When selecting cloud infrastructure, one of the most common and tempting metrics to rely on is the number of vCPUs a virtual machine instance offers. At face value, more vCPUs mean more compute capacity and better performance. However, the reality behind the scenes is far more complex—especially when considering **shared CPU models** and how different cloud providers implement and enforce CPU sharing rules.

In this post, I'll unpack why relying solely on vCPU counts is often misleading, especially for always-on small services that quietly inflate your cloud costs. I'll delve into how AWS, Azure, and other clouds define and regulate shared CPU usage, the pitfalls of using only average CPU metrics for right-sizing, and the importance of measuring workload peaks with the correct observation window and percentile metrics. Finally, we'll touch on how tools like AWS Compute Optimizer and Azure Advisor can help—but only when you understand their underlying assumptions and limitations.

Understanding vCPU Context: What Does It Really Mean?

First, let's clarify what vCPU means across providers. A *vCPU* typically represents a hardware thread (hyper-thread) on a physical CPU core. However, how these virtual CPUs are scheduled and shared between tenants varies substantially.

- **Dedicated vCPUs:** Instances guarantee exclusive CPU capacity, so if you have 4 vCPUs, you effectively own 4 hardware threads' worth of compute.
- **Shared vCPUs:** Virtual CPUs are time-sliced over physical CPUs among multiple noisy neighbors, leading to variable performance.

Different cloud providers implement shared vCPUs differently:

AWS Shared CPU Instances

AWS offers burstable instances through its T-family (e.g., t3, t4g), where vCPUs have a baseline CPU allocation plus a credit system https://computingforgeeks.com/shared-cpu-cloud-waste-migration-guide/ for bursting. AWS documents this as *sustained share* CPU credit mechanics—your instance earns CPU credits when idle and spends them to burst above baseline.

- When CPU credits run out, the instance throttles to baseline performance, often much lower than peak capacity.
- AWS's Compute Optimizer can suggest resizing based on CPU utilization, but it largely focuses on average metrics.

Azure Shared CPU Instances

Azure's B-series burstable VMs follow a similar concept but with different specifics on burst rules, credit accumulation, and throttling thresholds.

- The Azure Advisor recommends optimizations, but like AWS, it often relies on average CPU utilization for its calculations.
- Azure defines baseline and burst CPU percentage by VM size, and bursting is limited by accumulated credits.

Key Takeaway:

Equivalent vCPU counts do not mean equivalent performance. Your 2 vCPU T3 instance on AWS won't necessarily behave the same as a 2 vCPU B-series VM on Azure during CPU bursts or sustained loads.

Why This Matters: Always-On Small Services Hide Cloud Waste

Small always-on services, such as internal tools, feature flagging backends, or monitoring agents, often fall through the cracks in cloud waste audits. Because they are small, teams mistakenly size them based on either average CPU or the nominal vCPU count instead of performing detailed peak usage analysis.

What does that lead to?

- Unnecessary allocation of burstable or even larger dedicated instances "just in case."
- Instances running at zero or near-zero average CPU but costing the same in baseline hourly pricing.
- Failure to catch CPU credit exhaustion, leading to throttling and degradation masked by average metrics.

This hidden waste becomes expensive as you scale. For example, a fleet of 100 small burstable instances running with exhausted CPU credits 60% of the time is severely underperforming yet consuming capacity you pay for.

Measure the Right Metrics: P95, P99, and Spike Duration > Averages

Most tools provide average CPU utilization over a selected period. Using this alone is dangerously misleading because:

- Average CPU can mask short spikes crucial for latency-sensitive workloads.
- CPU burst credit exhaustion depends on the *duration* and *intensity* of CPU spikes, not just average usage.

The Importance of Percentiles

Instead, measure your CPU utilization using percentiles like P95 and P99 to capture near-worst-case usage:

Metric What It Represents Benefit Average CPU Mean CPU utilization over interval Easy to understand but hides spikes P95 CPU CPU utilization level exceeded only 5% of the time Shows typical upper-bound usage, useful to capacity plan for peak loads P99 CPU CPU utilization level exceeded only 1% of the time Identifies rare bursts that affect latency or reliability

Spike Duration Matters

Next, consider how long sustained CPU usage spikes last. Short sub-second spikes may be irrelevant, but multi-minute sustained bursts might exhaust credits.

To analyze properly:

1. Collect CPU usage telemetry at a fine granularity (1-min or better).
2. Calculate the distribution of spike durations — how often does CPU exceed baseline or credit thresholds, and for how long?
3. Compare observed burst patterns to provider-defined *burst rules* and credit limits.

Only by combining percentile metrics with spike duration analysis can you confidently right-size shared CPU instances.

How AWS Compute Optimizer and Azure Advisor Fit In

Both **AWS Compute Optimizer** and **Azure Advisor** are powerful cost and performance optimization tools with sophisticated machine learning models analyzing utilization patterns. However, my experience shows some caveats:

- **AWS Compute Optimizer:** Primarily optimizes for average CPU utilization and memory metrics. It may recommend downsizing burstable instances based on average CPU, leading to unexpected throttling if P95/P99 spikes aren't considered.
- **Azure Advisor:** Similarly reports recommendations using average CPU thresholds and doesn't expose detailed burst credit exhaustion signals.

