Migrating workloads to shared CPU instances in public clouds such as AWS and Azure can bring significant cost savings, but it's not a move to be made by looking at CPU utilization alone. Many teams get fixated on average CPU numbers and proceed with changes that backfire due to hidden bottlenecks or overlooked resource demands.

Over my 12 years running cloud infrastructure and SRE teams, performing countless cost reviews and migrations across AWS, Azure, and Google Cloud, I've learned that measuring the right signals with the right granularity—especially for always-on small services—prevents waste and performance hiccups.

This post digs into what you should be measuring besides CPU before you take the plunge on shared CPU instance families. I'll also walk through nuances in CPU sharing definitions across providers, the criticality of choosing proper observation windows, and how to use AWS Compute Optimizer and Azure Advisor without getting misled.

Why Always-On Small Services Hide Cloud Waste

Small services that run 24/7 and have modest CPU profiles frequently become the "invisible" cost bucket in your cloud bill. Individually, they may look trivial, but collectively these services can consume a sizable chunk of your budget. Because they mostly operate with low CPU at a glance, teams often pick smaller shared CPU instances hastily.

However, many of these always-on services experience transient CPU peaks, memory spikes, or network bursts during background tasks, cache warmups, or occasional request surges that average CPU metrics hide.

Being blind to these peaks invites problems:

- **Throttling on shared CPU:** High-CPU bursts may get throttled if the underlying vCPU allocation is too small or constrained by hypervisor scheduling.
- **Memory starvation:** Insufficient memory headroom can cause OOM kills or excessive swapping, which kills performance but doesn't always show in average CPU metrics.
- **Network bottlenecks:** Network egress constraints can escalate latency and external costs, especially when services sporadically redistribute data or sync with external APIs.

Shared CPU Definitions Differ by Provider — Know What You're Getting

"Shared CPU" is a generic term covering varied implementations with subtle but critical differences between cloud providers. Let's take a quick look at how AWS and Azure define shared CPU behavior in their instance types.

Provider Shared CPU Instance Concept How CPU Credit/Sharing Works Common Gotchas AWS (T Series, T3, T4g) CPU Credits accumulated during idle periods allow controlled CPU bursts

- Earn CPU credits when idle
- Spend credits during bursts
- Unlimited mode allows bursting beyond baseline but may incur charges
- Long CPU spikes can exhaust credits leading to throttling
- Baseline CPU is not a performance guarantee

- Ephemeral performance variations depending on other host tenants

Azure (B-series) Similar CPU credit model with accrual during low usage

- CPU credits accumulate up to a max
- Spend credits to burst above baseline CPU
- Limits on max credits and burst duration

- Long sustained CPU load causes throttling
- Misunderstanding of baseline leading to performance surprises
- Different instance sizes yield varying CPU/memory ratios

Other cloud providers may use completely different approaches (e.g., Google's shared vCPU is often fine-grained overcommit with different throttling). This underlines the importance of understanding micro-level behavior, not just top-level instance specs.

Measure Peaks with the Right Observation Window

What many cost optimization exercises miss is the need to analyze the *peaks* of resource usage rather than averages — and to do so over a time window that matches your application's operating characteristics.

Here's why averages are misleading:

- **CPU:** Average CPU utilization may hover under 10%, but spikes to 80-90% lasting 30 seconds to a few minutes can break shared CPU credit budgets.
- **Memory:** Average memory usage can look fine, but short-lived spikes may cause swapping or OOM kills, hurting stability.
- **Network egress:** Infrequent bursts of heavy egress traffic, if unaccounted, lead to unexpected overage costs and latency spikes.

So, what observation window should you use?

