Российская сервисная платформа виртуализации РУСТЭК

CPU-Pining, PCI-E проброс и NUMA

Релизы 2.5 и 2.6

2023

Оглавление

1 Получение информации об узле

Для начала нужно выяснить топологию CPU, NUMA и расположение PCI-E устройств.

Для просмотра процессоров и NUMA узлов на физическом узле выполните команду `lscpu`:

```
fc-n3 ~ # lscpu
Architecture:                    x86_64
CPU op-mode(s):                  32-bit, 64-bit
Byte Order:                      Little Endian
Address sizes:                   46 bits physical, 48 bits virtual
CPU(s):                          40
On-line CPU(s) list:             0-39
Thread(s) per core:              2
Core(s) per socket:              10
Socket(s):                       2
NUMA node(s):                    2   <--- Количество NUMA узлов на узле
Vendor ID:                       GenuineIntel
CPU family:                      6
Model:                           79
Model name:                      Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz
Stepping:                        1
CPU MHz:                         2835.275
CPU max MHz:                     3100.0000
CPU min MHz:                     1200.0000
BogoMIPS:                        4401.31
Virtualization:                  VT-x
L1d cache:                       640 KiB
L1i cache:                       640 KiB
L2 cache:                        5 MiB
L3 cache:                        50 MiB
NUMA node0 CPU(s):               0-9,20-29 <--- Ядра и HT потоки на первом узле
NUMA node1 CPU(s):               10-19,30-39 <--- Ядра и HT потоки на втором узле
Vulnerability Itlb multihit:     KVM: Vulnerable
Vulnerability L1tf:              Mitigation; PTE Inversion; VMX conditional cache flushes, SMT
vulnerable
Vulnerability Mds:               Mitigation; Clear CPU buffers; SMT vulnerable
Vulnerability Meltdown:          Mitigation; PTI
Vulnerability Spec store bypass: Mitigation; Speculative Store Bypass disabled via prctl and
seccomp
Vulnerability Spectre v1:        Mitigation; usercopy/swapgs barriers and __user pointer
sanitization
Vulnerability Spectre v2:        Mitigation; Retpolines, IBPB conditional, IBRS_FW, STIBP
conditional, RSB filling
Vulnerability Srbds:             Not affected
Vulnerability Tsx async abort:   Mitigation; Clear CPU buffers; SMT vulnerable
Flags:                           fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm
constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf pni pcl
                                 mulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16
xtpr pdcm pcid dca sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand
lahf_lm abm 3dnowprefetch cpuid_fault epb cat_l3 cdp_l3 invpcid_single pti intel_
                                 ppin ssbd ibrs ibpb stibp tpr_shadow vnmi flexpriority ept vpid
ept_ad fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm cqm rdt_a rdseed adx smap
intel_pt xsaveopt cqm_llc cqm_occup_llc cqm_mbm_total cqm_mbm_local dtherm ida arat pl
                                 n pts md_clear flush_l1d
```

Для просмотра ядер и привязанных к ним потоков выполните скрипт:

```
fc-n3 ~ # cat $(find /sys/devices/system/cpu -regex ".*cpu[0-
9]+/topology/thread_siblings_list") | sort -n | uniq
0,20
1,21
2,22
3,23
4,24
5,25
6,26
7,27
8,28
9,29
10,30
11,31
12,32
13,33
14,34
15,35
16,36
17,37
18,38
19,39
```

Для просмотра PCI-E сетевых карт и распределение их по NUMA узлам выполните скрипт:

```
fc-n3 ~ # for i in  /sys/class/net/*/device; do
    pci=$(basename "$(readlink $i)")
    if [ -e $i/numa_node ]; then
        echo "NUMA Node: `cat $i/numa_node` ($i): `lspci -s $pci`" ;
    fi
done | sort
NUMA Node: 0 (/sys/class/net/eno1/device): 07:00.0 Ethernet controller: Intel
Corporation I350 Gigabit Network Connection (rev 01) <-- мы будем пробрасывать это
устройство
NUMA Node: 0 (/sys/class/net/eno2/device): 07:00.1 Ethernet controller: Intel
Corporation I350 Gigabit Network Connection (rev 01)
NUMA Node: 0 (/sys/class/net/enp4s0f0/device): 04:00.0 Ethernet controller: Intel
Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection (rev 01)
NUMA Node: 0 (/sys/class/net/enp4s0f1/device): 04:00.1 Ethernet controller: Intel
Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection (rev 01)
NUMA Node: 0 (/sys/class/net/ens9f0/device): 06:00.0 Ethernet controller: Intel
Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection (rev 01)
NUMA Node: 0 (/sys/class/net/ens9f1/device): 06:00.1 Ethernet controller: Intel
Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection (rev 01)
```

Здесь наша сетевая карта расположена на NUMA Node 0, поэтому для CPU-Pining будем использовать ядра, начиная с 1 по 9 и потоки от них с 21 по 29, ядро 0 и его поток 20 оставим для гипервизора.

