Codensity™ T408 & T432 Quickstart Guide

Reference Number: 19TD002-13
3/22/2021

Reference Number: 19TD002-13
3/22/2021

Table of Contents

1 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.

2 NETINT Overview

NETINT provides high density and efficient video transcoding solutions using the powerful video processing engines inside our Codensity G4 Application-Specific Integrated Circuit (ASIC). We can provide multiple stream transcoding functions and services directly to video content providers and Transcoding as a Service (TaaS) providers for integration into their video streaming systems and services. Our functions and services can be used for highly efficient Video-on-Demand file transcoding, as well as real-time live video streaming applications.

This quick start guide provides an overview of NETINT video transcoding. It describes some of the most important transcoding parameters and the ways they are used and configured when integrating with different levels and parts of solutions that NETINT provides.

2.1 The Codensity T408/T432 Video Transcoders

Video content is the number one source of traffic on the Internet. Video is often generated using the ubiquitous H.264 AVC video encoding standard. Newer H.265 HEVC video delivers equivalent quality with up to a 50% reduction in file size and bandwidth requirements, making it the codec of choice for newer video end points and devices. Codensity T408/T432 Video Transcoders (henceforth referred to as T4XX) deliver scalable video transcoding between H.264 AVC and H.265 HEVC formats with up to 8K UHD video resolution.

3 Installation and Compatibility.

The installation of HW, FW, and SW must follow the details provided in this section. The T4XX Video Transcoders have been tested with the configuration detail in this section. All component parts must conform to the details given here. This includes the versions of open source software and the hardware detailed in this section.

3.1 Compatibility

The following hardware and software compatibility pertain to this quick start guide.

Software

This guide is for NETINT T4XX Video Transcoder software release 2.6.0

Hardware

Release 2.6.0 supports NETINT T4XX Video Transcoder hardware.

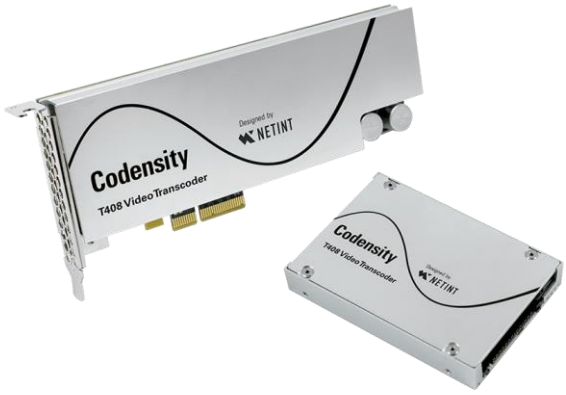
3.2 Hardware Installation

The T4XX Video Transcoders require minimal CPU resources when running video transcoding tasks. Typical CPU usage for 6x1080p30 streams on an i5 CPU is about 40%-50% per CPU thread. The minimum hardware requirement is:

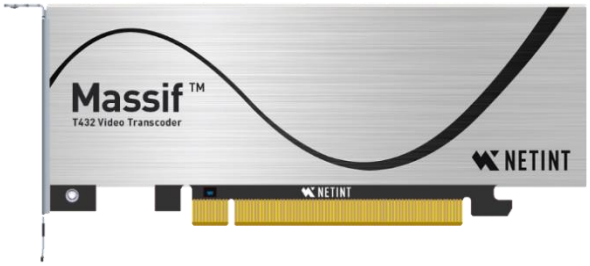
- Intel i5 CPU or equivalent
- 4GB DDR3 or DDR4
- T408 - Available U.2 interface or PCIe slot for add in card (AIC) versions
- T432 - Available PCIe 3.0 x16 PCIe slot with BIOS support for x4x4x4x4 bifurcation, and 300 LFM air flow
- T432 - GPU optimized servers are recommended (for better air flow cooling), e.g. Dell PowerEdge C4140

Some video transcoding applications may require deinterlacing, scaling or other processing done by FFmpeg which requires extra CPU resources. Also audio processing may require extra CPU resources. In these cases, or for a multiple T4XX transcoding system, the T4XX Video Transcoders can be installed in a host server with the following recommend requirements:

