APPS003-NUMA IO Performance Optimization Application Note

V2.0

11/30/2021

Table of Contents

1 Table of Abbreviations

Abbreviation:	Full Form:
NUMA	Non-Uniform Memory Access
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 provides a solution to optimize the instance IO efficiency on a multi core computer when operation/integrating the NETINT T408 transcoder.

3.1 Intended Audience

This document is intended 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 1.1.0 and up.

Hardware Compatibility

NETINT Codensity T408 Video Transcoder hardware.

Operation system

Linux system

4 What is NUMA

4.1 Computer with NUMA

NUMA (Non Uniform Memory Access) is a CPU/memory architecture on some of the computers. In such an architecture, each CPU has local memory attached to it which is fast, and it can also access other CPU's memory which is called remote memory but will take more time.

In such system, the CPU and memory are grouped as "Node" which is A physical CPU and attached memory and could be multiple CPUs (with off-chip memory controller)

Nodes are Interconnected to each other. Bus connects the various nodes together which is generally faster than memory bandwidth of a single node and can get overwhelmed by traffic from many nodes.

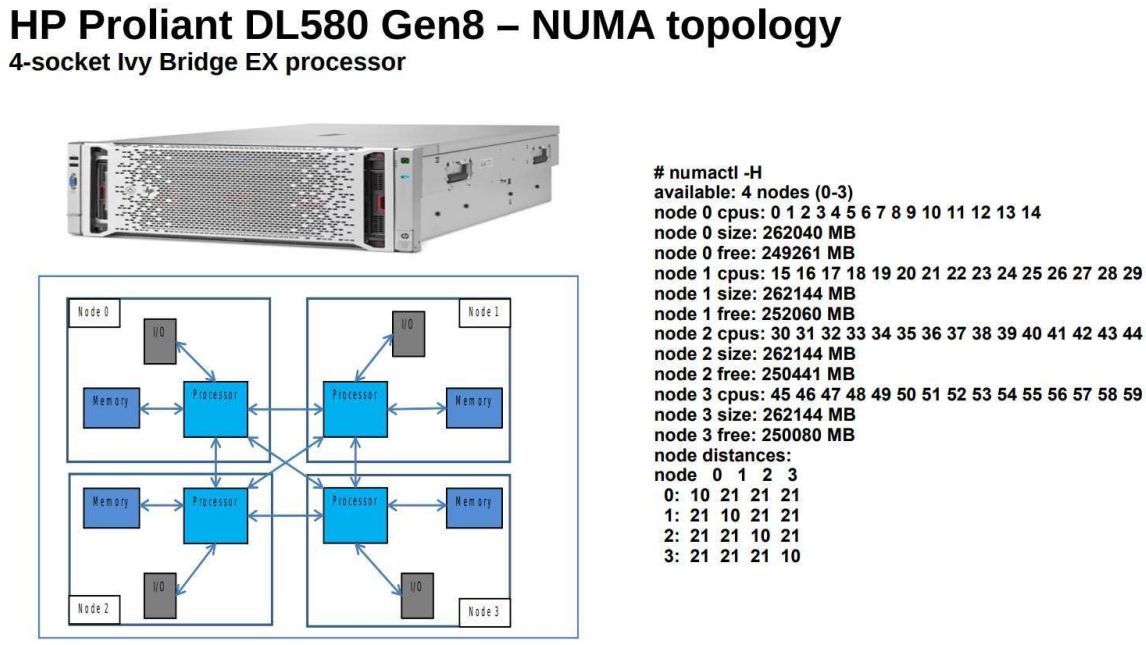

Figure 1 Example of Node (from: https://www.linux-kvm.org/images/7/75/01x07b-NumaAutobalancing.pdf)

In the example of HP Proliant DL580, there are 4 nodes in this system and each node has only 1 CPU, attached memory and IO.

NETINT t408 is a PCI-E device and will be associated with a node. If the CPU and memory are also allocated within the same node, then you will get optimized efficiency, i.e. less IO usage.

