Quad RTX3090 GPU Power Limiting with Systemd and Nvidia-smi

Posted on by Dr. Donald Kinghorn

Table of Contents

TL;DR: You can run 4 RTX3090's in a system under heavy compute load using a single PSU and without without overloading your power line, as shown in a previous post. This can be done automatically at system boot time using a script with nvidia-smi commands and a startup service configured with Systemd. A script and systemd unit file are provided (and explained) below.

Introduction

This is a follow up post to "Quad RTX3090 GPU Wattage Limited "MaxQ" TensorFlow Performance". In that post I presented TensorFlow ResNet50 performance results over a range of GPU wattage limits. The goal there was to find a power limit that would give 95% of the total performance at a system power load that is acceptable with a single PSU running on a US 110V 15A power line. It turns out that limiting the RTX3090's to 270W or 280W does that! That means that it should be reasonable to setup a Quad RTX3090 system for machine learning workloads. Performance was outstanding!

In the testing in the post mentioned above I used the NVIDIA System Management Interface tool, **nvidia-smi**, to set GPU power limits in the testing scripts. This post will show you a way to have GPU power limits set automatically at boot by using a simple script and a **systemd Unit file**.

I used Ubuntu 20.04 server as the OS for the performance testing and for the startup service testing in this post. However, any modern Linux distribution using systemd should be OK.

nvidia-smi commands and a script to set a power limit on RTX30 GPUs

Here are the needed/useful nvidia-smi commands,

Persistence Mode

"When persistence mode is enabled the NVIDIA driver remains loaded even when no active clients, such as X11 or nvidia-smi, exist." (only available on Linux) This keeps the NVIDIA kernel modules from unloading. It is designed to lower job startup latency but I believe it is a good idea to set this on boot so that your power setting don't get accidentally "un-set" from a module reload.

```
sudo  nvidia-smi -pm 1
or
sudo nvidia-smi --persistence-mode=1
```

This should set all of the GPUs. You can use the "-i" flag to explicitly specify the GPUs by id. For example `-i 0,1,2,3` for the first 4 GPUs in the system.

Set GPU Power Limits

Setting the GPU power limit wattage can be done with, (Setting a 280W limit on the 350W default RTX3090 GPU as an example)

```
sudo nvidia-smi -pl 280
or
sudo nvidia-smi  --power-limit=280
```

After you have made changes you can monitor power usage during a job run with, ("-q" query, "-d" display type, "-l 1" loop every 1 second)

```
nvidia-smi -q -d POWER -l 1 | grep "Power Draw"
```

Please see the NVIDIA nvidia-smi documentation for details. It's a very powerful and useful tool!

We will use a systemd Unit file to call a script to set GPU power limits at system startup. Here is a simple script to set the limits.

/usr/local/sbin/nv-power-limit.sh

```
#!/usr/bin/env bash

# Set power limits on all NVIDIA GPUs

# Make sure nvidia-smi exists
command -v nvidia-smi &> /dev/null || { echo >&2 "nvidia-smi not found ... exiting."; exit 1; }

POWER_LIMIT=280
MAX_POWER_LIMIT=$(nvidia-smi -q -d POWER | grep 'Max Power Limit' | tr -s ' ' | cut -d ' ' -f 6)

if [[ ${POWER_LIMIT%.*}+0 -lt ${MAX_POWER_LIMIT%.*}+0 ]]; then
    /usr/bin/nvidia-smi --persistence-mode=1
    /usr/bin/nvidia-smi  --power-limit=${POWER_LIMIT}
else
    echo 'FAIL! POWER_LIMIT set above MAX_POWER_LIMIT ... '
    exit 1
fi

exit 0
```

I like to use the "/usr/local" directory hierarchy for my own added system level applications, libraries and config files. I placed the above power limit script in /usr/local/sbin/nv-power-limit-sh You will have to be root (use sudo) to write in that directory. File permissions are set with,

```
chmod 744 /usr/local/sbin/nv-power-limit.sh
```

root has read, write, and execute permissions and "group" and "other" have read permission. You only want root to be able to modify or run this script!

