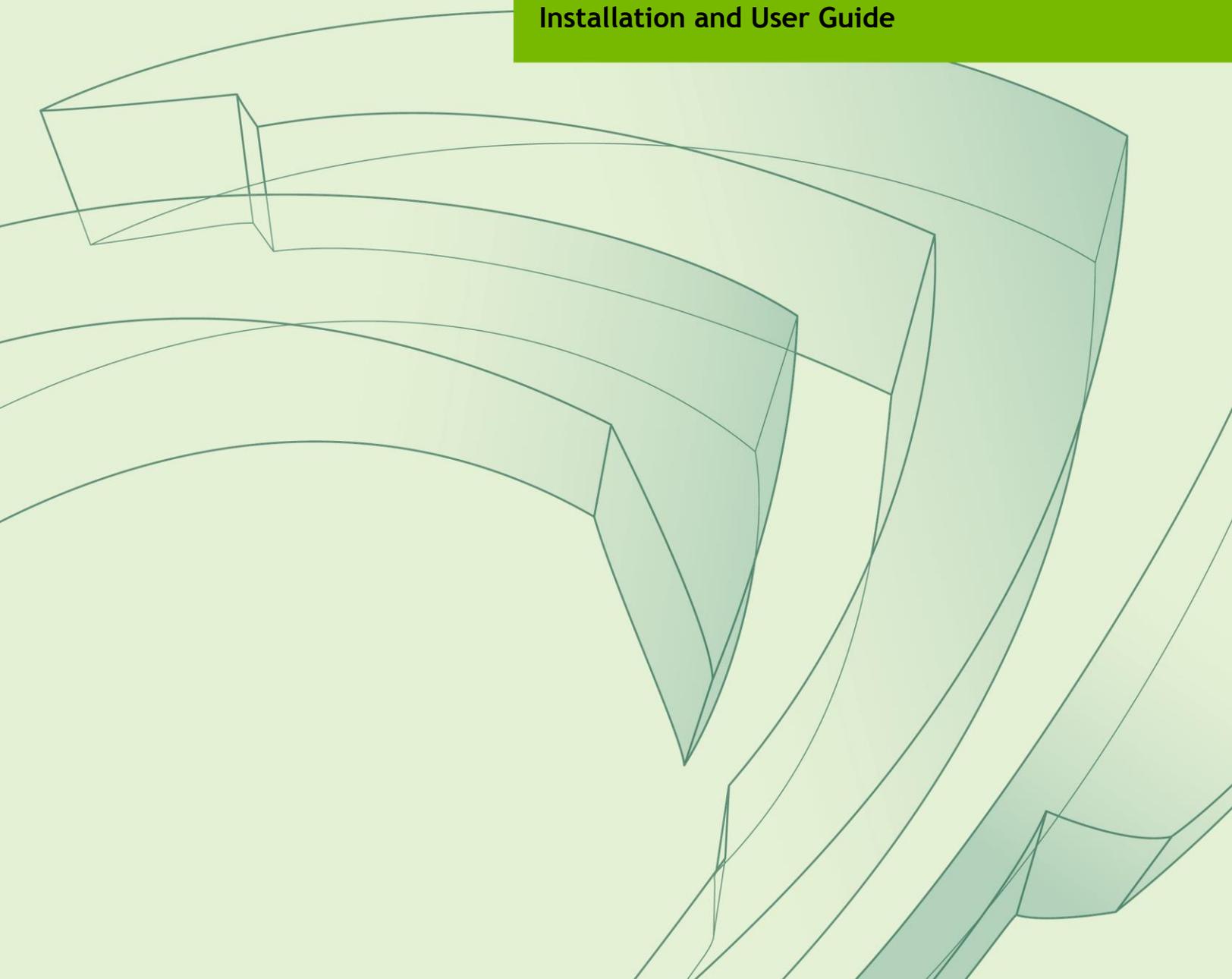




FFMPEG WITH NVIDIA ACCELERATION ON UBUNTU LINUX

DU-07857-001_v01 | November 2015

Installation and User Guide



DOCUMENT CHANGE HISTORY

DU-07857-001_v01

Version	Date	Authors	Description of Change
01	11/09/2015	ER	Initial release

TABLE OF CONTENTS

Step-by-Step Setup and Installation	1
Initial Setup	2
Install the Display Driver	3
NVENC SDK	7
CUDA Utility	7
Open Source Libraries.....	8
Build It All Together... ..	9
Using FFMPEG with NVENC	11
Transcode Performance.....	11
Measuring CPU and GPU Utilization.....	12
CPU Utilization	12
GPU Utilization	12
Multiple outputs from an input (1:n)	13
Resize Example	14
Transcode Quality	16
Presets	16
VBV Buffer	16
B-Frames.....	17
Group of Pictures (GOP).....	17
Adjusting Bit Budget Ratio Between I, P, & B Frames.....	18
Setting Quantization Limits	18

LIST OF FIGURES

Figure 1 - Selecting the 64bit Ubuntu 14.04 LTS network Ubuntu package	4
Figure 2 - Selecting the 64bit Ubuntu 14.04 LTS local Ubuntu package	5
Figure 3 - A typical 1:n resize scenario.....	13

STEP-BY-STEP SETUP AND INSTALLATION

This chapter describes how to obtain and install the necessary software for using FFmpeg with Ubuntu Linux (from 14.04 on). **The tasks must be completed in the order that they appear.** When finished, you'll have FFmpeg with support for the following:

- ▶ NVIDIA
 - NVENC (NVIDIA H/W Fixed Function video encoder for h.264 and HEVC)
 - GPU zero-copy engine
 - GPU accelerated resizer
- ▶ libx264 (x264 open source video codec for h.264/AVC)

INITIAL SETUP

These instructions assume a fresh install of x86_64 (64bit) Ubuntu 14.04 LTS.

If you are using another Linux distribution make sure that the NVIDIA display driver is at least r352.39. There are no other host specific requirements other than being x86_64 (64bit).

1. Install Ubuntu workstation (or server*).

*not tested

2. Boot the PC.

3. Log in as your username.

You'll need **internet access** and **sudo privilege** to run this install sequence.