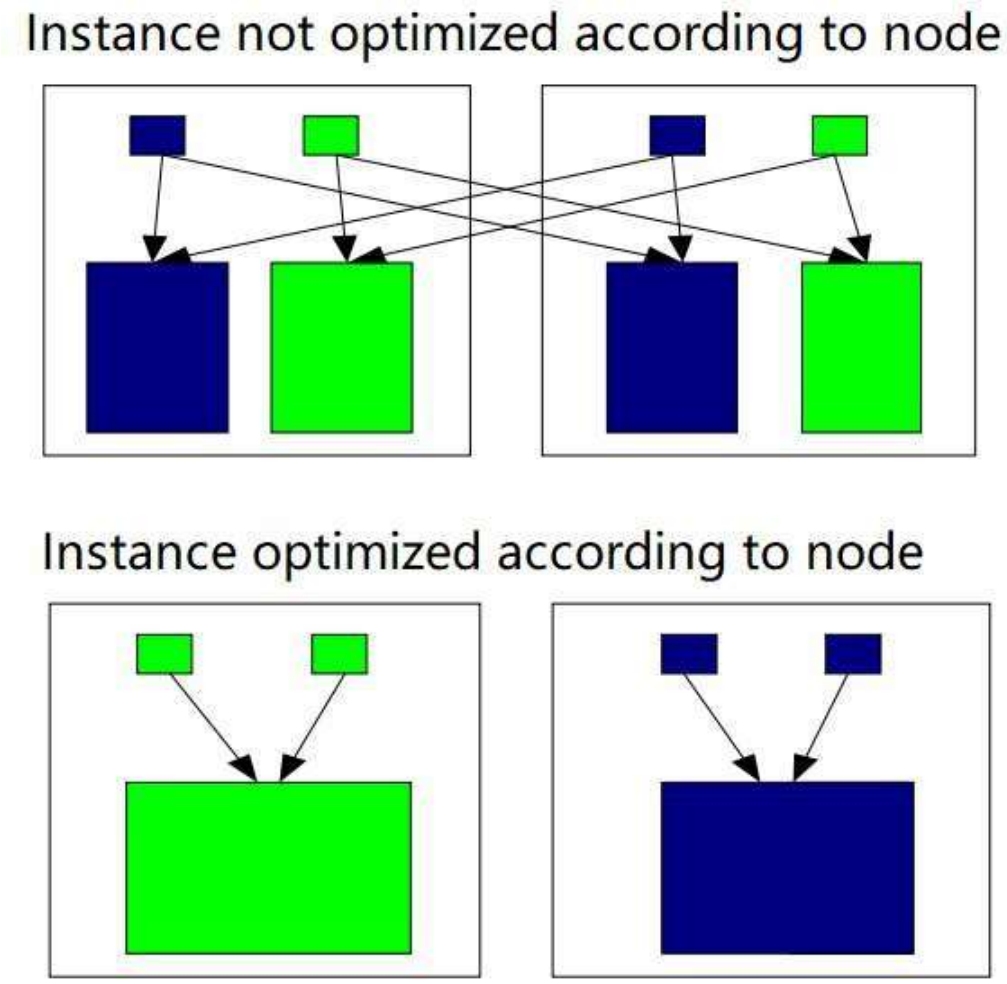

Figure 2 instance optimization according to node

4.2 Check the Node of your computer

Under linux, you can use “numactl -H” command to check the configuration of your system.

For example, on a single node computer:

```
available: 1 nodes (0)

node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
22 23 24 25 26 27 28 29 30
31

node 0 size: 31922 MB

node 0 free: 514 MB
```

```
node distances:
node 0
 0: 10
```

Another example on the Inspur server:

```
available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3 4 5 6 7 16 17 18 19 20 21 22 23
node 0 size: 15568 MB
node 0 free: 4366 MB
node 1 cpus: 8 9 10 11 12 13 14 15 24 25 26 27 28 29 30 31
node 1 size: 16124 MB
node 1 free:
5456 MB node
distances:
node 0 1
 0:  10  21
 1:  21  10
```

This system has two nodes.

5 Procedure to optimize NUMA

5.1 Install software tool

Install the tools needed:

```
sudo apt-get install hwloc numactl
```

Now we need to figure out which PCI card has which PCI ID and belongs to which NUMA node.

