When sizing cloud infrastructure for always-on, small services, picking the right instance type can be deceptively complex. Take https://computingforgeeks.com/shared-cpu-cloud-waste-migration-guide/ the Google **e2-small** as an example: its specs tout *2 vCPUs*, which on paper sounds like two full CPU threads. But is it really the equivalent of half a physical core sustained? Spoiler: not quite, and that discrepancy can hide cloud waste and lead to costly performance pitfalls if you don't dig deeper.

In this post, I'll break down:

- Why shared CPU definitions differ wildly between providers like Google, AWS, and Azure
- How to measure CPU usage correctly—not just average, but how spikes affect real-world performance
- The danger of ignoring 95th and 99th percentile CPU utilization in favor of averages
- Tools to guide right-sizing decisions, such as AWS Compute Optimizer and Azure Advisor

Understanding the Google e2-small's 50% Physical Core Share

Google's **e2-small** is advertised as having 2 vCPUs and 2 GB of memory, at a very low hourly price point, making it a tempting choice for lightweight workloads. But here's the catch: each e2 vCPU provides a *fractional share of a physical core*. In fact, an e2-small's vCPUs combined equate roughly to **half a physical core sustained**, not two fully dedicated hyperthreads.

This matters because the e2 family is built on *shared-core* technology. Rather than giving you dedicated cores or threads, Google splits physical cores' compute cycles among multiple instances dynamically. This means your workloads run on time-shared CPUs that can be interrupted or throttled based on overall host utilization.

Comparing Cloud Providers' Shared-Core Models

Provider Shared-Core Instance Examples CPU Allocation Performance Assumptions Google Cloud (E2) e2-small, e2-micro Fractional CPU shares summing to 0.5 - 1 vCPU sustained Variable, with dynamic time-sharing and potential bursting AWS T2.nano, T3.micro Defined baseline CPU credits with bursting capability Bursts when credits available; sustained CPU limited by credits Azure B1s, B2s CPU credits with baseline and burst capacity Bursts consume credits; baseline limits guaranteed compute

Notice that although AWS and Azure also use shared-core concepts, their models revolve around credit accumulation and bursting, whereas Google's E2 series is more dynamic with fractional shares, making effective CPU availability highly dependent on host load and co-tenancy.

Why Average CPU Utilization Metrics Are Misleading

One of my biggest pet peeves in cloud sizing conversations is basing decisions on average CPU usage alone. Imagine a microservice running on an e2-small that clocks 10% average CPU usage. Sounds cheap and under control, right? But what if that 10% average masks frequent short bursts to 70% or 90% utilization? Those spikes can impact latency, timeouts, and user experience severely.

Always ask __"What do the P95 and P99 CPU utilization percentiles look like?"__ The 95th or 99th percentile shows the CPU load during peak demand, not just the long tail of low usage. For shared-core instances like e2-small, these percentiles might reveal sustained, repeated bursts that exceed the true physical CPU share (half a core) and lead to throttling or queuing delays.

Measuring Spikes with the Right Observation Window

The width of your observation window matters too. Cloud monitoring tools like Google's Cloud Monitoring, AWS CloudWatch, or Azure Monitor often default to 5-minute or 1-minute granularity metrics. But very short spikes often last seconds, not minutes.

So, __use the finest-granularity CPU usage data available__, typically 1-minute or less, and consider integrating custom metrics with sub-minute sampling when possible. This helps to catch those spike durations that matter most for user-facing components.

Percentile Metrics Over Averages: Better Data-Driven Decisions

- **P50 (Median):** Shows typical load
- **P95:** Shows load during sustained bursts you can't ignore
- **P99:** Captures extreme peaks that affect tail latency

For small services on shared CPU, monitoring P95 and P99 gives you visibility into whether the workload is hitting limits of the allocated physical share, causing throttling or increased latency. Averages wash out these vital performance signals.

Lessons from AWS Compute Optimizer and Azure Advisor

Both AWS Compute Optimizer and Azure Advisor are free tools that analyze your usage metrics and provide recommendations for instance type resizing and cost savings. They embody the best practices of using percentile data and burst behavior analysis to tailor recommendations beyond just averages.

AWS Compute Optimizer

- Analyzes CPU, memory, disk I/O, and network metrics
- Uses P95 utilization to estimate if a resource is over or under-provisioned
- Considers burst credits for T-series instances before suggesting size changes
- Flags when sustained CPU usage may require moving off burstable instances to dedicated cores

Azure Advisor

- Monitors performance metrics and aggregates patterns over 14 days by default
- Incorporates burst credit models for B-series VMs
- Suggests rightsizing based on percentile peak usage—not averages alone
- Highlights cost optimization opportunities especially where always-on instances show low utilization

These tools reflect how cloud providers have acknowledged that simple “number of vCPUs” alone doesn’t tell the whole story. Always inspect how CPU credits, fractional core shares, and bursting affect your app’s performance and cost.

Always-On Small Services: The Unseen Cloud Waste

One subtle source of cloud waste is treating "small" consistently-on workloads as trivial. For instance, an e2-small might appear cheap and “small,” but if its 2 vCPUs are actually 50% physical core share, sustained work running near peak can either cause latency or force you to upgrade. Meanwhile, you waste money running oversized instances that don’t improve tail performance.

Rarely do workloads sit at average utilization all day. Instead, many services—APIs, internal tooling, background services—experience spike-driven loads. Ignoring this pattern leads to:

- Overprovisioned instances wasting money on idle cores
- Underprovisioned instances causing throttling and slower processing
- Hardware-level resource contention between noisy neighbors on shared cores

Don’t fall into the trap of assuming 2 vCPUs means “twice the performance of 1 CPU.” Instead, analyze your workload’s actual behavior and adjust instance selection accordingly.

Guidelines for Evaluating Shared-Core Instances Like Google e2-small

1. **Start with workload profiling.** Use fine-grained monitoring to identify CPU utilization spikes (P95, P99) and duration. Don't trust average CPU alone.
2. **Understand the CPU share model.** Research your provider's CPU sharing or burst credit system. With Google e2-small, expect roughly half a physical core across 2 vCPUs.
3. **Benchmark under expected peak load.** Run load tests to confirm actual CPU throttling or latency impact during spikes.
4. **Run pilot migrations.** Deploy a test fleet with different instance types to observe real-world behavior before committing.
5. **Use cloud advisory tools.** AWS Compute Optimizer and Azure Advisor provide actionable insights, but also complement with your own monitoring.
6. **Define rollback criteria in advance.** For example, roll back if P99 CPU spikes exceed 70% for more than 5 minutes or if request latency increases.

Summary

\u201c2 vCPUs\u201d on a Google e2-small doesn't mean "two half-physical cores" sustained equally as you might expect. The truth is that these are fractional, time-shared CPU slices equating roughly to **50% of a physical core** aggregated across two threads under shared-core scheduling.

Ignoring this nuance leads to either cloud waste or poor performance if the peak demand patterns aren't observed correctly. Always measure with appropriate *percentiles* (P95, P99), use fine-granularity monitoring to catch spike durations, and consult provider-specific optimizer tools for data-driven scaling recommendations.

With shared-core instances like Google e2-small, never treat vCPU counts as performance guarantees. Instead, focus on how sustained workload peaks align with physical CPU share, and adjust instance sizing accordingly for your always-on small services.

Author: A 12-year cloud infrastructure and SRE practitioner specializing in cost and performance optimization across AWS, Azure, and Google Cloud environments.