NETINT

APPS002-Codensity^{TM} T408 Operation Troubleshooting Application Note

V2.0

2021-12-09

Table of Contents

1 Table of Abbreviations

Abbreviation:	Full Form:
AI	Artificial Intelligence
API	Application Programming Interface
AUD	Access Unit Delimiters
AV	Audio Video
AV1	Alliance Open Media Video 1 Codec
CABAC	Context Adaptive Binary Arithmetic Coding
CBR	Constant Bit Rate
CPU	Central Processing Unit
CRF	Constant Rate Factor
DSP	Digital Signal Processing
EOF	End Of File
FIXQB	Fixed Quantization Parameter
FF	Form Factor
FPS	Frames Per Second
GOP	Group of pictures
HDR	High Dynamic Range
HHHL	Half Height, Half Length
HHFL	Half Height, Full Length
HLG	Hybrid Log Gamma
HRD	Hypothetical Reference Decoder
IE	Inference Engine
Mbps	Mega Bit per second
MBps	Mega Byte per second
NLP	Natural Language Processing
NPU	Neural Processing Unit
NVMe	Non-Volatile Memory express
PCIe	Peripheral Component Interconnect express
PPS	Picture Parameter Set
QP	Quoted Printable
RGB	Red Green Blue
RGBA	Red Green Blue Alpha
RDO	Optimization
ROI	Region of interest
SDK	Software Development Kit
SEI	Supplemental Enhancement Information
SRIOV	Single Root I/O Virtualization
SPS	Sequence Parameter Set
VBR	Variable Bit Rat
VCL	Video Coding Layer
VPS	Video Parameter Set
VUI	Video Usability Information

2 References

Technical Notes:

Specifications:

3 Background

This document records common problems and solutions that may be encountered when operating/integrating the NETINT T408 transcoder.

3.1 Intended Audience

This document is intended for to help engineers/technicians/developers operating NETINT transcoders troubleshoot common problems.

3.2 Compatibility

Software Compatibility

This guide is intended to be used with NETINT Codensity T408 Video Transcoder software Release 1.5.0 or newer release.

Hardware Compatibility

This guide is intended to be used with NETINT Codensity T408 Video Transcoder firmware Release 1.5.0 or newer release.

Operation system

-

4 FFmpeg-n3.1.1 cannot start in background

4.1 Affected scope

FFmpeg-n3.1.1 installations in all NETINT releases

4.2 Symptoms

When starting an FFmpeg-n3.1.1 transcoding command with T408 in background, the process will freeze before processing any frames. Ex:

```
sudo ffmpeg -nostdin -y -hide_banner -c:v h264_ni_dec -dec
0 -i 1280x720p_Basketball.264 -c:v h265_ni_enc -enc 0
1280x720p_Basketball_NETINT_xcod.265 &
```

4.3 Problem

When a process is run it opens 3 stdIO pipes: stdin, stdout and stderr. Typically, each pipe has a set memory buffer size that can hold up to a certain amount of data (4KB on Linux). If the pipe becomes full, it will block the writing process until data is read out of the pipe.

FFmpeg-n3.1.1 is prone to stdIO pipe blockage when starting as a background process. This is an issue native to FFmpeg-n3.1.1.

4.4 Solutions

FFmpeg 3.1.1 processes can be put into background after it has started processing frames:

1. Start FFmpeg-n3.1.1 process in foreground
2. Press keys: <ctrl> + z
3. Place into background by entering: bg

FFmpeg-n3.1.1 processes can be started in background by redirecting its stdIO pipes to null. Ex:

```
sudo ffmpeg -nostdin -y -hide_banner -c:v h264_ni_dec -dec
0 -i 1280x720p_Basketball.264 -c:v h265_ni_enc -enc 0
1280x720p_Basketball_NETINT_xcod.265 > /dev/null 2>&1 <
/dev/null &
```

5 NVMe Multipathing Incompatibility

5.1 Affected scope

All NETINT releases up to and including v1.1.0

Possibility to affect systems deploying more than one NETINT T408

5.2 Symptoms

When starting an FFmpeg command using the T408, the process will close citing 'Conversion failed!'.

The output of the following command:

```
ls /sys/class/block | grep nvme
```

Shows that the nvme controller index does not match the controller index recorded in the nvme namespace ID. Ex:

```
netint@transcoder:~$ ls -l /sys/class/block | grep nvme

lrwxrwxrwx 1 root root 0 Sep 10 11:33 nvme0n1 ->
../../devices/pci0000:10/0000:10:01.1/0000:11:00.0/0000:12:
02.0/0000:15:00.0/nvme/nvme2/nvme0n1

lrwxrwxrwx 1 root root 0 Sep 10 11:33 nvme1n1 ->
../../devices/pci0000:10/0000:10:01.1/0000:11:00.0/0000:12:
00.0/0000:13:00.0/nvme/nvme0/nvme1n1

lrwxrwxrwx 1 root root 0 Sep 10 11:33 nvme2n1 ->
../../devices/pci0000:10/0000:10:01.1/0000:11:00.0/0000:12:
03.0/0000:16:00.0/nvme/nvme3/nvme2n1

lrwxrwxrwx 1 root root 0 Sep 10 11:33 nvme3n1 ->
../../devices/pci0000:10/0000:10:01.1/0000:11:00.0/0000:12:
01.0/0000:14:00.0/nvme/nvme1/nvme3n1
```

Note here that linux sysfs maps nvme2 to nvme0n1, nvme0 to nvme1n1, nvme3 to nvme2n1, nvme1 to nvme3n1. This is due to nvme-multipathing.