5.2 Find out the PCI BUS ID of your T408 card

if we run “lspci -v” we get the following output (here just showing output for one card, there will be a lot of information we don’t need but we need to filter by our PCI BUS ID , to get only what we need)

```
68:00.0 Non-Volatile memory controller: Device 1d82:0202 (prog-if 02
[NVM Express])
Subsystem: Device
1d82:0000 Physical
Slot: 17-1
Flags: bus master, fast devsel, latency 0,
IRQ 82, NUMA node 6 Memory at dbf00000 (64-
bit, prefetchable) [size=16K] Capabilities:
<access denied>
Kernel driver in
use: nvme Kernel
modules: nvme
```

The above output shows that this is our T-408 card (Device 1d82:0202 is out PCI ID), and is belongs to NUMA Node 6, also it has a PCI BUS ID of 68:00:0.

What we need to save is only Numa node number and PCI BUS ID.

Now we can corelate this card to the /dev/nvme device number.

5.3 Find the NVME device of this card

We do this by running the following command for this particular PCI BUS ID:

```
nvme@nvme-cli71:~$ grep "68:00.0"
/sys/class/nvme/*/device/uevent
```

/sys/class/nvme/nvme22/device/uevent:PCI_SLOT_NAME=0000:68:00.0

As you can see, this device in the system is /dev/nvme22

Now we need to figure out which dec/encoder number belongs to this device.

5.4 Find dec or enc ID of this card

For this we need to query our Resource manager:

```
nvme@nvme-cli71:~$ sudo
ni_rsrc_list Num decoders:
24
Decoder #2
DeviceID:
/dev/nvme3 H/W
ID: 0
MaxNumInstances: 32
ActiveNumInstances: 0
Max1080pFps: 240
CurrentLoad:
0
H.264Capabil
ities:
Supported: yes
MaxResolution:
8192x4096
MinResolution:
16x16
Profiles: Baseline Profile, Constrained Baseline Profile, Main
Profile, High Profile, High10 Profile level: Level 5.2
additional info:
```

H.265Capabilities:

Supported: yes

MaxResolution:

8192x4096

MinResolution: 8x8

Profiles: Main Profile,

Main10 Profile level:

Level 5.1 Main-Tier

additional info:

num. s/w instances: 0

……

……

……

Decoder #21

DeviceID:

/dev/nvme22 H/W

ID: 0

MaxNumInstances: 32

ActiveNumInstances: 0

Max1080pFps: 240

CurrentLoad:

0

H.264Capabil

ities:

Supported: yes

MaxResolution:

8192x4096

MinResolution:

16x16

Profiles: Baseline Profile, Constrained Baseline Profile, Main Profile, High Profile, High10 Profile level: Level 5.2

additional info:

H.265Capabilities:

Supported: yes

MaxResolution:

8192x4096

MinResolution: 8x8

Profiles: Main Profile,

Main10 Profile level:

Level 5.1 Main-Tier

additional info:

num. s/w instances: 0

Decoder #22

DeviceID:

/dev/nvme23 H/W

ID: 0

MaxNumInstances: 32

ActiveNumInstances: 0

Max1080pFps: 240

CurrentLoad:

0

H.264Capabil

ities:

Supported: yes

MaxResolution:

8192x4096

MinResolution:

16x16

Profiles: Baseline Profile, Constrained Baseline Profile, Main Profile, High Profile, High10 Profile level: Level 5.2

additional info:

H.265Capabilities:

Supported: yes

MaxResolution:

8192x4096

MinResolution: 8x8

Profiles: Main Profile,

Main10 Profile level:

Level 5.1 Main-Tier

additional info:

num. s/w instances: 0

……

……..

…….

