PVE GPU Passthrough Guide (Threadripper/High-End X3D)

This is a step-by-step walkthrough to turn your desktop computer into a Proxmox VE host with a workstation or gaming VM with GPU and USB controller passthrough.

Testing environment 1

- CPU: AMD Threadripper 7970x
- GPU: Nvidia RTX 4090 (ASUS)
- Motherboard: Gigabyte TRX50 AI TOP
  - Benchmark results listed in 7970X Passthrough Results

Testing environment 2

- CPU: AMD Ryzen 7970X3D
- GPU: Nvidia RTX 4090 (ASUS)
- Motherboard: ASUS ROG Strix X670E-E
  - Benchmark results match a 7800X3D + 7700X CPU (Depending on affinity setting)

Prerequisites

- A CPU and motherboard with IOMMU support:
  - Intel VT-d or AMD-Vi **enabled** in BIOS
  - For systems with multiple CCDs (7970X/7950X3D), use NPS2/NPS4 respectively.
  - Other Bios optimizations listed in the 7970XBiosOptimizations File
- Proxmox VE (tested on 8.x)
- A target VM OS ISO (e.g. Windows 11)
- `sudo` or `root` privileges on the host, I do not recommend this if this is not a dedicated system for this task

Additional Information

- 7970X 3DMark TimeSpy Benchmark Results
- 7970X BIOS Optimizations for Gaming
- Breakdown of CPU Arguments and Topology
- Breakdown of Network Optimizations
- Script for enabling microcode on host system

Why would you want to do this?

- Allows you to increase performance of VMs which cannot utilize the amount of cores you have available (experience L3 thrashing with >2 NUMA Nodes)
- Multi-User Single-Computer Gaming with physical display/connections and no remoting
- Quick switching of operating systems without a full reboot
- Live backups of your virtual machines
- You want a proper virtualization host but need an updated kernel
- You have too much time and can't use a computer like a normal person

---

1. Host Kernel & GRUB Configuration

1. Edit your GRUB command-line:

   ```
   nano /etc/default/grub
   ```

2. Add IOMMU & hugepages options. amd_iommu=on is not necessary and should not be added. On Intel systems, you need to use the equivalent intel_iommu=on appended within the DEFAULT line.

```
- GRUB_CMDLINE_LINUX_DEFAULT="quiet"
+ GRUB_CMDLINE_LINUX_DEFAULT="quiet iommu=pt hugepagesz=1G default_hugepagesz=1G hugepages=8 kvm.ignore_msrs=1"
```

3. Apply:

```
update-grub
```

---

2. Preload VFIO Modules

This is necessary to ensure that the vfio drivers can be applied to the passthrough USB/GPU devices. If this step is skipped, the blacklisted drivers will load instead.

1. Open (or create) `/etc/initramfs-tools/modules` and add:

```
vfio
vfio_iommu_type1
vfio_pci
vfio_virqfd
```

2. Rebuild initramfs and reboot:

```
update-initramfs -u
reboot
```

---

3. Blacklist Native Device Drivers

Under `/etc/modprobe.d/`, create the following files.

3.1 USB Host Controllers

Ensure your networking is configured properly, this will make your system fully headless. If this is messed up you will need to chroot from recovery to repair the system. `blacklist-usb.conf` **

```
blacklist xhci_hcd
blacklist xhci_pci
blacklist ehci_hcd
blacklist uhci_hcd
blacklist ohci_hcd
```

3.2 Allow Unsafe IRQs

`iommu_unsafe_interrupts.conf` **

```
options vfio_iommu_type1 allow_unsafe_interrupts=1
```

