When optimizing cloud costs and right-sizing infrastructure, moving workloads onto *shared CPU* instances is an increasingly popular tactic. Shared CPU — often branded as burstable or spot-like instances — can yield significant savings for low-utilization services. However, these instance types come with nuances that require careful performance measurement and analysis before and after pilots.

In this post, we'll walk through exactly **what to record** to validate whether switching to shared CPU makes sense in your environment. Along the way, we'll clarify how different cloud providers define "shared CPU" differently, why averages lie, and how using the AWS Compute Optimizer and Azure Advisor tools can guide your decisions. If you care about *latency metrics*, *queue depth*, and *error rates* — and you should — you're in the right place.

Why Shared CPU Pilots Must Be Data-Driven

Before diving into specific measurements, it's important to clarify why pilots on shared CPU instances demand rigorous data collection:

- **Always-On Services Mask Wasted Resources:** Many enterprises have small, always-on services running medium-sized, dedicated instances. These can be huge cost sinks since instances never truly max out their CPU allocation.
- **Shared CPU is Not Uniform Across Clouds:** AWS T-series burstable instances allocate CPU credits differently than Azure B-series or Google Cloud's E2 family. A shared CPU instance on one vendor might behave very differently than the next.
- **Peak Metrics Are More Informative Than Averages:** Average CPU utilization hides spikes that cause latency degradations or errors. You must look at percentiles (P95, P99) and spike duration to avoid performance traps.

In my 12+ years running cost reviews and SRE ops, pilots that rely on average CPU data alone almost always lead to false confidence.

Understanding Shared CPU Across Cloud Providers

Each cloud vendor's definition of a "shared CPU" or burstable instance varies, impacting performance measurement strategies.

Cloud Provider Example Instance Family CPU Sharing Model Key Notes AWS T3, T4g CPU Credits accumulate during idle time, allowing burst above baseline CPU credits burn down with spikes; limited CPU credits cause throttling; hardware is dedicated but CPU cycles are shared in burst models Azure B-series Earn credits when CPU is under baseline; spend credits to burst beyond Credits reset on VM restart, can accumulate up to a max; baseline varies by instance Google Cloud E2 Shared physical cores with allocation based on vCPU count vCPUs share physical CPUs dynamically; no concept of credits but CPU usage is limited and monitored

This variance means you cannot assume a one-size-fits-all metric approach. Instead, tailor your observations to account for how burstiness is controlled by your particular cloud.

Key Metrics to Record Before a Shared CPU Pilot

You must capture a baseline before switching your service to shared CPU. Collect these primary metrics in a monitoring system with at least 1-minute granularity over a representative window (more on the window below):

1. **Latency Metrics**
    - P95 and P99 response latency for all user-facing endpoints or APIs.
    - Spike duration: how long latency remains elevated beyond acceptable SLO thresholds.
2. **Queue Depth**
    - Length of work queues, pending requests, or job backlogs.
    - Spike patterns in queue length, especially during traffic bursts.
3. **Error Rates**
    - HTTP 5xx rates or equivalent service errors.
    - Correlate spikes or sustained periods of errors with CPU usage peaks.
4. **CPU Utilization Histogram and Percentiles**
    - Not just average CPU %, but full distribution, including P50, P75, P95, and P99 at minimum.
    - Duration and frequency of spikes above the baseline CPU committed by the instance.
5. **Memory Usage** – to confirm that memory is not a bottleneck during bursts.
6. **Network Throughput** – to ensure no sudden egress or ingress bottlenecks coincide with CPU usage.

Why the Observation Window Matters

Sampling metrics over a 24-72 hour period that includes peak traffic windows and maintenance cycles is essential. Short duration pilots (less than 1 business cycle or peak window) risk missing critical performance patterns. Also ensure that your monitoring captures high frequency spikes rather than smoothed averages.

How to Use AWS Compute Optimizer and Azure Advisor in Your Pilot

While manual metric collection is vital, leverage cloud-native recommendations to complement your analysis.

- **AWS Compute Optimizer:** After baseline collection, run Compute Optimizer to get instance right-sizing recommendations based on actual utilization, including CPU, memory, and networking. It can point out

underutilized instances that are prime for burstable instance conversion.