Encoder #21

DeviceID:

/dev/nvme22 H/W

ID: 1

MaxNumInstances: 32

ActiveNumInstances: 0

Max1080pFps: 240

```
CurrentLoad:
0
H.264Capabil
ities:
Supported: yes
MaxResolution:
8192x4096
MinResolution:
16x16
Profiles: Baseline Profile, Constrained Baseline Profile, Main
Profile, High Profile, High10 Profile level: Level 5.2
additional info:
H.265Capabilities:
Supported: yes
MaxResolution:
8192x4096
MinResolution:
256x128
Profiles: Main Profile, Main10 Profile, Main
Still Picture Profile level: Level 5.1 Main-
Tier
additional info:
num. s/w instances: 0

…..

…..

…..
```

So now we need to save that card with PCI BUS ID 63:00:0 belongs to NUMA Node 6 and is used to access Encoder/Decoder 21

5.5 Find CPU in the same node of this card

[OPTIONAL]

Now we need to figure out which CPU cores belong to Numa Node 6, for this we run the following command:

```
nvme@nvme-cli71:~$ numactl -
-hardware available: 8 nodes
(0-7)
node 0 cpus: 0 1 2 3 32 33 34 35
node 0 size: 15819 MB
node 0 free: 15261 MB
node 1 cpus: 4 5 6 7 36 37 38 39
node 1 size: 16124 MB
node 1 free: 15715 MB
node 2 cpus: 8 9 10 11 40 41 42 43
node 2 size: 16124 MB
node 2 free: 15674 MB
node 3 cpus: 12 13 14 15 44 45 46 47
node 3 size: 16124 MB
node 3 free: 15643 MB
node 4 cpus: 16 17 18 19 48 49 50 51
node 4 size: 16124 MB
node 4 free: 15949 MB
node 5 cpus: 20 21 22 23 52 53 54 55
node 5 size: 16124 MB
node 5 free: 15909 MB
node 6 cpus: 24 25 26 27 56 57 58 59
node 6 size: 16124 MB
```

```
node 6 free: 15916 MB

node 7 cpus: 28 29 30 31 60 61 62 63

node 7 size: 16101 MB

node 7 free:

15920 MB node

distances:

node 0 1 2 3 4 5 6 7

0:  10  16  16  16  32  32  32  32

1:  16  10  16  16  32  32  32  32

2:  16  16  10  16  32  32  32  32

3:  16  16  16  10  32  32  32  32

4:  32  32  32  32  10  16  16  16

5:  32  32  32  32  16  10  16  16

6:  32  32  32  32  16  16  10  16

7:  32  32  32  32  16  16  16  10

nvme@nvme-cli71:~$
```

5.6 Commands for encoding and decoding

```
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.o
rig -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.

6 Test result

This experiment is done with an AMD Gigabyte 24 solt server with 4 T408 cards, each card has 1080pX6 Limelight scenario test.

6.1 Before NUMA optimization

total streams: 24

total fps: 720

average fps: 30.00

min fps: 30 (output0-0.txt)

max fps: 30 (output0-0.txt)

Figure 3 CPU Utilization before NUMA optimization