- 2x Intel Xeon Silver 4208 or equivalent
- 64GB DDR4



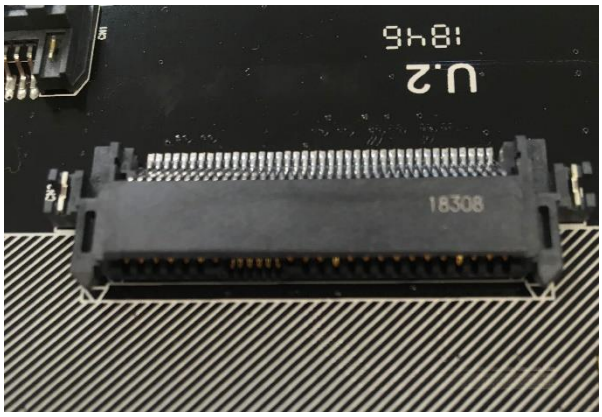
Codensity T408



T432 Video Transcoder

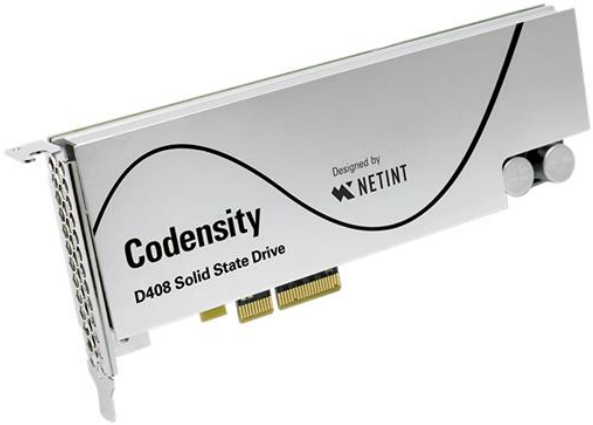
The T408 leverages NVMe server technology and is designed to directly plug into host servers equipped with NVMe U.2 bays. NVMe (non-volatile memory express) is a host controller interface and storage protocol for high speed data transfer over a server's high-speed Peripheral Component Interconnect Express (PCIe) bus. U.2 is a computer interface standard that encompasses the physical connector, electrical characteristics, and communication protocols. It uses up to four PCI Express lanes.

T408 modules are available in both U.2 and add-in-card (AIC) formats. T432 modules are only available in an add-in-card (AIC) format.



Host server U.2 interface

Host servers not equipped with U.2 interfaces can utilize the AIC (add in card) variant of the T408 or the T432.



T408 Add in card

3.3 Operating Systems and Software

The following software, including operating systems, is required for the operation of the T4XX Video Transcoders.

3.3.1 Operating Systems

A host server with one of the following operating systems installed is recommended:

- OS: Ubuntu 16.04.3 LTS; kernel: 4.10.0-28-generic

- OS: Ubuntu 16.04.3 LTS; kernel: 4.15.0-64-generic
- OS: Ubuntu 18.04.2 LTS; kernel: 4.15.0-45-generic
- OS: CentOS 7.2.1511; kernel: 3.10.0-327.el7.x86_64
- OS: CentOS 7.5.1804; kernel: 3.10.0-862.11.6.el7.x86_64
- OS: CentOS 7.6.1810; kernel: 3.10.0-957.el7.x86_64

If host server is utilizing Ubuntu operating system, run the following command to install prerequisite packages:

```
$ sudo apt-get install -y yasm pkg-config git gcc make
```

If host server is utilizing CentOS operating system, use the following steps to configure environment variables and install prerequisite packages:

1. Install prerequisite packages by running the commands:

   ```
   $ sudo yum --enablerepo=extras install -y epel-release
   ```

   ```
   $ sudo yum install -y make gcc redhat-lsb-core yasm git
   pkgconfig wget pciutils
   ```

2. Add the following lines in the file, /etc/bashrc:

   ```
   export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig/
   ```

   ```
   export LD_LIBRARY_PATH=/usr/local/lib/
   ```

3. Then run the command:

   ```
   $ source /etc/bashrc
   ```

4. Add the following line in the file, /etc/ld.so.conf:

   ```
   /usr/local/lib
   ```

5. Then run the command:

   ```
   $ sudo ldconfig
   ```

For any Linux operation system, check that the /etc/sudoers file is configured to run the programs to be installed:

1. In the file /etc/sudoers find the line:

   ```
   Defaults    secure_path =
   ```

2. Note its entries are separated by ':'. If /usr/local/sbin and /usr/local/bin are not in the secure_path add it by appending the following to the secure_path line:

   ```
   :usr/local/sbin:/usr/local/bin
   ```

3. Add the following line in the file, /etc/sudoers:

```
Defaults    env_keep += "PKG_CONFIG_PATH"
```

3.3.2 Verify T408/T432 Hardware Installation

lspci is a utility for displaying information about PCI buses in the system and devices connected to them.

To verify the installation, run the command:

```
$ lspci -d 1d82: | wc -l
```

The number returned is the number of T4XX devices installed and detected on the system.

3.3.3 NVME CLI Download and Install

The following material defines the NVMe CLI application download and install.

1. Download the NVMe CLI application from Git repository and untar it:

```
$ wget https://github.com/linux-nvme/nvme-
cli/archive/v1.6.tar.gz

$ tar -xzf v1.6

$ cd nvme-cli-1.6/
```

2. Install the NVMe CLI with the following command:

```
$ sudo make && sudo make install
```

3.3.4 Codensity T408/T432 Firmware Update

The following material explains the Codensity T4XX firmware update for both Ubuntu and CentOS.

