APPS040-Codensity™ T408 Docker Interworking Application Note

V2.0

Table of Contents

1 Table of Abbreviations

Abbreviation	Full Form
T408	Codensity T408 Video Transcoder
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 demonstrates the configuration and usage of T408 device in a Docker container.

The installation and configuration of a Linux host is not in scope of this document.

3.1 Intended Audience

This document is intended to help engineers/technicians/developers operating NETINT transcoders better understand the usage of T408 in Docker environment.

3.2 Compatibility

Software Compatibility

This guide is intended to be used with NETINT Codensity T408 Video Transcoder software Release 2.5.0 or newer.

Hardware Compatibility

Release 1.6.0 or newer supports NETINT Codensity T408 Video Transcoder hardware.

Operation system

All OS supporting T408 Video Transcoder.

4 Pre-requisite T408 Environment on Host

The Linux Host has the following working environment.

1] T408 libxcoder and FFmpegXcoder/FFmpeg-n4.2.1 source installed and compiled successfully. FFmpeg 4.2.1 version is referenced in this document. The user can use any other supported FFmpeg version of choice.

2] Running 'ffmpeg' gives a dump similar to sample below.

```
ffmpeg version 4.2.1 Copyright (c) 2000-2019 the FFmpeg developers
  built with gcc 4.8.5 (GCC) 20150623 (Red Hat 4.8.5-44)
  configuration: --pkg-config-flags=--static --enable-gpl --enable-nonfree --extra-ldflags=-lm --extra-ldflags=-ldl --enable-libxcoder --enable-pthreads --extra-libs=-lpthread --enable-encoders --enable-decoders --enable-avfilter --enable-muxers --enable-demuxers --enable-parsers --enable-x86asm --disable-debug --disable-ffplay --disable-ffprobe --disable-libx264 --disable-libx265 --disable-cuda-nvcc --disable-cuda --disable-cuvid --disable-nvdec --disable-nvenc --disable-libvmaf --enable-static --disable-shared --extra-cflags=-UNIENC_MULTI_THREAD
  libavutil      56. 31.100 / 56. 31.100
  libavcodec     58. 54.100 / 58. 54.100
  libavformat    58. 29.100 / 58. 29.100
  libavdevice    58.  8.100 / 58.  8.100
  libavfilter     7. 57.100 /  7. 57.100
  libswscale      5.  5.100 /  5.  5.100
  libswresample   3.  5.100 /  3.  5.100
  libpostproc    55.  5.100 / 55.  5.100
Hyper fast Audio and Video encoder
usage: ffmpeg [options] [[infile options] -i infile]... {[outfile options] outfile}...

Use -h to get full help or, even better, run 'man ffmpeg'
```

3] Running 'sudo nvme list' displays the installed T408 NVMe device.

```
Node                  SN                   Model
Namespace Usage                            Format           FW Rev
--------------- -------------------- ---------------------------------------- --------- -------------------------- ----------------
--- --------
/dev/nvme0n1          TU05-07-02-B04-0959    T408-U2
1             536.87    GB / 536.87    GB       512      B +    0 B      250X1F10
```

4] Running ‘ni_rsrc_mon’ or ‘sudo ni_rsrc_mon’ provide a dump similar to sample below.

```
NI resource init'd already ..
**************************************************
1 devices retrieved from current pool at start up
Mon May    3 22:33:33 2021 up 00:00:00 v250R1F10
Num decodes: 1
BEST INDEX LOAD MODEL_LOAD MEM    INST DEVICE              NAMESPACE
L       0        0       0              0       0       /dev/nvme0       /dev/nvme0n1
Num encodes: 1
BEST INDEX LOAD MODEL_LOAD MEM    INST DEVICE              NAMESPACE
L       0        0       0              0       0       /dev/nvme0       /dev/nvme0n1
**************************************************
```

5 Let's create a Docker Image titled t408xc

1] Install Docker module on your Linux Host. Refer links in Section 4.1 of this document for installation steps based on the OS type of your Linux Host.

