scope: done.
>>> Dev[1] Lane[5] eye scope: start.
>>> Dev[1] Lane[5] eye scope: done.
>>> Dev[1] Lane[6] eye scope: start.
>>> Dev[1] Lane[6] eye scope: done.
>>> Dev[1] Lane[7] eye scope: start.
>>> Dev[1] Lane[7] eye scope: done.
.....
>>> Dev[1] Lane[14] eye scope: start.
>>> Dev[1] Lane[14] eye scope: done.
>>> Dev[1] Lane[15] eye scope: start.
>>> Dev[1] Lane[15] eye scope: done.
>>> Dev[1] done.
```



命令说明：

vaqual pcie reboot -d 0,1

vaqual pcie eye -d 0,1

-d 选项用来指定 device index id； 进行多卡测试时，device index id 之间用逗号隔开。如果不加-d 参数，默认是测试所有的卡 。

Pass 标准：

测试的眼图结果位于/opt/va/vaststream/tools/vaqual/bin/result/{测试日期命名文件夹}/pcie_eye_diagram_test/{时间命名文件夹}/路径下，只需查看眼图图片，即*.Scope.png 文件。检查 PCIe 每个 Lane 的眼高大于 15mV，眼宽大于 0.3UI。

注意：

PCIe 眼图测试完成，应执行 vaqual pcie reboot 对 PCIe 进行重启。否则会影响其他业务功能测试。

3.2.10 DDR 带宽测试

在容器测试环境下，通过 cd 命令切换到/opt/va/vaststream/tools/vaqual/bin 目录下，参考如下测试步骤开始 DDR 带宽测试。

```
[root@localhost bin]# ./vaqual pcie reboot
[Pcie reboot]
start pcie reboot...
>>> pcie reboot success in dev0<pfn=0>
[root@localhost bin]# ./vaqual pcie reboot_ddrbw
Pcie reboot success!
[Reboot ddrbw]
Reboot ddrbw success, use [ddr bw] cmd to get ddr bandwidth
[root@localhost bin]# ./vaqual ddr bw
```

```
[DDR bandwidth test]
dev             bandwidth
0             109.36 GB/s
1             109.37 GB/s
============================
dev             bandwidth
0             109.37 GB/s
1             109.36 GB/s
============================
dev             bandwidth
0             109.37 GB/s
1             109.36 GB/s
============================
```

命令说明：

vaqual pcie reboot -d 0,1

vaqual pcie reboot_ddrbw -d 0,1

vaqual ddr bw -d 0,1

-d 选项用来指定 device index id；进行多卡测试时，device index id 之间用逗号隔开。如果不加 -d 参数，默认是测试所有的卡 。如果要停止测试，可通过按 Ctrl+c 组合键来中断测试。

PASS 标准：

DDR 带宽理论值为 200 GB/s，其实测值应达到理论值的 50%以上即大于 100 GB/s。

注意：

PCIe 带宽测试完成，应执行 vaqual pcie reboot 进行 PCIe 重启。否则会影响其他业务功能测试。

3.2.11 AI 算力测试(INT8)

在容器测试环境下，通过 cd 命令切换到/opt/va/vaststream/tools/vaqual/bin 目录下。首先，执行 vaqual tops config 进行配置，然后设置环境变量，最后开始执行算力测试。测试完成后需 reboot PCIe，详细测试步骤参考如下。

```
root@cf4ff0e48785:/opt/va/vaststream/tools/vaqual/bin# ./vaqual tops config
[tops config]
root@cf4ff0e48785:/opt/va/vaststream/tools/vaqual/bin# source ../setenv.sh
set tops env
LD_LIBRARY_PATH=/opt/va/vaststream/tools/third_party/vacl/lib/aarch64
root@cf4ff0e48785:/opt/va/vaststream/tools/vaqual/bin# ./vaqual tops
[115938]: Initializing VACM logger with default config.
Trace: Construct stream : 1
vaclInvokeStreamOp
Tops monitor:
   dev             Tops
------------------------------------
    0            18.01
============================
<1/1>
Tops monitor:
```