```
0 B/s | Total DISK WRITE :       0.00 B/s
0 B/s | Actual DISK WRITE:       0.00 B/s
 READ   DISK WRITE   SWAPIN     IO>    COMMAND
0 B/s    0.00 B/s  0.00 % 37.54 % ffmpeg -hide_banner -nostdin -c:v h264_ni_~e=6400000:frameR
0 B/s    0.00 B/s  0.00 % 37.37 % ffmpeg -hide_banner -nostdin -c:v h264_ni_~e=6400000:frameR
0 B/s    0.00 B/s  0.00 % 35.91 % ffmpeg -hide_banner -nostdin -c:v h264_ni_~e=6400000:frameR
0 B/s    0.00 B/s  0.00 % 35.48 % ffmpeg -hide_banner -nostdin -c:v h264_ni_~e=6400000:frameR
0 B/s    0.00 B/s  0.00 % 35.31 % ffmpeg -hide_banner -nostdin -c:v h264_ni_~e=6400000:frameR
0 B/s    0.00 B/s  0.00 % 35.17 % ffmpeg -hide_banner -nostdin -c:v h264_ni_~e=6400000:frameR
0 B/s    0.00 B/s  0.00 % 35.16 % ffmpeg -hide_banner -nostdin -c:v h264_ni_~e=6400000:frameR
0 B/s    0.00 B/s  0.00 % 35.15 % ffmpeg -hide_banner -nostdin -c:v h264_ni_~e=6400000:frameR
0 B/s    0.00 B/s  0.00 % 34.86 % ffmpeg -hide_banner -nostdin -c:v h264_ni_~e=6400000:frameR
0 B/s    0.00 B/s  0.00 % 34.55 % ffmpeg -hide_banner -nostdin -c:v h264_ni_~e=6400000:frameR
0 B/s    0.00 B/s  0.00 % 34.37 % ffmpeg -hide_banner -nostdin -c:v h264_ni_~e=6400000:frameR
0 B/s    0.00 B/s  0.00 % 33.81 % ffmpeg -hide_banner -nostdin -c:v h264_ni_~e=6400000:frameR
