YOYOWORKS

XPU User Guide

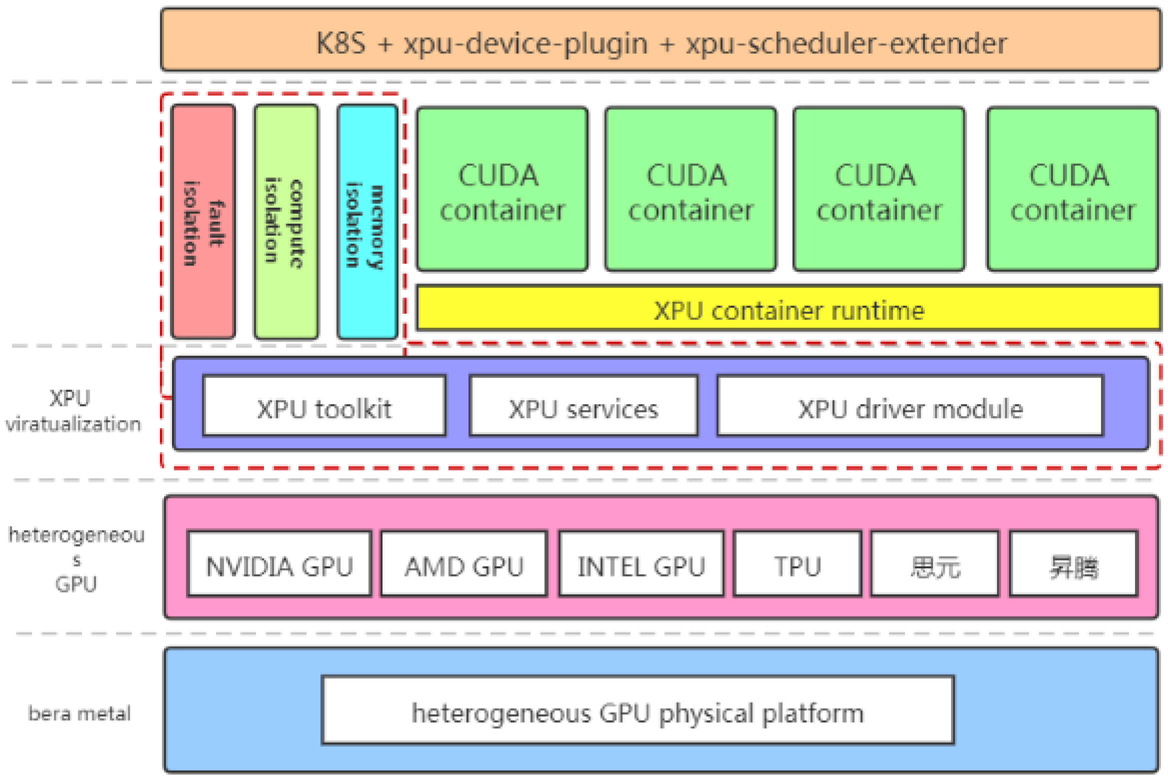
Version：2.0.21

Date：2021-09-09

Table of Contents

1 .XPU Introduction

The XPU is a GPU virtualization product that provides virtual GPUs for applications based on NVIDIA GPUs, including CUDA applications, OpenGL/EGL applications, and other scientific applications. NVIDIA GPU applications, which previously run on physical GPUs, can now run on virtual GPUs assigned to one container without the need to mount a physical GPU (see XPU architecture below).



XPU architecture

The XPU has two components: the host-side component and the container-side component. The host-side component includes: XPU toolkit, XPU service & XPU (kernel) driver module. XPU toolkit and service take charge of virtual GPU management, and XPU driver module virtualizes one physical GPU to many vGPUs which can be assigned and used by containers. It is the host-side component that enforces fault isolation, compute isolation & memory isolation among containers. The container-side component includes: XPU container runtime. XPU container runtime helps the container recognize the virtualized GPUs just as physical GPUs, so the AI application inside the container can use the virtual GPUs even without knowing those are virtual GPUs.

