readyset

How the AI Buildout Is Repricing the Rest of Your Infrastructure.

A briefing for infrastructure leaders.

Published by **Readyset**

Strategic analysis, 2026

Contents

	Executive summary	2
01	The demand shock	2
02	The supply squeeze	3
03	The canary on the retail shelf	5
04	The next domino: managed infrastructure	6
05	The hedge: engineer for efficiency now	8
	The bottom line	11
	Sources	13

Executive summary

The AI story most people are pricing into a 2027 budget is about capability. The one that will land on the invoice sooner is financial: what the AI buildout is doing to the cost of ordinary compute and memory, the resources that run the rest of the business in the background.

At the time of writing, three things have already happened:

- **April 2026 — Microsoft.** CFO Amy Hood told investors that $25 billion of the company's record $190 billion 2026 capital-expenditure plan was attributable to higher component prices, and that Microsoft expected to remain capacity-constrained on GPUs, CPUs, and storage through at least the end of the year.[1]
- **Q1 2026 — TrendForce.** The firm revised its forecast for conventional DRAM contract prices upward from 55–60 percent QoQ to 90–95 percent, and reported that 1Q26 DRAM industry revenue rose about 81 percent QoQ.[2]
- **January 2026 — AWS.** The company raised the price of its EC2 Capacity Blocks for ML by roughly 15 percent, widely described as the first list-price increase in the service's roughly two-decade history.[3]

The pressure is concentrated in memory, and the database is where memory pressure hits an infrastructure bill first, because a database instance is largely a memory-and-compute unit that runs continuously, scales by replication rather than by efficiency, and is the part of the stack most teams have the least visibility into.

> For close to two decades, the safe assumption in infrastructure planning was that cloud unit costs would fall every year. That assumption no longer holds, and analysts do not expect meaningful new memory-manufacturing capacity to come online before 2027 or 2028. The cheapest server is the one you do not have to buy, and most organizations sit on more unclaimed efficiency headroom than they realize.

This briefing traces how the AI buildout pushes cost into ordinary infrastructure, why the database absorbs it first, and which fixes pay back at today's prices, let alone next year's. Nearly all of them require the same starting point: a clear view of what your database does at a query-by-query level and what it costs to do it.

01 The demand shock

Consumption is compounding

The comforting narrative for the past two years has been that AI gets cheaper over time. On a per-token basis that is largely true, and it is exactly why total spend is exploding. When a resource gets cheaper, organizations find more uses for it and total spend grows.

Uber is the clearest early data point:

- **December 2025 — Rollout.** The company rolled Claude Code out to roughly 5,000 engineers.
- **February to March 2026 — Adoption.** Usage climbed from 32 percent of engineers to 84 percent, and by spring roughly 95 percent were using AI tools monthly, with about 70 percent of committed code originating from those

tools.[4]

- **April 2026 — Budget exhausted.** Four months in, Uber had burned through its entire 2026 AI coding-tools budget, and CTO Praveen Neppalli Naga told The Information the company was back to the drawing board on its assumptions.[4] Reported costs per engineer ran between $500 and $2,000 a month on power users.[4]

Uber has a real engineering budget and real financial discipline. This is one of the early indications that even a sophisticated finance organization struggled to model token-metered AI consumption at engineering scale. Gartner projects AI agent software spending to reach nearly $207 billion in 2026, up more than 139 percent from an $86.4 billion 2025 base.[5]

It is evident that we are underestimating the demand for AI for enterprises. A 2027 budget should treat AI as a compounding input cost, one that keeps pulling on the same finite pool of physical infrastructure as the rest of the business. Per-unit pricing keeps falling and total bills keep climbing. This is Jevons paradox: when a resource gets cheaper to use, consumption rises faster than the price falls, and total spend goes up. Every efficiency gain at the model layer is spent on more inference, not less hardware.

