Different CUDA versions shown by nvcc and NVIDIA-smi

Asked 3 years, 7 months ago   Modified 4 months ago   Viewed 142k times

224

87

I am very confused by the different CUDA versions shown by running `which nvcc` and `nvidia-smi`. I have both cuda9.2 and cuda10 installed on my ubuntu 16.04. Now I set the PATH to point to cuda9.2. So when I run

```
$ which nvcc
/usr/local/cuda-9.2/bin/nvcc
```

However, when I run

```
$ nvidia-smi
Wed Nov 21 19:41:32 2018
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 410.72       Driver Version: 410.72       CUDA Version: 10.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce GTX 106...  Off  | 00000000:01:00.0 Off |                  N/A |
| N/A   53C    P0    26W /  N/A |    379MiB /  6078MiB |      2%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0      1324      G   /usr/lib/xorg/Xorg                           225MiB |
|    0      2844      G   compiz                                       146MiB |
|    0     15550      G   /usr/lib/firefox/firefox                       1MiB |
|    0     19992      G   /usr/lib/firefox/firefox                       1MiB |
|    0     23605      G   /usr/lib/firefox/firefox                       1MiB |
```

So am I using cuda9.2 as `which nvcc` suggests, or am I using cuda10 as `nvidia-smi` suggests? I saw this answer but it does not provide direct answer to the confusion, it just asks us to reinstall the CUDA Toolkit, which I already did.

cuda

Share Follow

edited Aug 28, 2021 at 4:05
James Graham
39.4k ●42 ●170 ●245

asked Nov 22, 2018 at 0:44
yuqli
3,681 ●7 ●24 ●42

71 I think I've seen this exact question come up multiple times over the last couple days. But I can't seem to find a duplicate now. The answer is: nvidia-smi shows you the CUDA version that your driver supports. You have one of the recent 410.x drivers installed which support CUDA 10. The version the driver supports has nothing to do with the version you compile and link your program against. A driver that supports CUDA 10.0 will also be able to run an application that was built for CUDA 9.2... – Michael Kenzel Nov 22, 2018 at 1:12

1 @MichaelKenzel I see. Thanks for the clarification! Guess I'm using CUDA9.2 then.. – yuqli Nov 22, 2018 at 1:39

A similar question is here. @MichaelKenzel if you want to add an answer I would upvote. – Robert Crovella Nov 24, 2018 at 17:37

1 @RobertCrovella yes, that was the one I was looking for. I only learned the answer from your comment there, so if anyone deserves an upvote then it is you yourself ;) – Michael Kenzel Nov 24, 2018 at 18:13

4 Answers

Sorted by:
Trending sort available

Highest score (default)

292

CUDA has 2 primary APIs, the runtime and the driver API. Both have a corresponding version (e.g. 8.0, 9.0, etc.)

The necessary support for the driver API (e.g. libcuda.so on linux) is installed by the GPU driver installer.

The necessary support for the runtime API (e.g. libcudart.so on linux, and also `nvcc`) is installed by the CUDA toolkit installer (which may also have a GPU driver installer bundled in it).

In any event, the (installed) driver API version may not always match the (installed) runtime API version, especially if you install a GPU driver independently from installing CUDA (i.e. the CUDA toolkit).

The `nvidia-smi` tool gets installed by the GPU driver installer, and generally has the GPU driver in view, not anything installed by the CUDA toolkit installer.

Recently (somewhere between 410.48 and 410.73 driver version on linux) the powers-that-be at NVIDIA decided to add reporting of the CUDA Driver API version installed by the driver, in the output from `nvidia-smi`.

This has no connection to the installed CUDA runtime version.

`nvcc`, the CUDA compiler-driver tool that is installed with the CUDA toolkit, will always report the CUDA runtime version that it was built to recognize. It doesn't know anything about what driver version is installed, or even if a GPU driver is installed.

Therefore, by design, these two numbers don't necessarily match, as they are reflective of two different things.

If you are wondering why `nvcc -V` displays a version of CUDA you weren't expecting (e.g. it displays a version other than the one you think you installed) or doesn't display anything at all, version wise, it may be because you haven't followed the mandatory instructions in step 7 (prior to CUDA 11) (or step 6 in the CUDA 11 linux install guide) of the cuda linux install guide

Note that although this question mostly has linux in view, the same concepts apply to **windows** CUDA installs. The driver has a CUDA driver version associated with it (which can be queried with `nvidia-smi`, for example). The CUDA runtime also has a CUDA runtime version associated with it. The two will not necessarily match in all cases.