Before upgrading firmware the following requirements must be met:

1. Codensity T4XX devices are detected by command:

```
$ sudo nvme list
```

2. Codensity T4XX firmware update package (i.e. T4XX_V2.0.0.tar.gz)

Once requirements are verified, use the following steps to upgrade firmware:

1. Use the GNU Tar to extract update script and binaries from the package tarball (i.e. T4XX_V2.0.0.tar.gz):

```
$ tar -xvf <path_to_update_package>
```

2. From a command line in the extracted folder, execute the BASH upgrade script as following:

```
$./t4xx_auto_upgrade.sh
```

3. The upgrade script scans the host system to look for Codensity T4XX devices, and proceeds to prompt user for upgrade confirmation:

```
$ To proceed, please press [Y/y] to begin the upgrade.
```

NOTE: User must stop all transcoding processes before performing upgrade, otherwise it may lead to device malfunction.

4. The upgrade script will perform an integrity check on the firmware binary file prior to performing the actual upgrade

```
$ {DEVICE} -- Checksum has been found!
$ {DEVICE} -- Firmware image integrity check succeeded!
```

NOTE: User must NOT interrupt the upgrade process when it is ongoing, doing so may lead to device malfunction. In the event the cold upgrade fails, a system reboot may resolve this issue and the process should be repeated.

5. The following console message indicates the completion of the upgrade process, user may proceed to reboot the host machine:

```
$ Firmware Update Completed! You may now reboot the system to
activate the firmware.
```

NOTE: Additional information logs on the upgrade process can be found in upgrade_log.txt generated by the upgrade script.

The t408_auto_upgrade.sh script defaults to cold upgrades but a warm upgrade can be performed using the following command.

From a command line, execute the BASH upgrade script with an additional argument as following:

```
$ ./t4xx_auto_upgrade.sh -w
```

NOTE: The behavior of the upgrade process is identical to that of cold upgrade, except a system reboot is no longer required. In the event the warm upgrade fails, a system reboot will resolve this issue and the process should be repeated or cold upgrade used.

3.3.5 FFmpeg Download and Installation

Interfacing with a T4XX is most easily done with *FFmpeg*. NETINT supports the following FFmpeg versions: 2.7.1, 3.4.2, 4.1.3, 4.2.1, 4.3, 4.3.1, 4.3.2 and 4.4. The FFmpeg stock versions are stored in FFmpeg repo, e.g.:

- 4.2.1 https://github.com/FFmpeg/FFmpeg/tree/n4.2.1
- 2.7.1 https://github.com/FFmpeg/FFmpeg/tree/n2.7.1

To clone the GitHub repository from the command line, run the following command based on the `<version>` (2.7.1, 3.4.2, 4.1.3, 4.2.1, 4.3, 4.3.1, 4.3.2 and 4.4) you want:

```
$ git clone -b n<version> --depth=1
https://github.com/FFmpeg/FFmpeg.git FFmpeg
```

ex.

```
$ git clone -b n3.4.2 --depth=1
https://github.com/FFmpeg/FFmpeg.git FFmpeg
```

NOTE: The GitHub repositories contain the current versions. If you use the Linux repositories from your operating system you will get the latest version of FFmpeg which might not be compatible with the NETINT configuration.

3.3.6 Apply FFmpeg Patch

The following instructions need to be done in sequence to prepare FFmpeg for use with the Codensity T4XX Video Transcoder.

1. Untar the Codensity T4XX Software Release package:

   ```
   $ tar -zxf Codensity_T4XX_Software_Release_V*.tar.gz
   ```

2. Copy the *libxcoder/* folder from the release package to the parent folder of FFmpeg (i.e. same level as FFmpeg):

   ```
   $ cp -r release/libxcoder ./
   ```

3. Copy the FFmpeg patch file from the release package to the *FFmpeg/* folder depending on what official `<version>` (2.7.1, 3.4.2, 4.1.3, 4.2.1, 4.3, 4.3.1, 4.3.2 and 4.4) of FFmpeg was downloaded earlier:

   ```
   $ cp release/FFmpeg-n<version>_t4xx_patch FFmpeg/
   ```

   ex.

   ```
   $ cp release/FFmpeg-n3.4.2_t4xx_patch FFmpeg/
   ```

4. Change directories to the *FFmpeg/* folder:

```
$ cd FFmpeg/
```

5. Apply the patch depending on the `<version>` (2.7.1, 3.4.2, 4.1.3, 4.2.1, 4.3, 4.3.1, 4.3.2 and 4.4) of FFmpeg previously selected:

```
$ patch -t -p 1 < FFmpeg-n<version>_t4xx_patch
```

ex.

```
$ patch -t -p 1 < FFmpeg-n3.4.2_t4xx_patch
```

The following instructions need to be done in sequence to prepare FFmpeg for use with the Codensity T4XX Video Transcoder. These instructions are applied to the software package delivered as part of the NETINT purchase.