3.3 VFIO-PCI Device IDs

1. Run `lspci -vnn` and note the IDs of GPU, audio, USB controllers, etc.
2. For example, grab the two from the following 10de:2684,10de:22ba:

```
81:00.0 VGA compatible controller [0300]: NVIDIA Corporation AD102 [GeForce RTX 4090] [10de:2684] (rev a1) (prog-if 00 [VGA controll
        Subsystem: ASUSTeK Computer Inc. AD102 [GeForce RTX 4090] [1043:88e2]
        Flags: bus master, fast devsel, latency 0, IRQ 91, NUMA node 0, IOMMU group 16
        Memory at f0000000 (32-bit, non-prefetchable) [size=16M]
        Memory at 10800000000 (64-bit, prefetchable) [size=32G]
        Memory at 11000000000 (64-bit, prefetchable) [size=32M]
        I/O ports at 3000 [size=128]
        Expansion ROM at f1000000 [disabled] [size=512K]
        Capabilities: [60] Power Management version 3
        Capabilities: [68] MSI: Enable+ Count=1/1 Maskable- 64bit+
        Capabilities: [78] Express Legacy Endpoint, MSI 00
        Capabilities: [b4] Vendor Specific Information: Len=14 <?>
        Capabilities: [100] Virtual Channel
        Capabilities: [258] L1 PM Substates
        Capabilities: [128] Power Budgeting <?>
        Capabilities: [420] Advanced Error Reporting
        Capabilities: [600] Vendor Specific Information: ID=0001 Rev=1 Len=024 <?>
        Capabilities: [900] Secondary PCI Express
        Capabilities: [bb0] Physical Resizable BAR
        Capabilities: [c1c] Physical Layer 16.0 GT/s <?>
        Capabilities: [d00] Lane Margining at the Receiver <?>
        Capabilities: [e00] Data Link Feature <?>
        Kernel driver in use: vfio-pci
        Kernel modules: nvidiafb, nouveau

81:00.1 Audio device [0403]: NVIDIA Corporation AD102 High Definition Audio Controller [10de:22ba] (rev a1)
        Subsystem: ASUSTeK Computer Inc. AD102 High Definition Audio Controller [1043:88e2]
        Flags: bus master, fast devsel, latency 0, IRQ 43, NUMA node 0, IOMMU group 16
        Memory at f1080000 (32-bit, non-prefetchable) [size=16K]
        Capabilities: [60] Power Management version 3
        Capabilities: [68] MSI: Enable- Count=1/1 Maskable- 64bit+
        Capabilities: [78] Express Endpoint, MSI 00
        Capabilities: [100] Advanced Error Reporting
        Capabilities: [160] Data Link Feature <?>
        Kernel driver in use: vfio-pci
        Kernel modules: snd_hda_intel
```

3. ** `vfio.conf` ** (replace the IDs below):

```
options vfio-pci ids=10de:2684,10de:22ba
```

3.4 Blacklist GPU & Audio Devices

`blacklist-gpu.conf` **

```
blacklist amdgpu
blacklist nouveau
blacklist nvidiafb
blacklist snd_hda_intel
```

3.5 (Optional) Blacklist Wi-Fi Devices

I recommend this as you can then use your passed through device for Bluetooth/connecting to other networks/etc. `blacklist-wifi.conf` **

```
blacklist iwlwifi
```

3.6 Update Initramfs and Reboot

```
update-initramfs -u
reboot now
```

4. Install Microcode

Open the attached Microcode installer and execute. Otherwise, you can also use the Microcode script available at: https://community-scripts.github.io/ProxmoxVE/scripts?id=microcode

```
chmod +x InstallMicrocode.sh
./InstallMicrocode.sh amd|intel
```

5. GPU ROM (for proper device resets)

This ensures that when your VM is powered on, it will properly reload and reset your GPU as if it was baremetal. This is necessary on iGPUs like the 7800X3D/7950X3D.

1. Download your GPU's VBIOS. Grab yours from here: https://www.techpowerup.com/vgabios or extract your ROM directly. Move or wget your ROM into the /usr/share/kvm folder on your PVE host.

```
wget https://www.techpowerup.com/vgabios/251157/Asus.RTX4090.24576.221014.rom
mv Asus.RTX4090.24576.221014.rom /usr/share/kvm/409024576.rom
```

2. **Ensure** it matches your card's firmware version.

6. (Optional) Network Stack Tuning

Tune your network stack for a workstation/gaming targeted environment. Additional information and explanation for these changes can be found in the NetworkOptimizations.md file.

Append to `/etc/sysctl.conf`:

```
net.core.rmem_max = 268435456
net.core.wmem_max = 268435456
net.ipv4.tcp_rmem = 4096 87380 134217728
net.ipv4.tcp_wmem = 4096 65536 268435456
net.ipv4.tcp_congestion_control = bbr
net.core.default_qdisc = fq
net.ipv4.tcp_fastopen = 3
net.ipv4.tcp_window_scaling = 1
net.ipv4.tcp_sack = 1
net.ipv4.tcp_no_metrics_save = 1
net.ipv4.tcp_mtu_probing = 1
net.ipv4.tcp_low_latency = 1
net.ipv4.ip_local_port_range = 10240 65535
```

Execute changes with:

```
sysctl -p
```

7. Example VM Configuration

For a virtual machine targeted as Windows for gaming, you will need to apply a few tweaks for ensuring EAC/similar tools will not block your device from use. Look at the breakdown of args in the CPUBreakdown file.

