

RHEL Week - VirtIO-Win-NetKVM - UDP Segmentation Offload

本文档将使用中文完整记录测试流程和结果，方便大家查阅。

RHEL Week - VirtIO-Win-NetKVM - UDP Segmentation Offload

一、客户端配置

1. 系统环境
2. 网卡驱动版本
3. iperf3 测试工具
4. Wireshark & Winpcap 抓包工具

二、服务端配置

1. 系统环境
2. 开启 Libvirt 服务
3. 开启 iperf3 server 端
4. 调整 MTU 数值

三、效果测试

- 1/3: 测试**关闭** UDP Segmentation Offload 的 UDP 带宽
- 2/3: 测试**打开** UDP Segmentation Offload 的 UDP 带宽 with 8972B data (Standard Mode)
- 3/3: 测试**打开** UDP Segmentation Offload 的 UDP 带宽 with 65507B data (Stress-test mode)

四、可能的提问。

Q0: CPU 利用率的问题

A0: 关闭 USO 后 CPU 利用率 8%

A0: 打开 USO 后 CPU 利用率 7%

Q1: RHEL 的 UDP offload 支持吗？

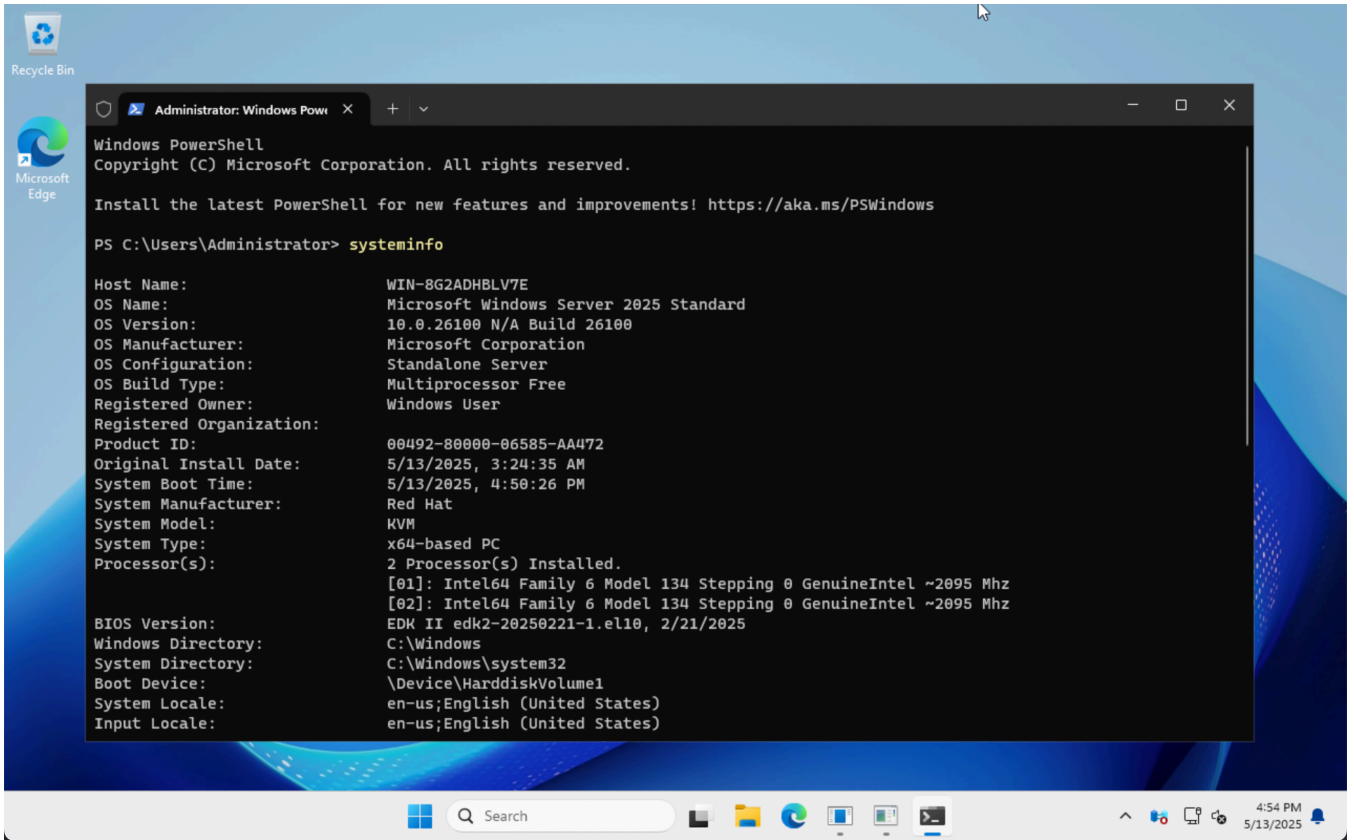
A1: 我不测试 RHEL 不是很清楚。昨天和 RHEL network 的 feature owner 确定还不支持。可以通过 ethtool 列举网卡的选项。