In most cases, if `nvidia-smi` reports a CUDA version that is numerically equal to or higher than the one reported by `nvcc -V`, this is not a cause for concern. That is a defined compatibility path in CUDA (newer drivers/driver API support "older" CUDA toolkits/runtime API). For example if `nvidia-smi` reports CUDA 10.2, and `nvcc -V` reports CUDA 10.1, that is generally not cause for concern. It should just work, and it does not necessarily mean that you "actually installed CUDA 10.2 when you meant to install CUDA 10.1"

If `nvcc` command doesn't report anything at all (e.g. `Command 'nvcc' not found...`) or if it reports an unexpected CUDA version, this may also be due to an incorrect CUDA install, i.e the mandatory steps mentioned above were not performed correctly. You can start to figure this out by using a linux utility like `find` or `locate` (use man pages to learn how, please) to find your `nvcc` executable. Assuming there is only one, the path to it can then be used to fix your PATH environment variable. The CUDA linux install guide also explains how to set this. You may need to adjust the CUDA version in the PATH variable to match your actual CUDA version desired/installed.

Similarly, when using docker, the `nvidia-smi` command will generally report the driver version installed on the base machine, whereas other version methods like `nvcc --version` will report the CUDA version installed inside the docker container.

Similarly, if you have used another installation method for the CUDA "toolkit" such as Anaconda, you may discover that the version indicated by Anaconda does not "match" the version indicated by `nvidia-smi`. However the above comments still apply. Older CUDA toolkits installed by Anaconda can be used with newer versions reported by `nvidia-smi`, and the fact that `nvidia-smi` reports a newer/higher CUDA version than the one installed by Anaconda does not mean you have an installation problem.

Here is another question that covers similar ground. The above treatment does not in any way indicate that this answer is only applicable if you have installed multiple CUDA versions instentionally or unintentionally. The situation presents itself *any time you install CUDA*. The version reported by `nvcc` and `nvidia-smi` may not match, and that is **expected** behavior and in most cases quite normal.

Share Follow

edited Nov 20, 2021 at 19:03

community wiki
11 revs, 2 users 98%
Robert Crovella

4 Difference between the driver and runtime APIs according to Nvidia. – HongboZhu Sep 26, 2019 at 14:34

1 @Rober Crovella Thanks for the clearity. I have the same situation nvidia-smi shows CUDA version 10.1 and nvcc shows 9.1. Now is it okay to train the network or does the installation is okay still works? – Dhiren Hamal Dec 6, 2019 at 8:43

I followed the post-installation steps, but still, nvcc and nvidia-smi are showing different cuda version – BeingMIAkashs Jan 25, 2020 at 18:08

That's entirely possible. If you installed the latest driver (causing `nvidia-smi` to display CUDA 10.2, currently) but an earlier version of CUDA (say, 10.1) then there will be a difference reported by `nvidia-smi` as compared to `nvcc`. It's not a problem in that case. – Robert Crovella Jan 25, 2020 at 18:53

1 The comment is helpful, but doesn't explain what happens if `nvcc` reports a higher version (say 10.2) than `nvidia-smi` (say 10.1). In that case, the Cuda tries to compile it as 10.2 and tries to run it on 10.1. This usually results in runtime error, `"RuntimeError: CUDA error: no kernel image is available for execution on the device"` as an example. – TheSaurus Oct 30, 2020 at 19:23

7

`nvcc` is in the CUDA bin folder - as such check if the CUDA bin folder has been added to your `$PATH`.

Specifically, ensure that you have carried out the CUDA Post-Installation actions (e.g. from here):

1. Add the CUDA Bin to `$PATH` (i.e. add the following line to your `~/.bashrc`)

```
export PATH=/usr/local/cuda-10.1/bin:/usr/local/cuda-10.1/NsightCompute-
2019.1${PATH:+:${PATH}}
```

> PS. Ensure the following two paths above, exist first: `/usr/local/cuda-10.1/bin` and `/usr/local/cuda-10.1/NsightCompute-2019.1` (the NsightCompute path could have a slightly different ending depending on the version of Nsight compute installed...

2. Update `$LD_LIBRARY_PATH` (i.e. add the following line to your `~/bashrc`).

```
export LD_LIBRARY_PATH=/usr/local/cuda-10.1/lib64\
                         ${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
```

After this, both `nvcc` and `nvidia-smi` (or `nvtop`) report the same version of CUDA...