> **Recommendation:** Power off the VM before editing its `.conf`.

```
 # VM 100: Gaming VM for 7970X/Multi NUMA node CPU (Use Primary cores of first two NUMA nodes, change accordingly)
affinity: 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
agent: 1,fstrim_cloned_disks=1
args: -smp '16,cores=16,threads=1,sockets=1,maxcpus=16' -cpu 'host,host-cache-info=on,l3-cache=on,topoext=on,+invtsc,+kvm_pv_unhalt,
balloon: 0
bios: ovmf
boot: order=sata0
cpu: host,hidden=1,flags=+pdpe1gb;+hv-tlbflush
hostpci0: 0000:81:00.0,pcie=1,romfile=409024576.rom # x-vga=1 is not necessary. With the ROM added, booting your VM will provide a b
hostpci0: 0000:81:00.1,pcie=1
hostpci3: 0000:4b:00.0,pcie=1
# …add other USB controllers as hostpci*
hotplug: disk,network
machine: pc-q35-9.2+pve1
memory: 49152 # Ensure this is the sum of your NUMA 0-X bindings
meta: creation-qemu=9.2.0,ctime=1746507489
name: vmname
net0: e1000e=00:00:5E:14:0A:E1,bridge=vmbr0
numa: 1 # Enable NUMA if necessary by your system
numa0: cpus=0-7,hostnodes=0,memory=24576,policy=bind # Specify the NUMA memory configuration if necessary with your system
numa1: cpus=8-15,hostnodes=1,memory=24576,policy=bind
ostype: win11
sata0: local-lvm:vm-100-disk-0,cache=writeback,discard=on,size=500G,ssd=1 # You can also pass through dedicated hardware as storage
scsihw: lsi # LSI controller is necessary to bypass EAC, QEMU Storage will be blocked
smbios1: uuid=USE-EXISTING-UUID,manufacturer=R2lnYWJ5dGUgVGVjaG5vbG9neSBDby4sIEx0ZC4=,product=VFJYNTAgQUkgVE9Q,version=RGVmYXVsdCBzd
sockets: 1
tablet: 0
tpmstate0: local-lvm:vm-100-disk-1,size=4M,version=v2.0
vga: none # Don't confuse the VM by adding a second display adapter.
vmgenid: USE-EXISTING-GENID
vmstatestorage: local-lvm
```

7.1 SMBIOS Configuration (smbios1)

Extract the SMBIOS header from your PVE host using the dmidecode command. This is required to get past EAC. Add this information under VM -> Options -> smbios1:

```
 dmidecode -t 2
# dmidecode 3.4
Getting SMBIOS data from sysfs.
SMBIOS 3.6.0 present.
# SMBIOS implementations newer than version 3.5.0 are not
# fully supported by this version of dmidecode.

Handle 0x0002, DMI type 2, 15 bytes
Base Board Information
        Manufacturer: Gigabyte Technology Co., Ltd.
        Product Name: TRX50 AI TOP
        Version: x.x
        Serial Number: Default string
        Asset Tag: Default string
        Features:
                Board is a hosting board
                Board is replaceable
        Location In Chassis: Default string
        Chassis Handle: 0x0003
        Type: Motherboard
        Contained Object Handles: 0
```

7.2 VM Configuration

Below are examples for configuring CPU affinity/NUMA(if used by your motherboard/CPUs), more information can be found in the CPUBreakdown file.

7.2.1 Gaming VM

On this system (7970X with 4x 32GB of memory), you will want to pass specific CPUs/cores for certain scenarios for the best performance. From testing the best result for gaming was the following - this matches what Ryzen Master does for Core disablement but allows the rest of your CPU to be used:

```
affinity: 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
numa: 1
numa0: cpus=0-7,hostnodes=0,memory=24576,policy=bind
numa1: cpus=8-15,hostnodes=1,memory=24576,policy=bind
```

7.2.2 Workstation VM

For the workstation VM, you can allocate both SMT/HT as well as primary cores. The typical ordering is to use the main core/hyperthread, so on. For Node 0 on the numactl command, you can see that 0-7 are primary cores and 32-39 are HT/SMT, these must be interleaved for the best performance as it is what the guest VM will expect for ordering.

```
affinity: 0,32,1,33,2,34,3,35,4,36,5,37,6,38,7,39,8,40,9,41,10,42,11,43,12,44,13,45,14,46,15,47,16,48,17,49,18,50,19,51,20,52,21,53,
numa: 1
numa0: cpus=0-15,hostnodes=0,memory=24576,policy=bind
numa1: cpus=16-31,hostnodes=1,memory=24576,policy=bind
numa2: cpus=32-47,hostnodes=2,memory=24576,policy=bind
numa3: cpus=48-63,hostnodes=3,memory=24576,policy=bind
```