Q2: Windows 的 TCP offload 支持吗？

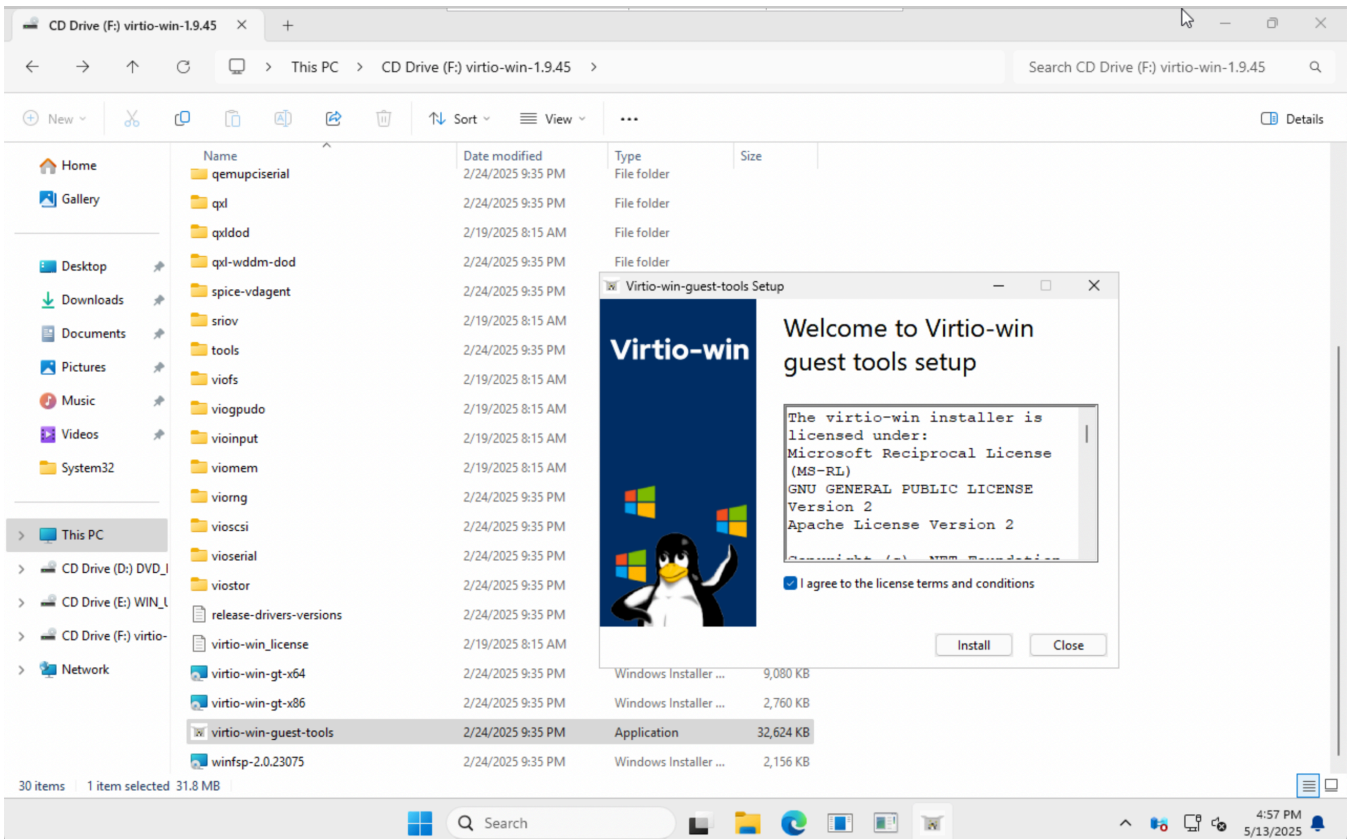
A2: 支持的

一、客户端配置

1. 系统环境

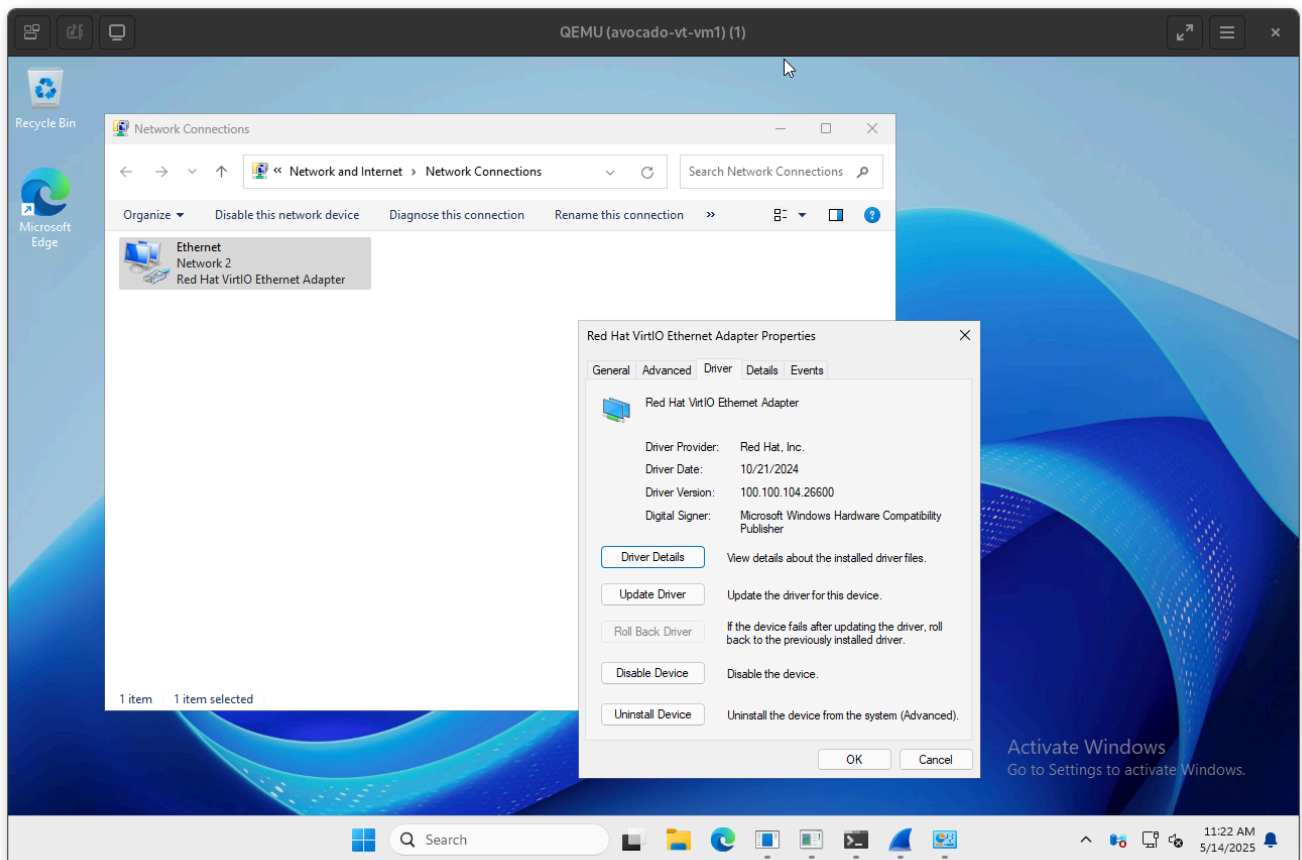


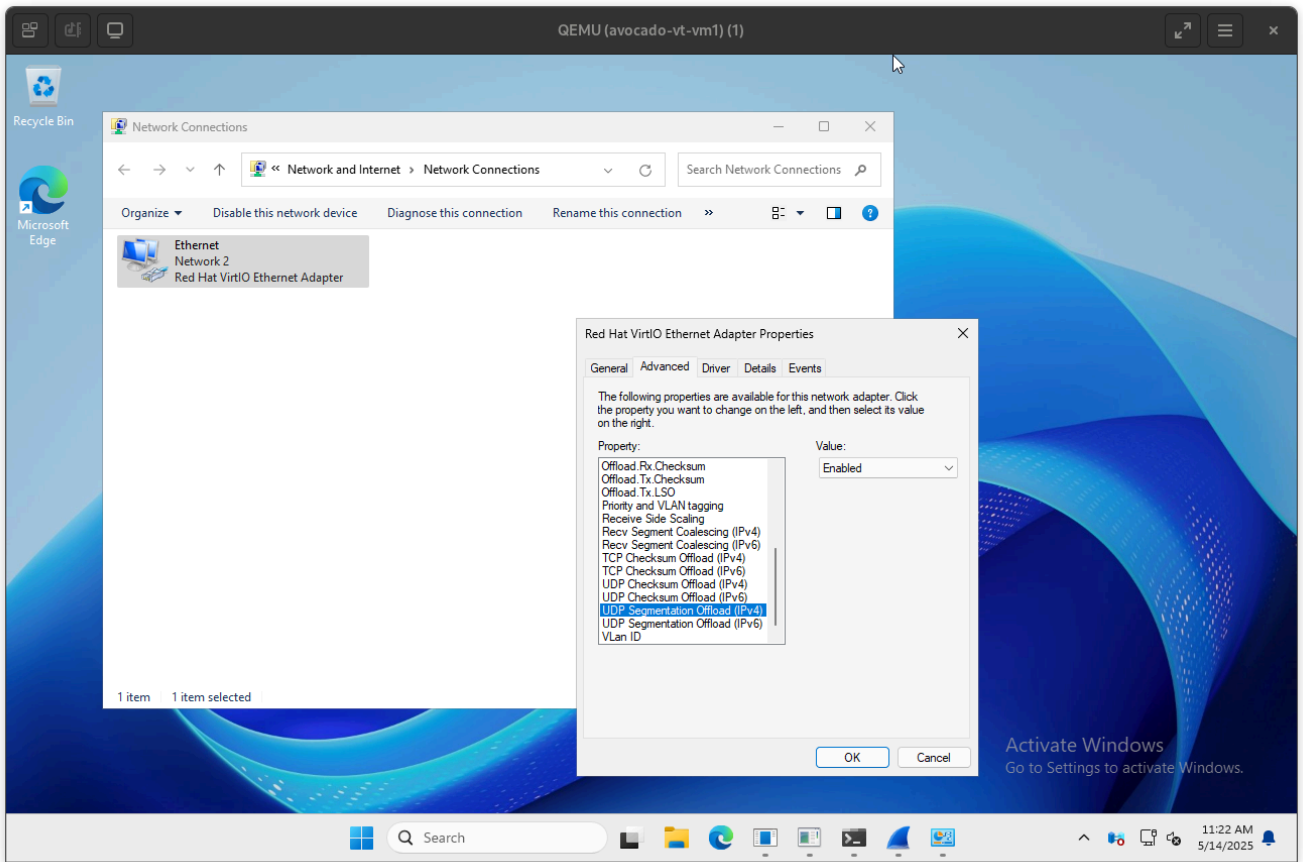
2. 网卡驱动版本



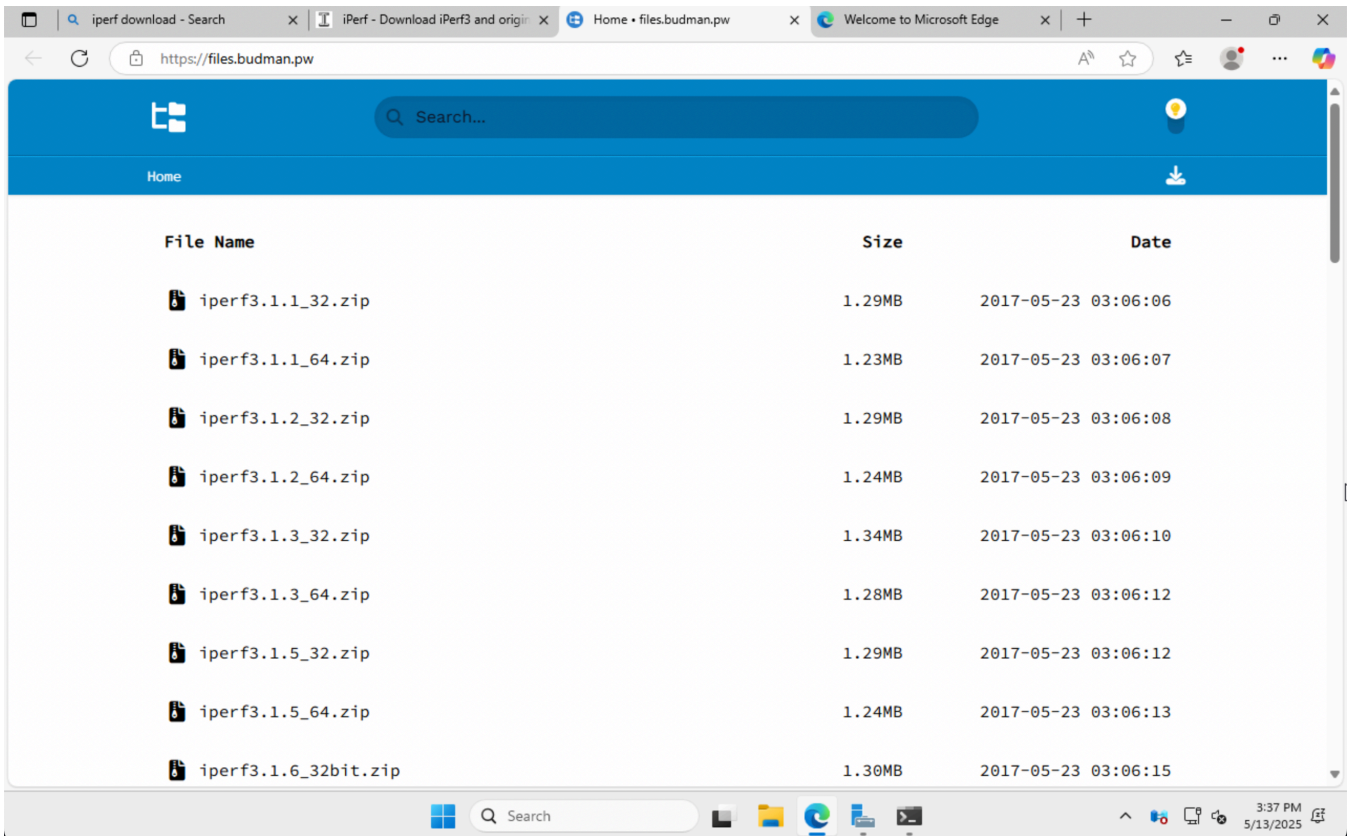
Tips: 目前客户可以拿到的最新的版本是：`virtio-win-1.9.45-0.el10_0.rpm` 需要将它解压后得到 `virtio-win-1.9.45-0.el10_0.iso` 并挂载到虚拟机内。

```
Name       : virtio-win
Version    : 1.9.45
Release    : 0.el10_0
Architecture : noarch
Size       : 258 M
Source     : virtio-win-1.9.45-0.el10_0.src.rpm
Repository : beaker-AppStream
Summary    : VirtIO para-virtualized drivers for Windows(R)
URL        : http://www.redhat.com/
License    : Apache-2.0 AND BSD-3-Clause AND GPL-2.0-only AND GPL-2.0-or-later
Description : VirtIO para-virtualized Windows(R) drivers for 32-bit and 64-bit
           : Windows(R) guests.
```