Systemd Unit file to start nv-power-limit.service at boot time

The following systemd unit file will be placed in /usr/local/etc/systemd That subdirectory may not exist, you can create it (as root) with,

```
sudo mkdir /usr/local/etc/systemd
```

/usr/local/etc/systemd/nv-power-limit.service

```
[Unit]
Description=NVIDIA GPU Set Power Limit
After=syslog.target systemd-modules-load.service
ConditionPathExists=/usr/bin/nvidia-smi

[Service]
User=root
Environment="PATH=/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin"
ExecStart=/usr/local/sbin/nv-power-limit.sh

[Install]
WantedBy=multi-user.target
```

This file should have permissions set to 644 i.e. root has read and write permission and group and other have read permission.

```
sudo chmod 644 /usr/local/etc/systemd/nv-power-limit.service
```

With the unit service file in place we need to link it into the /etc/systemd/system directory so systemd can find it.

```
sudo ln -s /usr/local/etc/systemd/nv-power-limit.service /etc/systemd/system/nv-power-limit.service
```

Do an "ls -l /etc/systemd/service" and check that you got the link right.

After the power limit script is in place and the systemd unit file linked correctly you can check that it's working properly with,

```
sudo systemctl start nv-power-limit.service
```

and

```
sudo systemctl status nv-power-limit.service
```

With the system configuration used in this post "status" output looks like,

```
kinghorn@pslabs-ml1:~$ sudo systemctl status nv-power-limit.service
● nv-power-limit.service - NVIDIA GPU Set Power Limit
     Loaded: loaded (/usr/local/etc/systemd/nv-power-limit.service; linked; vendor preset: enabled)
     Active: inactive (dead)

Nov 23 16:11:25 pslabs-ml1 systemd[1]: Started NVIDIA GPU Set Power Limit.
Nov 23 16:11:27 pslabs-ml1 nv-power-limit.sh[14583]: Enabled persistence mode for GPU 00000000:53:00.0.
Nov 23 16:11:27 pslabs-ml1 nv-power-limit.sh[14583]: All done.
Nov 23 16:11:27 pslabs-ml1 nv-power-limit.sh[14587]: Power limit for GPU 00000000:53:00.0 was set to 280.00 W from
350.00 W.
Nov 23 16:11:27 pslabs-ml1 nv-power-limit.sh[14587]: All done.
Nov 23 16:11:27 pslabs-ml1 systemd[1]: nv-power-limit.service: Succeeded.
```

The last thing to do is to "enable" the service so that it will start at boot time.

```
sudo systemctl enable nv-power-limit.service
```

which should output,

```
Created symlink /etc/systemd/system/multi-user.target.wants/nv-power-limit.service → /usr/local/etc/systemd/nv-
power-limit.service.
```

When you restart your system the GPUs should be set to the power limit you configured in nv-power-limit.sh. It would be good to double check with,

```
nvidia-smi -q -d POWER
```

Conclusion

That's it! You should be able to run your Quad RTX3090 ML/AI rig with a single PSU and a reasonable power load.

If you try this be sure you understand the script and systemd unit file. Make changes as appropriate. I hope this post is helpful for everyone who is wanting to put 4 of these powerful RTX3090 to work. If you have suggestions on how things could be done better please put a note in the comments!

Happy computing! –dbk @dbkinghorn

Configure a System!

Related Content

- NVIDIA RTX4090 ML-AI and Scientific Computing Per...
- AMD Ryzen 7950X Scientific Computing Performanc...
- WSL2 vs Linux (HPL HPCG NAMD)
- Molecular Dynamics Benchmarks GPU Roundup GR...

View All Related Content

Latest Content

- Install Golang In Your Home Directory And Configure ...
- How To Create A Docker Container For AMD AOCCv4...
- Puget Systems' 2023 Event Schedule
- What H.264 and H.265 Hardware Decoding is Suppor...

View All

Why Choose Puget Systems?

Built specifically for you

Rather than getting a generic workstation, our systems are designed around your unique workflow and are optimized for the work you do every day.

We're Here, Give Us a Call!

We make sure our representatives are as accessible as possible, by phone and email. At Puget Systems, you can actually talk to a real person!

Fast Build Times

By keeping inventory of our most popular parts, and maintaining a short supply line to parts we need, we are able to offer an industry leading ship time.

Lifetime Labor & Tech Support

Even when your parts warranty expires, we continue to answer your questions and even fix your computer with no labor costs.
Click here for even more reasons!

Puget Systems Hardware Partners

Tags: Machine Learning, NVIDIA, RTX30 series, TensorFlow