4. Install the build infrastructure packages.

```
~ $ sudo apt-get install build-essential git yasm unzip wget sysstat
```



Note: Before you begin to run commands to install FFmpeg you'll need to get some NVIDIA Ubuntu packages and the NVENC SDK.

5. Download the NVIDIA Ubuntu packages into a directory called *Ubuntu*.

```
cd ~  
~ $ mkdir Ubuntu  
~ $ cd Ubuntu  
~/Ubuntu $
```

INSTALL THE DISPLAY DRIVER

1. To use NVENC on Linux the display driver must be version 352.39 or later. GPUs based on the [Kepler](#) or [Maxwell](#) architecture are supported.
2. There are two mechanisms to install NVIDIA display drivers.
 - Within the Ubuntu application management system (via .deb files)
 - A standalone installer (via .run files)

These cannot be mixed together. Since we are assuming a fresh install in these instructions we will continue to use the Ubuntu system.

NVIDIA Display Driver Ubuntu packages are released with each CUDA SDK. The CUDA 7.5 SDK repository from the NVIDIA developer's site has Display Driver r352.39 integrated.

We will download the CUDA 7.5 SDK to use that driver.

 **Note:** If you wish to use a more current version of the Display Driver you can download it from <http://www.nvidia.com/Download/index.aspx>. Before installation you must first uninstall all NVIDIA Ubuntu packages. See step 9 below for instructions on stopping the desktop, unloading and uninstalling the Ubuntu driver.

All the commands you will need to run are in **bold** below.

 **Note:** ONLY download the version that matches your OS.

3. You can manually download the driver following this link.
<https://developer.nvidia.com/cuda-downloads>

4. Below is an example of downloading the Ubuntu 14.04 LTS deb (network) version. The network version is a small (2.1k) package that will go to the internet for the actual packages only when those parts are installed.

Select Target Platform ?

Click on the green buttons that describe your target platform. Only supported platforms will be shown.