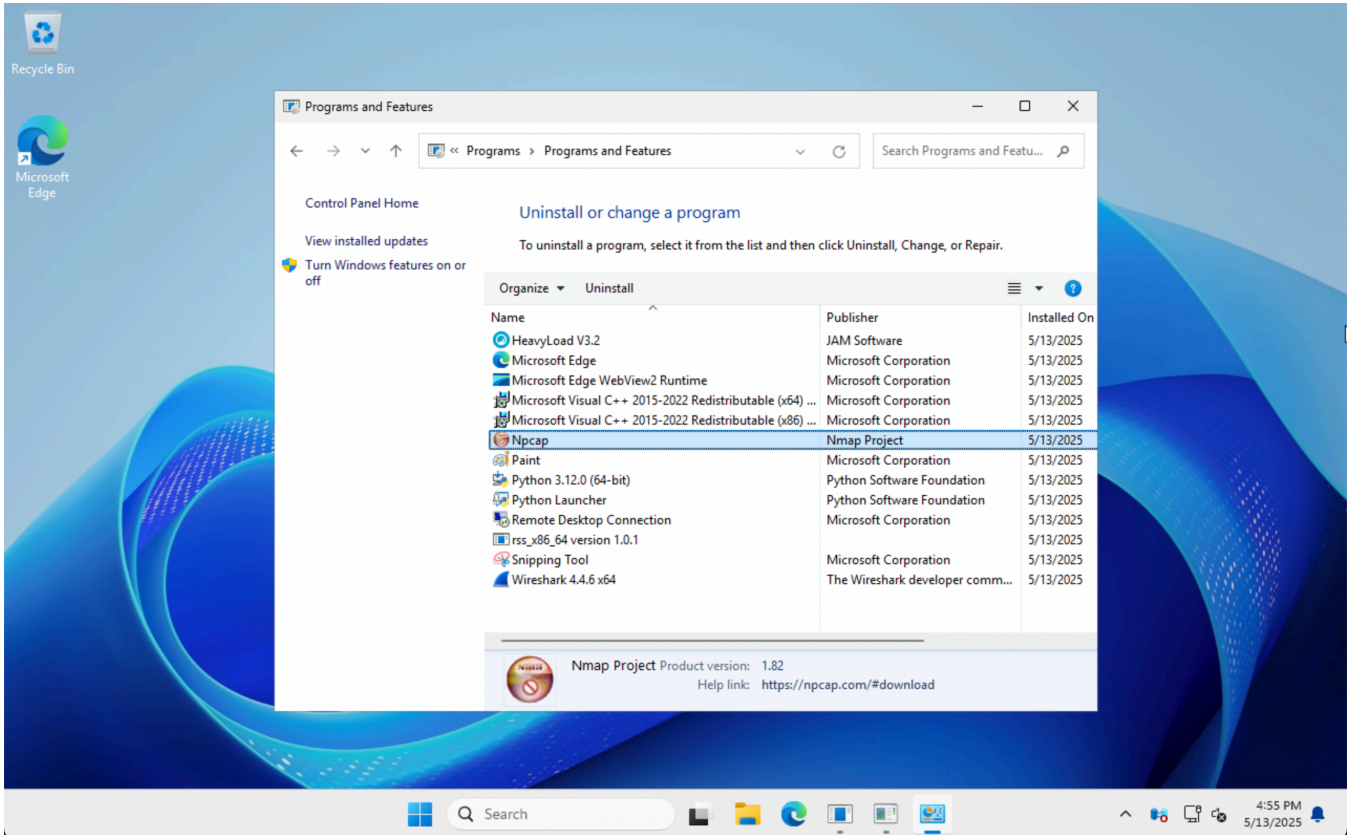




3. iperf3 测试工具



4. Wireshark & Winpcap 抓包工具



二、服务端配置

1. 系统环境

```
[root@dell-per750-51 ~]# cat /etc/redhat-release
Red Hat Enterprise Linux release 10.1 Beta (Coughlan)
[root@dell-per750-51 ~]# uname -r
6.12.0-74.el10.x86_64
[root@dell-per750-51 ~]# rpm -q qemu-kvm
qemu-kvm-10.0.0-1.el10.x86_64
```

2. 开启 Libvirt 服务

```
[root@dell-per750-51 ~]# systemctl enable libvirtd # 设置开机自启
[root@dell-per750-51 ~]# systemctl start libvirtd # 立即启动服务
[root@dell-per750-51 ~]# virsh net-list # 检查 libvirt Network 是否存在
Name      State    Autostart  Persistent
-----
default   active  yes        yes

[root@dell-per750-51 ~]# ip addr show dev virbr0 # Linux 软件桥接 (bridge) 设备
16: virbr0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc htb state UP group default qlen
1000
    link/ether 52:54:00:a7:62:4c brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global noprefixroute virbr0
        valid_lft forever preferred_lft forever
[root@dell-per750-51 ~]#
```

3. 开启 iperf3 server 端

```
[root@dell-per750-51 ~]# iperf3 -s -p 12345
```

```
-----
Server listening on 12345 (test #1)
-----
```

4. 调整 MTU 数值

“MTU”是“Maximum Transmission Unit”的缩写，中文通常称为“最大传输单元”。链路层 MTU 决定了“单帧”在一个网络跳（交换机→路由器→网卡）上传输时的最大长度（例如 1500 B 或 9000 B）。它指的是在一条链路层（例如以太网、Wi-Fi）上，单个数据帧（或包）的最大大小（以字节为单位），超过这个大小就必须拆分或分段才能发送。

- 默认 MTU: 1500 字节

这是绝大多数传统以太网接口（包括大部分交换机和路由器）出厂时的标准设置，适用于 IPv4/IPv6 的一般流量。

- Jumbo Frame（巨型帧） MTU: 常见 9000 字节

许多现代网卡和交换机支持将 MTU 调到更大，一般在 9000 B 左右，以降低分片开销、提升大流量吞吐（例如存储网络、虚拟化环境中常用）。

只要网络全链路都支持 jumbo frame，把 MTU 调大就能显著提速。

```
[root@dell-per750-51 ~]# ip link set virbr0 mtu 1500
[root@dell-per750-51 ~]# ip addr show dev virbr0
16: virbr0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc htb state UP group default qlen 1000
    link/ether 52:54:00:a7:62:4c brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global noprefixroute virbr0
        valid_lft forever preferred_lft forever
[root@dell-per750-51 ~]#
[root@dell-per750-51 ~]# ip link set tap0 mtu 1500
[root@dell-per750-51 ~]# ip addr show dev tap0
31: tap0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq master virbr0 state UNKNOWN
group default qlen 1000
    link/ether 7e:59:e2:17:88:2e brd ff:ff:ff:ff:ff:ff
    inet6 fe80::7c59:e2ff:fe17:882e/64 scope link proto kernel_ll
        valid_lft forever preferred_lft forever
```

Tips: QEMU/KVM 给虚拟机分配网卡时，在 Host 创建一个 tap 二层设备，让 VM 的虚拟网卡被桥接到宿主机网络。没有开启虚拟机之前将不会存在。