NOTE: Run the patch according to your FFmpeg version.

3.3.7 Build FFmpeg with NETINT Codec Library

The following instructions need to be done in sequence to build FFmpeg with the NETINT Codec Library.

1. From the *FFmpeg/* folder, go to the *libxcoder/* directory with the following command:

```
$ cd ../libxcoder
```

2. Build and install libxcoder with one of the below commands:

- For Ubuntu or CentOS (newer than 6.5):

```
$ bash build.sh
```

- For CentOS 6.5 (or kernels with nvme driver older than v0.10):

```
$ bash build.sh -o
```

3. Load the libxcoder shared library into ldconfig with the following command:

```
$ sudo ldconfig
```

4. Go to the *FFmpeg/* directory with the following command:

```
$ cd ../FFmpeg
```

5. Run the `build_ffmpeg.sh` script with the following commands:

```
$ sudo make clean
```

```
$ bash build_ffmpeg.sh
```

6. Install FFmpeg with the following command:

```
$ sudo make install
```

3.3.8 Setup Verification

After completing the installation steps for both the hardware and software, you can follow the steps below to activate and verify your setup.

1. Check the NVMe device presence with the following command:

   ```
   $ sudo nvme list
   ```

   This will output a list of NVMe drives on the system. Look for drives with Model number T408/T432. See the example below:

```
[nvme@nvme-cli145 ~]$ sudo nvme list
Node             SN                   Model                                    Namespace Usage                      Format           FW Rev
---------------- -------------------- ---------------------------------------- --------- -------------------------- ---------------- --------
/dev/nvme0n1     TU05-06-02-C01-0085  T408-U2                                  1         393.62  MB / 393.62  MB      4 KiB +  0 B   224X1B03
/dev/nvme1n1     TA16-09-03-C21-0101A T432-AIC                                 1           4.10  TB /   4.10  TB    512   B +  0 B   220X1B03
/dev/nvme2n1     TA16-09-03-C21-0101B T432-AIC                                 1           4.10  TB /   4.10  TB    512   B +  0 B   220X1B03
/dev/nvme3n1     TA16-09-03-C21-0101C T432-AIC                                 1           4.10  TB /   4.10  TB    512   B +  0 B   220X1B03
/dev/nvme4n1     TA16-09-03-C21-0101D T432-AIC                                 1           4.10  TB /   4.10  TB    512   B +  0 B   220X1B03
/dev/nvme5n1     TA16-09-03-C21-0091A T432-AIC                                 1           4.10  TB /   4.10  TB    512   B +  0 B   220X1B03
/dev/nvme6n1     TA16-09-03-C21-0091B T432-AIC                                 1           4.10  TB /   4.10  TB    512   B +  0 B   220X1B03
/dev/nvme7n1     TA16-09-03-C21-0091C T432-AIC                                 1           4.10  TB /   4.10  TB    512   B +  0 B   220X1B03
/dev/nvme8n1     TA16-09-03-C21-0091D T432-AIC                                 1           4.10  TB /   4.10  TB    512   B +  0 B   220X1B03
```

   Note, each T408 card will show up as one device/node but each T432 card will be 4 devices/nodes.

2. To run FFmpeg without root privilege, the user needs to be added to the user ‘disk’ group:

```
$ sudo usermod -a -G disk <user>
```

   If you want to run FFmpeg as a root, you can skip this step and use *sudo* init rsrc to initialize the T4XX device.

3. Reboot the server:

   ```
   $ sudo reboot now
   ```

4. Initialize T4XX devices with the following command:

```
$ init_rsrc
```

If the host is on an older FW and FFmpeg version than 2.4.0, the host needs to be rebooted.

4 Monitoring Load

To monitor load of the processing load on T4XX devices, use the NETINT resource monitor utility:

```
$ sudo ni_rsrc_mon
NI resource init'd already ..
************************************************
5 devices retrieved from current pool at start up
Thu Sep 30 13:26:37 2021 up 00:00:00 v25NR1E09
Num decoders: 5
BEST INDEX LOAD MODEL_LOAD MEM  INST DEVICE         NAMESPACE
L    0     55   50         23   4    /dev/nvme0     /dev/nvme0n1
     1     56   50         23   4    /dev/nvme1     /dev/nvme1n1
     2     55   50         23   4    /dev/nvme2     /dev/nvme2n1
     3     55   50         23   4    /dev/nvme4     /dev/nvme4n1
     4     55   50         23   4    /dev/nvme5     /dev/nvme5n1
Num encoders: 5
BEST INDEX LOAD MODEL_LOAD MEM  INST DEVICE         NAMESPACE
L    0     100  72         23   8    /dev/nvme0     /dev/nvme0n1
     1     99   72         23   8    /dev/nvme1     /dev/nvme1n1
     2     100  72         23   8    /dev/nvme2     /dev/nvme2n1
     3     100  72         23   8    /dev/nvme4     /dev/nvme4n1
     4     100  72         23   8    /dev/nvme5     /dev/nvme5n1
************************************************
```

