In cloud cost optimization, it's tempting to switch every low-utilization instance to burstable or shared CPU options to save money. Tools like AWS Compute Optimizer and Azure Advisor reinforce this by recommending instance types with shared CPU resources based on average utilization. But before you blindly follow utilization metrics, pause and reconsider.

Many always-on, small services have tight latency targets and hard deadlines requiring predictable performance that shared CPUs may not reliably guarantee. In this article, I'll share insights from 12 years of cloud infrastructure experience, focusing on why you should sometimes keep dedicated CPU allocations even if the average utilization looks low.

Understanding Shared CPU and Dedicated CPU Across Providers

One of the most common misunderstandings in cloud CPU management is treating vCPU counts as strict performance guarantees. The actual CPU resource availability depends heavily on the instance type model—shared or dedicated—and how the cloud provider defines "shared." Let's look at definitions across major clouds.

Shared CPU Definitions Differ by Provider

- **AWS Burstable Instances (T3, T4g, etc.):** Use CPU credits that accumulate during idle time and can be spent during CPU bursts. You get limited baseline CPU performance, but performance can spike generally only up to 100% of a vCPU for a limited time unless credits last.
- **Azure B-Series VMs:** Provide burstable CPU with credits, similar to AWS. When credits deplete, CPU throttling happens.
- **Google Cloud Shared-core VMs:** Virtual CPUs are time-shared among multiple tenants, with no credit system but inherently non-dedicated CPU.

computingforgeeks.com

Dedicated CPU instances provide a fixed allocation of physical CPU cores to your VM, removing noisy neighbor concerns and providing predictable compute availability.

Why Average Utilization Can Be Misleading

Compute optimizer tools often generate recommendations based on average CPU utilization over some observation window. While valuable as a starting point, average metrics alone can be dangerously misleading because:

- The **average smooths peaks**, hiding infrequent but critical CPU spikes.
- Short latency-critical jobs can require bursts of CPU that average usage doesn't capture.
- Different workloads show different CPU utilization patterns: steady background, spiky bursts, or periodic batch jobs.

For services with tight latency targets or real-time constraints, occasional CPU starvation due to bursting limits can mean deadline misses and degraded user experience.

Always-On Small Services Hide Cloud Waste

Small services with “always on” infrastructure are particularly easy to overlook. Even if CPU appears mostly idle, the requirement to keep the service responsive means you cannot scale down or burst aggressively without risking latency explosions. These small instances tend to multiply, each silently wasting valuable cloud spend if bursting isn’t a good fit.

Measure Peaks with the Right Observation Window

To determine whether you can safely switch from dedicated to shared CPU instances, you need more detailed CPU usage insights:

1. **Observation Window Duration:** Use windows long enough to catch rare spikes but short enough to retain temporal resolution (e.g., 5 minutes rather than 1 hour).
2. **P95 and P99 CPU Usage Percentiles:** These provide a better picture of peak CPU demand. For example, if P99 CPU usage nears 100%, bursting risks throttling.
3. **Spike Duration and Frequency:** Look at how long CPU spikes last. Short, infrequent spikes may be okay on shared CPU, but sustained spikes likely need dedicated CPU.

As an SRE engineer, you should also ask:

- What do the P95 and P99 CPU utilization curves look like during peak hours?
- Are CPU spikes correlated to traffic bursts or user actions with hard deadlines?
- How much time does the instance spend near 100% CPU usage?