Operating System	Windows	Linux	Mac OSX
Architecture ?	x86_64	ppc64le	
Distribution	Fedora	OpenSUSE	RHEL
	SteamOS	Ubuntu	CentOS
Version	15.04	14.04	
Installer Type ?	runfile [local]	deb [local]	deb [network]

Download Target Installer for Linux Ubuntu 14.04 x86_64

cuda-repo-ubuntu1404_7.5-18_amd64.deb (md5sum:
f3e66fa414120f672dc46368816117ca)

[Download \[2.1 KB\]](#)

Installation Instructions:

1. ``sudo dpkg -i cuda-repo-ubuntu1404_7.5-18_amd64.deb``
2. ``sudo apt-get update``
3. ``sudo apt-get install cuda``

For further information, see the [Installation Guide for Linux](#) and the [CUDA Quick Start Guide](#).

Figure 1 - Selecting the 64bit Ubuntu 14.04 LTS network Ubuntu package

5. Save the above file in the `~/Ubuntu` directory or use the command line utility `wget` to download the file using this URL.

```
~/Ubuntu $ wget
```

```
http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1404/x86\_64/cuda-repo-ubuntu1404\_7.5-18\_amd64.deb
```

6. If your computer will not always have internet access you can download the local version instead. It has the display driver and all the CUDA development packages so it is a sizable 1.9GB.

Figure 2 - Selecting the 64bit Ubuntu 14.04 LTS local Ubuntu package

7. Save the above file in the `~/Ubuntu` directory or use the command line utility `wget` to download the file using this URL.

```
~/Ubuntu $ wget
http://developer.download.nvidia.com/compute/cuda/7.5/Prod/local\_installers/cuda-repo-ubuntu1404-7-5-local\_7.5-18\_amd64.deb
```

8. Add the repository you downloaded to your local repository.

```
~/Ubuntu $ sudo dpkg -i cuda-repo-ubuntu1404*_7.5-18_amd64.deb
~/Ubuntu $ sudo apt-get update
```

9. Before we can install the NVIDIA display driver we will need to turn off the desktop and go back to a terminal prompt.

```
~/Ubuntu $ sudo service lightdm stop
```

10. If you do not see a text prompt you will need to change to another virtual console. To do that press Alt-F1 or Alt-F2. You will then see a login prompt. Login as usual.
11. We now need to unload and remove the existing NVIDIA drivers. First we will see what is loaded.

```
~/Ubuntu $ lsmod | grep nvidia
nvidia_uvm          76757  0
nvidia              8604684  1 nvidia_uvm
drm                 303102  1 nvidia
```

Remove each module in the reverse order of use. In the printout above you can see that *nvidia* is used by *nvidia_uvm* so we have to remove *nvidia_uvm* that first then *nvidia*.

```
~/Ubuntu $ sudo rmmod nvidia_uvm
```

```
~/Ubuntu $ sudo rmmod nvidia
```

12. Once no NVIDIA modules are loaded we can uninstall the driver

```
~/Ubuntu $ sudo apt-get remove nvidia*
```



Note: If you are using a newer display driver skip step 13 below and run the standalone installer instead. Click Accept and click through all the default options.

e.g. `~/Ubuntu $ sudo ./NVIDIA-Linux-x86_64-352.55.run`

13. Install the NVIDIA display driver bundled in the CUDA 7.5 SDK repository you downloaded from above.

```
~/Ubuntu $ sudo apt-get install nvidia-352
```

14. Once installed restart the desktop.

```
~/Ubuntu $ sudo service lightdm start
```

```
~/Ubuntu $ sudo restart
```

NVENC SDK

1. Get the NVENC SDK from the NVIDIA developer's site.

<https://developer.nvidia.com/nvidia-video-codec-sdk>

By clicking the link below or downloading via “*wget*”, you are confirming that you have read and agree to be bound by the [NVIDIA VIDEO CODEC SDK LICENSE AGREEMENT](#).

http://developer.download.nvidia.com/compute/nvenc/v5.0/nvenc_5.0.1_sdk.zip

2. Copy the NVENC SDK to ~/Development.

```
~ $ mkdir Development
```

```
~ $ cd Development
```

```
~/Development $ wget
```

```
http://developer.download.nvidia.com/compute/nvenc/v5.0/nvenc\_5.0.1\_sdk.zip
```

```
~/Development $ unzip nvenc_5.0.1_sdk.zip
```

3. Copy the NVENC headers to */usr/local/include* to make it easier later.

```
~/Development $ sudo cp nvenc_5.0.1_sdk/Samples/common/inc/*.h
/usr/local/include
```

