DBC computing node deployment

Pre-installation preparation (based on the configured fixed public network ip address), deploy the KVM installation environment

Note: Please uninstall the installed graphics driver before starting, this operation cannot have graphics driver

```
sudo apt-get update
sudo apt-get upgrade -y
sudo apt-get  install qemu-kvm libvirt-clients libvirt-daemon-system bridge-utils
virt-manager ovmf cpu-checker vim -y
```

create and mount the XFS file system

1. Check the hard disk partition

`lsblk`

2. Create a data disk folder, format the hard disk, and mount the hard disk (the data disk mounting directory must be /data)

```
sudo mkdir /data
sudo apt-get install xfsprogs -y
sudo mkfs.xfs -n ftype=1 -f /dev/sdb   (Whether it is sdb here depends on the
situation of lsblk)
sudo mount  -o pquota /dev/sdb /data
sudo chmod 777 /data
sudo echo "/dev/sdb /data     xfs pquota 0 1" >> /etc/fstab
sudo mount -a
```

Determine whether the machine supports virtualization

1. Turn on hardware support

> BIOS open VT-d (search according to the motherboard type browser)
> VT (VT-x) and VT-d support, you need to set related support to enable, which is enabled by default

> Path under normal circumstances: Processor—IIO Configuration—Intel@ VT for Directed I/O(VT-d)

2. Environment dependence, check whether the CPU supports virtualization and whether KVM is available

`egrep -c '(svm|vm)' /proc/cpuinfo`

> CPU detection, if it is displayed as 0, virtualization is not supported

`kvm-ok`

> Check if kvm is available
>
> display INFO: /dev/kvm exists
> KVM acceleration can be used
> Indicates that subsequent operations can be performed. If the display does not match it, please check whether VT-d is turned on correctly

enable system grouping

1. Configure intel_iommu

```
sudo vim /etc/default/grub

#Add in the GRUB_CMDLINE_LINUX_DEFAULT field
intel_iommu=on iommu=pt rd.driver.pre=vfio-pci
#Add in GRUB_CMDLINE_LINUX field
intel_iommu=on iommu=pt rd.driver.pre=vfio-pci
```

2. Configure the module file

```
sudo vim  /etc/modules
#Add the following content:
pci_stub
vfio
vfio_iommu_type1
vfio_pci
kvm
kvm_intel

#Update grub.cfg file
sudo update-grub

#Restart the machine, check whether iommu is correctly enabled (or restart and check after subsequent operations)
dmesg | grep -i iommu

#Display is similar to [3.887539] pci 0000:83:00.1: Adding to iommu group 46 means successful activation
```

isolate GPU resources

1. Set up a blacklist to prevent the card from being occupied

```
sudo vim /etc/modprobe.d/blacklist.conf
#Finally add content:
blacklist snd_hda_intel
blacklist amd76x_edac
blacklist vga16fb
blacklist nouveau
blacklist rivafb
blacklist nvidiafb
blacklist rivatv
```

2. Collect PCI device information

```
lspci -nnv | grep NVIDIA
#Display similar to
17:00.0 VGA compatible controller [0300]: NVIDIA Corporation TU104 [GeForce RTX
2080] [10de:1e82] (rev a1) (prog-if 00 [VGA controller])
17:00.1 Audio device [0403]: NVIDIA Corporation TU104 HD Audio Controller
[10de:10f8] (rev a1)
17:00.2 USB controller [0c03]: NVIDIA Corporation TU104 USB 3.1 Host Controller
[10de:1ad8] (rev a1) (prog-if 30 [XHCI])
17:00.3 Serial bus controller [0c80]: NVIDIA Corporation TU104 USB Type-C UCSI
Controller [10de:1ad9] (rev a1)
65:00.0 VGA compatible controller [0300]: NVIDIA Corporation TU104 [GeForce RTX
2080] [10de:1e82] (rev a1) (prog-if 00 [VGA controller])
65:00.1 Audio device [0403]: NVIDIA Corporation TU104 HD Audio Controller
[10de:10f8] (rev a1)
65:00.2 USB controller [0c03]: NVIDIA Corporation TU104 USB 3.1 Host Controller
[10de:1ad8] (rev a1) (prog-if 30 [XHCI])
65:00.3 Serial bus controller [0c80]: NVIDIA Corporation TU104 USB Type-C UCSI
Controller [10de:1ad9] (rev a1)

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>
#Record all device codes and PCI id (repeated codes are only taken once)
#E.g:
#equipment number:
10de:1e82,10de:10f8,10de:1ad8,10de:1ad9     (Repeat only once)
#PCI interface id (The PCI interface of each machine is different, please note
the record)
17:00.0
17:00.1
17:00.2
17:00.3
65:00.0
65:00.1
65:00.2
65:00.3
```