A help text with description of how to use the program can be accessed with the command:

```
$ ni_rsrc_mon -h
-------- ni_rsrc_mon v25NR1E09 --------
The ni_rsrc_mon program provides a real-time view of NETINT T408
resources running on the system.
return 0 on success
return 1 on failure
Usage: sudo ni_rsrc_mon [OPTIONS]
-n      Specify reporting interval in one second interval.
        If 0 or no selection, report only once.
        Default: 0

-r      Init transcoder card resource regardless firmware release
        version to libxcoder version compatibility.
        Default: only init cards with compatible firmware version.

-t      Set timeout time in seconds for device polling. Program will
        exit with failure if timeout is reached without finding at
        least one device. If 0 or no selection, poll indefinitely
        until a T408 device is found.
        Default: 0

-l      Set loglevel of libxcoder API.
        [none, fatal, error, info, debug, trace]
        Default: info

-i      Do not refresh the devices list.

-h      Open this help message.

Reporting columns
BEST          flag showing card of lowest realtime load and to be
              selected for next auto allocated job
INDEX         index number used by resource manager to identify the
              resource
LOAD          realtime load
MODEL_LOAD    estimated load based on framerate and resolution
INST          number of job instances
DEVICE        path to NVMe device file handle
NAMESPACE     path to NVMe namespace file handle
```

5 Optimisations related to NUMA enabled hardware

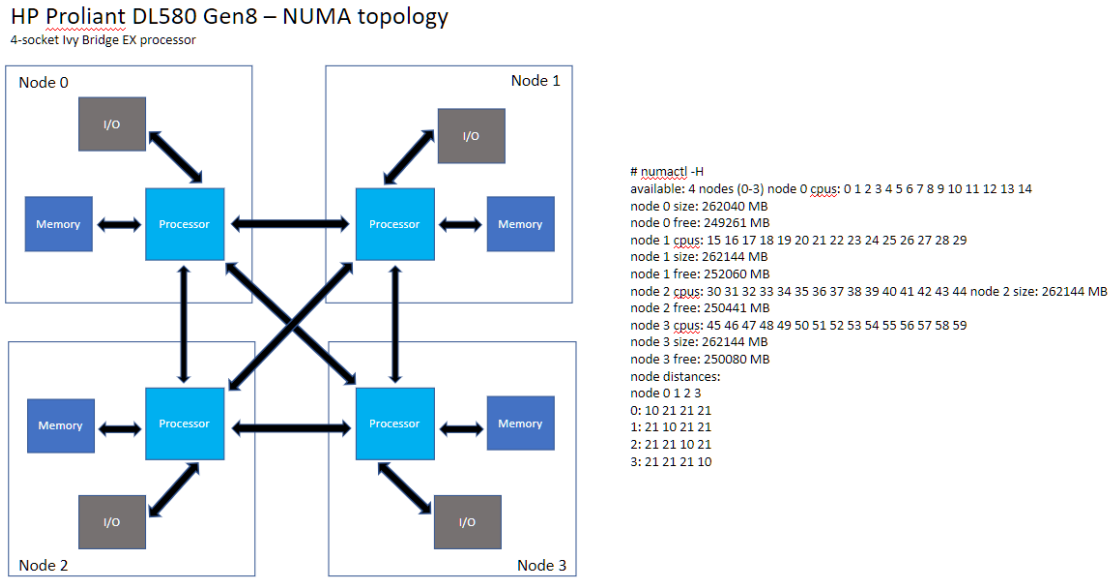
5.1 Introduction

Note: Section 5 does not apply to Intel based CPUs and only applies to host servers utilizing AMD EPYC Gen 1 CPUs due to specific details in their CPU architecture. Users with AMD EPYC Gen 2 CPU hosts should contact NETINT support for recommendations on how to maximize performance.

NUMA (Non-Uniform Memory Access) is a CPU/memory architecture where the memory/IO access time depends on the memory/IO location relative to a particular processor. Under NUMA, a processor can access its own local memory faster than non-local memory (memory local to another processor or memory shared between processors). The benefits of NUMA are limited to specific types workloads, notably on servers where the data is often associated strongly with certain tasks.