2 Настройка службы Nova Compute

Nova обрабатывает процессоры узла, используемые для незакрепленных ВМ иначе, чем те, которые используются ВМ с закрепленными ЦП. Первые отслеживаются при размещении с использованием типа ресурса **VCPU** и могут использовать переподписку физических ресурсов ЦП, а вторые отслеживаются с использованием типа ресурса **PCPU**. По умолчанию nova будет сообщать обо всех ЦП узла как о VCPU. Это поведение можно настроить с помощью параметров конфигурационного файла:

- **compute/cpu_shared_set** — указать, какие ЦП узла следует использовать для инвентаризации **VCPU** и запуска незакрепленных ВМ;
- **compute/cpu_dedicated_set** — указать, какие ЦП узла следует использовать для инвентаризации **PCPU** и запуска закрепленных ВМ.

На нашем узле необходимо внести изменения в конфигурационный файл `/etc/nova/nova.conf`. Добавим в секцию `compute` следующие строчки и перезапустим сервис `nova-compute` при помощи команды `/etc/init.d/nova-compute restart`:

```
[compute]
cpu_dedicated_set=1-9,21-29
cpu_shared_set=0,10-20,30-39
```

Мы указываем, что для ВМ с CPU-Pining будем использовать ядра с 1 по 9 и их потоки с 21 по 29. Для остальных ВМ используем ядро 0 и его поток 20, а также ядра с 10 по 19 и их потоки с 30 по 39.

Проверим, какие ресурсы теперь отдаёт наш гипервизор:

```
fc-n3 ~ #  openstack --os-cloud rustack_system hypervisor list --matching fc-n3 -c ID -f value
7ed7d761-e050-4234-a669-149a9033f106

fc-n3 ~ # openstack --os-cloud rustack_system resource provider inventory list  7ed7d761-e050-
4234-a669-149a9033f106
+----------------+------------------+----------+----------+----------+-----------+-------+
| resource_class | allocation_ratio | min_unit | max_unit | reserved | step_size | total |
+----------------+------------------+----------+----------+----------+-----------+-------+
| VCPU           |              8.0 |        1 |       22 |        0 |         1 |    22 |
| MEMORY_MB      |              2.0 |        1 |    64370 |    33766 |         1 | 64370 |
| DISK_GB        |              1.0 |        1 |      118 |        0 |         1 |   118 |
| PCPU           |              1.0 |        1 |       18 |        0 |         1 |    18 |
+----------------+------------------+----------+----------+----------+-----------+-------+
```

Видно, что у нас осталось 22 VCPU с возможностью переподписки, и появился новый ресурс PCPU в количестве 18 штук и без переподписки.

3 Настройка ядра системы

Чтобы системные процессы гипервизора не могли использовать наши ядра, выделенные для ВМ с CPU-Pining, мы должны указать их номера при загрузке ядра ОС.

Для этого нам необходимо в файле `/boot/grub/grub.cfg` найти строчку загрузки ядра и добавить в нее параметр `isolcpus` с указанием ядер для CPU-Pining. Рекомендуется менять параметр загрузки только для секции `menuentry 'Rustack 2021.2.5 GNU/Linux'`. В остальных подсекциях параметры загрузки ядра менять не надо.

```
linux   /vmlinuz-5.4.197 root=UUID=896dea2a-907d-4039-bf74-fd38243c446a ro
scsi_mod.scan=sync intel_iommu=on iommu=pt transparent_hugepage=never cma=128M tsx=off
kvm.nx_huge_pages=off isolcpus=1-9,21-29
```

После этого нам необходимо перегрузить ВМ.

После перезагрузки можно проверить передачу параметра в ядро при помощи следующих команд:

```
fc-n3 ~ # cat /var/log/dmesg | grep isol
[    0.000000] Command line: BOOT_IMAGE=/vmlinuz-5.4.197 root=UUID=896dea2a-907d-4039-
bf74-fd38243c446a ro scsi_mod.scan=sync intel_iommu=on iommu=pt
transparent_hugepage=never cma=128M tsx=off kvm.nx_huge_pages=off isolcpus=1-9,21-29
[    0.359333] Kernel command line: BOOT_IMAGE=/vmlinuz-5.4.197 root=UUID=896dea2a-
907d-4039-bf74-fd38243c446a ro scsi_mod.scan=sync intel_iommu=on iommu=pt
transparent_hugepage=never cma=128M tsx=off kvm.nx_huge_pages=off isolcpus=1-9,21-29
[    0.585226] Kernel/User page tables isolation: enabled
fc-n3 ~ # cat /sys/devices/system/cpu/isolated
1-9,21-29
```

Теперь необходимо создать нужную нам конфигурацию, которая будет использовать CPU-Pining, для этого используются метаданные `hw:cpu_policy.` Существует три политики: выделенная (**dedicated**), смешанная (**mixed**) и общая (**shared**). Общая (shared) политика используется по умолчанию и указывает на то, что у ВМ нет закрепленных ЦП.