三、效果测试

1/3: 测试关闭 UDP Segmentation Offload 的 UDP 带宽

```
Administrator: Windows Powe...
PS C:\> Get-NetAdapterAdvancedProperty -Name "Ethernet" | Where-Object DisplayName -Match "UDP Segmentation Offload"

Name                DisplayName                DisplayValue                RegistryKeyword RegistryValue
-----                -
Ethernet            UDP Segmentation Offload (I... Disabled                *UseIPv4         {0}
Ethernet            UDP Segmentation Offload (I... Disabled                *UseIPv6         {0}

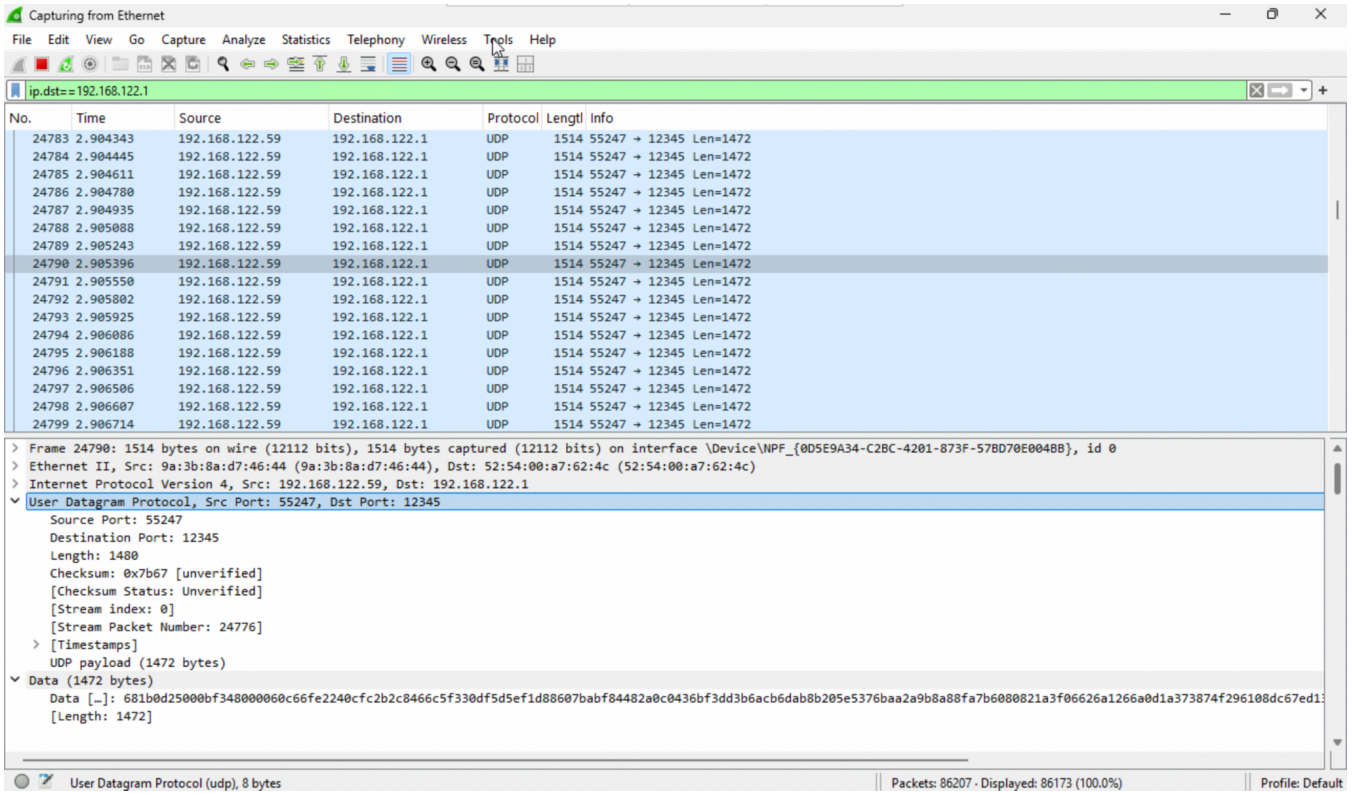
PS C:\> .\iperf3.exe -c 192.168.122.1 -u -l 1472 -b 1G -p 12345
Connecting to host 192.168.122.1, port 12345
[ 4] local 192.168.122.59 port 53937 connected to 192.168.122.1 port 12345
[ ID] Interval            Transfer            Bandwidth            Total Datagrams
[ 4] 0.00-1.00 sec          14.3 MBytes        120 Mbits/sec        10189
[ 4] 1.00-2.00 sec          14.6 MBytes        123 Mbits/sec        10408
[ 4] 2.00-3.00 sec          14.7 MBytes        123 Mbits/sec        10478
[ 4] 3.00-4.00 sec          14.6 MBytes        122 Mbits/sec        10401
[ 4] 4.00-5.00 sec          14.4 MBytes        121 Mbits/sec        10255
[ 4] 5.00-6.00 sec          14.8 MBytes        124 Mbits/sec        10559
[ 4] 6.00-7.00 sec          13.7 MBytes        115 Mbits/sec        9743
[ 4] 7.00-8.00 sec          14.3 MBytes        120 Mbits/sec        10212
[ 4] 8.00-9.00 sec          14.3 MBytes        120 Mbits/sec        10206
[ 4] 9.00-10.00 sec         14.3 MBytes        120 Mbits/sec        10163
-----
[ ID] Interval            Transfer            Bandwidth            Jitter            Lost/Total Datagrams
[ 4] 0.00-10.00 sec         144 MBytes         121 Mbits/sec        0.001 ms          0/102614 (0%)
[ 4] Sent 102614 datagrams

iperf Done.
PS C:\>
```