```
   dev             Tops
-----------------------------------
    0            18.01
============================
root@cf4ff0e48785:/opt/va/vaststream/tools/vaqual/bin# ./vaqual pcie reboot
[Pcie reboot]
start pcie reboot...
root@cf4ff0e48785:/opt/va/vaststream/tools/vaqual/bin#
```

命令说明：

vaqual tops -d 0

-d 选项用来指定 device index id；进行多卡测试时，device index id 之间用逗号隔开。如果不加 -d 参数，默认测试所有的卡 。如果要停止测试，可通过按 Ctrl+c 组合键来中断测试。

PASS 标准：

AI (20TOPS)算力值大于 17.5 TOPS，小于 20 TOPS 为测试 Pass。

备注：

开始算力测试的进程需要时间启动，前几秒打印算力值请忽略，以稳定后的数值作为判断结果。

3.2.12 功耗测试

功耗测试不需要在 docker 容器环境下进行，参考如下步骤在服务器系统环境下进行环境搭建和功耗测试。

1. 拷贝 powervr.ini 文件到系统下的/etc 文件下；
2. 拷贝工具包（tools-3.0.1-20240112-74bc36c-linux-aarch64-fusion-tools.tar.gz）到系统中的任意目录下，使用 tar -zxvf ******.tar.gz 对工具压缩包进行解压缩。解压完成在当前工作目录解压缩出一个名为 tools-3.0.1-20240112-74bc36c-linux-aarch64-fusion 的文件。
3. 拷贝 DDK 包到系统任意目录下，执行 tar -zxvf SGPU100_GPU_Linux_DDK_Driver_*********.tar.gz 进行解压缩，解压缩完成后会解压缩出一个名为 ddk_pack 的文件。切换到 ddk_pack 文件下，执行./install_ddk.sh -i 安装 DDK 驱动，再执行./install_ddk.sh -s_pwr 开启 ddk 为功耗测试。如果执行./install_ddk.sh -s_pwr 开启 ddk 功耗测试失败的情况，可执行./install_ddk.sh -u 卸载 DDK，重新执行./install_ddk.sh -i 和./install_ddk.sh -s_pwr 安装 DDK 驱动和开启 ddk 功耗测试。
4. 通过 cd 命令切换到 vaqual 工具所在路径下；

   cd tools-3.0.1-20240112-74bc36c-linux-aarch64-fusion/tools/vaqual_sg/bin
5. 执行 vaqual stress -d 0,1 开始对 device id 为 0 和 1 的设备进行功耗测试。

```
root@va:/home/lianjie/tools/vaqual_sg/bin# ./vaqual stress -d 0,1
[stress test]
Power and temperature monitor [20240124071604]:
```

```
   dev       power(W)       temp(C)
---------------------------------------
    0          64.6          51.2
    1          65.3          58.3
=======================================
<1/1>
Power and temperature monitor [20240124071605]:
   dev       power(W)       temp(C)
---------------------------------------
    0          64.6          52.2
    1          64.4          59.9
=======================================
```

3.2.13 渲染算力测试

渲染算力不需要使用 docker 环境测试，在系统环境下进行渲染算力测试。下表列出了渲染算力测试已适配的 OS 和对应的 Kernel 版本。

Platform	OS	Kernel	DDK Package
x86-64	Ubuntu 20.04	5.4.0-156-generic	SGPU100_GPU_Linux_DDK_Driver_Mainline_x86_Ubuntu_20240117.tar.gz
aarch64	Ubuntu 20.04	5.4.153-5.4.153-ampere	SGPU100_GPU_Linux_DDK_Driver_Mainline_aarch64_Ubuntu_20240117.tar.gz
aarch64	OpenEuler 22.03 (LTS-SP2)	5.10.0-153	SGPU100_GPU_Linux_DDK_Driver_Mainline_aarch64_OpenEular_20240117.tar.gz

3.2.13.2 安装 DDK 驱动

进入系统后，开始安装 DDK 驱动，DDK 驱动安装方法参考如下步骤；

1) 将 DDK 包（SG100_GPU_Linux_DDK_Driver_Mainline_aarch64_Ubuntu_****.tar.gz）拷贝到系统的/home 或者其他目录下；
2) 执行"tar -xvf SG100_GPU_Linux_DDK_Driver_Mainline_aarch64_Ubuntu_****.tar.gz "进行解压缩，解压缩完成后解压缩出一个名为 ddk_pack 的文件；
3) 在 ddk_pack 文件下，执行"./install_ddk.sh -i" 安装 ddk；
4) 在 ddk_pack 文件下，执行"./install_ddk.sh -s" 启动 ddk。