Для правильного выделения ресурсов рекомендуется указывать ресурс PCPU в необходимом количестве — это позволит избежать создания дополнительных агрегатов.

За использование потоков отвечают метаданные `hw:cpu_thread_policy.` Существует три политики: требуется SMT (**require**), без SMT (**isolate**) и предпочтительно (**prefer**), в этом случае ВМ можно запустить как на узле с SMT так и без.

Создадим тестовую конфигурацию со следующими параметрами, указав необходимость SMT и выделения двух pCPU:

```
fc-n1 ~ # openstack --os-cloud rustack_system flavor create --vcpus 2 --ram 1024 --public --property
hw:cpu_thread_policy=require --property resources:PCPU=2 cpu_pinned
+----------------------------+-------------------------------------------------------------------------------+
| Field                      | Value                                                                         |
+----------------------------+-------------------------------------------------------------------------------+
| OS-FLV-DISABLED:disabled   | False                                                                         |
| OS-FLV-EXT-DATA:ephemeral  | 0                                                                             |
| description                | None                                                                          |
| disk                       | 0                                                                             |
| id                         | 4cc07dac-46cb-4902-9a94-49a6d07923d5                                          |
```

```
| name                        | cpu_pinned                                                                                |
| os-flavor-access:is_public | True                                                                                      |
| properties                  | hw:cpu_thread_policy='require', resources:PCPU='2' |
| ram                         | 1024                                                                                      |
| rxtx_factor                 | 1.0                                                                                       |
| swap                        |                                                                                           |
| vcpus                       | 2                                                                                         |
+-----------------------------+-------------------------------------------------------------------------------------------+
```

Теперь можно создать тестовую ВМ из портала с указанием этой конфигурации. После создания узнаем её ID в панели информации и посмотрим на ВМ статистику по CPU:

```
fc-n3 ~ # virsh vcpuinfo 19c290ba-ddbe-434c-bd57-3dc63d018e47
VCPU:           0
CPU:            1
State:          running
CPU time:       11.2s
CPU Affinity:   -y--------------------------------------

VCPU:           1
CPU:            21
State:          running
CPU time:       16.3s
CPU Affinity:   --------------------y------------------


fc-n3 ~ # taskset -cp `pgrep -f 19c290ba-ddbe-434c-bd57-3dc63d018e47` <-- дополнительно можно
посмотреть и через такие команды
pid 33022's current affinity list: 1,21
```

Мы видим привязку vCPU к 1 и 21 pCPU. Наш vCPU привязан к одному физическому ядру 1 и к его потоку 21.

4 Проброс PCI-E устройств

Для проброса PCI-E устройства внутрь ВМ необходимо узнать его Vendor и Product ID, это можно сделать, используя команду `lspci`:

```
fc-n3 ~ # lspci -nn | grep Ether
04:00.0 Ethernet controller [0200]: Intel Corporation 82599ES 10-Gigabit SFI/SFP+
Network Connection [8086:10fb] (rev 01)
04:00.1 Ethernet controller [0200]: Intel Corporation 82599ES 10-Gigabit SFI/SFP+
Network Connection [8086:10fb] (rev 01)
06:00.0 Ethernet controller [0200]: Intel Corporation 82599ES 10-Gigabit SFI/SFP+
Network Connection [8086:10fb] (rev 01)
06:00.1 Ethernet controller [0200]: Intel Corporation 82599ES 10-Gigabit SFI/SFP+
Network Connection [8086:10fb] (rev 01)
07:00.0 Ethernet controller [0200]: Intel Corporation I350 Gigabit Network Connection
[8086:1521] (rev 01)
07:00.1 Ethernet controller [0200]: Intel Corporation I350 Gigabit Network Connection
[8086:1521] (rev 01)
```

Мы собираемся пробрасывать Intel Corporation I350 Gigabit Network Connection, у которого Vendor и Product ID 8086:1521.