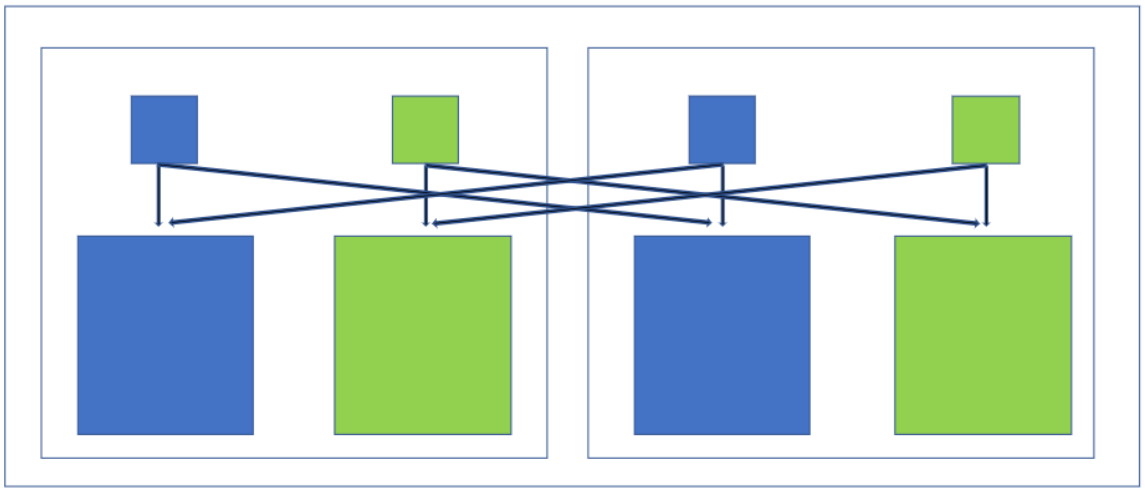
5.2 NUMA Example

In a system utilizing the NUMA architecture, the CPU and memory are grouped as a “Node” which consists of a physical CPU and local memory directly attached to this CPU or multiple CPUs (with off-chip memory controller).

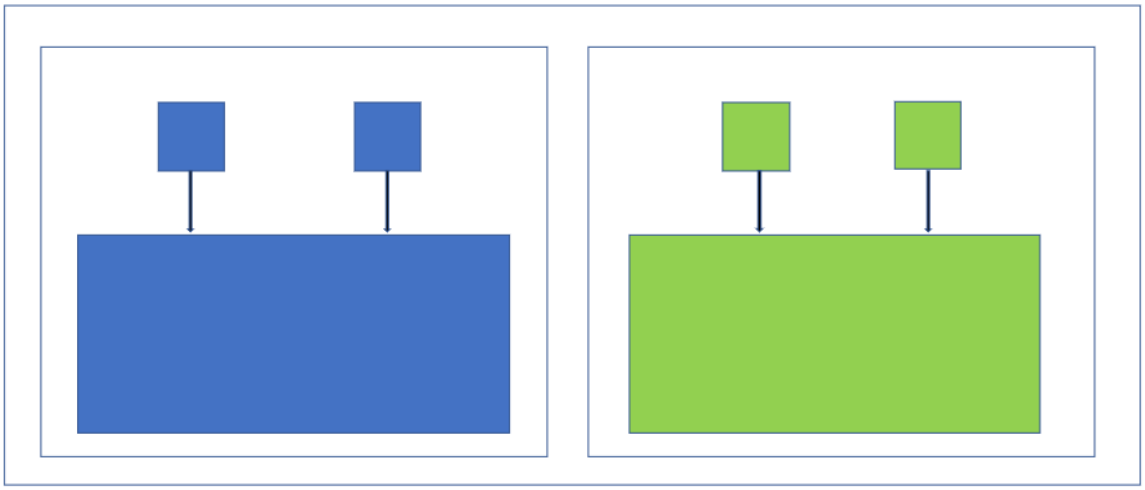


Example of node grouping (from: www.linux-kvm.org/images/7/75/01x07b-NumaAutobalancing.pdf)

In the example of an HP Proliant DL580 server, there are 4 nodes in this system and each node has only 1 CPU with multiple cores, attached memory and IO. The NETINT T4XX is a PCI-E device and will be associated with a node. Maximum performance and reduced IO latency can be achieved If the transcoding software process is also forced to use both the processing core(s) of the CPU and the memory that belong to this NUMA node



Task Grouping and Placement Example – Tasks outside of NUMA Nodes



Task Grouping and Placement Example – Tasks scheduled on corresponding NUMA nodes

5.3 NUMA topology discovery tools

To efficiently work with NUMA enabled hardware, NETINT provides a set of tools to identify and present the systems NUMA node layout and their relationship to T4XX devices present in the system. The involves running the t408_numa_associations.sh script which gathers various information from the system and creates both a human readable report as well as a CSV formatted output which can be consumed by other scripts to efficiently schedule decoding/encoding/transcoding sessions on T4XX devices that belong to specific NUMA nodes in the system.