3. Set up vfio and isolate the GPU for pass-through

```
sudo vim /etc/modprobe.d/vfio.conf
#Write the device code information collected above (if repeated, just write it once):
options vfio-pci ids=10de:1e82,10de:10f8,10de:1ad8,10de:1ad9

sudo vim /etc/modules-load.d/vfio-pci.conf
#Write the following
vfio-pci kvmgt vfio-iommu-type1 vfio-mdev

#Restart the machine
sudo reboot
```

4. Check the GPU status (all interfaces must be queried to prevent it from being occupied by vfio-pci)

```
#Please pay attention to the replacement of PCI interface content!
lspci -vv -s <PCI interface> | grep driver
#E.g:
lspci -vv -s 17:00.0 | grep driver
lspci -vv -s 17:00.1 | grep driver
lspci -vv -s 17:00.2 | grep driver
lspci -vv -s 17:00.3 | grep driver

#No output means there is no driver.
#If Kernel driver in use: vfio-pci is displayed, the isolation is successful
#If the display is similar to Kernel driver in user: snd_hda_intel indicates that the device is occupied by other drivers
```

> If there is a PCI that is not occupied by vfio-pci, please continue to execute, if it has been successfully occupied by vfio-pci, you can skip the next step

If the driver query is Kernel driver in use: vfio-pci, there is no need to operate the following content, please continue to execute if the binding is not successful

1. Unbind the device

> If the driver query shows a non-Kernel driver in user: vfio-pci, unbind the device (each group of IDs must be unbound, the following is only an example, please modify it according to your own query pci interface)

```
#Please pay attention to the replacement of the content, the following command is only for demonstration (need to unbind all occupied graphics card pci interfaces)
sudo -i
sudo echo 0000:17:00.0 > /sys/bus/pci/devices/0000\:17\:00.0/driver/unbind
sudo echo 0000:83:00.0 > /sys/bus/pci/devices/0000\:83\:00.0/driver/unbind
```

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

#Restart the host and check if the GPU is isolated in a different IOMMU group and the vfio driver is being used
#Execute the command to check whether the GPU is isolated in different IOMMU groups
find /sys/kernel/iommu_groups/*/devices/*
#Display grouping is normal

#Re-query PCI (pay attention to replace), if vfio-pci is still not queried or other content is displayed, please perform a next step
lspci -vv -s 17:00.0 | grep driver
```

2. Manually bind the GPU

```
#Execute the command to bind (Note: the content after echo is the ID of the graphics card that the machine queried) The already occupied PCI does not need to be manually bound
sudo -i
sudo echo 10de 1e82 > /sys/bus/pci/drivers/vfio-pci/new_id
sudo echo 10de 2206 >> /sys/bus/pci/drivers/vfio-pci/new_id
……


#Check again after binding (check all items of each card)
lspci -vv -s 17:00.0 | grep driver
#If Kernel driver in use: vfio-pci appears, the binding is successful. If still unsuccessful, please go back and check
```

After confirming that the graphics card of the machine is occupied by vfio-pci, start the libvirtd service and set the boot to start automatically

1. Turn on the virt tcp monitoring service:

```
sudo vim /etc/libvirt/libvirtd.conf
#After the arrow is the modified content: remove the # in front of these three lines, and change sasl to none