02 The supply squeeze

The buildout is consuming the memory supply

To meet AI demand, the largest cloud providers are spending at an unprecedented scale, and concentrating that spend on a small number of suppliers. Amazon, Microsoft, Google, and Meta have collectively guided to roughly $725 billion in 2026 capital expenditure, up about 77 percent from 2025's record of ~$410 billion.[6] Microsoft's CFO Amy Hood attributed roughly $25 billion of Microsoft's $190 billion 2026 plan to higher component prices alone, and told investors Microsoft expected to remain capacity-constrained on GPUs, CPUs, and storage through at least the end of 2026.[1]

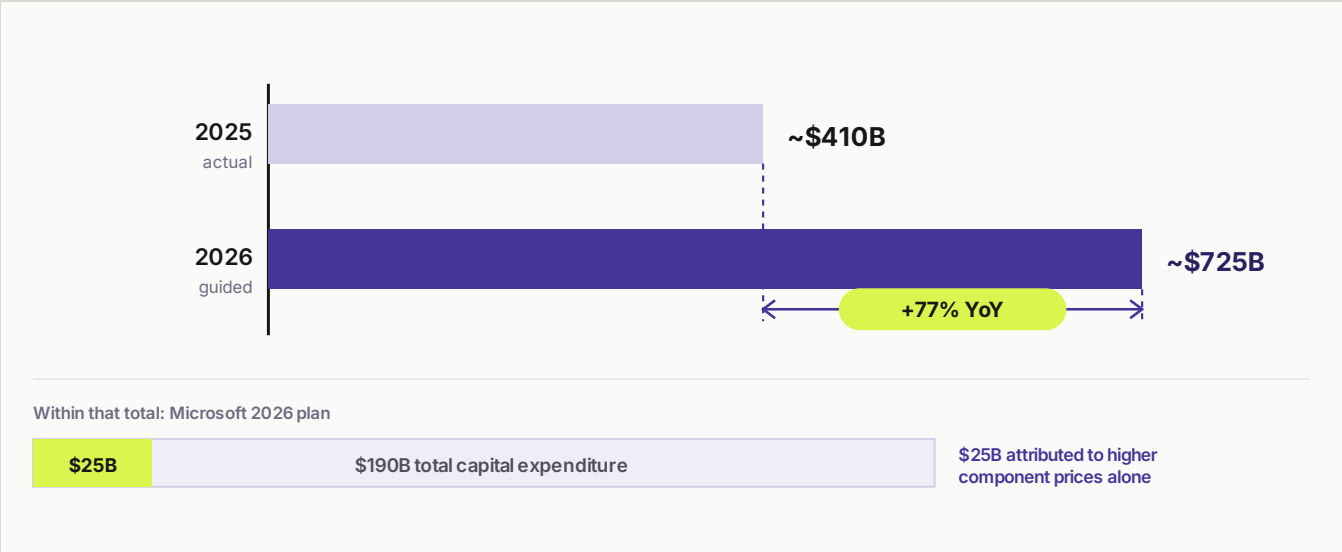


Figure 1. Combined 2026 capex from the four largest hyperscalers is up ~77 percent year over year, with Microsoft alone attributing ~$25B to component-price increases.

When the companies operating the cloud are themselves rationing capacity and passing through cost increases, the pressure reaches customers on a timeline the customer does not control.

The constraint is physical, and it is concentrated in memory

The mechanism is the memory supply chain. High-bandwidth memory (HBM) for AI accelerators consumes far more wafer capacity per bit than the conventional DRAM used in a database server, and carries far higher margins, so manufacturers have reallocated capacity toward it. TrendForce revised its Q1 2026 conventional DRAM contract price forecast from 55-60 percent QoQ to 90-95 percent, projected server DRAM up around 90 percent QoQ, and reported PC DRAM prices rising more than 100 percent QoQ, all records for the categories.[2]

