Migrating from dedicated CPU instances to shared CPU (burstable or t-shirt sizes offering partial vCPU allocation) can initially look like a straightforward cost optimization. Vendors pitch shared CPU as a way to leverage idle periods without paying for full dedicated capacity. But as any cloud practitioner with years in infrastructure and SRE will tell you, the devil is in the details — especially when those "details" hide subtle performance regressions that silently degrade customer experience.

In this post, I walk through how to identify when your move to shared CPU might be causing more harm than good—and how to use tools like AWS Compute Optimizer and Azure Advisor to make smarter decisions. Spoiler alert: Always dig deeper than average CPU utilization numbers. Measure peak usage and spike duration with proper observation windows and use your percentiles (P95, computingforgeeks P99) instead of averages before pulling that trigger on instance type changes.

Understanding Shared CPU: What Does It Even Mean?

The first pitfall is to treat shared CPU instances as simply "less powerful CPUs." In reality, each cloud provider defines shared CPU in ways that aren't directly comparable:

- **AWS T-series (e.g., t3, t4g):** These burstable instances accumulate CPU credits when idle, enabling them to burst above their baseline for short periods.
- **Azure B-series:** Similar burstable capability but with different credit accrual and consumption models.
- **Google Cloud's E2 shared-core instances:** These provide fractional vCPU shares with noisy neighbor management and less deterministic performance.

The takeaway? Don't just compare vCPU counts or core fractions across providers or even within your own account. "2 vCPU shared" in AWS is not the same guarantee as "2 vCPU shared" in Azure. And measuring performance impact requires more nuanced metrics than CPU alone.

Why Average CPU Usage Is a Trap

It is common to rely on average CPU utilization to make resizing decisions. If your average CPU hovers around 20%, it's tempting to downsize to a shared CPU instance and hope the peak usage will fit the bursting model.

However, this is a shortcut to disaster because:

- Average CPU does not reflect burst spikes and their duration.
- Batch jobs, worker queues, and asynchronous workloads often have short, intense CPU bursts that break SLAs.
- Performance regressions manifest in error rates, latency spikes, and throughput drops—not just CPU numbers.

An engineer who "always asks what the P95 and P99 look like before touching instance types" will examine:

1. Peak CPU at P95 and P99 across a full week (to capture weekdays and weekend variation).
2. Duration of CPU spikes (are they 10 seconds? 10 minutes?).
3. Latency and error rate correlation with CPU bursts.

Signs You Should Really Consider Reverting to Dedicated CPU

Let's move beyond the theoretical. What are the real-world indicators your shared CPU instances are causing trouble and you should rollback?

Error Rate Increase

Spikes in error rates (HTTP 5xx spikes, background job failures, queue processing timeouts) are the canary in the coal mine. Since shared CPU instances may throttle your bursts after credit depletion, you might experience:

- Timeouts due to CPU starvation when demand spikes.
- Failed API calls correlated with CPU credit exhaustion periods.

Use monitoring tools tied into your APM to look for error rate deviations alongside CPU and credit metrics.

Latency Regression

Latency percentiles jump. You'll see P95 and especially P99 latencies spike, even if median latency looks stable. For example:

Metric Before Shared CPU After Shared CPU Noted Change Median latency (ms) 50 55 +10% P95 latency (ms) 120 350 +191% P99 latency (ms) 250 900 +260%

This is a classic sign of resource contention during CPU credits depletion or noisy neighbor interference in shared environments.

Throughput Drop

Another key symptom is a measurable drop in overall throughput (requests per second, messages processed per minute). Even if average CPU looks "in the green," the shared CPU throttling or noisy neighbors will limit the ability to burst when the workload demands.