8. GPU Passthrough

- In the PVE GUI: VM → **Hardware** → **Add** → **PCI Device** , then pick your GPU as well as the audio device for your GPU. These are found with the lspci command.

```
lspci
...
4b:00.0 USB controller: Intel Corporation Thunderbolt 4 NHI [Maple Ridge 4C 2020]
5d:00.0 USB controller: Intel Corporation Thunderbolt 4 USB Controller [Maple Ridge 4C 2020]
6f:00.0 USB controller: Advanced Micro Devices, Inc. [AMD] 600 Series Chipset USB 3.2 Controller (rev 01)
...
81:00.0 VGA compatible controller: NVIDIA Corporation AD102 [GeForce RTX 4090] (rev a1)
81:00.1 Audio device: NVIDIA Corporation AD102 High Definition Audio Controller (rev a1)
...
```

Add these devices accordingly. For the GPU, you will need to edit the `.conf` file under /etc/pve/qemu-server/. to specify the romfile that you had downloaded/added earlier. Append the following to your hostpci0(GPU) line `,romfile=409024576.rom`. Ensure PCI-Express is enabled for all PCI devices. This cannot be done in the PVE GUI.

```
- hostpci0: 0000:81:00.0,pcie=1
+ hostpci0: 0000:81:00.0,pcie=1,romfile=409024576.rom
```

9. USB Passthrough

- In the PVE GUI: VM → **Hardware** → **Add** → **PCI Device** , then pick your USB controller(s).
- Or in the VM's `.conf` as additional `hostpci*` lines (see above).

9. Windows-Side Tweaks

- Use **e1000e** NIC and **LSI** storage controller to avoid EAC detection.
- Do not install the balloon driver, it will degrade performance. Don't install the serial driver, it's detected by EAC. After installing **qemu-guest-agent** + drivers, uninstall any `virtio-*` drivers in Device Manager. Mostly the virtio serial driver, if you left it.
- Change the NICs MAC address to a generic OUI: `00:00:5E:XX:XX:XX`. If you leave the BC:24:11 it can be detected as a VM.
- Consider disabling Windows mitigations (e.g. Spectre/Meltdown) if it's a single-tenant gaming host.
- Disable Hardware Accelerated GPU Scheduling for best performance.

10. Troubleshooting

- `dmesg | grep -i vfio` for errors
- Verify your IOMMU groups: `find /sys/kernel/iommu_groups/ -type l`
- Ensure no driver is grabbing the device: `lspci -k -s 81:00.0`, you should see that the driver in use is vfio-*** for instance:

```
 Kernel driver in use: vfio-pci
Kernel modules: nvidiafb, nouveau
```

If something still doesn't work, double-check IDs from `lspci -vnn`, confirm your ROM file, and make sure grub/initramfs changes applied correctly. I am by no means an expert at this, so some of this information could be incorrect. If you see a problem please submit a issue and describe what needs corrected. I mostly am providing this documentation as a means to duplicate this easier in the future.

Threadripper 7970X BIOS Settings for a Gaming-Optimized VM

Here is a categorized breakdown of the key BIOS tweaks you'll want on a TR 7970X or similar host when you're running a GPU-accelerated gaming VM:

> **Tip:** After applying these, double-check with `hwinfo`, `lscpu`, and your VM's `/proc/cpuinfo` that your guest sees the expected topology and frequencies. This profile aggressively favors consistent low-latency performance over power savings—ideal for gaming VMs running close to bare-metal speeds.

---

1. CPU & Virtualization

Setting	Value	Why it helps
SVM	Enabled	AMD's Secure Virtual Machine—must be on for nested paging and VM performance
SR-IOV (IOMMU)	Enabled	Allows direct PCIe device assignment (GPU, USB controllers) without driver conflicts
Local APIC Mode	x2APIC	Faster inter-processor interrupts and better scaling across many vCPUs

---

2. Memory & XMP

Setting	Value	Why it helps
XMP/EXPO Profile	XMP1	Loads your RAM at its rated high frequency and timings
XMP/EXPO High Bandwidth Support	Enabled	Optimizes memory controller for maximum throughput
Memory Context Restore	Enabled	Saves/restores full memory controller state on sleep/wake—avoids timing recalcs under load
Determinism Control	Manual	Unlocks manual tuning of memory timing determinism
Determinism Enable	Performance	Prioritizes lowest latency and highest stability in timing loops
Spread Spectrum Control / FCH	Disabled	Locks clock generators—eliminates tiny frequency dithering that can hurt real-time performance
RAM Power Down Enable	Disabled	Keeps DRAM fully powered to avoid wake-up latency