The user can run the t408_numa_associations.sh file to get all the information necessary to be able to schedule transcoding jobs in the most efficient manner according to the system`s NUMA layout

```
nvme@cli70:~/FFmpegXcoder/tools/amd_multi_inst_demo$ ./t408_numa_associations.sh
Total T-408 devices in the system: 10
+-------------+-------------+-------------+
|  DEC index  |  ENC index  |  NUMA node  |
+-------------+-------------+-------------+
|           0 |           0 |           0 |
|           1 |           1 |           0 |
|           2 |           2 |           1 |
|           3 |           3 |           1 |
|           4 |           4 |           2 |
|           5 |           5 |           2 |
|           6 |           6 |           6 |
|           7 |           7 |           6 |
|           8 |           8 |           6 |
|           9 |           9 |           6 |
+-------------+-------------+-------------+
Example FFMpeg command for H264->H265 transcoding on dec/enc 9 with numa node pinning to numa node 6:
    sudo numactl --cpubind=6 --membind=6 ffmpeg -c:v h264_ni_dec -dec 9 -i <INPUT H264 FILE PATH> -c:v h2
65_ni_enc -enc 9 -f <OUTPUT H265 FILE PATH>
```

Default output of t408_numa_associations.sh script

The same report can be generated to provide the CSV formatted output that can be used within the user's script to programmatically perform the optimized process scheduling

```
nvme@cli70:~/FFmpegXcoder/tools/amd_multi_inst_demo$ ./t408_numa_associations.sh -c
0,0,0
1,1,0
2,2,1
3,3,1
4,4,2
5,5,2
6,6,6
7,7,6
8,8,6
9,9,6
```

CSV style out put of t408_numa_associations.sh script

Passing the “-h” parameter to the script will print the usage and all the options the tool offers

```
nvme@cli70:~/FFmpegXcoder/tools/amd_multi_inst_demo$ ./t408_numa_associations.sh -h
Usage: ./t408_numa_associations.sh [OPTIONS]
Report the dec/enc index to numa node mapping.
Note, to use NUMA node pinning the dependencies hwloc, and numactl must be installed.

Options:
-h, --help
    Print this help
-v, --verbose
    Verbose output
-d, --install_deps
    Install dependencies needed for the script, then exit
-m, -c, --csv
    Output is produced in the form of a csv array with each line containing
    information regarding one card. The format of each line return is:
    [decoder#][encoder#][numa node#]
    Ex. for a system with 2 T408 cards on 2 numa nodes (50 and 52):
        0,0,50
        1,1,52
```

All the available command line options of t408_numa_associations.sh script

Once the NUMA nose and enc/dec pairing is generated, user can use “numactl” to select NUMA nose for each FFmpeg instance.

For example:

sudo numactl --cpubind=6 --membind=6 ffmpeg -hide_banner -nostdin -c:v h264_ni_dec -dec 21-re -f concat -i ~/golden_data/ref/avc/demo_1920x1080p30_5757_br3800_b0.h264.list.orig -c:v h265_ni_enc -enc 21 -xcoder-params "gopPresetIdx=5:intraPeriod=92:RcEnable=1:RcInitDelay=3000:bitrate=6400000:frameRate=30" -f null /dev/null
This command will make the transcoding instance run in node 6 which will optimize the performance.

A well configurated system can have better performance and lower CPU usage.

6 FW Authentication and Bridging

6.1 Introduction to FW Authentication

Netint cryptographically signs firmware images to prevent the installation of corrupted or tampered firmware.

When Netint compiles T408/T432 firmware a digital signature is generated for the firmware image using the Netint private key. This signature is embedded in the firmware image file (nor_flash.bin, fw.img, etc.).

Within Netint's firmware is Netint's public key. When a firmware image is transferred from the host to the T408/T432 for upgrade consideration the firmware on the card will check the digital signature of the firmware image using the public key within the firmware on the card. If contents of the firmware image to be loaded is all good, it will be loaded by the T408/T432 as the next firmware image. A power-cycle or reset command for cold-upgrade or warm-upgrade respectively will cause the next (new) firmware image to be loaded.

By this way, the private key used to sign the new firmware image must be compatible with the public key stored in the old (previously loaded) firmware image. This protection prevents T408/T432 from being loaded with unofficial or corrupted firmware.

6.2 "Bridging"

Sometimes it is necessary to migrate T408/T432 cards from one public-key private-key pair to another public-key private-key pair. This process involves loading special firmware containing the new public key but signed with the old private key. This special firmware is calling bridging firmware typically suffixed with '_bridge'. (eg. nor_flash_v2.2.0-T408_bridge.bin)

A key consolidation event occurred during T408/T432 release v2.1.1 during which the public-key private-key pair of regular Netint release firmware changed. To facilitate this change, bridging firmware releases were created on v2.1.1 and v2.2.0.