CUDA UTILITY

1. Download and install a light-weight library to communicate with the CUDA display driver.

```
~/Development/ $ wget
```

```
http://developer.download.nvidia.com/compute/redist/ffmpeg/1511-patch/cudautils.zip
```

2. Copy the CUDA utility to ~/Development/.

```
~/Development/ $ unzip cudautils.zip
```

```
~/Development/cudautils/ $ cd cudautils
```

3. Build the CUDA utility.

```
~/Development/cudautils/ $ make
```

```
~/Development/cudautils/ $ cd ..
```

```
~/Development/ $
```

OPEN SOURCE LIBRARIES

Download and install all the open source libraries.

1. Get x264.

```
~/Development $ git clone git://git.videolan.org/x264.git
```

```
~/Development $ cd x264
```

2. Configure x264.

```
~/Development/x264 $ ./configure \  
  
--disable-cli \  
  
--enable-static \  
  
--enable-shared \  
  
--enable-strip
```

3. Build x264.

```
~/Development/x264 $ make -j 10
```

4. Install x264.

```
~/Development/x264 $ sudo make install
```

```
~/Development/x264 $ sudo ldconfig
```

```
~/Development/x264 $ cd ..
```

```
~/Development/ $
```

BUILD IT ALL TOGETHER...

1. Get FFmpeg.

```
~/Development/ $ git clone git://source.ffmpeg.org/ffmpeg.git
```

2. Download the NVIDIA acceleration.

```
~/Development/ $ wget
http://developer.download.nvidia.com/compute/redist/ffmpeg/1511-
patch/ffmpeg\_NVIDIA\_gpu\_acceleration.patch
```

```
~/Development/ $ cd ffmpeg
```

3. Apply the NVIDIA acceleration patch. Note that this patch was created against the git master commit:

```
commit b83c849e8797fbb972ebd7f2919e0f085061f37f
Date: Tue Nov 10 04:14:55 2015 +010
```

```
~/Development/ffmpeg $ git apply ../ffmpeg_NVIDIA_gpu_acceleration.patch
```

4. Configure FFmpeg with NVENC, NVRESIZE and x264 support.

```
~/Development/ffmpeg $ cd ..
```

```
~/Development/ $ mkdir ffmpeg_build
```

```
~/Development/ $ cd ffmpeg_build
```

```
~/Development/ffmpeg_build $ ../ffmpeg/configure --enable-nonfree \
--enable-nvenc \
--enable-nvresize \
--extra-cflags=-I../cudautils \
--extra-ldflags=-L../cudautils \
--enable-gpl \
--enable-libx264
```

5. Build FFmpeg.

```
~/Development/ffmpeg_build $ make -j 10
```

6. Check that FFmpeg works. If NVENC and libx264 built properly you should get them in this list of encoders. We can filter the list down to h.264 encoders with “*grep 264*”.

```
~/Development/ffmpeg_build $ ./ffmpeg -encoders | grep 264
ffmpeg version N-76328-g1b82a00 Copyright (c) 2000-2015 the FFmpeg developers
built with gcc 4.8 (Ubuntu 4.8.4-2ubuntu1~14.04)
configuration: --enable-nonfree --enable-nvenc --enable-nvresize --extra-cflags=-I../cudautils
               --extra-ldflags=-L../cudautils --enable-gpl --enable-libx264
libavutil      55.  4.100 / 55.  4.100
libavcodec     57. 12.100 / 57. 12.100
libavformat    57. 11.100 / 57. 11.100
libavdevice    57.  0.100 / 57.  0.100
libavfilter     6. 14.100 /  6. 14.100
libswscale     4.  0.100 /  4.  0.100
libswresample  2.  0.100 /  2.  0.100
libpostproc   54.  0.100 / 54.  0.100
V..... libx264                libx264 H.264 / AVC / MPEG-4 AVC / MPEG-4 part 10 (codec h264)
V..... libx264rgb             libx264 H.264 / AVC / MPEG-4 AVC / MPEG-4 part 10 RGB (codec h264)
V..... nvenc                  NVIDIA NVENC h264 encoder (codec h264)
V..... nvenc_h264             NVIDIA NVENC h264 encoder (codec h264)
```