Thus, to avoid premature wrong-sizing:

1. **Review Compute Optimizer and Advisor recommendations.** Use them as a starting point, not an oracle.
2. **Analyze your own CPU utilization percentiles and burst durations.** Correlate with instance baseline and burst credit specs.
3. **Run pilots with explicit rollback criteria.** For example, monitor latency and error rates carefully when resizing burstable instances.

Comparing Shared CPU Definitions and Rules by Provider

Provider Shared CPU Model Sustained Share / Baseline Burst Credit Rules Key Gotcha AWS (T3, T4g) Baseline CPU % with burst credits earned during idle Varies by instance size, e.g. t3.nano baseline 5% Credit accrual and spend tracked per-minute, throttles aggressively when credits run out Credit exhaustion causes drastic throttling; average CPU misses this Azure (B-series) Baseline CPU % with credits for bursting Varies by VM size; e.g., B1s baseline 10% Credits accumulate when under baseline; bursting limited by credit balance Long CPU bursts deplete credits; average CPU often hides this Google Cloud (e2-micro, f1-micro) Shared physical CPU with bursting capabilities Lower baseline CPU, burst up to 100% No explicit credits, but performance contention occurs Performance can vary significantly; vCPU alone is unreliable

Recommendations for Practitioners

Based on all this, here are some practical next steps to responsibly approach vCPU-based sizing given varying shared CPU implementations:

1. **Understand your application's CPU burst tolerance.** What are acceptable latency and throughput degradation windows during CPU throttling? Write rollback criteria for pilot resizing projects.
2. **Collect high-resolution CPU metrics and calculate P95/P99 percentiles and spike durations.** Use these to model actual credit usage against provider burst rules.
3. **Don't trust vCPU counts as absolute performance guarantees.** Always couple with CPU credit availability or baseline sustained CPU percent.
4. **Monitor CPU credit metrics if provider exposes them (e.g., AWS's CloudWatch metrics for CPU credit balance).** Address credit exhaustion proactively via alerting or autoscaling.
5. **Use AWS Compute Optimizer and Azure Advisor as inputs, not the final word.** Augment recommendations with your detailed burst analysis.
6. **Prefer dedicated or unlimited CPU instances for latency-sensitive or sustained workloads.** The cost premium often offsets the unpredictable outage or throttling impact.

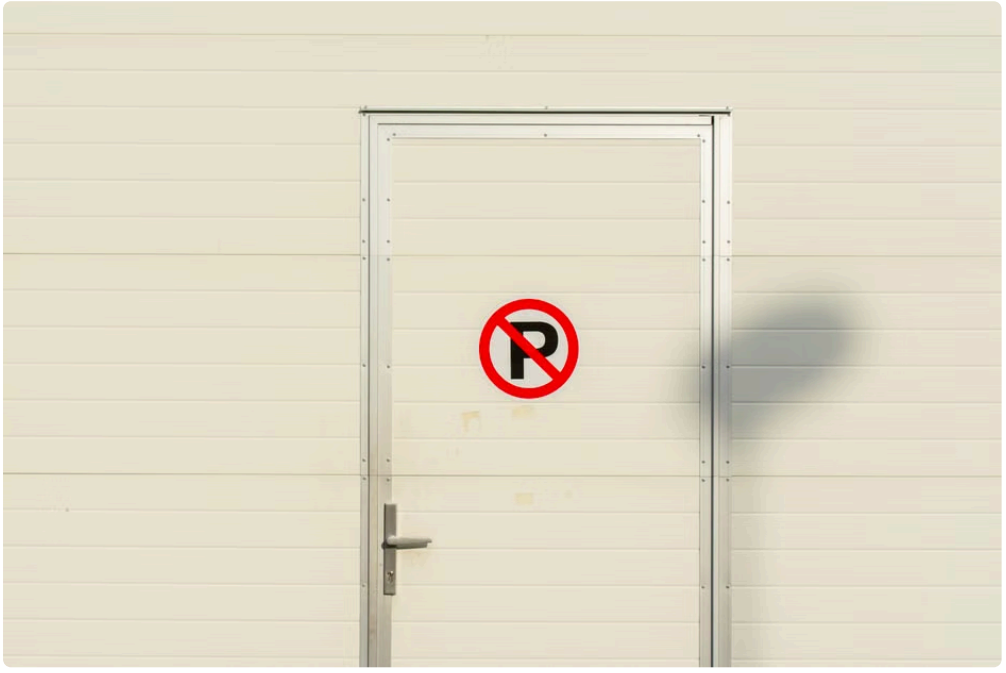
Conclusion

vCPU counts are a tempting shorthand for sizing compute resources, but when providers implement shared CPU using diverse burst and credit schemes, naive reliance on vCPU alone leads to hidden cloud waste and unpredictable performance.

Always remember:

- Shared CPU and burst rules differ markedly by provider.
- Average CPU utilization is an insufficient metric; analyze at P95/P99 percentile level and consider spike durations.
- Tools like AWS Compute Optimizer and Azure Advisor help but don't replace detailed workload profiling and explicit rollback criteria.
- Sustained share vs. burst credits are critical to understanding how "performance" maps onto hourly cost.

By embracing these engineering discipline norms, you avoid the trap of overpaying for nominal vCPU capacity that doesn't translate into the real, usable compute your applications demand.



If you're embarking on cloud cost optimization or migration initiatives, start by asking: *what do the P95 and P99 CPU utilizations look like, and how long do bursts last?* Without that insight, any resizing or switching instance types is a shot in the dark.