2] Install Python Docker package.

```
sudo pip install docker
```

3] Create a text file titled 'Dockerfile' with the following contents. This file will be used for creating a Docker image on the host and installing minimum Docker environment to be able to detect the T408 NVMe device and run transcoding. This file also includes the minimum environment you will need to run a compile of libxcoder and ffmpeg in FFmpegXcoder.

```
FROM centos:centos7


# Install Libraries

RUN yum --enablerepo=extras install -y epel-release

RUN yum install -y make gcc gcc-c++ redhat-lsb-core yasm git
pkgconfig wget pciutils sudo.x86_64 nvme-cli.x86_64

ENV PKG_CONFIG_PATH /usr/local/lib/pkgconfig/

ENV LD_LIBRARY_PATH /usr/local/lib/

RUN echo "/usr/local/lib" >> /etc/ld.so.conf

RUN echo "Defaults
secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/s
bin:/bin:/snap/bin"" >> /etc/sudoers

RUN ldconfig

RUN yum install -y nvme-cli.x86_64

RUN yum install -y python-devel python3-pip python-pip

RUN python3 -m pip install --upgrade --user pip
```

```
EXPOSE 80

RUN ldconfig
```

4] Run the following command to create an image titled t408xc. Ensure the file 'Dockerfile' exists in your current folder.

```
cd ~

ls -al Dockerfile

sudo docker build --tag t408xc .
```

A successful image creation will end in a message as per sample below.

```
Successfully built 63d276aeef81

Successfully tagged centos7:latest
```

5] Confirm Docker image is created

```
sudo docker images

REPOSITORY      TAG         IMAGE ID        CREATED         SIZE

t408xc          latest      ab2aa2788f70    2 hours ago     1.07GB
```

5.1 Useful Links

1] https://linuxhint.com/install_configure_docker_ubuntu/

2] https://docs.docker.com/engine/install/centos/

3] https://download.docker.com/linux/centos/8/x86_64/stable/Packages/

6 Docker container – T408 interworking

6.1 Scope

Scope of this section is for demonstrating the T408 NVMe device transcoding session in a Docker container.

6.2 Launch Docker container with T408 device

Run the following command to start a container using Docker image t408xc.

```
sudo docker run -it --privileged -v ~/FFmpegXcoder:/FFmpegXcoder
--device /dev/nvme0n1 --device /dev/nvme0 t408xc /bin/sh
```

Note – The '-v' option can be updated to point to the host folder where FFmpeg release folder is installed. For example: ~/FFmpeg, ~/FFmpegXcoder etc.

You should get a shell prompt to demonstrate a successful container start.

```
Sh-4.2#
```

Running 'sudo nvme list' should list the T408 device.

```
sh-4.2# sudo nvme list
Node                  SN                   Model
Namespace Usage                      Format           FW Rev
---------------- -------------------- ---------------------------------------- --------- -------------------------- ----------
--- --------
/dev/nvme0n1          TU05-07-02-B04-0959  T408-U2
1            536.87  GB / 536.87  GB      512   B +  0 B    250X1F10
/dev/nvme1n1          21012B802361         WDS100T3X0C-00SJG0
1              1.00  TB /   1.00  TB      512   B +  0 B    111110WD
```

6.3 Compile ffmpeg in the Docker container

Section 5.2 has started a container with host volume ~/FFmpegXcoder mounted. Proceed to compile the libxcoder and ffmpeg same as you would compile on the host.

Once the ffmpeg is compiled successfully, run 'ni_rsrc_mon'.

```
sh-4.2# sudo ni_rsrc_mon

NI resource not init'd, continue ..

Reading device file: nvme1

Reading device file: nvme0

Libxcoder SW ver: 250R1F10

Compatible minimum FW ver: 250, FW API flavors: 1F, minimum API ver: 10

1. /dev/nvme0 /dev/nvme0n1 num_hw: 2

Device /dev/nvme1 /dev/nvme1n1 not added as it is not an xcoder, or has incompatible FW rev:

Creating shm_name: NI_SHM_CODERS     lck_name: /dev/shm/NI_LCK_CODERS

0. nvme0

decoder h/w id 0 create

Creating shm_name: NI_shm_d0 , lck_name /dev/shm/NI_lck_d0

ni_rsrc_get_one_device_info written out.

decoder h/w id 0 update

encoder h/w id 1 create

Creating shm_name: NI_shm_e0 , lck_name /dev/shm/NI_lck_e0

ni_rsrc_get_one_device_info written out.

encoder h/w id 1 update

1. nvme1

Device /dev/nvme1 not added as it is not an xcoder, or has incompatible FW rev:

**********************************************

1 devices retrieved from current pool at start up
```