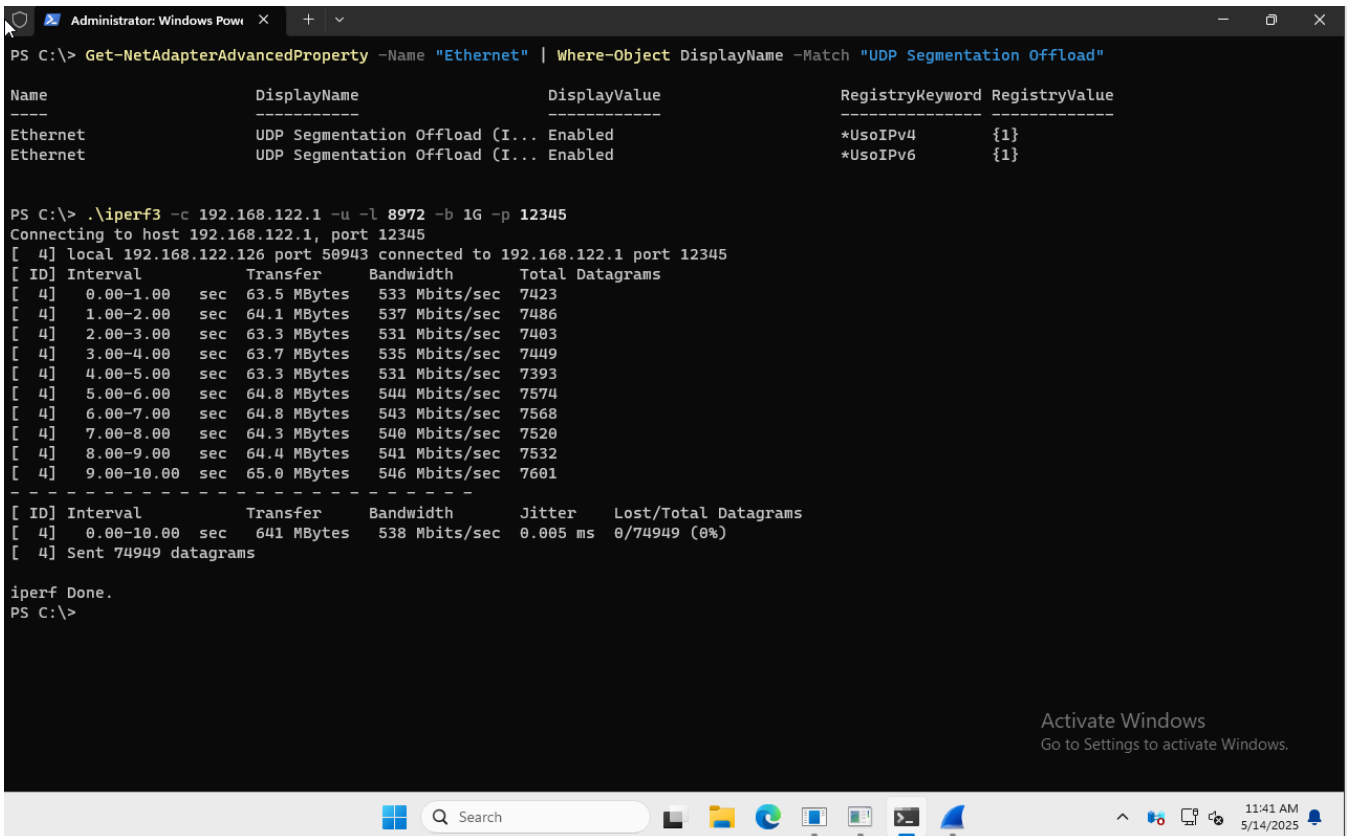
Attention: UDP 报文（包括 IP 头+UDP 头+UDP 载荷）总长度 ≤ 本跳链路的 MTU，就不会触发 IP 分片。

IPv4 + UDP

- IP 头 20 B, UDP 头 8 B
- 最大 UDP 有效载荷 ≈ 1500 - 20 - 8 = **1472 B**



2/3: 测试打开 UDP Segmentation Offload 的 UDP 带宽 with 8972B data (Standard Mode)



Attention: UDP 报文 (包括 IP 头+UDP 头+UDP 载荷) 总长度 \leq 本跳链路的 MTU, 就不会触发 IP 分片。

- IP 头 20 B, UDP 头 8 B
- 最大 UDP 有效载荷 $\approx 9000 - 20 - 8 = 8972$ B

The screenshot shows a Wireshark capture of network traffic. The main pane displays a list of UDP packets. The selected packet (No. 4471) has a source of 192.168.122.126 and a destination of 192.168.122.1. The protocol is UDP, and the length is 134 bytes. The info pane shows the details of the User Datagram Protocol (UDP) packet, including the source and destination ports (60952 and 12345) and the payload length (8972 bytes).

No.	Time	Source	Destination	Protocol	Length	Info
4471	10.057762	192.168.122.126	192.168.122.1	UDP	134	60952 → 12345 Len=8972
4471	10.057904	192.168.122.126	192.168.122.1	UDP	134	60952 → 12345 Len=8972
4471	10.058045	192.168.122.126	192.168.122.1	UDP	134	60952 → 12345 Len=8972
4471	10.058188	192.168.122.126	192.168.122.1	UDP	134	60952 → 12345 Len=8972
4471	10.058327	192.168.122.126	192.168.122.1	UDP	134	60952 → 12345 Len=8972
4471	10.058473	192.168.122.126	192.168.122.1	UDP	134	60952 → 12345 Len=8972
4471	10.058618	192.168.122.126	192.168.122.1	UDP	134	60952 → 12345 Len=8972
4471	10.058759	192.168.122.126	192.168.122.1	UDP	134	60952 → 12345 Len=8972
4471	10.058900	192.168.122.126	192.168.122.1	UDP	134	60952 → 12345 Len=8972
4471	10.059040	192.168.122.126	192.168.122.1	UDP	134	60952 → 12345 Len=8972
4471	10.059180	192.168.122.126	192.168.122.1	UDP	134	60952 → 12345 Len=8972
4471	10.059321	192.168.122.126	192.168.122.1	UDP	134	60952 → 12345 Len=8972
4472	10.059467	192.168.122.126	192.168.122.1	UDP	134	60952 → 12345 Len=8972
4472	10.059609	192.168.122.126	192.168.122.1	UDP	134	60952 → 12345 Len=8972
4472	10.059749	192.168.122.126	192.168.122.1	UDP	134	60952 → 12345 Len=8972
4472	10.059889	192.168.122.126	192.168.122.1	UDP	134	60952 → 12345 Len=8972

Frame 447190: 134 bytes on wire (1072 bits), 134 bytes captured (1072 bits) on interface \Device\NPF_{905C832E-D8FD-46DC-BCFC-C207B4182FD2}, id 0
 Ethernet II, Src: 9a:08:21:9b:1b:15 (9a:08:21:9b:1b:15), Dst: 52:54:00:a7:62:4c (52:54:00:a7:62:4c)
 Internet Protocol Version 4, Src: 192.168.122.126, Dst: 192.168.122.1
 User Datagram Protocol, Src Port: 60952, Dst Port: 12345
 Data (8972 bytes)
 Data [..]: 68241140000220750000f98a519df7ee6b0ba801fac50de656b27c916b3710e185199fed8ba5e9ce2231b34c112222c1dca2d188f3afee5ec7a7757b187d6c9d96fc7b129254e0c4969400a7b6236982...
 [Length: 8972]

3/3: 测试打开 UDP Segmentation Offload 的 UDP 带宽 with 65507B data (Stress-test mode)

The screenshot shows a Windows command prompt window. The first command is used to check the configuration of UDP Segmentation Offload for the Ethernet adapter. The output shows that it is enabled for both IPv4 and IPv6.

```
PS C:\> Get-NetAdapterAdvancedProperty -Name "Ethernet" | Where-Object DisplayName -Match "UDP Segmentation Offload"
```