---

3. NUMA & Interleaving

Setting	Value	Why it helps

Setting	Value	Why it helps
NUMA Nodes (NPS)	NPS4	Consolidates memory into fewer NUMA regions for lower cross-node latency
ACPI SRAT L3 Cache As NUMA Domain	Disabled	Prevents each L3 slice being treated as its own NUMA node
Memory Interleaving	Disabled	Keeps each NUMA node's pages contiguous—better for pinned workloads

4. PCIe & I/O

Setting	Value	Why it helps
PCIEX16_1 (GPU Slot)	Gen 4 (max your GPU supports)	Full PCIe bandwidth for GPU → lowest latency, highest throughput
PCIE Speed Controller	Disabled	Avoids dynamic speed shifts under load
PCIE Speed PMM Control	Static Target Link Speed (GEN4 or GEN5 as available)	Locks link at max supported speed
CPPC	Enabled	Collaborative Processor Performance Control—OS can pick best cores
CPPC Preferred Cores	Enabled	Pin high-priority threads to top-performing cores

5. Prefetchers & C-States

Setting	Value	Why it helps
L1 Stream HW Prefetcher	Enabled	Proactively loads sequential data into L1 for smoother memory access
L1 Stride Prefetcher	Enabled	Detects and preloads strided access patterns
L2 Region Prefetcher	Enabled	Preloads larger blocks at L2 level for bigger working sets
L2 Stream HW Prefetcher	Enabled	Similar to L1 stream but at L2 cache
L2 Up/Down Prefetcher	Enabled	Dynamically adjusts L2 prefetch depth
DF C-States	Disabled	Disables deep fabric power-down—avoids unpredictable wake-up latencies
DF P-State Frequency Optimizer	Disabled	Stops dynamic fabric frequency changes—keeps interconnect steady
DF P-State Latency Optimizer	Disabled	Avoids added latency from power-state transitions on the data fabric
APBDIS	Enabled	Disables CPU power-gating—ensures cores stay in their highest performance state
HSMP Support	Disabled	Turns off AMD's internal sensor network—removes any extra overhead

3DMark Timespy Benchmark Results (Threadripper 7970X)

Below are benchmark runs, grouped by category. Scores are taken directly from Timespy; "Changes" summarizes the key configuration tweaks between each run.

Notes:

- "Primary" denotes scheduling only the first hardware thread of each core; "Hyperthreads" denotes scheduling only the SMT sibling threads.
- NPS# refers to custom NUMA policy settings (e.g. NPS4 = policy #4).
- Affinity mappings like "0–7 & 32–39" bind the processes to those physical/core ranges using Process Lasso.
- Early results (Tests 1–6) focus on BIOS/kernel tweaks; later tests explore CCD pass-through and fine-tuned CPU/memory pinning.
    - These have additional BIOS optimizations as stated in the BIOS Optimizations file, unless stated otherwise.

1. Initial Tuning (Tests 1–6)

Test	Graphics	CPU	Changes
1	30542	12543	• Enabled PCIe multifunction passthrough • Enabled `x-vga=1` • Applied NPS4 NUMA policy
2	31048	12251	• Enabled `hidden=1` hypervisor flag
3	30969	12316	• Added `amd_pstate=active` to kernel boot • Disabled Process Lasso service
4	31037	12649	• Disabled Windows hardware-accelerated GPU scheduling
5	31424	12323	• Manual memory context restore • Determinism: control=manual, mode=performance • Disabled DF P-state latency optimizer • Switched GPU to PCIe Gen4, disabled 3D V-cache • Disabled APBDIS & PCIe speed controller • Added VM flags `hv-vendor-id=AMDVENDOR, hv-crash`
6	31384	12387	• Disabled spread-spectrum & FCH spread • Set PBO limit to 80C Level 3 • Set NUMA to NPS4 • Disabled SATA & RAS • Disabled RAM power-down

---

2. CCD Passthrough & Hyperthreading (Tests 7–14)

Test	Graphics	CPU	Changes
7	33520	9188	• Passthrough CCD0 & CCD1 • Use hyperthreads on CCD0 for Timespy
8	29001	9171	• Inverted hyperthread pairs • Passthrough CCD0 & CCD1 • CCD0 primary • Explicit affinity mapping inverted to `32→0,33→1,…`
9	20842	12346	• CCD0 & CCD1 as primary threads
10	20806	12222	• Hyperthreads only on CCD0 & CCD1
11	28870	9336	• Hyperthreads only on CCD0
12	29386	9247	• Hyperthreads only on CCD1
13	18173	11904	• CCD1 primary only
14	11619	10933	• Full pass-through of CCD0 & CCD1