7. Check that FFmpeg has the NVRESIZE video filter. We can filter the list down with “*grep nvresize*”.

```
~/Development/ffmpeg_build $ ./ffmpeg -filters | grep nvresize
ffmpeg version N-76328-g1b82a00 Copyright (c) 2000-2015 the FFmpeg developers
built with gcc 4.8 (Ubuntu 4.8.4-2ubuntu1~14.04)
configuration: --enable-nonfree --enable-nvenc --enable-nvresize --extra-cflags=-I../cudautils
               --extra-ldflags=-L../cudautils --enable-gpl --enable-libx264
libavutil      55.  4.100 / 55.  4.100
libavcodec     57. 12.100 / 57. 12.100
libavformat    57. 11.100 / 57. 11.100
libavdevice    57.  0.100 / 57.  0.100
libavfilter     6. 14.100 /  6. 14.100
libswscale     4.  0.100 /  4.  0.100
libswresample  2.  0.100 /  2.  0.100
libpostproc   54.  0.100 / 54.  0.100
... nvresize      V->N          GPU accelerated video resizer.
```

8. Install FFmpeg.

```
~/Development/ffmpeg_build $ sudo make install

~/Development/ffmpeg_build $ sudo ldconfig

~/Development/ffmpeg_build $ cd ..

~/Development/ $ cd ..

~/ $
```

USING FFMPEG WITH NVENC

The following command lines will compare NVENC to x264:

TRANSCODE PERFORMANCE

This comparison will measure the time taken to transcode an input file to h.264 @5Mbps. It will copy the audio track to the output (if present).

In the interest of space and avoiding line wrapping in this document the Linux “continue command on next line” operator will be used (`\`) to wrap the command to the next line.

Using NVENC:	Using x264:
<pre>~/ \$ time ffmpeg -y -i <mp4 input file> \ -vcodec nvenc -b:v 5M \ -acodec copy \ <OUTPUT.mp4></pre>	<pre>~/ \$ time ffmpeg -y -i <INPUT.mp4> \ -vcodec libx264 -b:v 5M \ -acodec copy \ OUTPUT.mp4</pre>

MEASURING CPU AND GPU UTILIZATION

When comparing NVENC to x264 it is useful to monitor the CPU and GPU utilization. To do this we will use two command line tools.

CPU Utilization

The important column is the cpu “us” (user) utilization.

```
ubuntu@localmachine:~$ vmstat -w -n 1
```

```
procs -----memory----- --swap-- ----io---- -system-- -----cpu-----
 r b      swpd      free      buff      cache    si   so   bi   bo   in   cs  us  sy  id  wa  st
 9  0    29748   187236   170928   445720    0    0   384    0 4522 10101 81  0 19  0  0
 5  0    29748   183020   170928   449664    0    0  2432    0 4668 11630 80  0 19  0  0
12  0    29748   179424   170936   452716    0    0  2304   12 4571 11792 75  0 25  0  0
 9  0    29748   176820   170936   455496    0    0  1408   52 4809 10742 83  0 17  0  0
14  0    29748   174092   170936   458180    0    0  1536    0 4925 10767 84  0 15  0  0
15  1    29748   170992   170936   460920    0    0  1664    0 4652 10504 80  0 20  0  0
```

GPU Utilization

The NVENC utilization can be seen in the “enc” column. The “sm” column is the CUDA workload. We will use this later when doing GPU resize.

```
ubuntu@localmachine:~$ nvidia-smi dmon -i 0
```

```
# gpu   pwr   temp   sm    mem    enc    dec   mclk  pclk
# Idx   W     C     %     %     %     %     MHz  MHz
   0    82   35    12    4     86    0   3304 1151
   0    82   35    11    4     90    0   3304 1151
   0    83   35    11    4     93    0   3304 1151
   0    83   35    12    5     92    0   3304 1151
   0    83   36    11    4     94    0   3304 1151
   0    83   36    9     4     96    0   3304 1151
```

MULTIPLE OUTPUTS FROM AN INPUT (1:N)

In many scenarios multiple output formats are created at the same time from the input format.

Software resize is CPU intensive and quickly bottlenecks the ability to encode. For that reason, NVIDIA has implemented a GPU zero-copy engine to share frames between plugins as well as a video filter that does GPU resize (“nvresize”).