1. **Understand your request/load profile:** Look at your P95 and P99 request latency and throughput over time—these align to service spikes and impact resource demand.
2. **Align metrics window accordingly:** If spikes typically last 1-5 minutes, measure CPU, memory, and network utilization in one-minute intervals or finer.
3. **Don't rely solely on 5-minute averages:** Cloud-native monitoring systems (CloudWatch, Azure Monitor) often default to 5-minute intervals or more, which bluntly smooth spikes.

   computingforgeeks.com
4. **Use high-frequency metrics if available:** For example, AWS EC2 detailed monitoring at 1-minute or even 1-second granularity.

Use Percentiles and Spike Duration, Not Averages

When analyzing resource usage, use these best practices:

- **Percentiles:** Look beyond P50 (median) to P90, P95, and P99. These indicate how often your service experiences unusually high resource demands.
- **Spike duration:** Identify how long spikes last. A brief spike of 10 seconds may be tolerable on a shared CPU instance, whereas sustained 5-minute spikes will exhaust CPU credits.

- **Multi-dimensional insights:** Don't just look at CPU percentiles in isolation. Combine with memory headroom, request latency percentiles, and network egress spikes to get the full picture.

Key Metrics Beyond CPU You Must Measure

Memory Headroom

Memory is often the silent culprit behind performance regression in shared CPU migrations. Because instance sizes come with fixed memory, squeezing into a smaller shared CPU instance with less RAM will cause:

- Increased garbage collection pauses (if using JVM or similar runtimes)
- Frequent swapping or OOM crashes if memory headroom is inadequate
- Higher latency through reduced buffer or cache sizes

How to measure: Track the *memory utilization percentiles (P90, P95, P99)* over your chosen observation window. Also monitor swapping rates and OOM event alerts in logs. You should have at least 10-20% free memory headroom during peak periods to be safe.

Request Latency (P95, P99)

Ultimately, end-user experience is the best proxy for whether your migration is viable. Latency spikes often tell a more nuanced story than resource metrics alone.

Look at:

- **P95 and P99 request latency percentiles:** Are they stable or showing upward trends during previously known CPU/memory peak windows?
- **Spike correlation:** Do latency spikes correspond to CPU credit exhaustions, memory pressure, or network bottlenecks?
- **Duration:** How long do latency peaks last, and is this acceptable for your SLAs?

Network Egress

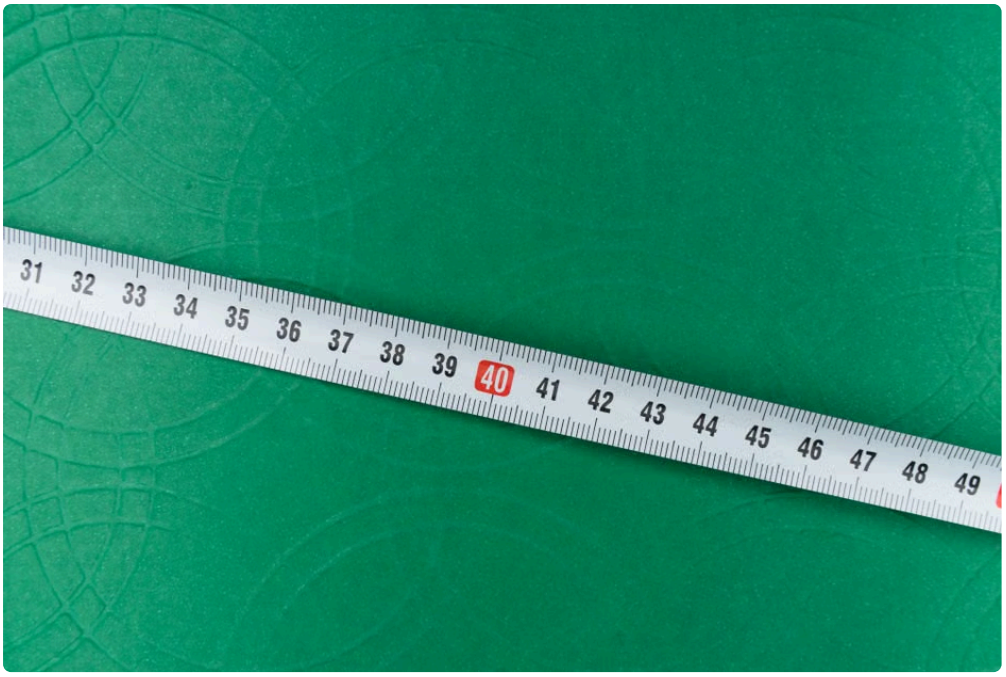
Network egress impacts cost and performance but is often overlooked in CPU-focused migrations. Services with irregular but heavy network egress can:

- Drive up cloud bills, especially across availability zones or regions
- Cause transient latency if bandwidth limits or bursts exceed capacity
- Induce CPU load spike moments due to network I/O processing

Measurement tips: Use your cloud provider's network egress reporting to analyze P95 and P99 spikes. Look for burst patterns correlated with scheduled jobs, cache repopulation, or data syncs.

How to Use AWS Compute Optimizer and Azure Advisor Effectively

Both AWS Compute Optimizer and Azure Advisor provide recommendations for right-sizing instances, including notes on when shared CPU instances may be a fit. But here's what you need to keep in mind:



AWS Compute Optimizer

- It considers CPU utilization, memory utilization (if enhanced logging is enabled), and EBS volume metrics.
- Check Compute Optimizer's *recommendations vs actual P95/P99 utilization*—if you only look at average CPU, you risk picking a smaller instance that throttles during bursts.
- Enable Compute Optimizer's enhanced infrastructure metrics collection to get memory usage insights, which is crucial for migration decisions.
- Remember its feedback loop can take 24-48+ hours, so run it over periods with realistic load, including known peak times.

Azure Advisor

- Provides recommendations for virtual machine sizing, including B-series burstable VM relevance.
- Its performance metrics, however, are often aggregated over 5 or 10 minutes, which can mask spikes.
- Pair Azure Advisor insights with Azure Monitor's *metrics explorer* using 1-minute granularity and custom alerting on high CPU, memory, and network percentiles.
- Don't blindly trust a "better size" suggestion without cross-checking application-level latency and error rates.

Practical Rollout Tips and Rollback Criteria

Treat your shared CPU migration as a controlled experiment:

1. **Define rollback triggers up-front:** For example:
    - P99 latency increase > 15% for 10 minutes
    - OOM kills or swapping alerts within 1 hour
    - CPU throttling events visible in instance metrics
2. **Run a pilot with canary traffic:** Select a subset of services or instances first, measure all key metrics.
3. **Observe patterns under real peak usage:** Don't just test during quiet hours.

4. **Continue monitoring after rollout:** High-frequency metrics dashboards with alerts on P95/P99 CPU, memory, latency, and network egress.

Summary Checklist: What to Measure for Shared CPU Migration

Metric Recommended Aggregation Why It Matters Measurement Tools CPU utilization P95, P99 in 1-minute granularity Identify burst CPU demands that impact CPU credits AWS CloudWatch (detailed monitoring), Azure Monitor Memory utilization / headroom P90, P95 with swap/oom event detection Prevent OOMs and swapping that degrade performance CloudWatch with enhanced monitoring, Azure Monitor Request latency P95, P99 Actual user experience reflection Application monitoring (Datadog, New Relic, Cloud monitoring) Network egress P95, P99 with burst analysis Impacts cost and latency; correlates with CPU spikes Cloud provider network metrics dashboards

Final Thoughts

Moving to shared CPU cloud instances isn't just about chopping your CPU hours down. You must measure and understand the full resource dynamics of your workload—especially memory headroom, request latency percentiles, and network egress behavior—using the right observation windows and avoiding smoothing over critical spikes.

Trust but verify recommendations from tools like AWS Compute Optimizer and Azure Advisor by cross-referencing with your application-level SLAs and workload characteristics. Always pilot safely with clear rollback criteria before widespread rollouts.

Following these guidelines will ensure you unlock cost savings while preserving the performance and reliability your users expect.