0 B/s    0.00 B/s  0.00 % 33.51 % ffmpeg -hide_banner -nostdin -c:v h264_ni_~e=6400000:frameR
0 B/s    0.00 B/s  0.00 % 32.98 % ffmpeg -hide_banner -nostdin -c:v h264_ni_~e=6400000:frameR
0 B/s    0.00 B/s  0.00 % 32.52 % ffmpeg -hide_banner -nostdin -c:v h264_ni_~e=6400000:frameR
0 B/s    0.00 B/s  0.00 % 32.35 % ffmpeg -hide_banner -nostdin -c:v h264_ni_~e=6400000:frameR
0 B/s    0.00 B/s  0.00 % 32.15 % ffmpeg -hide_banner -nostdin -c:v h264_ni_~e=6400000:frameR
0 B/s    0.00 B/s  0.00 % 31.90 % ffmpeg -hide_banner -nostdin -c:v h264_ni_~e=6400000:frameR
0 B/s    0.00 B/s  0.00 % 31.54 % ffmpeg -hide_banner -nostdin -c:v h264_ni_~e=6400000:frameR
0 B/s    0.00 B/s  0.00 % 31.48 % ffmpeg -hide_banner -nostdin -c:v h264_ni_~e=6400000:frameR
0 B/s    0.00 B/s  0.00 % 31.35 % ffmpeg -hide_banner -nostdin -c:v h264_ni_~e=6400000:frameR
0 B/s    0.00 B/s  0.00 % 30.81 % ffmpeg -hide_banner -nostdin -c:v h264_ni_~e=6400000:frameR
0 B/s    0.00 B/s  0.00 % 30.29 % ffmpeg -hide_banner -nostdin -c:v h264_ni_~e=6400000:frameR
0 B/s    0.00 B/s  0.00 % 29.88 % ffmpeg -hide_banner -nostdin -c:v h264_ni_~e=6400000:frameR
0 B/s    0.00 B/s  0.00 %  0.00 % init
0 B/s    0.00 B/s  0.00 %  0.00 % [kthreadd]
0 B/s    0.00 B/s  0.00 %  0.00 % [kworker/0:0H]
0 B/s    0.00 B/s  0.00 %  0.00 % [kworker/u256:0]