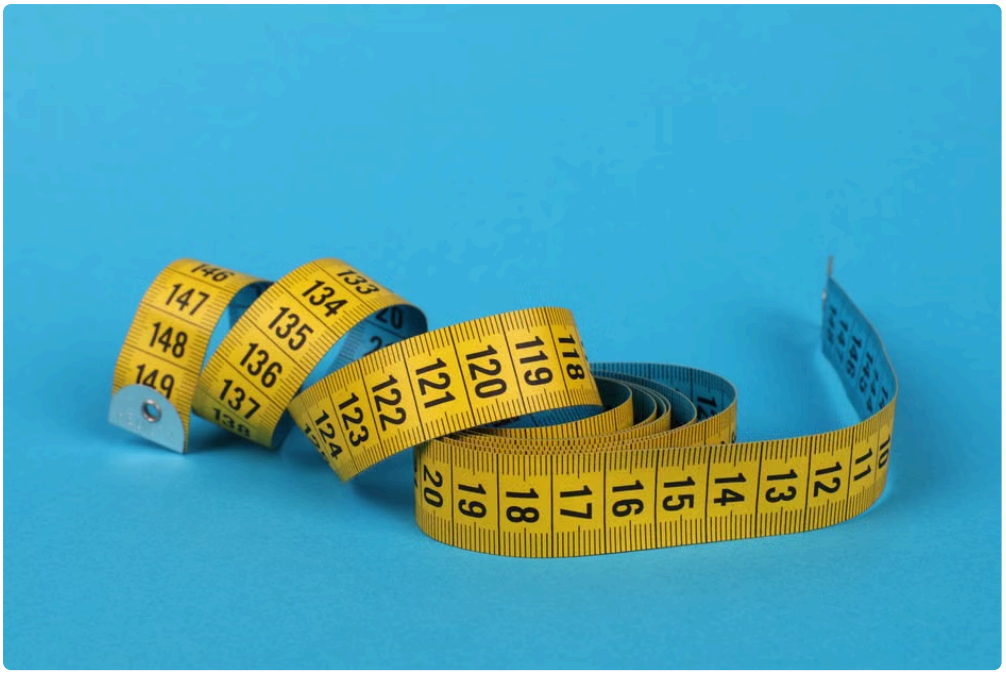
Look for sustained throughput degradation, especially during peak load periods:

- Worker queues backing up.
- Retries or dropped requests increasing.

- Observed CPU throttling metrics (AWS CloudWatch for T series credit exhaustion, Azure Metrics for B-series bursts).

How AWS Compute Optimizer and Azure Advisor Can Help—But Don't Stop There

Both AWS Compute Optimizer and Azure Advisor provide data-driven recommendations on right-sizing, including suggestions for switching between shared and dedicated CPU families.



- **AWS Compute Optimizer:** Analyzes utilization and makes recommendations based on 14 days of historical CPU, memory, and network usage. It also highlights performance risks, including those relating to burst credit availability.
- **Azure Advisor:** Integrates with Azure Monitor to assess VM utilization patterns and recommend VM size adjustments, factoring in burstable VM series utilization.

However, both tools use averages and smoothed utilization data at intervals that might mask short spikes or credit depletion bursts. They rarely factor in error rates or latency regressions directly.

This is why I always recommend:

1. Correlate cloud cost optimizer recommendations with your application performance monitoring (APM) and logs.
2. Explicitly measure P95 and P99 CPU usage, not averages, across at least a 7-day window including peak and off-peak hours.
3. Track CPU spike duration—burstable instances might handle short spikes but choke on longer sustained load.
4. Watch error rates, latency, and throughput metrics to validate that performance SLAs hold after changes.
5. Define rollback criteria in advance — e.g., >20% increase in P99 latency sustained for >30 minutes triggers pod/container scaling back to dedicated CPU.

Practical Steps Before Reverting

1. **Run a pilot:** Isolate a small set of replicas or service shards on shared CPU.

2. **Instrument rigorously:** Enable detailed CPU credit and throttling metrics, plus enhanced monitoring of error, latency, and throughput.
3. **Observe for a full week:** Capture weekday and weekend load variation.
4. **Compare percentiles and spike durations to the dedicated baseline.**
5. **Document rollback criteria:** For example, sustained >15% error rate increase or P99 latency increase over baseline for 30+ minutes.
6. **Automate rollback:** Use IaC or fleet management tools to quickly revert instances or scale pools up when criteria hit.

Summary Checklist: Signs You Need to Revert

Symptom Measurement / Indicator Reason to Revert? Error rate increase Unexpected 5xx spikes beyond baseline during CPU credit depletion Yes Latency regression P95 and P99 latency doubles or more vs dedicated baseline Yes Throughput drop Sustained decrease in processed transactions/request rate at peak Yes High CPU utilization but no credit available CPU credit metrics consistently near zero Yes Long duration CPU spikes Spike duration > burst credit replenishing interval Yes Inconsistent or jittery performance Periodic latency and CPU variability not explained by workload Consider rollback

Final Thoughts

Shared CPU instances can be a powerful cost-saving lever—especially for always-on small services with predictable, low CPU requirements. But don't let the allure of cheaper pricing lead you to ignore signs of degraded performance.

Before you commit to shared CPU fleets, always:

- Measure the right metrics at the right percentiles over the right intervals.
- Correlate performance degradation (error rate jumps, latency spikes, throughput drops) with CPU credit depletion and throttling.
- Use cloud optimization tools like AWS Compute Optimizer and Azure Advisor as starting points, not the final word.
- Define rollback triggers before launching pilots.

When the symptoms pile up — especially error rate increases, latency regressions, and throughput drops tightly correlated with shared CPU throttling — it's a clear signal to revert and reconsider your approach.

Remember: CPU count is not performance guarantee. Use P95 and P99, not averages. Measure spike duration, not just peak levels. And always question hand-wavy cost estimates that ignore storage and egress—because your customers notice when performance degrades, and so should you.