---

3. Primary vs. Hyperthread-Only Trials (Tests 15–24)

Test	Graphics	CPU	Changes
15	11807	11519	• CCD0 primary; reverted HT/core order, switched NUMA policy to NPS1
16	11830	11521	• CCD1 primary only
17	27002	9202	• CCD0 primary only

Test	Graphics	CPU	Changes
18	29251	9312	• CCD1 primary only
19	27937	9238	• Hyperthreads only on CCD0
20	28304	9312	• Hyperthreads only on CCD1
21	7853	11560	• CCD0 & CCD1 primary
22	7603	11437	• CCD0 & CCD1 hyperthreads
23	23024	11660	• Passthrough CCD0 only; NPS4 policy applied
24	31298	11971	• CCD1 primary only

4. Extended CCD & Mixed-Mode Tests (Tests 25–33)

Test	Graphics	CPU	Changes
25	33421	9204	• CCD0 primary only
26	33583	9407	• CCD1 primary only
27	33559	9397	• CCD0 hyperthreads
28	33829	9201	• CCD1 hyperthreads
29	19754	12711	• CCD0 & CCD1 primary
30	22010	12974	• CCD0 & CCD1 hyperthreads
31	23023	12404	• CCD0 primary + CCD1 hyperthreads
32	21747	12864	• CCD1 primary + CCD0 hyperthreads
33	31261	12414	• Full passthrough of CCD0; NPS4 policy

5. Full Pass-Through & Core Splits (Tests 34–41)

Test	Graphics	CPU	Changes
34	34822	9411	• Primary threads only
35	34916	9272	• Hyperthreads only
36	34978	9241	• First half of cores (0–31)
37	35040	9247	• Second half of cores (32–63)
38	30332	12270	• Full passthrough on CCD1
39	31930	14171	• Full passthrough on CCD0 & CCD1 primary-only
40	13012	12237	• Full passthrough on all CCDs (0–3) primary-only
41	32734	9639	• Full passthrough on all CCDs; CPU affinity pinned to core 0

6. Affinity & Final Adjustments

Test	Graphics	CPU	Changes
45	9224	7426	• Disabled hardware-accelerated GPU scheduling • CPU set 0–15, affinity 0–63
48	31877	9689	• Used CPU cores 0–7 only
49	8414	14963	• Used cores 0–7 & 32–39 • SMP: cores=8, threads=2, sockets=4
51	34324	9586	• Remapped VM NUMA from nodes [0–7,32–39] to [0–15] • Affinity for Steam & related processes to cores 0–7
53	31787	13926	• Repeat of Test 39 (full CCD0&1 primary-only)
54	30823	14684	• Removed NUMA node 1 • VM: sockets=1, cores=16

Recommended Arguments for Gaming and Workstations

CPU Arguments ideal for both workstations and gaming, ensures the virtual machine cannot tell it is not running on baremetal:

```
-cpu 'host,host-cache-info=on,l3-cache=on,topoext=on,+invtsc,+kvm_pv_unhalt,+kvm_pv_eoi,hv_spinlocks=0x1fff,hv_vapic,hv_time,hv_rese
```

Arguments for full CPU passthrough - ideal for workstations and creative works (7970X):

```
-smp '64,cores=8,threads=2,sockets=4,maxcpus=64'
```

Arguments for partial CPU passthrough - ideal for gaming:

```
-smp '16,cores=16,threads=1,sockets=1,maxcpus=16'
```

The main goal with creating the gaming system is to prevent L3 thrashing, to do this you ideally will have less than 2 NUMA nodes and will limit the total amount of cores as well. Hyperthreads/SMT are not as effective as the main cores (1->0.5 effectively), disabling these and providing a max of two CCDs with primary cores will provide the best results.

To get the performance using a multi NUMA node/CCD system you need to use the CCD attached to the GPU PCI lane and only use the memory attached to that CCD.

Breakdown of `args:` lines

---

1. SMP Topology (`-smp …`)

Parameter	16-core VM (Gaming)	64-core VM (Workstation)	What it is	Why it helps
`-smp`	`16`	`64`	Total number of vCPUs presented to the guest	Matches the guest's CPU count to the host for max performance
`cores`	`16`	`8`	Cores per socket	Defines how many "real" cores each virtual socket has—aligns with host cache topology
`threads`	`1`	`2`	Threads per core (SMT/hyperthreading)	More threads can boost throughput on SMT-capable CPUs
`sockets`	`1`	`4`	Virtual CPU sockets	Splitting into multiple sockets can help NUMA-aware OSes optimize locality
`maxcpus`	`16`	`64`	Maximum hot-plug-able vCPUs	Lets you add vCPUs at runtime up to this limit (must match total vCPU count)

Splitting the standard configuration of the NPS1 on a Threadripper system to NPS4 (or forcing the VM to make the assumption) significantly helps the operating system best perform as it understands it's environment. The Infinity Fabric is fast but not as fast as the IPC on a NUMA Node communicating to itself. Forcing a 7970X to assume it is effectively 4x 7700X CPUs will provide better performance.