2 . XPU Requirements

Hardware & Software Requirements

Hardware	Software
● CPU: Intel/AMD x86_64 only ● MEM: 16GB (at least) ● GPU: NVIDIA GPU ● NIC: 1Gbps (at least)	● OS: RHEL 7.x (7.9 recommended) RHEL 8.x (8.4 recommended) CentOS 7.x (7.9 recommended) CentOS 8.x (8.4 recommended) Ubuntu Server 18.04 LTS Version Ubuntu Server 20.04 LTS Version ● Docker (Version 19.03 or later) ● NVIDIA GPU Driver (Version 440.x.x or later)
Note : 1. XPU only supports Pascal or later GPUs (Tesla/Quadro/RTX/GeForce supported)	Note : 1. XPU only supports 64 bit OS 2. XPU only supports 64 bit app

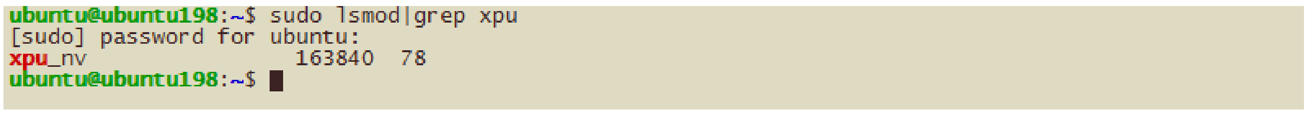
NOTE :XPU is available together with the EC2 instance type of 'Accelerated Computing' in the form of AWS AMI. Please be informed that all the XPU packages and its requirements have all been installed and setup in the XPU image.

3 . Check XPU Status

XPU AWS version is released and generally available via AWS AMI. All the requirements and XPU packages have been provisioned and setup completely. You can use XPU immediately after your EC2 GPU instances are booted. Before you use XPU, we recommend you check XPU status.

1. run `lsmod |grep xpu` to check the XPU driver module , and it should print something like below:

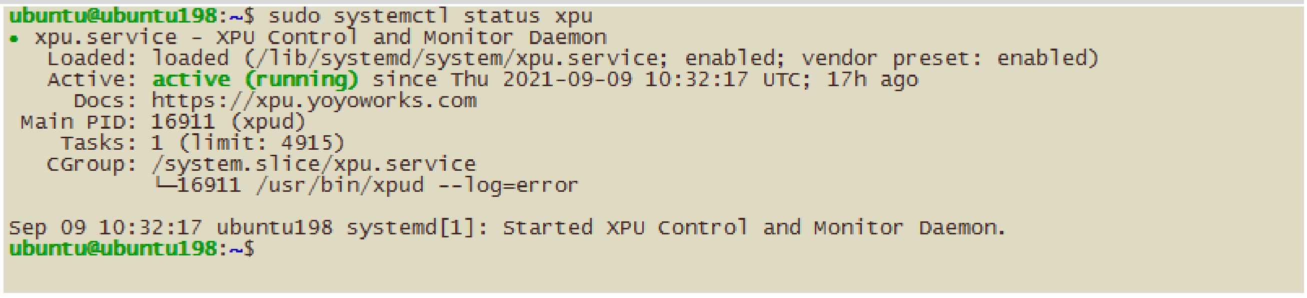
```
$>sudo lsmod |grep xpu
```



```
ubuntu@ubuntu198:~$ sudo lsmod|grep xpu
[sudo] password for ubuntu:
xpu_nv                 163840  78
ubuntu@ubuntu198:~$
```

2. run `systemctl status xpu` to check the XPU status , and it should print something like below:

```
$>sudo systemctl status xpu
```



```
ubuntu@ubuntu198:~$ sudo systemctl status xpu
● xpu.service - XPU Control and Monitor Daemon
   Loaded: loaded (/lib/systemd/system/xpu.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2021-09-09 10:32:17 UTC; 17h ago
     Docs: https://xpu.yoyoworks.com
 Main PID: 16911 (xpud)
    Tasks: 1 (limit: 4915)
   CGroup: /system.slice/xpu.service
           └─16911 /usr/bin/xpud --log=error

Sep 09 10:32:17 ubuntu198 systemd[1]: Started XPU Control and Monitor Daemon.
ubuntu@ubuntu198:~$
```

4 . Use XPU

Brief introduction

XPU virtualizes one physical GPU into many shares (including GPU memory and GPU compute cores) , which are assigned and used by vGPU. You can assign a particular shares to a container via Docker environment variables.