Name	DisplayName	DisplayValue	RegistryKeyword	RegistryValue
Ethernet	UDP Segmentation Offload (I...	Enabled	*UseIPv4	{1}
Ethernet	UDP Segmentation Offload (I...	Enabled	*UseIPv6	{1}

The second command runs iperf3 in stress-test mode, connecting to 192.168.122.1 on port 12345 with a window size of 65507 bytes and a payload size of 16 bytes.

```
PS C:\> .\iperf3.exe -c 192.168.122.1 -u -l 65507 -b 16 -p 12345
```

The output shows the connection details and the performance results over a 10-second interval.

```
Connecting to host 192.168.122.1, port 12345
[ 4] local 192.168.122.59 port 65296 connected to 192.168.122.1 port 12345
[ ID] Interval      Transfer    Bandwidth  Total Datagrams
[ 4] 0.00-1.00 sec  101 MBytes  844 Mbits/sec  1611
[ 4] 1.00-2.00 sec  118 MBytes  993 Mbits/sec  1895
[ 4] 2.00-3.00 sec  118 MBytes  989 Mbits/sec  1885
[ 4] 3.00-4.00 sec  118 MBytes  987 Mbits/sec  1884
[ 4] 4.00-5.00 sec  116 MBytes  970 Mbits/sec  1851
[ 4] 5.00-6.00 sec  116 MBytes  973 Mbits/sec  1858
[ 4] 6.00-7.00 sec  117 MBytes  981 Mbits/sec  1871
[ 4] 7.00-8.00 sec  117 MBytes  985 Mbits/sec  1878
[ 4] 8.00-9.00 sec  119 MBytes  1.00 Gbits/sec  1910
[ 4] 9.00-10.00 sec 117 MBytes  978 Mbits/sec  1867
-----
[ ID] Interval      Transfer    Bandwidth  Jitter    Lost/Total Datagrams
[ 4] 0.00-10.00 sec 1.13 GBytes  970 Mbits/sec  0.037 ms  0/18510 (0%)
[ 4] Sent 18510 datagrams
```

The test concludes with the message "iperf Done." and the prompt "PS C:\> |".

Attention: UDP 报文 (包括 IP 头 + UDP 头 + UDP 载荷) 总长度 > 本跳链路的 MTU, 会触发 IP 分片。

IPv4 + UDP

- IP 头 20 B, UDP 头 8 B
- 最大 UDP 有效载荷 $\approx 65\,535 - 20 - 8 = 65\,507\text{ B}$

The screenshot shows a Wireshark interface with a capture filter of 'ip.dst==192.168.122.1'. The packet list pane displays a series of UDP packets from source 192.168.122.59 to destination 192.168.122.1, all with source port 55248 and destination port 12345. The selected packet (No. 4212) is expanded to show its structure: Ethernet II, Internet Protocol Version 4, and User Datagram Protocol. The UDP payload is 65507 bytes, and the data field contains a long hexadecimal string.

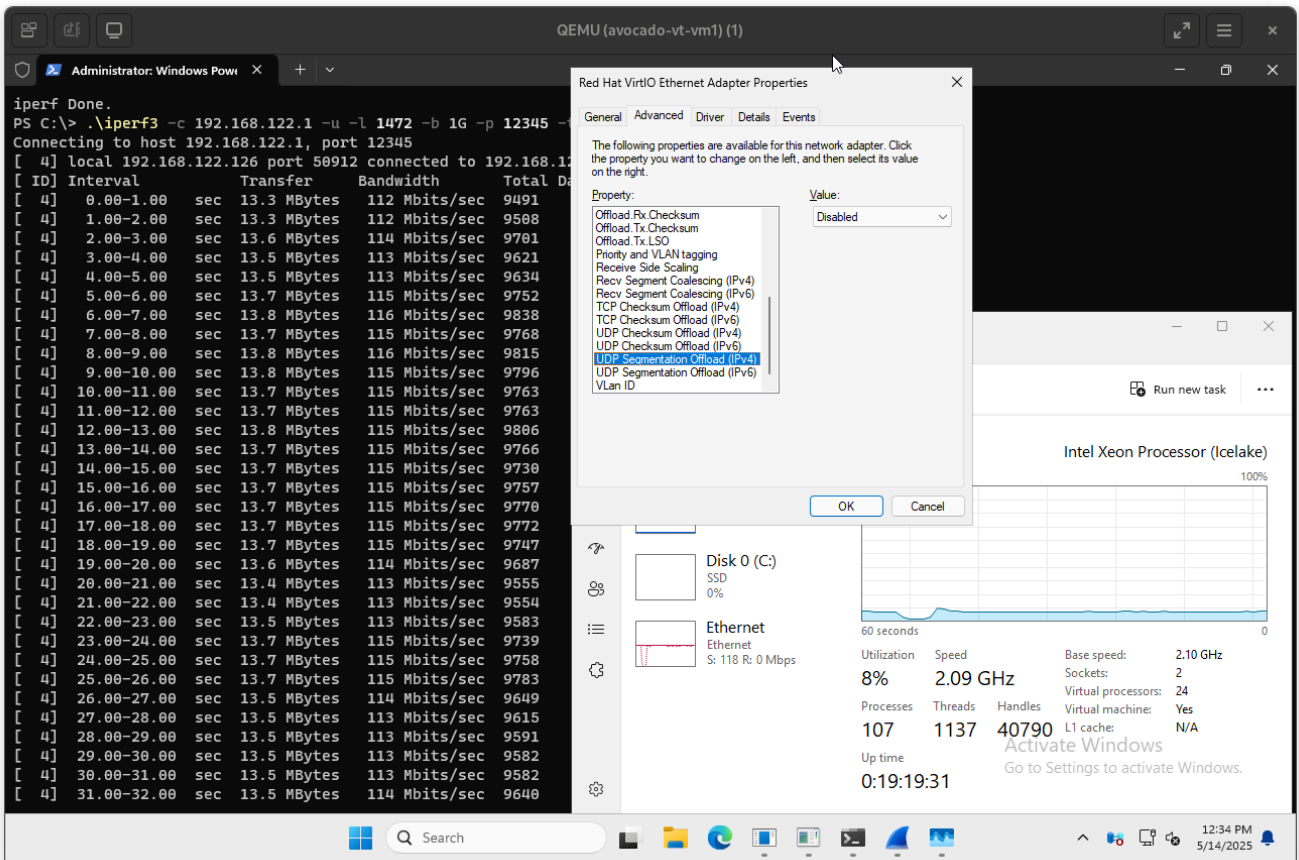
No.	Time	Source	Destination	Protocol	Length	Info
4209	112.806586	192.168.122.59	192.168.122.1	UDP	429	55248 → 12345 Len=65507
4210	112.808466	192.168.122.59	192.168.122.1	UDP	429	55248 → 12345 Len=65507
4211	112.809065	192.168.122.59	192.168.122.1	UDP	429	55248 → 12345 Len=65507
4211	112.809606	192.168.122.59	192.168.122.1	UDP	429	55248 → 12345 Len=65507
4212	112.811499	192.168.122.59	192.168.122.1	UDP	429	55248 → 12345 Len=65507
4212	112.812076	192.168.122.59	192.168.122.1	UDP	429	55248 → 12345 Len=65507
4213	112.812611	192.168.122.59	192.168.122.1	UDP	429	55248 → 12345 Len=65507
4214	112.814429	192.168.122.59	192.168.122.1	UDP	429	55248 → 12345 Len=65507
4214	112.815025	192.168.122.59	192.168.122.1	UDP	429	55248 → 12345 Len=65507
4214	112.815554	192.168.122.59	192.168.122.1	UDP	429	55248 → 12345 Len=65507
4215	112.817388	192.168.122.59	192.168.122.1	UDP	429	55248 → 12345 Len=65507
4216	112.817967	192.168.122.59	192.168.122.1	UDP	429	55248 → 12345 Len=65507
4216	112.818501	192.168.122.59	192.168.122.1	UDP	429	55248 → 12345 Len=65507
4217	112.820369	192.168.122.59	192.168.122.1	UDP	429	55248 → 12345 Len=65507
4217	112.820966	192.168.122.59	192.168.122.1	UDP	429	55248 → 12345 Len=65507
4218	112.821523	192.168.122.59	192.168.122.1	UDP	429	55248 → 12345 Len=65507
4219	112.823429	192.168.122.59	192.168.122.1	UDP	429	55248 → 12345 Len=65507

```
> Frame 421283: 429 bytes on wire (3432 bits), 429 bytes captured (3432 bits) on interface \Device\NPF_{0D5E9A34-C2BC-4201-873F-57BD70E0048B}, id 0
> Ethernet II, Src: 9a:3b:8a:d7:46:44 (9a:3b:8a:d7:46:44), Dst: 52:54:00:a7:62:4c (52:54:00:a7:62:4c)
> Internet Protocol Version 4, Src: 192.168.122.59, Dst: 192.168.122.1
  > User Datagram Protocol, Src Port: 55248, Dst Port: 12345
    Source Port: 55248
    Destination Port: 12345
    Length: 65515
    Checksum: 0xe3c5 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 3]
    [Stream Packet Number: 6632]
  > [Timestamps]
  > UDP payload (65507 bytes)
  > Data (65507 bytes)
    Data [-]: 681b0d93000a85d10000297e2a0296efd22e2605fa7b362ac17906e26bc24a5d23825cbce508b67e812c46ab2edc9a010ac006043c3c2efdb635e021f72a7e1aacdbd692e38d1064b9560fe733a9e83d69e
    [Length: 65507]
```