#listen_tls = 0 =======>    listen_tls = 0
#listen_tcp = 1 =======>    listen_tcp = 1
#auth_tcp = "sasl"  ======> auth_tcp = "none"

sudo vim /etc/default/libvirtd
#Corresponding modification to the following configuration
libvirtd_opts="-l"
```

2. Start libvirtd and set it to start at boot

sudo systemctl start libvirtd.service
sudo systemctl enable libvirtd.service

Create a dbc user

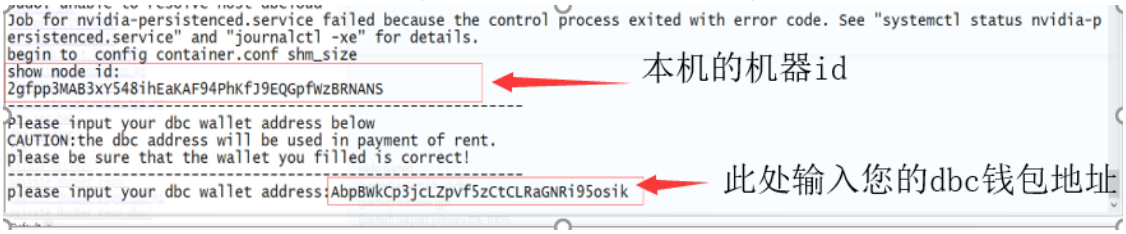
```
sudo wget http://111.44.254.179:22244/install_dbc_ry_machine.sh
sudo chmod +x install_dbc_ry_machine.sh
sudo bash install_dbc_ry_machine.sh

sudo wget http://116.85.24.172:20444/static/add_dbc_user.sh
sudo chmod +x add_dbc_user.sh
sudo ./add_dbc_user.sh dbc
#dbcUser password is set by yourself
```

Install the DBC node program

> Note: need to switch to dbc user installation

```
su - dbc
sudo mkdir install && cd install
sudo wget http://111.44.254.179:22244/install_dbc_ry_machine.sh
sudo bash ./install_dbc_ry_machine.sh -d
sudo bash ./install_dbc_ry_machine.sh -i /home/dbc
#During the installation process, there will be a selection path, select the /data folder (be sure to select /data, otherwise the virtual machine will not be successfully created)
```

> 输入您的钱包地址（如果机器被租用，dbc将直接转入该钱包地址，请确保地址信息正确）



> The first item in the picture is the machine id and the second item enters your wallet address

Restart DBC program & service status check

```
sudo systemctl stop dbc
sudo systemctl start dbc
sudo systemctl status dbc
```

Download the mirror template

Under production optimization, this step can be skipped temporarily, and the latest mirroring will be synchronized after completion

Back up the machine id and private key (very important)

Back up the contents of the following file: vi /home/dbc/0.3.7.3/dbc_repo/dat/node.dat, put it in a safe place, and use it later if you reinstall the system or reinstall DBC

Viewing personal wallet address

`cat /home/dbc/0.3.7.3/dbc_repo/conf/core.conf`

Parameter check

```
#Check the memory, hard disk, graphics card, IP, if you do not see the content of the following picture on the website, it means that the system does not detect the memory or hard disk, you need to manually execute a check command:
sudo bash   /home/dbc/0.3.7.3/dbc_repo/tool/node_info/node_info.sh