Use Percentiles and Spike Duration, Not Averages, for Right-Sizing

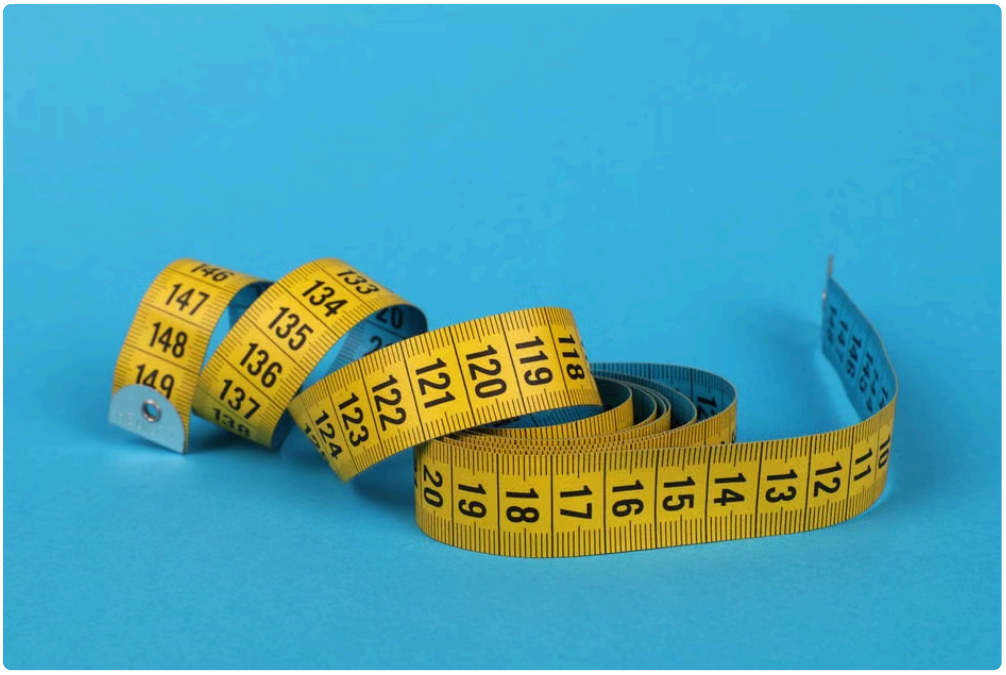
Let’s say you run a service with a tight latency target—say 10ms API response time—and hard deadlines. If the CPU hits 95–100% utilization, even rarely, your latency may spike dramatically or requests queue up, violating SLAs.

By incorporating percentile-based CPU metrics, you get a fuller picture of your CPU needs. Example:

Metric Value Interpretation Average CPU Utilization 25% Looks low; would suggest downsizing P95 CPU Utilization 85% High spikes; potential risk if moved to burstable P99 CPU Utilization 98% Almost full CPU near

constantly at rare times; might need dedicated CPU Spike Duration Up to 5 minutes Spike lasts longer than one-minute CPU credits; bursting might not suffice

In this example, the average looks low enough to consider bursting, but percentile data and spike length indicate dedicated CPU is warranted for predictable performance.



When to Choose Dedicated CPU Despite Low Average Utilization

Here are concrete scenarios where keeping dedicated CPUs is the better engineering choice:

- **Tight Latency Targets:** Services that need low, predictable response times can't tolerate CPU throttling delays.
- **Hard Deadlines:** Jobs or requests that must complete within a fixed time window require consistent CPU availability.
- **Spiky But Short-Duration Peaks:** If CPU usage spikes to 100% for brief periods that coincide with deadline-critical operations.
- **Multi-threaded Workloads with High vCPU Demand:** Shared-core burstable instances may provide only baseline CPU time insufficient for multi-core parallelism.
- **Services Where CPU Bursting Credits Can Exhaust:** Constant high CPU bursts can deplete credits, causing throttling unknown to naive average-based monitoring.

Practical Tips for Evaluating Your Fleet

1. **Enable Detailed Monitoring:** Use CloudWatch with 1-minute granularity on AWS, Azure Monitor for VM insights, or Google Operations Suite. Don't rely on default 5-minute average metrics.
2. **Analyze Percentiles:** Check P90, P95, and P99 CPU usage and visualize over different time frames.
3. **Quantify Spike Durations:** Count how long CPU utilization stays above selected thresholds.
4. **Review SLA and Latency Metrics:** Correlate CPU spikes to latency outliers and SLA misses.
5. **Run Pilot with Rollback Criteria:** Before switching to burstable instances, define rollback triggers, e.g., latency increases beyond X%, error rates spike, or CPU throttling detected.

6. **Understand Provider Limitations:** AWS Compute Optimizer and Azure Advisor recommendations are helpful but don’t substitute for workload-specific analysis.

Summary: Balancing Cost and Performance

Optimization is about balancing cost savings with operational risk. Dedicated CPUs come at a premium but provide predictable compute capacity critical for many production workloads. Always validate recommendations from cloud advisors or compute optimizer tools by digging into percentile CPU metrics and understanding your workload's latency and deadline requirements.

Remember these key points:

- **Averages lie:** Make decisions based on P95, P99, and spike duration, not average CPU utilization.
- **Shared CPU semantics vary:** Know your cloud's bursting and sharing model before downsizing.
- **Latency and deadlines matter:** Prioritize dedicated CPUs for any service with hard deadlines and tight latency targets.
- **Implement pilots and rollback plans:** Don’t switch instance sizes blindly—test rigorously in production-like environments.

By combining robust observation windows, percentile metrics, and a clear understanding of your workload’s performance profile, you’ll make smarter decisions that optimize cost while avoiding unexpected latency degradations.

Further Reading

- AWS Compute Optimizer Documentation
- Azure Advisor Cost Recommendations
- Google Cloud: Avoid VM Performance Challenges on Shared-Core Machines
- Brendan Gregg's CPU Monitoring Best Practices