Check VPS CPU steal time and container throttling

A slow application with a modest average CPU graph is worth investigating at several layers. One busy thread, a container quota, storage waits or time unavailable to a virtual CPU can produce different symptoms. This guide collects a short sample without launching a benchmark or changing scheduling settings.

Record a useful sample

1. Note the UTC time and the exact slow action, such as a delayed page request or server tick.
2. Inside the Linux VPS, run a short sample:

```
date -u
vmstat 1 10
top
```

In `vmstat`, the first report generally represents averages since boot; use the later interval rows for the current incident. Read `us`, `sy`, `id`, `wa` and `st` together. Steal time reports time a virtual CPU did not receive from the hypervisor while it could have run. It is distinct from time used by your application and from an application-level CPU quota.

Do not diagnose the host from one brief spike. Repeat the sample during normal operation and during the slowdown. A zero steal reading also does not rule out all provider-side issues. Per-core views in an interactive process monitor can reveal a single busy core hidden by a whole-machine average.

Check the workload’s CPU allowance

If the application runs in Docker, inspect only the relevant container:

```
docker inspect --format 'NanoCpus={{.HostConfig.NanoCpus}} CpuQuota={{.HostConfig.CpuQuota}}
CpuPeriod={{.HostConfig.CpuPeriod}} Cpuset={{.HostConfig.CpusetCpus}}' my-container
```

Replace `my-container`. These settings describe different ways to constrain CPU access; compare them with the deployment configuration instead of assuming the guest's total vCPU count is available to every container. Avoid pasting a full inspection output into public chat because unrelated configuration can contain secrets.

Compare throttling counters

On cgroup v2, a workload's `cpu.stat` exposes bandwidth-throttling counters and `cpu.max` describes its quota. For a systemd service, obtain the path first:

```
systemctl show myapp.service -p ControlGroup
cat /sys/fs/cgroup/ACTUAL-GROUP/cpu.max
cat /sys/fs/cgroup/ACTUAL-GROUP/cpu.stat
```

Replace the unit and `ACTUAL-GROUP` with the returned path. Read counters twice around the slow action; old cumulative throttling is not proof of present throttling. Parent cgroup constraints can matter too. Missing files require checking the cgroup layout, not adding files manually.

Report the right evidence

Send support the timed interval sample if repeated steal time coincides with the incident. Send quota configuration and increasing throttle counters to the application administrator when workload limits are implicated. High I/O wait needs a storage investigation instead. After an agreed change, repeat the same action and compare latency as well as CPU readings. Do not run a stress test on a busy production VPS or raise all quotas blindly to manufacture a better graph.

Related CloudExa guides

- Collect storage latency evidence
- Check another source of application stalls

Technical references

- procps vmstat manual
- Kernel CPU accounting fields
- Docker CPU constraints
- Kernel cgroup v2 CPU interface

Revision #1
Created 2026-11-10 10:51:41 UTC by CloudExa Hosting Admin
Updated 2026-11-10 10:51:41 UTC by CloudExa Hosting Admin