Далее необходимо проверить права на устройства для работы с ВМ (это пользователь **nova)**:

```
# ls -al /dev/vfio/
total 0
drwxr-xr-x  2 root root      100 мар 29 15:59 .
drwxr-xr-x 15 root root     3940 мар 29 15:59 ..
crw-------  1 root root 240,   0 мар 29 15:59 94
crw-------  1 root root 240,   1 мар 29 15:59 95
crw-rw-rw-  1 root root  10, 196 мар 29 15:57 vfio
```

При отсутствии прав необходимо создать файл `/etc/udev/rules.d/vfio.rules` с содержимым:

```
SUBSYSTEM=="vfio", OWNER="root", GROUP="nova", MODE="0666"
```

Теперь для корректной работы проброса необходимо отключить системный драйвер для этой сетевой карты и вместо него подключить драйвер `vfio.` Для этого необходимо добавить в файл `/etc/modprobe.d/vfio.conf` строчку с указанием нашего устройства:

```
fc-n3 ~ # echo "options vfio-pci ids=8086:1521" >> /etc/modprobe.d/vfio.conf
```

После чего перегрузить ВМ.

В случае с сетевым картами рекомендуется отключить драйвер скриптом. Для этого необходимо создать файл `/etc/local.d/unbind-igb.start`, установить на него права на запуск и добавить следующее содержимое:

```
#!/bin/sh
echo "0000:07:00.0" > /sys/bus/pci/devices/0000\:07\:00.0/driver/unbind
echo "vfio-pci"  >  /sys/bus/pci/devices/0000\:07\:00.0/driver_override
echo "0000:07:00.0" > /sys/bus/pci/drivers_probe


echo "0000:07:00.1" > /sys/bus/pci/devices/0000\:07\:00.1/driver/unbind
echo "vfio-pci"  >  /sys/bus/pci/devices/0000\:07\:00.1/driver_override
echo "0000:07:00.1" > /sys/bus/pci/drivers_probe
```

```
fc-n3 ~ # chmod a+x /etc/local.d/unbind-igb.start
```

После чего перезагрузить ВМ. Проверить правильный ли драйвер используется можно следующим образом:

```
fc-n3 ~ # lspci -nnk -d 8086:1521
07:00.0 Ethernet controller [0200]: Intel Corporation I350 Gigabit Network Connection
[8086:1521] (rev 01)
        DeviceName:  Onboard LAN
        Subsystem: Super Micro Computer Inc I350 Gigabit Network Connection
[15d9:1521]
        Kernel driver in use: vfio-pci <-- Должен быть vfio-pci
        Kernel modules: igb
07:00.1 Ethernet controller [0200]: Intel Corporation I350 Gigabit Network Connection
[8086:1521] (rev 01)
        DeviceName:  Onboard LAN
        Subsystem: Super Micro Computer Inc I350 Gigabit Network Connection
[15d9:1521]
        Kernel driver in use: vfio-pci <-- Должен быть vfio-pci
        Kernel modules: igb
```

Теперь необходимо настроить службу Nova на проброс устройства. Для этого на вычислительном узле, где находится карта, в файл `/etc/nova/nova.conf` нужно добавить следующие строки:

```
[pci]
passthrough_whitelist = { "vendor_id": "8086", "product_id": "1521" }
alias = { "vendor_id":"8086", "product_id":"1521", "device_type":"type-PCI",
"name":"igb" }
```

После чего перезапустить службу nova-compute при помощи команды `/etc/init.d/nova-compute restart`.

На контроллерах, если у них не будет проброса, достаточно добавить в файл `/etc/nova/nova.conf` следующие строчки:

```
[pci]
alias = { "vendor_id":"8086", "product_id":"1521", "device_type":"type-PF",
"name":"igb" }
```

Если контроллеры тоже участвуют в пробросе, то конфигурация должна быть как на физическом узле.

Важно отметить добавление поля `device_type`. Это необходимо, поскольку данное PCI-устройство поддерживает SR-IOV. Служба **nova-compute** классифицирует устройства по одному из трёх типов, в зависимости от возможностей, о которых сообщают устройства:

- **type-PF** — устройство поддерживает SR-IOV и является родительским или корневым устройством;
- **type-VF** — устройство является дочерним устройством устройства, поддерживающего SR-IOV;
- **type-PCI** — устройство не поддерживает SR-IOV.

После чего перезапустить службу nova-api командой `/etc/init.d/uwsgi.nova-api restart.`

В планировщике nova (nova-scheduler) необходимо указать дополнительный фильтр. Для этого в файле `/etc/nova/nova.conf` на контроллерах, в строчке `enabled_filters` надо добавить через запятую значение `PciPassthroughFilter` и перезапустить службу nova-scheduler командой `/etc/init.d/nova-scheduler restart.`

Теперь добавим в нашу конфигурацию запрос на выделение устройства `igb` с помощью команды:

```
openstack --os-cloud=rustack_system flavor set cpu_pinned --property
"pci_passthrough:alias"="igb:1"
```

Попробуем создать ВМ с этой конфигурацией из портала и проверим проброс устройства.