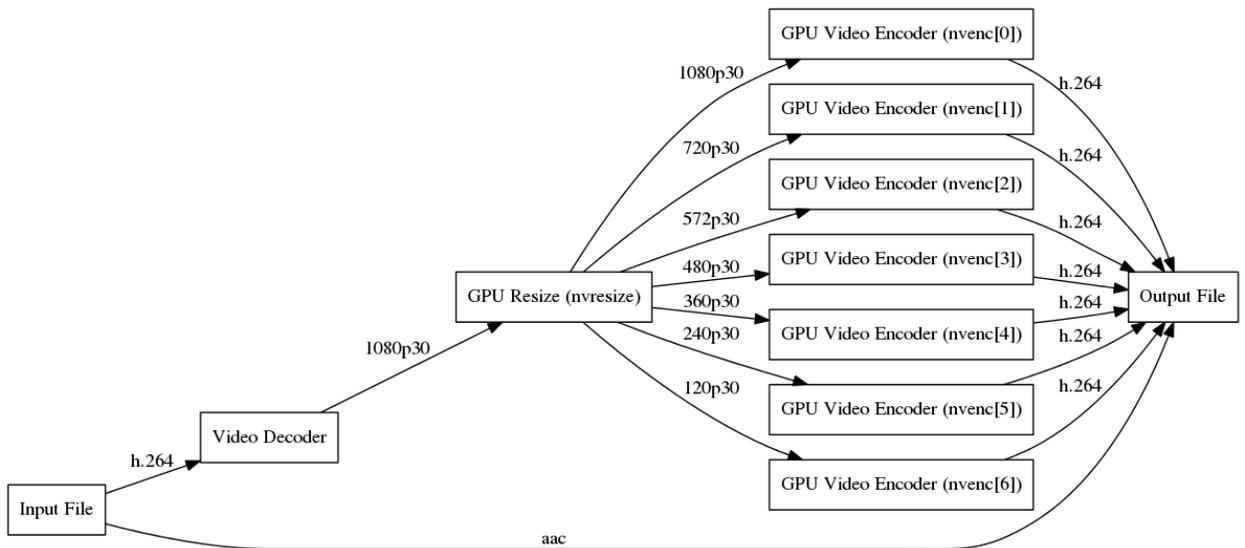


Figure 3 - A typical 1:n resize scenario

In the Figure above the video is resized into 7 formats and combined as different video streams in a single output container. The audio stream is copied from the input container to the new output container.

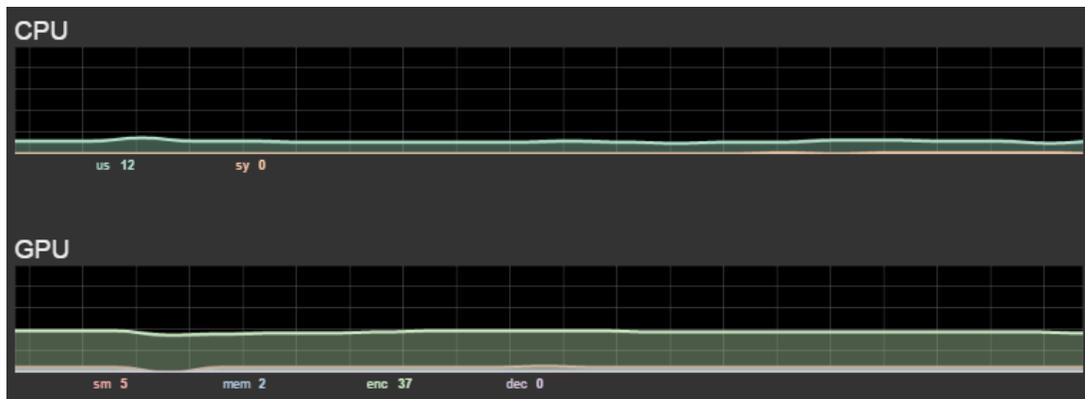
Resize Example

In the following example we will take a 1080p30 input file and downsize it to 5 formats, each stream is encoded with NVENC and then the output stream is put in its own container output file along with a copy of the audio (if present).