```
root@va:/home# tar -xvf SGPU100_GPU _****.tar.gz
ddk_pack/
ddk_pack/KM/
ddk_pack/KM/target_aarch64/
ddk_pack/KM/target_aarch64/install_km.sh
```

```
ddk_pack/KM/target_aarch64/drm_nulldisp.ko
ddk_pack/KM/target_aarch64/va_gfx.ko
ddk_pack/install_ddk.sh
ddk_pack/UM/
...
root@va:/home/lianjie/ddk_pack# ./install_ddk.sh -i
/home/lianjie/ddk_pack/UM/target_aarch64 /home/lianjie/ddk_pack
/usr/lib/aarch64-linux-gnu
Installing user components for architecture target_aarch64
shared_library libPVRScopeServices.so -> /usr/lib/aarch64-linux-
gnu/libPVRScopeServices.so.1.18.6276027
executable eglconfigs -> /usr/local/bin/eglconfigs
executable egldmabuf -> /usr/local/bin/egldmabuf
executable eglinfo -> /usr/local/bin/eglinfo
executable gles1image_external -> /usr/local/bin/gles1image_external
executable gles1test1 -> /usr/local/bin/gles1test1
executable gles2test1 -> /usr/local/bin/gles2test1
executable gles3_render_to_image -> /usr/local/bin/gles3_render_to_image
executable gles3image_external -> /usr/local/bin/gles3image_external
executable gles3test1 -> /usr/local/bin/gles3test1
executable gles3tritest1 -> /usr/local/bin/gles3tritest1
executable gltest1 -> /usr/local/bin/gltest1
executable gltest2 -> /usr/local/bin/gltest2
executable gltest3 -> /usr/local/bin/gltest3
executable hwperfbin2jsont -> /usr/local/bin/hwperfbin2jsont
...
Installing firmware components for architecture target_neutral
firmware rgx.fw.35.4.1632.23 -> /lib/firmware/rgx.fw.35.4.1632.23

/usr/lib/aarch64-linux-gnu
Installing kernel components for architecture target_aarch64
install_km.sh: line 3: check_module_directory: command not found
kernel_module drm_nulldisp.ko -> /lib/modules/5.4.153-5.4.153-
ampere/extra/drm_nulldisp.ko
kernel_module va_gfx.ko -> /lib/modules/5.4.153-5.4.153-ampere/extra/va_gfx.ko

