CPU

Threads

NUMA (**Non-Uniform Memory Access)**

On a typical two-socket server, each processor has RAM attached to it. A core can access RAM attached to either processor, but its local RAM is usually faster to reach. Linux represents these groups as *NUMA nodes*. That makes two placement decisions matter: which core runs your thread, and where its data lives.

Say you have **one HPC compute node** with two physical CPUs and **512 GiB of RAM**. A simple layout would be:

```
One HPC compute node (one server)
├── NUMA node 0 → CPU socket 0 (32 cores) + 256 GiB of local RAM
└── NUMA node 1 → CPU socket 1 (32 cores) + 256 GiB of local RAM
```

Check NUMA settings passed to kernal at boot:

```
tr ' ' '\n' < /proc/cmdline | grep '^numa'
```

On a NUMA system, pinning a thread to a core usually means that any new memory pages it first writes to are allocated in that core's local NUMA node, assuming the default local allocation policy. Changing CPU affinity by itself leaves existing memory pages where they are.

CPU Affinity

CPU affinity specifies which CPU cores a thread is allowed to run on.

- **CPU affinity uses logical CPU IDs.** On your Pi, IDs `0–3` map one-to-one to physical cores. On a server with simultaneous multithreading, two logical CPUs can share one physical core. Pinning a thread to a CPU still allows other threads to use that CPU.
- `taskset` **applies to individual threads.** With a process PID, `taskset -pc 0 1234` changes the main thread. Use `taskset -apc 0 1234` to change all existing threads in that process, or `taskset -c 0 command` to launch a new command with that affinity.
-

taskset is the main command used for checking and changing process affinity settings.

Flags	Description
-pc $PID	Check the affinity settings for a PID
-pc $CORE $PID	Set a specific core affinity for a PID. IE `taskset -pc 0 1234`
-pc $CORE-$CORE $PID	Set a range of cores affinity for a PID. IE `taskset -pc 0-3 1234`
-pc $CORE, $CORE, $CORE $PID	Set a list of specified cores for a PID. IE `taskset -pc 1,2,3 1234`

NICE

Niceness tells Linux how to share CPU time among competing threads on the cores they’re allowed to use, with lower nice values generally getting a larger share.

In Linux, the nice value is a user-space and process-level mechanism that influences the scheduling priority of a process. It helps determine how much CPU time a process receives relative to other processes. The concept of "niceness" ranges from -20 (most favorable to the process) to 19 (least favorable to the process), with the default being 0.

Setting Nice Value for a new process

Nice value can be set when initially running the command

```
nice -n 15 command
```

Setting Nice value for an existing process

1234 needs to be replaced with the PID

```
renice -n 5 -p 1234
```

Revision #4
Created 2026-11-25 12:01:27 UTC by Daniel

Updated 2026-11-25 13:04:05 UTC by Daniel