Mon May 3 22:23:27 2021 up 00:00:00 v250R1F10

Num decodes: 1

BEST INDEX LOAD MODEL_LOAD MEM INST DEVICE NAMESPACE

L 0 0 0 0 0 /dev/nvme0 /dev/nvme0n1

Num encodes: 1

BEST INDEX LOAD MODEL_LOAD MEM INST DEVICE NAMESPACE

L 0 0 0 0 0 /dev/nvme0 /dev/nvme0n1

****

6.4 Transcoding

Run a transcoding operation using ffmpeg. Sample below is for avc to hevc transcode on a 1920x1080p_ParkScene.264 clip.

Note – There are some test clips in the FFmpegXcoder/libxcoder/test folder

sh-4.2# ffmpeg -y -nostdin -hide_banner -vsync 0 -c:v h264_ni_dec -dec 0 -i 1920x1080p_ParkScene.264 -vf scale=1920:1080 -xcoder-params intraQP=27:intraPeriod=120 -c:v h265_ni_enc -enc 0 -xcoder-params intraQP=27:intraPeriod=120 1920x1080p_ParkScene-0-0.265

Input #0, h264, from '1920x1080p_ParkScene.264':

Duration: N/A, bitrate: N/A

Stream #0:0: Video: h264 (High), yuv420p(progressive), 1920x1080, 24 fps, 24 tbr, 1200k tbn, 48 tbc

Stream mapping:

Stream #0:0 -> #0:0 (h264 (h264_ni_dec) -> hevc (h265_ni_enc))

[h264_ni_dec @ 0x22ac3c0] ff_xcoder_dec_receive: resolution changed: 1920x1080 to 1920x1088

[h265_ni_enc @ 0x22ace00] Session state: 0 allocate frame fifo.

[h265_ni_enc @ 0x22ace00] dts offset: 2, gop_offset_count: 0

Output #0, hevc, to '1920x1080p_ParkScene-0-0.265':

Metadata:

encoder : Lavf58.29.100

Stream #0:0: Video: hevc (h265_ni_enc), yuv420p, 1920x1080, q=2-31, 200 kb/s, 24 fps, 24 tbn, 24 tbc

Metadata:

encoder : Lavc58.54.100 h265_ni_enc

frame= 240 fps=155 q=-0.0 Lsize= 2859kB time=00:00:09.83 bitrate=2381.8kbits/s speed=6.35x

video:2859kB audio:0kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: 0.000000%

Decoder HW[0] INST[127]-average usage:52%

Encoder HW[0] INST[127]-average usage:66%

sh-4.2#

7 Revision History

Version	Change	Create Date	Author
0.1	Document Created	July 22, 2021	Balwinder Sidhu
1.0	Issued Document	July 22, 2021	Balwinder Sidhu
2.0	Updated as per new app note template	December 7, 2021	Balwinder Sidhu

8 Legal Notice

Information in this document is provided in connection with NETINT products. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by this document. Except as provided in NETINT's terms and conditions of sale for such products, NETINT assumes no liability whatsoever and NETINT disclaims any express or implied warranty, relating to sale and/or use of NETINT products including liability or warranties relating to fitness for a particular purpose, merchantability, or infringement of any patent, copyright or other intellectual property right.

A "Mission Critical Application" is any application in which failure of the NETINT Product could result, directly or indirectly, in personal injury or death. Should you purchase or use NETINT's products for any such mission critical application, you shall indemnify and hold NETINT and its subsidiaries, subcontractors and affiliates, and the directors, officers, and employees of each, harmless against all claims costs, damages, and expenses and reasonable attorney's fees arising out of, directly or indirectly, any claim of product liability, personal injury, or death arising in any way out of such mission critical application, whether or not NETINT or its subcontractor was negligent in the design, manufacture, or warning of the NETINT product or any of its parts.

NETINT may make changes to specifications, technical documentation, and product descriptions at any time, without notice. The information here is subject to change without notice. Do not finalize a design with this information. The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications.

NETINT, Codensity, and NETINT Logo are trademarks of NETINT Technologies Inc. All other trademarks or registered trademarks are the property of their respective owners.