- ▶ Software resize provided resized frames at 75fps (375fps total - using 37% NVENC utilization).
- ▶ GPU resize provided resized frames at 190fps (950fps total - capped by 100% NVENC utilization).

Software based resize:

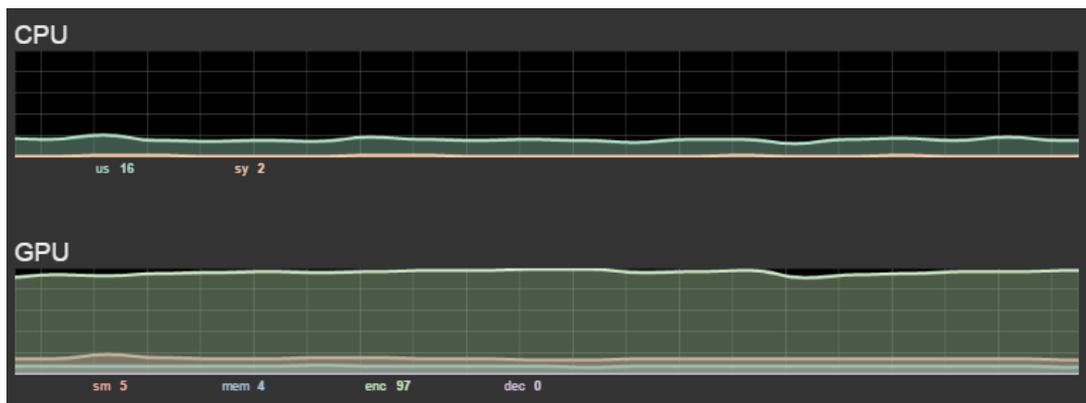
```
~/ $ time ffmpeg -y -i INPUT.mp4 \
    -acodec copy -vcodec nvenc -b:v 5M -s hd1080 out1sw.mkv \
    -acodec copy -vcodec nvenc -b:v 4M -s hd720 out2sw.mkv \
    -acodec copy -vcodec nvenc -b:v 3M -s hd480 out3sw.mkv \
    -acodec copy -vcodec nvenc -b:v 2M -s wvga out4sw.mkv \
    -acodec copy -vcodec nvenc -b:v 1M -s cif out5sw.mkv
```



GPU Accelerated Resize

Note how the pipe character ‘|’ has to be escaped “\|” for a bash shell.

```
~/ $ time ffmpeg -y -i INPUT.mp4 -filter_complex \
    nvresize=5:s=hd1080\|hd720\|hd480\|wvga\|cif:readback=0[out0][out1]
    [out2][out3][out4] \
    -map [out0] -acodec copy -vcodec nvenc -b:v 5M out0nv.mkv \
    -map [out1] -acodec copy -vcodec nvenc -b:v 4M out1nv.mkv \
    -map [out2] -acodec copy -vcodec nvenc -b:v 3M out2nv.mkv \
    -map [out3] -acodec copy -vcodec nvenc -b:v 2M out3nv.mkv \
    -map [out4] -acodec copy -vcodec nvenc -b:v 1M out4nv.mkv
```



TRANSCODE QUALITY

The above performance tests relied heavily on the default parameters. The quality of the output has not been optimized. We can make the output better by adjusting some of the encoding parameters.

These parameters are generic Ffmpeg commands so they apply to NVENC and x264. The following commands can be run again using x264 for comparison to NVENC by changing the “vcodec” from “nvenc” to “libx264”.

In the following examples we will add a tuning one by one. Each change is highlighted in yellow.

Presets

First we will use a higher quality preset. The NVENC “slow” preset turns on 2-pass encoding. This improves the rate control as well as the quality. We’ll discuss rate control later in these examples.

```
~/ $ time ffmpeg -y -i INPUT.mp4 \
    -vcodec nvenc -preset slow -b:v 5M \
    -acodec copy \
    OUTPUT.mp4
```

VBV Buffer

The H.264 standard includes a section called the VUI information. This describes how fast the video stream can be transmitted and the size of the FIFO buffer on the target decoder. Defining the VUI buffer size (VBV) as well as the maximum rate it can be filled controls how much the current bitrate can deviate from the target bitrate. A good size for the VBV is 2 seconds of video - thus twice the size of the desired target bitrate. In this example, let’s allow it to fill at twice the target bitrate.

```
~/ $ time ffmpeg -y -i INPUT.mp4 \
    -vcodec nvenc -preset slow -b:v 5M \
    -maxrate 10M -bufsize:v 10M -bf 2 -ref 1 \
    -acodec copy \
    OUTPUT.mp4
```

B-Frames

Next we will add B-frames. These are the most efficient frames in the H.264 standard.