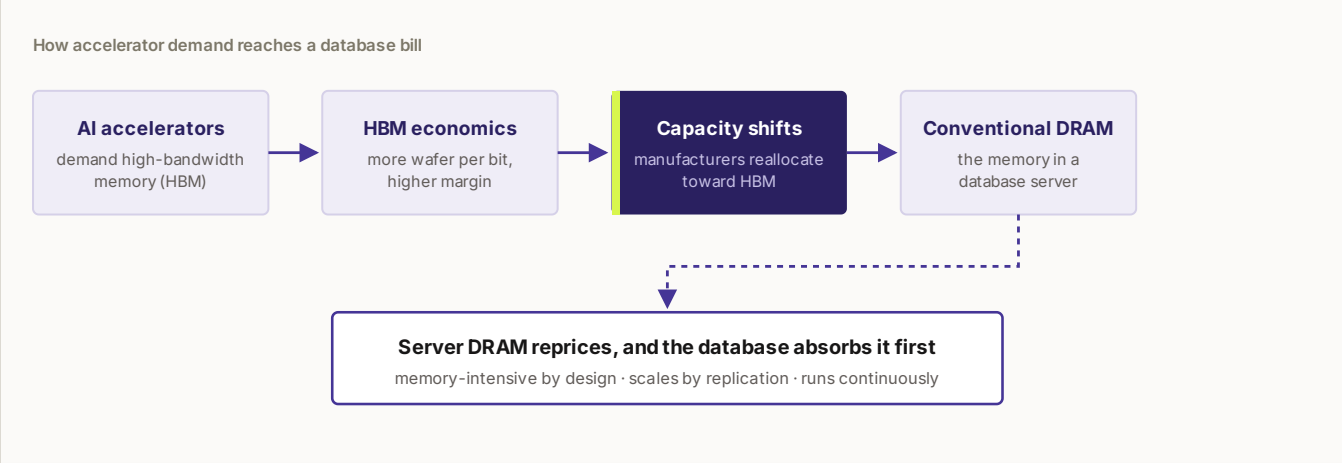


Figure 2. The mechanism behind the price move: accelerator demand pulls wafer capacity toward high-bandwidth memory, which squeezes the conventional DRAM a database server runs on.

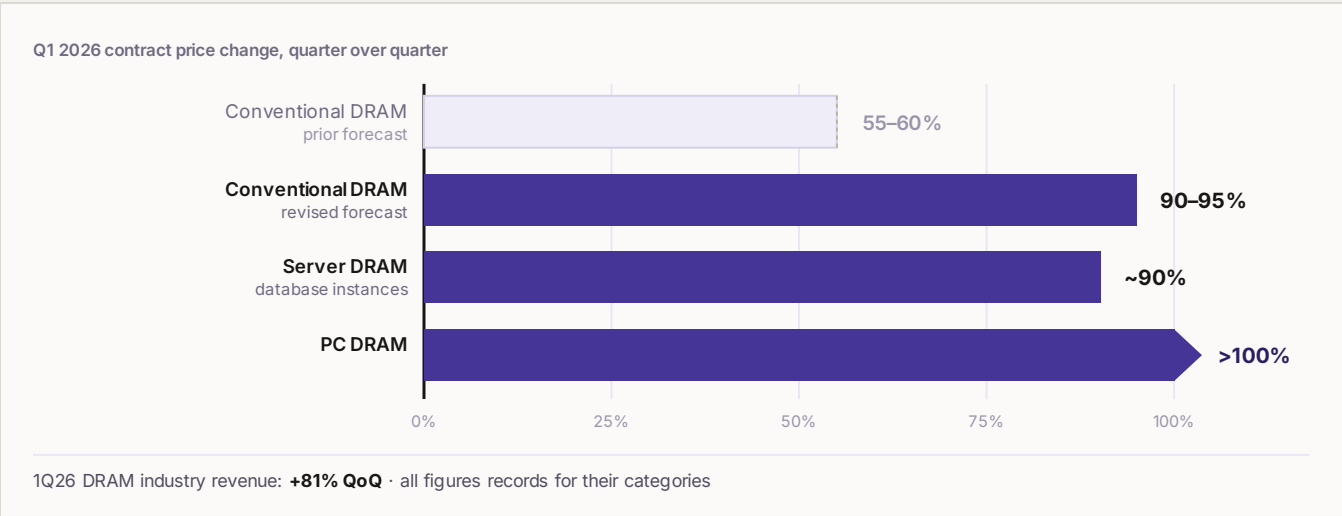


Figure 3. Q1 2026 DRAM contract prices reset across every category, driven by AI-server demand crowding out ordinary supply.

This shortage is driven by demand growth. The 2021 memory crunch resolved once factories reopened and logistics normalized; this one persists as long as AI demand keeps growing faster than supply can catch up. Analysts put 2026 DRAM and NAND supply growth at just 16 to 17 percent, below the historical norm, with meaningful new capacity not online until 2027 or 2028.[7]

03 The canary on the retail shelf

The bill is already arriving in price-transparent markets

The clearest way to see what AI-driven memory inflation does to a normal customer is to look at a market with no enterprise contracts to hide behind. Consumer electronics is that market. In the first half of 2026, the warnings turned into prices.

On June 17, 2026, Apple CEO Tim Cook told the Wall Street Journal that the company would raise prices to offset surging memory and storage costs.[8]

> "Unfortunately, price increases are unavoidable. We've been trying to shield our customers from the increases, but the situation has become unsustainable."
>
> **Tim Cook**, CEO, Apple. Wall Street Journal, June 17, 2026.