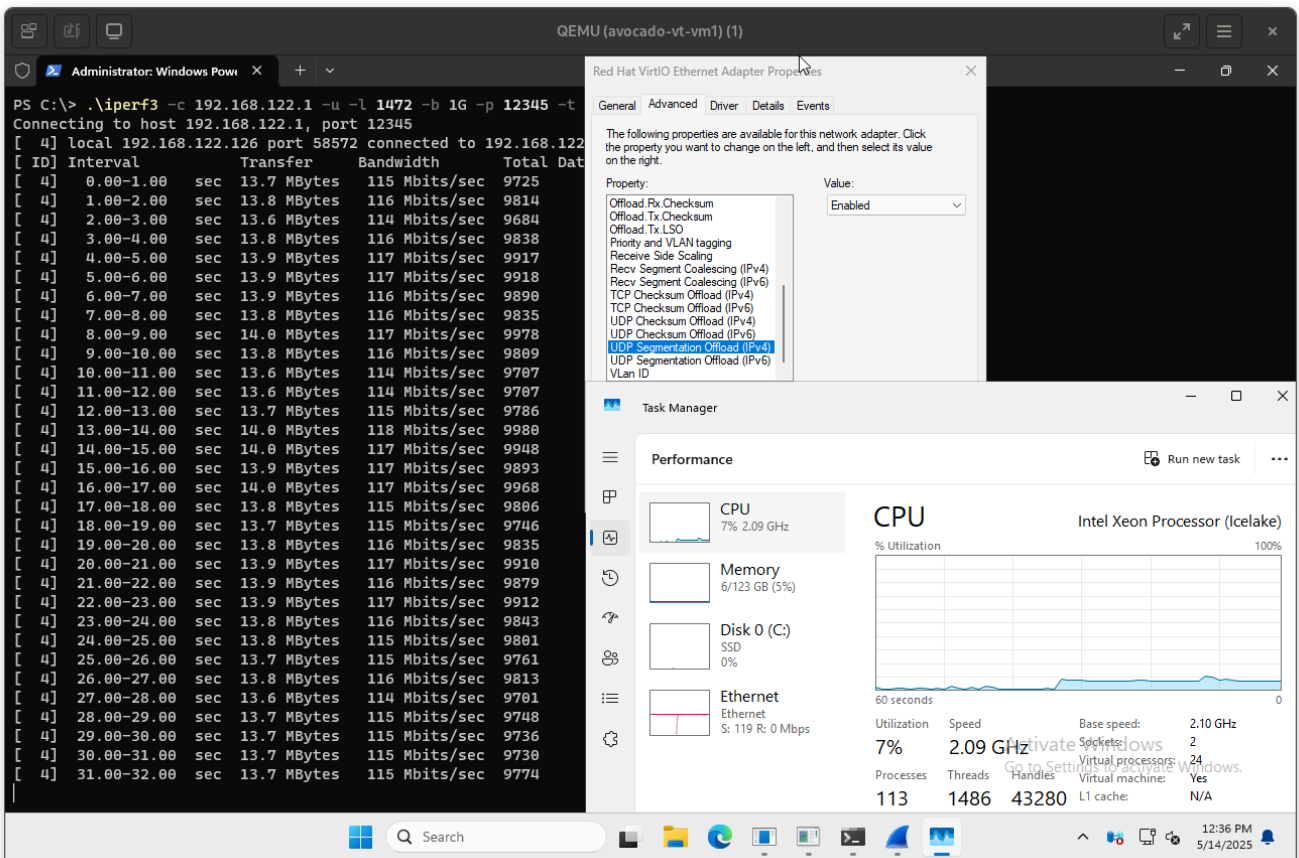
四、可能的提问。

Q0: CPU 利用率的问题

A0: 关闭 USO 后 CPU 利用率 8%



A0: 打开 USO 后 CPU 利用率 7%



Q1: RHEL 的 UDP offload 支持吗？

A1: 我不测试 RHEL 不是很清楚。昨天和 RHEL network 的 feature owner 确定还不支持。可以通过 ethtool 列举网卡的选项。

Q2. Windows 的 TCP offload 支持吗？

A2: 支持的