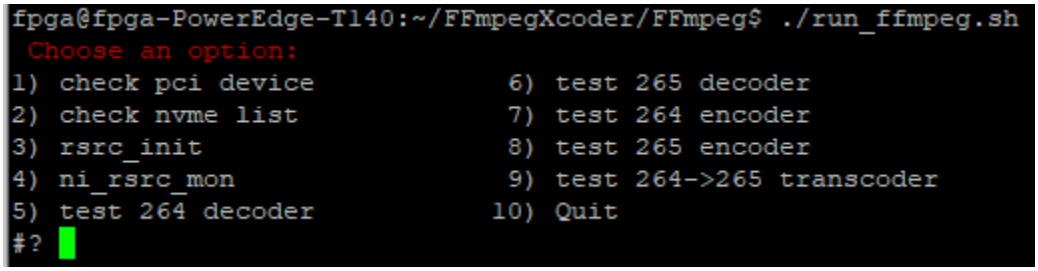
If upgrading a T408/T432 that currently has firmware older than v2.1.1 to firmware newer than v2.1.1 it is very likely necessary to require use of 'bridging' firmware. The process to upgrade from v2.1.0 to v2.3.0 would be:

1. Acquire the v2.2.0 bridging firmware (T4XX_V2.2.0-bridge.tar.gz)
2. Follow instructions in section ***3.3.4: Codensity T408/T432 Firmware Update*** to perform firmware upgrade
3. Acquire the regular v2.3.0 release firmware (T4XX_V2.3.0.tar.gz)
4. Follow instructions in section ***3.3.4: Codensity T408/T432 Firmware Update*** to perform firmware upgrade

7 Encoding, decoding and transcoding test

User can run the run_ffmpeg.sh file to test basic encoding, decoding and transcoding features.

```
$ ./run_ffmpeg.sh
```



Run_ffmepg.sh example

After running this bash file, user will see a menu to choose which test item to run from 1~9.

1. check pci device. This item will list all memory controllers on the PC.
2. check nvme list. This item will call “sudo nvme list”
3. rsrc_init, this item will call sudo ../libxcoder/build/rsrc_init to initialize the T4XX HW.
4. ni_rsrc_mon run resource monitor.
5. test 264 decoder, this item will test h264->yuv decoding function. After the decoding finished, it will display the output.yuv file size which should be 45619200.
6. test 265 decoder, this item will test h265->yuv decoding function. After the decoding finished, it will display the output.yuv file size which should be 45619200.
7. test 264 encoder, this item will test yuv->h264 encoding function. After the encoding finished, it will display the output.264 file size which should be 156106.
8. test 265 encoder, this item will test yuv->h265 encoding function. After the encoding finished, it will display the output.264 file size which should be 115018.
9. test 264->265 transcoder, this item will test H264 to H265 transcoding function. After the transcoding finished, it will display the output.264 file size which should be 8402285.
10. Quit: quit this menu.

FFmpeg output message of encoding, decoding or transcoding will also output to ffmpeg.log file in the same folder as a record.

Note: with different release, the encoded file size may have slight difference with the number recorded in the run_ffmpeg.sh. This is normal.

8 NETINT FFmpeg Patch Overview

This section gives an overview of the FFmpeg patch, its components, and how to compile it.

8.1 NETINT Codec Library

The *NETINT Codec Library* is a software module that provides a codec API for users to control and operate NETINT's ASIC codecs in a manner that abstracts from the lower level details. It is a standalone library that can be easily integrated into any application.

The source code is located in the directory *libxcoder/source/*.**.

Refer to the section “Build FFmpeg with NETINT Codec Library” for compilation instructions.

8.2 NETINT Libavcodec

The *NETINT Codec Library* is supported in the FFmpeg application through the standard FFmpeg libavcodec interface layer for the NETINT decoder and encoder. This FFmpeg NETINT libavcodec interface layer allows complete integration of the NETINT codec into the FFmpeg application. The NETINT Libavcodec source code is located in the *FFmpeg/libavcodec/* directory and contains the following files:

File	Description
nicodec.h	The main header file for NETINT decoder and encoder.
nicodec.c, nidec.c	The libavcodec source file for NETINT decoder.
nidec_h264.c, nidec_hevc.c	The libavcodec source file for NETINT decoder.
nienc_h265.c, nienc_h264.c, nienc.c	The libavcodec source file for NETINT encoder.

Refer to section “Build FFmpeg with NETINT Codec Library” for build details.

8.3 NETINT FFmpeg Changes

Netint has made fixes and updates to base FFmpeg code to add support for NETINT Libavcodec, backport support for features (eg. HLG), fix bugs, and improve performance. For a list of changes which NETINT applies to FFmpeg refer to the *NETINT_FFMPEG_CHANGE_LIST.txt* file in the software release package, *Codensity_T4XX_Software_Release_V*.tar.gz*.

9 Troubleshooting

When attempting decode/encode/transcode the following message appears:

```
Error shm_open SHM_CODERS ..: No such file or directory
```

Remember to initialize T4XX resources every time system is booted by running:

```
$ sudo init_rsrc
```

If it is necessary to forcefully re-initialize resources (ex. after libxcoder is updated without subsequent reboot):

```
$ sudo rm /dev/shm/NI_*
$ sudo init_rsrc
```