We will also limit to one reference frame (for broader H.264 player compatibility). Reference frames are the frames that can be referred to by B frames.

```
~/ $ time ffmpeg -y -i INPUT.mp4 \
    -vcodec nvenc -preset slow -b:v 5M \
    -maxrate 10M -bufsize:v 10M -bf 2 -ref 1 \
    -bf 2 -ref 1 \
    -acodec copy \
    OUTPUT.mp4
```

Group of Pictures (GOP)

Next we will define the order of I, P, and B frames. A collection of these is called a GOP. When you seek back and forth on video you must start at a GOP boundary. For internet video this is typically 5 seconds. For control over the experience, for example on Blu-ray, it's 1 second. Let's use a 5-second GOP in this example and assume the input is 30fps - thus the GOP is $30 \times 5 = 150$.

Setting the number of B-frames and the GOP defines the order of I, P, and B frames.

For example: B-frames = 2, GOP = 15 would result in

```
I P B B P B B P B B P B B P I
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
```

```
~/ $ time ffmpeg -y -i INPUT.mp4 \
    -vcodec nvenc -preset slow -b:v 5M \
    -maxrate 10M -bufsize:v 10M -bf 2 -ref 1 \
    -bf 2 -ref 1 -g 150 \
    -acodec copy \
    OUTPUT.mp4
```

Adjusting Bit Budget Ratio Between I, P, & B Frames

Next we will adjust the ratio of bits used in I, P and B frames. By adjusting this bias we can tune the overall perceived quality. This parameter is content dependent. The values in this example are for real world videos not computer generated video content like animated stories or computer gameplay.

```
~/ $ time ffmpeg -y -i INPUT.mp4 \
    -vcodec nvenc -preset slow -b:v 5M \
    -maxrate 10M -bufsize:v 10M -bf 2 -ref 1 \
    -bf 2 -ref 1 -g 150 \
    -i_qfactor 1.1 -b_qfactor 1.25 \
    -acodec copy \
    OUTPUT.mp4
```

Setting Quantization Limits

Quantization is the complexity of each frame. By setting maximum and minimum limits we can control how wide the deviation is from the target bitrate. A quantization (“qp”) of 1 is basically lossless. Anything above 30 is really complex. For this example we’ll use a range of 1..50.

```
~/ $ time ffmpeg -y -i INPUT.mp4 \
    -vcodec nvenc -preset slow -b:v 5M \
    -maxrate 10M -bufsize:v 10M -bf 2 -ref 1 \
    -bf 2 -ref 1 -g 150 \
    -i_qfactor 1.1 -b_qfactor 1.25 \
    -qmin 1 -qmax 50 \
    -acodec copy \
    OUTPUT.mp4
```

Notice

The information provided in this specification is believed to be accurate and reliable as of the date provided. However, NVIDIA Corporation ("NVIDIA") does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This publication supersedes and replaces all other specifications for the product that may have been previously supplied.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and other changes to this specification, at any time and/or to discontinue any product or service without notice. Customer should obtain the latest relevant specification before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer. NVIDIA hereby expressly objects to applying any customer general terms and conditions with regard to the purchase of the NVIDIA product referenced in this specification.

NVIDIA products are not designed, authorized or warranted to be suitable for use in medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer's own risk.

NVIDIA makes no representation or warranty that products based on these specifications will be suitable for any specified use without further testing or modification. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to ensure the product is suitable and fit for the application planned by customer and to do the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this specification. NVIDIA does not accept any liability related to any default, damage, costs or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this specification, or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this specification. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA. Reproduction of information in this specification is permissible only if reproduction is approved by NVIDIA in writing, is reproduced without alteration, and is accompanied by all associated conditions, limitations, and notices.

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the NVIDIA terms and conditions of sale for the product.

Trademarks

NVIDIA and the NVIDIA logo are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

© 2015 NVIDIA Corporation. All rights reserved.