root@va:/home/lianjie/ddk_pack# ./install_ddk.sh -s
Loaded PowerVR consumer services.
root@vastai:/home/lianjie/ddk_pack#
```

3.2.13.3 安装 Clpeak

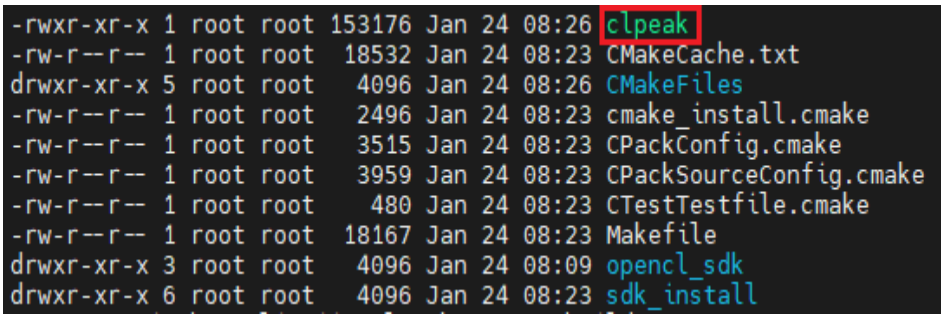
将 clpeak.zip 和 tools-**********.tar.gz 压缩包拷贝到系统中并解压缩到/home 目录下，执行以下两条指令分别解压缩 clpeak.zip 和 tools-**********.tar.gz;

- unzip clpeak.zip
- tar -xvf tools-*****.tar.gz

clpeak.zip 解压缩出来一个名未 clpeak-master 文件，通过 cd 命令切换至 clpeak_master 目录下进行 clpeak 的编译和安装。clpeak 的编译和安装方法请参考如下步骤;

1) git submodule update --init --recursive --remote
2) mkdir build
3) cd build
4) cmake ..
5) cmake --build .

安装完成后，在 build 目录下会生成一个名为 clpeak 的可执行文件。



注意：

在进行 clpeak 编译和安装时请使用 root 权限。

3.2.13.4 执行算力测试

测试环境搭建完成后，拷贝 4pf_mode_clpeak_test.sh 脚本至 clpeak 工具所在目录下，执行“chmod +x 4pf_mode_clpeak_test.sh”为脚本添加执行权限，执行”./4pf_mode_clpeak_test.sh”脚本开始渲染算力测试。

```
root@vastai:/home/lianjie/clpeak-master/build# ./4pf_mode_clpeak_test.sh
[gpu tops config]
DRI node exists, driver: 128:va
DRI node exists, driver: 129:va
DRI node exists, driver: 130:va
DRI node exists, driver: 131:va
/home/lianjie/clpeak-master/build
nodeId=128 nodeCnt=8
PF num: 4 in system.
export PVR_GPUIDX=0 , and start to compute clpeak for  single PF ...

Platform: VG1000
  Device: VastAI Graphics VG1000
    Driver version  : 1.18@6276027 (Linux ARM64)
    Compute units   : 4
    Clock frequency : 1400 MHz

    Single-precision compute (GFLOPS)
      float   : 1783.20
      float2  : 1681.59
      float4  : 1360.66
      float8  : 1690.41
      float16 : 1680.03
```

```
compute PF:0 finished.
export PVR_GPUIDX=1 , and start to compute clpeak for  single PF ...

Platform: VG1000
  Device: VastAI Graphics VG1000
    Driver version  : 1.18@6276027 (Linux ARM64)
    Compute units   : 4
    Clock frequency : 1400 MHz

    Single-precision compute (GFLOPS)
      float   : 1783.97
      float2  : 1682.66
      float4  : 1360.88
      float8  : 1692.10
      float16 : 1680.63

compute PF:1 finished.
export PVR_GPUIDX=2 , and start to compute clpeak for  single PF ...

Platform: VG1000
  Device: VastAI Graphics VG1000
    Driver version  : 1.18@6276027 (Linux ARM64)
    Compute units   : 4
    Clock frequency : 1400 MHz

    Single-precision compute (GFLOPS)
      float   : 1783.29
      float2  : 1683.43
      float4  : 1361.29
      float8  : 1692.50
      float16 : 1681.66

compute PF:2 finished.
export PVR_GPUIDX=3 , and start to compute clpeak for  single PF ...

Platform: VG1000
  Device: VastAI Graphics VG1000
    Driver version  : 1.18@6276027 (Linux ARM64)
    Compute units   : 4
    Clock frequency : 1400 MHz

    Single-precision compute (GFLOPS)
      float   : 1782.07
      float2  : 1682.15
      float4  : 1361.01
      float8  : 1691.23
      float16 : 1680.64

compute PF:3 finished.
```

结果说明：

DC1000 加速卡包含 4 个 PF，使用 clpeak 对 4 个 PF 进行渲染算力测试，4 个 PF 的算力值相加为板卡总渲染算力值。

PASS 标准：

渲染算力测试的值大于 6963 GFLOPS。依据 DC1000 白皮书定义的 7.6TFLOPS * 90%。

注意：

渲染算力测试结果正常输出，但测试一直处于未退出状态，需要等待测试进程自动结束并退出。