5.3 Problem

Currently, NETINT's libxcoder does not support nvme-multipathing. The controller ID referenced in the nvme controller device must match the controller ID referenced in the nvme namespace device as connected by the linux kernel in sysfs.

5.4 Solutions

Disable nvme-multipathing by running the following commands:

```
sudo sh -c "echo options nvme-core multipath=0 >
/etc/modprobe.d/nvme.conf";

sudo update-initramfs –u;

sudo reboot now;
```

Note, this workaround may be successful on some version of the linux kernel whilst unsuccessful on others. (eg. This solution is not successful on 4.15.0-29-generic but it successful on 4.15.0-64-generic)

6 FFmpeg "stream loop" feature does not work

6.1 Affected scope

FFmpeg-n3.1.1 and FFmpeg-n3.4 installations in all NETINT releases.

6.2 Symptoms

When starting an FFmpeg command using the T408 with the stream_loop <n> feature, the process will close after the first iteration citing 'Operation not permitted' while it is supposed to use the same input and loop n times.

Command line example:

```
sudo ffmpeg -stream_loop 3 -i 1920x1080p25_Pedestrian.264 -c:v
h264_ni_enc 1920x1080p25_Pedestrian.mp4
```

6.3 Problem

There are actually 2 different problems:
1- even with legacy SW ffmpeg it is not possible to loop if the input is a bitstream. An encapsulated file (ts, Mov, Flv ...) is required to have a correct continuity of time stamps when the stream is restarted. 2- using a mov file our HW encoder V4.1.3 was able to continue to next loop iterations but it was not the case for 3.4 and 3.1. In the literature it is mentioned that versions older than 4.0 have bug with the loop options (based on ffmpeg forum). {*Note that this option **was** buggy (see bug reports #5719, #6121, and #7043) so may not work for old builds (anything older than 2018 06-27 including the FFmpeg 4.0.x release branch)*}

6.4 Solutions

- For FFmpeg-4.1.3, just use an encapsulated file and it would work:

```
sudo ffmpeg -stream_loop 3 -i 1920x1080p25_Pedestrian.mp4 -c:v
h264_ni_enc 1920x1080p25_Pedestrian.mp4
```

- For FFmpeg<4.x, The ffmpeg concat format allows fixing the problem:

Create a file called 1920x1080p25_Pedestrian.txt with the following syntax:

```
file '1920x1080p25_Pedestrian.264'
file '1920x1080p25_Pedestrian.264'
file '1920x1080p25_Pedestrian.264'
```

{repeat as much as you need to loop}

Then:

```
sudo ffmpeg –f concat -i 1920x1080p25_Pedestrian.txt -c:v
h264_ni_enc  1920x1080p25_Pedestrian.mp4
```

7 Livestream audio and video does not start playing at same time or VLC debug messages show “unconvertible timestamp”

7.1 Affected scope

FFmpeg

7.2 Symptoms

When viewing an FFmpeg generated livestream (eg. mpegts/hls) the audio and vide beging playing at different times after seeking despite no obvious lipsync issues after both audio and video are playing. Another symptom may be that the video player crashes or VLC debugging messages show “unconvertible timestamps”.

Upon deeper analysis it is possible to verify that the audio and video stream packets are not well interleaved by PTS/DTS. Ie. the next audio packet after a video packet of PTS=N seconds has a PTS of N+8 seconds.

The FFmpeg command is typically also strictly set to use “-vsync 0” or “-af aresample=async=0”.

7.3 Problem

FFmpeg will attempt to evenly interleave all streams (audio, video, etc.) into the output stream based on DTS. This could fail if “-vsync 0” or “-af aresample=async=0” restricts FFmpeg from dropping/duplicating frames to match a constant output rate. This can lead to desynchronization in the output which is not well handled by default. For example, if “-vsync 0” is used in the transcoding FFmpeg command and the input video frame experiences packet loss in the input video signal then the DTS of the next frame will be corrected down but there will still be a PTS gap in the output. This gap will eventually build up in the muxing queue as the muxing interleaver interleaves all streams by DTS.

7.4 Solutions

When transcoding in a scenario where there may be packet drop at the input, or unexpected performance variations use the ffmpeg muxer parameter “-max_interleave_delta” or global parameter ”-vsync 1”.

The muxer parameter sets the maximum DTS delta in µs between the oldest and newest packet in the muxer queue before the oldest packet is flushed into the muxing interleaver immediately. This has the effect of closely tying the output packet interleaving uniformity to the input packet interleaving uniformity as all streams will only deviate from their input side by the set max_interleave_delta.

Example in command:

```
sudo ffmpeg -vsync 0 -i input_stream.ts -c:a copy -c:v
h265_ni_enc -f mpegts -max_interleave_delta 200000
output_stream.ts
```

Setting the global parameter “-vsync” to “-vsync 1” or “-vsync cfr” will allow FFmpeg to compensate for any PTS gaps in the input.

Example in command:

```
sudo ffmpeg -vsync 1 -i input_stream.ts -c:a copy -c:v
h265_ni_enc -f mpegts output_stream.ts
```

8 Revision History

Version	Change	Create Date	Author
1.0	Document Issued	January 28, 2020	Steven Zhou
2.0	Updated to new template	November 22, 2021	Balwinder Sidhu

9 Legal Notice