0 B/s    0.00 B/s  0.00 %  0.00 % [mm_percpu_wq]
0 B/s    0.00 B/s  0.00 %  0.00 % [ksoftirqd/0]
0 B/s    0.00 B/s  0.00 %  0.00 % [rcu_sched]
0 B/s    0.00 B/s  0.00 %  0.00 % [rcu_bh]
0 B/s    0.00 B/s  0.00 %  0.00 % [migration/0]
0 B/s    0.00 B/s  0.00 %  0.00 % [watchdog/0]
0 B/s    0.00 B/s  0.00 %  0.00 % [cpuhp/0]
0 B/s    0.00 B/s  0.00 %  0.00 % [cpuhp/1]
0 B/s    0.00 B/s  0.00 %  0.00 % [watchdog/1]
0 B/s    0.00 B/s  0.00 %  0.00 % [migration/1]
0 B/s    0.00 B/s  0.00 %  0.00 % [ksoftirqd/1]
0 B/s    0.00 B/s  0.00 %  0.00 % [kworker/1:0]
0 B/s    0.00 B/s  0.00 %  0.00 % [kworker/1:0H]
```

Figure 4 IO utilization before NUMA optimization

6.2 After NUMA optimization

total streams: 24

total fps: 719

average fps: 29.95

min fps: 29 (output0-0.txt)

max fps: 30 (output0-2.txt)

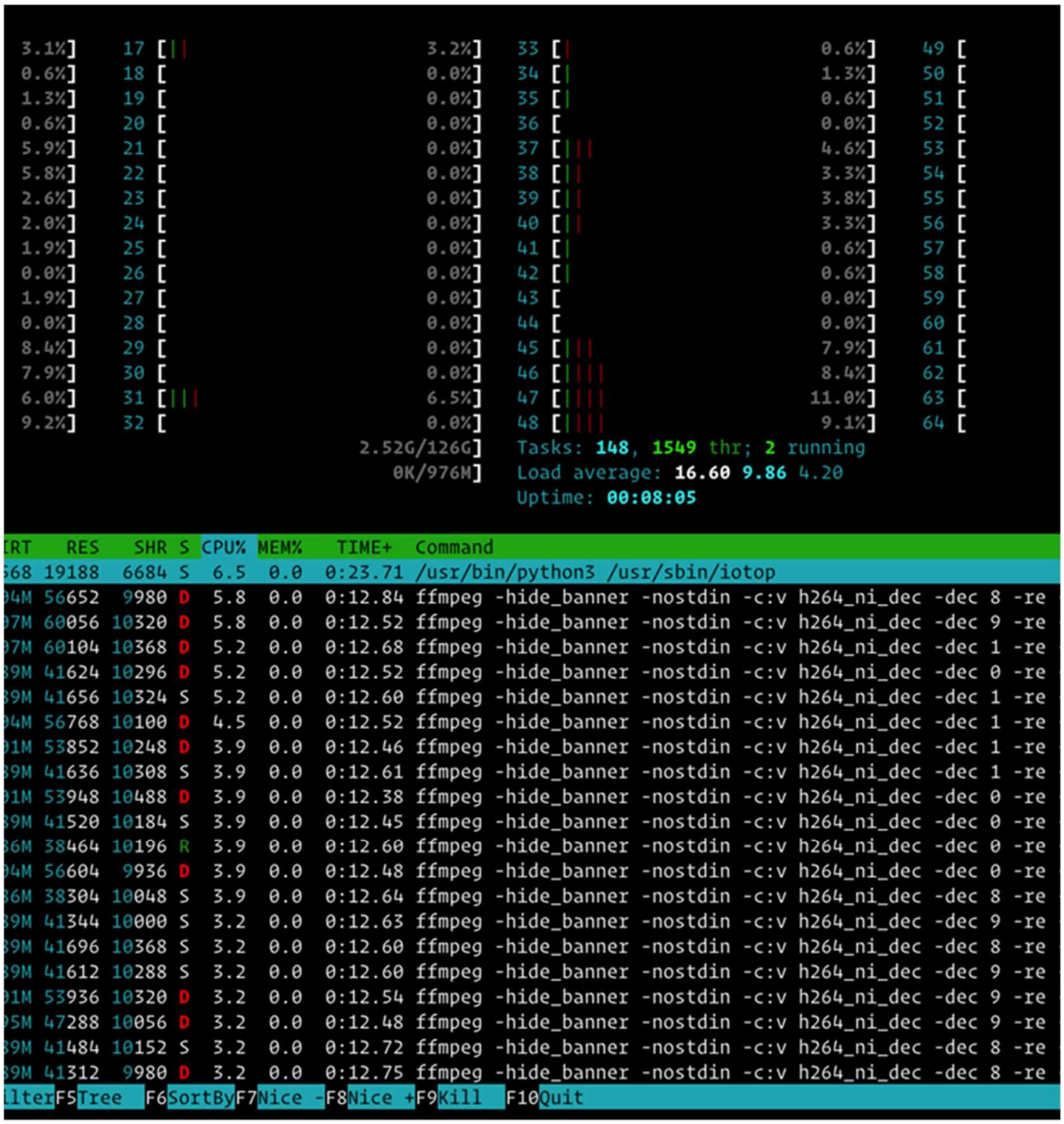

Figure 5 CPU usage after NUMA optimization

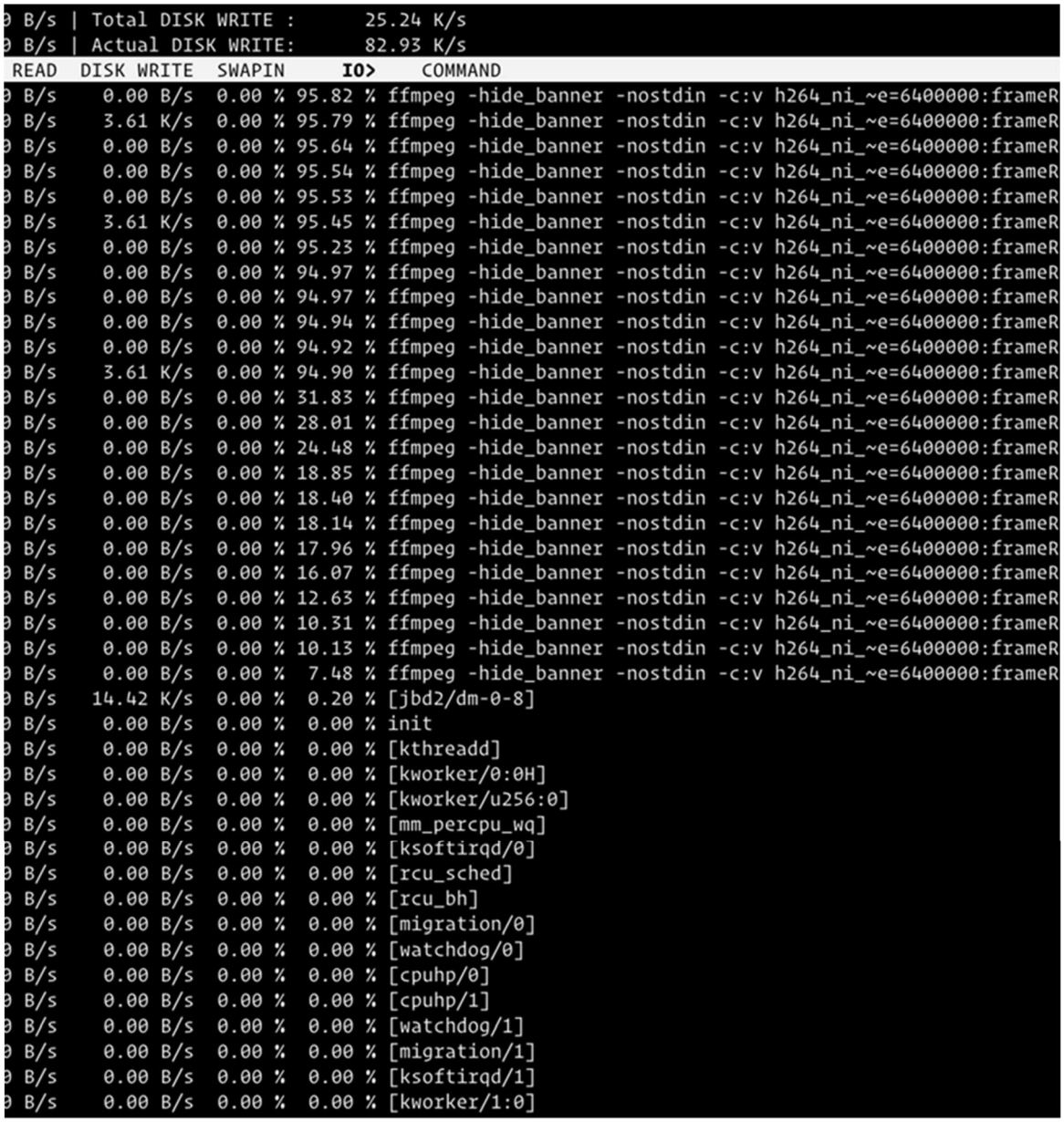

Figure 6 IO usage after NUMA optimization

After NUMA optimization, CPU Usage per ffmpeg instance is around 3.2%-6%.

Before NUMA optimization, CPU Usage per ffmpeg instance is around 13.6%-16%.

7 Revision History

Version	Change	Create Date	Author
1.0	Document Created	Oct 1,2019	Tom Han
2.0	Updated to new template	November 30, 2021	Balwinder Sidhu

8 Legal Notice