2. CPU Feature Flags (`-cpu …`)

Flag	What it is	Why it helps
`host`	Use the host's exact CPU model	Exposes every feature your physical CPU has, minimizing emulation overhead
`host-cache-info=on`	Expose host cache hierarchy (sizes, sharing)	Allows guest OS to optimize cache usage exactly like on bare-metal
`l3-cache=on`	Specifically expose L3 cache topology	Improves scheduling decisions for L3-sensitive workloads
`topoext=on`	Enable topology extensions	Lets the guest see extended CPU topology (e.g. dies, CCDs)
`+invtsc`	Invariant TSC (time stamp counter)	Provides a stable clock source across P-states, C-states, and VM migrations
`+kvm_pv_unhalt`	Paravirtualized halt/unhalt	Speeds up CPU idle and wake-up paths
`+kvm_pv_eoi`	Paravirtualized End-Of-Interrupt	Fast EOI handling—lowers interrupt exit latency
`hv_spinlocks=0x1fff`	Hypervisor-spinlock mask	Optimizes guest spin-lock loops to reduce VM-exit storms
`hv_vapic`	Virtualized local APIC	Faster interrupt routing inside the VM
`hv_time`	Paravirtualized clock	High-precision, low-overhead time keeping
`hv_reset`	Hypervisor-driven reset interface	Allows the host to reset the VM quickly
`hv_vpindex`	Virtual processor index	Guest can read its vCPU ID directly
`hv_runtime`	Paravirtualized runtime interface	Provides a generic hypervisor-guest API for improved sync
`hv_relaxed`	Relaxed VM exit conditions	Reduces exits on RDTSC and other timing instructions
`hv_crash`	Crash notification	Propagates host crashes to the guest OS immediately
`hv_ipi`	Paravirtualized inter-processor interrupts	Low-latency IPIs between vCPUs
`hv_vendor_id=AMDVENDOR`	Override CPUID vendor string	Masks "KVM" vendor, useful to bypass hypervisor checks (EAC, anti-cheat)
`host-phys-bits=true`	Expose full host physical address width	Lets the guest address all RAM on large-memory systems
`hypervisor=off`	Hide hypervisor bit in CPUID	Makes the guest believe it's on bare-metal
`kvm=off`	Hide KVM presence	Further masks virtualization, avoiding certain software checks

---

NUMA Nodes and CPU Affinity

To check your system to query which CPUs are ideal for passing/utilizing, use the following command: `bash numactl --hardware` Output: `bash available: 4 nodes (0-3) node 0 cpus: 0 1 2 3 4 5 6 7 32 33 34 35 36 37 38 39 node 0 size: 31723 MB node 0 free: 2760 MB node 1 cpus: 16 17 18 19 20 21 22 23 48 49 50 51 52 53 54 55 node 1 size: 32250 MB node 1 free: 3747 MB node 2 cpus: 24 25 26 27 28 29 30 31 56 57 58 59 60 61 62 63 node 2 size: 32250 MB node 2 free: 4960 MB node 3 cpus: 8 9 10 11 12 13 14 15 40 41 42 43 44 45 46 47 node 3 size: 32169 MB node 3 free: 4378 MB node distances: node 0 1 2 3 0: 10 12 12 12 1: 12 10 12 12 2: 12 12 10 12 3: 12 12 12 10`

Gaming VM

On this system (7970X with 4x 32GB of memory), you will want to pass specific CPUs/cores for certain scenarios for the best performance. From testing the best result for gaming was the following:

```
 affinity: 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
numa: 1
numa0: cpus=0-7,hostnodes=0,memory=24576,policy=bind
numa1: cpus=8-15,hostnodes=1,memory=24576,policy=bind
```

This utilizes two NUMA nodes, allocating the main cores of CCD0 (0-7) and CCD1 (8-15) to the VM. For memory, binding provides the most consistent results. This requires you to minimize the cross node communication for memory access. To do this it is best to use only the memory available on each NUMA node while leaving some available for the host. In my case, I allocated 24,576MiB from both CCD0 and CCD1. Enabling NUMA also ensures that the guest VM understands the layout and can schedule/allocate accordingly.