Cook, in his final weeks as CEO after more than 40 years in supply-chain roles, described the memory shortage as a "hundred-year flood."[8] TechInsights estimated the cost of memory and storage in an iPhone 17 Pro at roughly $50 last year, and projected the same components for the iPhone 18 Pro at about $200.[8]

Apple runs one of the most disciplined supply chains in the world and has for years absorbed cost shocks that other manufacturers passed through. When Apple says it can no longer absorb them, the signal is hard to dismiss.

A week earlier, on June 10, new Xbox CEO Asha Sharma published a memo on Xbox Wire titled "Next 100 Days: Xbox Reset" with an unusually specific disclosure.

> "We are in a hardware component crisis. When I joined as CEO in February, the price we paid for console storage components was over 2x as high as we paid last fall. These costs have since doubled again. And as we plan for the 2027 holiday season, we expect another significant increase, taking us over 5x the prices we paid only two years earlier. Memory costs have followed a broadly similar trajectory."
>
> **Asha Sharma**, CEO, Xbox. "Next 100 Days: Xbox Reset," Xbox Wire, June 10, 2026.[9]

The company most publicly attributing its component-cost pressure to the AI buildout is also helping drive that demand. Azure and Xbox ultimately compete for the same constrained memory supply, and Azure's purchasing scale is winning.[9]

Sony and Valve also raised hardware prices during 2026 despite no meaningful hardware changes.[10] Consumer devices, enterprise servers, and cloud databases all draw from the same constrained memory supply. Consumer hardware reprices first because it lacks the long-term contracts and margin cushion that let enterprise infrastructure absorb the shock.

04 The next domino: managed infrastructure

The parts of the cloud bill most exposed have not fully repriced yet

Cloud providers typically run a lag of three to six months between a change in procurement costs and a change in retail pricing, so the memory-cost shock that hit component suppliers in late 2025 and early 2026 points to pass-through hitting managed services somewhere in mid-2026 and continuing through the year. That timing is an estimate and worth stating as one, but the direction is not.

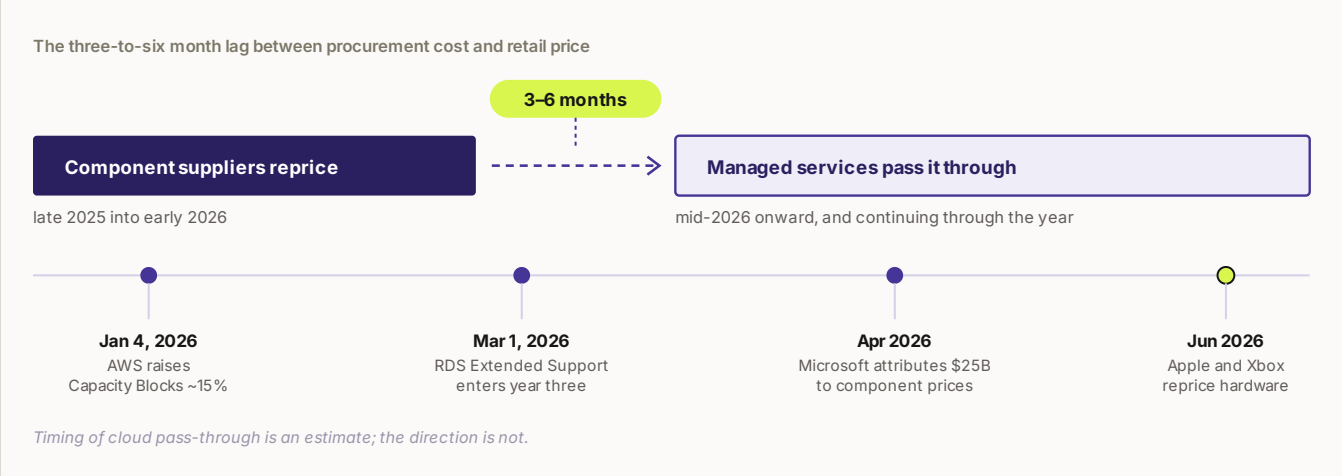


Figure 4. Procurement costs and retail prices move on different clocks. The dated events across 2026 fall either side of that lag.