- **Azure Advisor:** Provides insights to optimize AWS-style cost and performance recommendations for VMs including B-series burstable VMs. It highlights potential VM size changes, performance bottlenecks, and cost opportunities.

However, both tools rely on average metrics and aggregated utilization data, so use their output as a starting point rather than final decision inputs. Always cross-check with your service's tail latency and error behavior.

Key Metrics to Record After Moving Services to Shared CPU

After you pilot on shared CPU, redo your measurements with identical metrics and observation computingforgeeks.com windows:

1. **Latency Metrics**
    - Watch for extended latency spikes especially if CPU credits or allocation limits begin throttling performance.
2. **Queue Depth and Backlogs**
    - Spikes here may indicate CPU throttling or insufficient compute headroom.
3. **Error Rates**
    - Coach out any elevated 5xx spikes or connection drops.
4. **CPU Utilization Percentiles**
    - Look for throttling signs: time spent at 100% CPU usage, CPU credits depletion (AWS), or sustained maxed out CPU window.
5. **Cloud Provider-specific Metrics:**
    - AWS: Monitor CPU credit balance and CPU credit usage metrics.
    - Azure: Monitor burst credits remaining and baseline utilization metrics.

Comparing Before and After Data

Your success criteria must include:

- At least no statistically significant regression in P95/P99 latency.
- Queue depth spike frequency and duration remain stable or improve.
- Error rates stay within normal baseline variance.
- No consistent depletion of CPU credits or bursting capacity during peak.
- Sum total cost savings justify performance tradeoffs — be comprehensive by including storage and egress cost impacts.

Common Pitfalls When Measuring Shared CPU Performance

Beware these traps that can skew your pilot observations:

- **Relying on Average CPU Utilization Only:** Average CPU can mask short but impactful CPU contention episodes. Focus on high percentile utilization instead.

- **Ignoring Burst Credit Metrics:** AWS and Azure provide burst credit stats that are essential to understand throttle risk — never ignore them.
- **Short Observation Windows:** Sampling only during low-traffic times or cache-warmed periods misrepresents burst behavior.
- **Assuming Shared CPU = Always Bad Uptime:** If planned carefully and measured, shared CPU workloads can maintain uptime and performance at great cost savings.
- **Omitting Storage and Egress Modeling:** Compute instance changes often influence storage IO patterns and network egress costs. Factor those into your ROI calculations.

Summary: The Engineering Checklist Before and After Shared CPU Pilots

Stage Metric / Action Why It Matters Before Pilot Collect P95 / P99 latency metrics Understand high-end latency experience, catch spikes Before Pilot Measure queue depth and spike duration Detect backlogs that correlate with CPU bottlenecks Before Pilot Track error rates Set baseline for failure rates under current config Before Pilot CPU utilization percentiles, not average only Reveal peak usage patterns that matter for bursting Before Pilot Use AWS Compute Optimizer / Azure Advisor Identify initial right-sizing and cost recommendations After Pilot Repeat latency, queue, error measurements with same windows Compare changes and detect regressions or improvements After Pilot Monitor CPU burst credit balances and exhaustion Assess if service may throttle and degrade during spikes After Pilot Review cloud-native advisor recommendations post-change Validate or refine ongoing cost/performance strategy After Pilot Calculate full cost savings including storage and egress Confirm ROI beyond compute price alone

Closing Thoughts

Shifting to shared CPU instances can be a cost-saving win when done correctly — but only if your pilot is guided by rigorous data capturing the right metrics over appropriate time windows. Avoid hand-wavy cost estimates or decisions based solely on average CPU utilization. Instead, leverage percentiles, measure spike duration, and always correlate with latency, queue, and error rates.

Cloud vendor tools like AWS Compute Optimizer and Azure Advisor are valuable allies in this process, but they’re no substitute for your application-specific performance data—especially tail latency.

The engineering discipline of recording rollback criteria, measuring impact holistically, and respecting provider-specific definitions sets you up for a successful transition that avoids end-user experience degradation while delivering tangible cost efficiencies.

Happy piloting!