Workstation VM

For the workstation VM, you can allocate both SMT/HT as well as primary cores. The typical ordering is to use the main core/hyperthread, so on. For Node 0 on the numactl command, you can see that 0-7 are primary cores and 32-39 are HT/SMT, these must be interleaved for the best performance as it is what the guest VM will expect for ordering.

```
 affinity: 0,32,1,33,2,34,3,35,4,36,5,37,6,38,7,39,8,40,9,41,10,42,11,43,12,44,13,45,14,46,15,47,16,48,17,49,18,50,19,51,20,52,21,53,
numa: 1
numa0: cpus=0-15,hostnodes=0,memory=24576,policy=bind
numa1: cpus=16-31,hostnodes=1,memory=24576,policy=bind
numa2: cpus=32-47,hostnodes=2,memory=24576,policy=bind
numa3: cpus=48-63,hostnodes=3,memory=24576,policy=bind
```

Since this VM has all four NUMA nodes available, you can bind memory from all 4 NUMA domains. Make sure to leave some for the host as ballooning should be disabled.

Network Optimization Explanation

Below is a line-by-line walk-through of the settings, describing what each one changes, and when it helps on a **Proxmox host** that mainly:

- runs VMs and containers
- talks to fast local storage or Ceph
- moves a lot of data over 10–40 Gb Ethernet
- rarely acts as a public-facing web server

---

1 Socket buffer limits

(affects **any** program that uses TCP/UDP, including QEMU and Ceph OSDs)

Setting	Meaning (bytes)	Default	When to raise it	Caveat
net.core.rmem_max = 268435456	Max receive buffer a single socket may auto-grow to (256 MiB)	212 KiB	Long-haul or 40/100 GbE links with high bandwidth × latency product	Big cap alone ≠ bigger buffers — apps must ask for them or autotune must ramp up.
net.core.wmem_max = 268435456	Max send buffer (256 MiB)	212 KiB	Bulk senders (NFS server, iSCSI target) over high-speed LAN/WAN	Uses more kernel memory under heavy fan-out loads.
net.ipv4.tcp_rmem = 4096 87380 134217728	Min / default / max recv autotune steps	4 KiB / 85 KiB / 128 MiB	Keep defaults (85 KiB) but allow auto-growth to 128 MiB	Needs net.core.rmem_max ≥ max.
net.ipv4.tcp_wmem = 4096 65536 268435456	Min / default / max send autotune	4 KiB / 64 KiB / 256 MiB	Same logic	As above.

Should you keep them?

- **Yes** if you have 10 Gb + links **and** move multi-gigabyte files (VM images, Ceph replication).
- Otherwise the stock limits (16 MiB max) are fine and use less RAM.

---

2 Congestion-control & queuing

Setting	What it does	Default	Keep?
net.ipv4.tcp_congestion_control = bbr	Switches to Google BBR congestion algorithm. Gives faster ramp-up and steadier RTT than CUBIC on clean links.	cubic	• Great for WAN or oversubscribed switches.• Safe on LAN.
net.core.default_qdisc = fq	Sets the default queuing discipline to FQ (Fair Queue). Works best with BBR.	pfifo_fast	Yes – low latency, combats buffer-bloat.
net.ipv4.tcp_fastopen = 3	Enables TCP Fast Open for client (1) + server (2) = 3. Skips 1 RTT on connection setup.	1 (client only) or 0	Mostly helps web servers. Harmless but you need app support.

3 Window scaling / SACK / low-latency tweaks

Setting	Default	Meaning & advice
net.ipv4.tcp_window_scaling = 1	1	Always leave enabled; required for windows > 64 KiB.
net.ipv4.tcp_sack = 1	1	Selective ACKs – speeds up recovery; keep 1 unless a buggy firewall disables SACK.
net.ipv4.tcp_no_metrics_save = 1	0	Don’t cache old RTT/cwnd per destination. Can improve first connection after route changes; safe to enable.
net.ipv4.tcp_mtu_probing = 1	0	Actively probe for a working PMTU when ICMP is blocked. Good for WAN, harmless on LAN.
net.ipv4.tcp_low_latency = 1	0	Tells the TCP stack to favor lower latency over throughput in some heuristics. Minor effect; fine to enable.

4 Local port range

Setting	Default	Why bump it
net.ipv4.ip_local_port_range = 10240 65535	32768 60999	Allows ~10 × more concurrent outbound sockets before wrap-around; useful if the host launches many container/VM connections (SSH, Ceph, NFS, web). Safe.