Using XPU in Docker

Env variables	Type	Note	Example
YOYO_XPU_SHARES	string	the XPU shares of this container	As 4 GPU on 1 host, use nvidia-smi -L to list GPU device id and UUID Returned: GPU 0: Tesla A10 (UUID: GPU-3aec****) GPU 1: Tesla A10 (UUID: GPU-45bc****) GPU 2: Tesla A10 (UUID: GPU-e728****) GPU 3: Tesla A10 (UUID: GPU-403e****) Set env variables : YOYO_XPU_SHARES=0:2-0,2:4-4 Means the No.0 GPU with 2 shares of memory and all shares of computing, No.2 GPU with 4 shares of memory and 4 shares of computing. In case of all computing shares, it can be specified as YOYO_XPU_SHARES=0:2,2:4-4 as well.

To list the shares (all_shares or free_shares):

```
$>sudo cat /proc/xpu/nvidia0/all_shares
```

all shares available of nvidia0. It tells the total shares which can be used on this GPU.

```
$>sudo cat /proc/xpu/nvidia0/free_shares
```

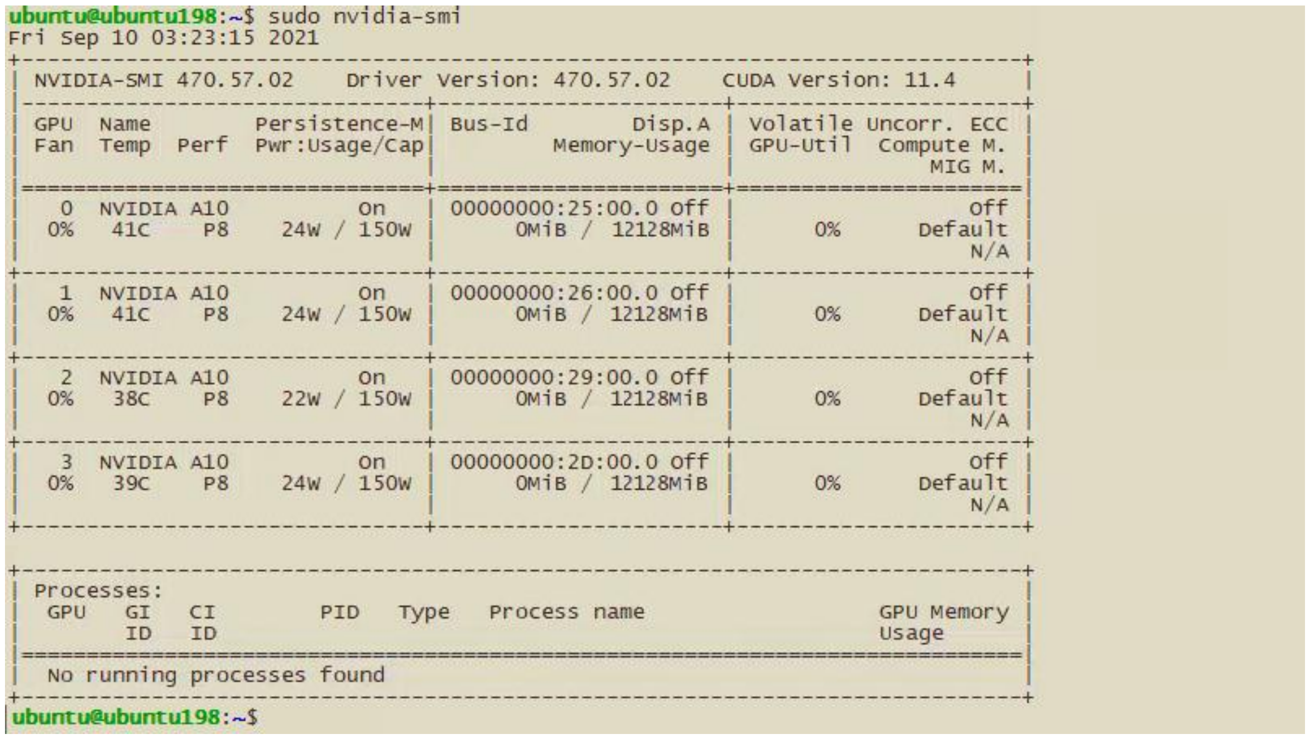
the current free shares of nvidia0. It tells the current free shares which can be used for new allocation.