# Restart DBC:
sudo systemctl restart dbc
```

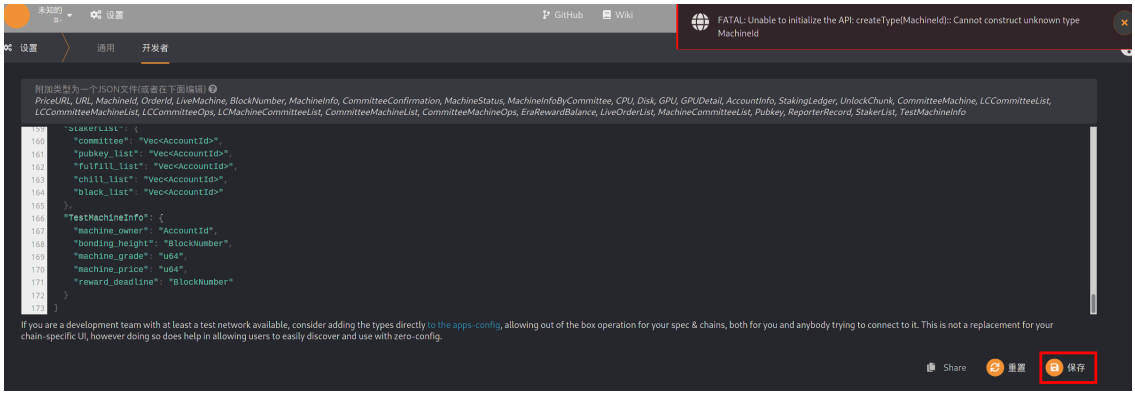
> Perform this step to check whether the parameter acquisition is normal
>
> cat /home/dbc/0.3.7.3/dbc_repo/.dbc_node_info.conf
>
> If the GPU part is displayed as N/A, it can be ignored. Other parts show N/A or empty, please correct it manually and restart DBC

Machine on the chain

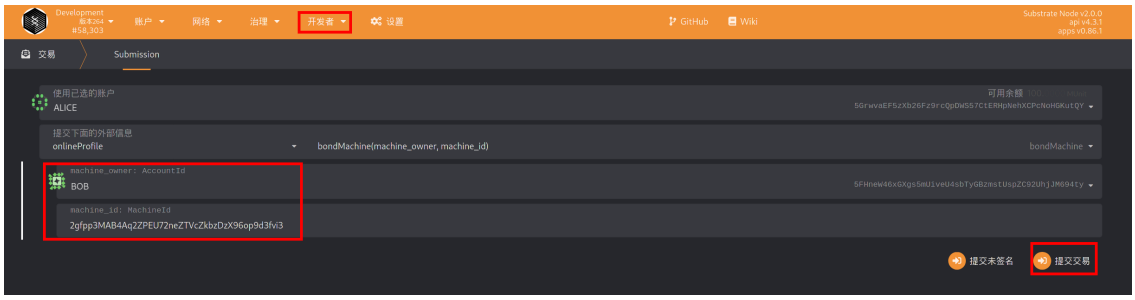
How to bind the machine to get online rewards

Method 1: Binding via web wallet

0. Before binding, please make sure that there is enough balance in the wallet. (Each card is estimated at 100,000 DBC).
1. Open the settings page of the web wallet: `https://www.dbcwallet.io/?rpc=wss%3A%2F%2Finnertest.dbcwallet.io#/settings/developer`
2. Open `https://github.com/DeepBrainChain/DeepBrainChain-MainChain/blob/feature/staking_v3.0.0_online_profile/types.json`, copy the content of `types.json`, and paste it to the web wallet settings page, click save .



3. Refresh the page and wait a while.
4. Navigate to: `Developer`--`Transaction`, and select the `bondMachine` method of the `onlineProfile` module as shown below. Among them, `machine_owner: AccountId` fill in the built-in wallet address in the machine; `machineId` fill in the machine ID you want to bind, and finally click to submit the transaction.



Method 2: add via script

```
git clone https://github.com/DeepBrainChain/DeepBrainChain-MainChain.git
cd DeepBrainChain-MainChain && checkout dev-example
yarn install
node sign_txs.js --port="wss://innertest.dbcwallet.io" --module onlineProfile --
func bondMachine --key "sample split bamboo west visual approve brain fox arch
impact relief smile" 5FHneW46xGXgs5mUiveU4sbTyGBzmstUspZC92UhjJM694ty
2gfpp3MAB4Aq2ZPEU72neZTVcZkbzDzX96op9d3fvi3
```

Among them, `--key` specifies the mnemonic phrase, and the last two parameters are the wallet address bound in the machine and the machine ID