To run a docker with XPU-based vGPU:

```
$>sudo docker run --gpus all --runtime=nvidia --name xpu -it –privileged -e YOYO_XPU_SHARES=0:2 nvcr.io/nvidia/cuda:10.2-devel-CentOS7 /bin/bash
```

To login the container to check：

```
$>sudo docker exec -it xpu /bin/bash

# Inside container>nvidia-smi
```



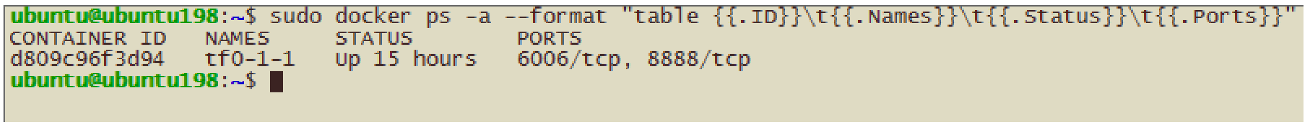
```
ubuntu@ubuntu198:~$ sudo nvidia-smi
Fri Sep 10 03:23:15 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.57.02    Driver Version: 470.57.02    CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA A10          On   | 00000000:25:00.0 Off |                  Off |
|  0%   41C    P8    24W / 150W |      0MiB / 12128MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  NVIDIA A10          On   | 00000000:26:00.0 Off |                  Off |
|  0%   41C    P8    24W / 150W |      0MiB / 12128MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   2  NVIDIA A10          On   | 00000000:29:00.0 Off |                  Off |
|  0%   38C    P8    22W / 150W |      0MiB / 12128MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   3  NVIDIA A10          On   | 00000000:2D:00.0 Off |                  Off |
|  0%   39C    P8    24W / 150W |      0MiB / 12128MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
ubuntu@ubuntu198:~$
```

5 . XPU Test

YOYOWORKS provides one simple test script, and you can download it for a quick test :

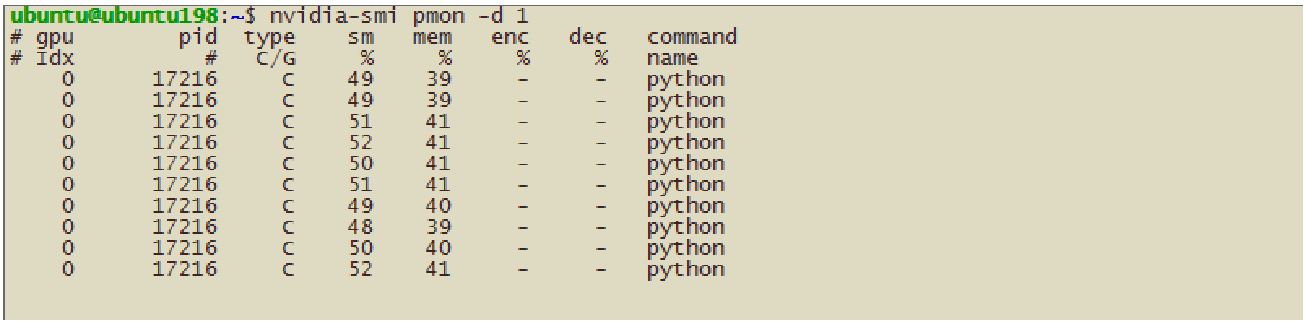
```
$>sudo curl http://www.yoyoworks.com/test/xpu-test.sh -o xpu-test.sh
$>sudo chmod +x ./xpu-test.sh
$>sudo ./xpu-test.sh
```

To check whether the container is running:



```
ubuntu@ubuntu198:~$ sudo docker ps -a --format "table {{.ID}}\t{{.Names}}\t{{.Status}}\t{{.Ports}}"
CONTAINER ID   NAMES     STATUS         PORTS
d809c96f3d94   tf0-1-1   Up 15 hours    6006/tcp, 8888/tcp
ubuntu@ubuntu198:~$
```

Run `nvidia-smi pmon -d 1` to check the vGPU task container status and utilization:



```
ubuntu@ubuntu198:~$ nvidia-smi pmon -d 1
# gpu        pid  type    sm   mem   enc   dec   command
# Idx          #   C/G     %     %     %     %   name
    0      17216     C    49    39     -     -   python
    0      17216     C    49    39     -     -   python
    0      17216     C    51    41     -     -   python
    0      17216     C    52    41     -     -   python
    0      17216     C    50    41     -     -   python
    0      17216     C    51    41     -     -   python
    0      17216     C    49    40     -     -   python
    0      17216     C    48    39     -     -   python
    0      17216     C    50    40     -     -   python
    0      17216     C    52    41     -     -   python
```