One signal has already broken the pattern. On January 4, 2026, AWS raised the price of its EC2 Capacity Blocks for ML, the reserved GPU capacity used for machine-learning training, by roughly 15 percent.[3] The p5e.48xlarge instance went from $34.61 to $39.80 per hour in most U.S. regions, and the change flowed straight through to customers on enterprise discount agreements because those discounts are calculated against list prices.[3] AWS reduced prices 134 times between its 2006 launch and September 2023, by its own count.[15] A company with that track record raising prices on a category as strategically important as GPU compute is a signal that the economics underneath the cloud have changed. Microsoft's own guidance names CPUs and storage alongside GPUs as capacity-constrained, so non-GPU workloads sit on the same curve.

A cost that has already landed on database teams

Alongside the broader trend, there is a more immediate increase that has already hit many MySQL and PostgreSQL teams. AWS RDS Extended Support keeps older major versions patched after community end of life for a per-vCPU-hour surcharge on top of normal instance costs. The surcharge starts at $0.100 per vCPU-hour for the first two years, and in year three it doubles to $0.200 per vCPU-hour.[11] MySQL 5.7 and PostgreSQL 11 both entered year-three pricing on March 1, 2026, so every team still running those versions saw their surcharge double on that date, and the two billing items are independent, so a multi-year reserved-instance commitment provides no offset against it.[11]

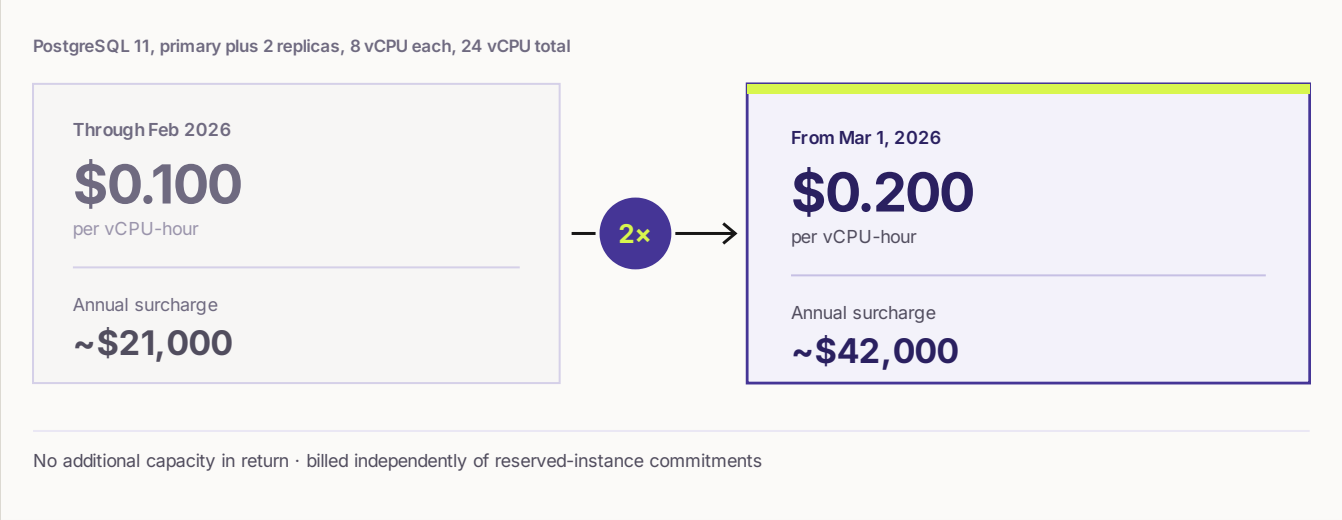


Figure 5. A modest PostgreSQL 11 fleet (primary + 2 replicas at 8 vCPUs each) saw its Extended Support surcharge double overnight on March 1, 2026, adding roughly $42,000 a year with no additional capacity in return.

This example inverts the old logic. Under falling prices, deferring the upgrade was defensible: the migration carried risk and engineering cost, and the surcharge was a known, bounded line item that might even shrink in relative terms as the rest of the bill fell. Under the year-three schedule, that bounded line item just doubled, and the migration that looked expensive last year now pays for itself in months. The price of waiting has changed even though the team's situation has not.

Why the database sits closest to the pressure

The database is typically among the largest infrastructure line items and among the most exposed to rising costs, for three compounding reasons. It is memory-intensive by design, which puts database instances in the exact category absorbing the steepest increases right now. It scales by replication, which turns every read-load problem into another full instance carrying continuous memory-and-compute cost, whether or not that instance is doing useful work. And it is hard to see into: most teams cannot name the queries driving their load, so the default response to a performance problem is to add capacity rather than diagnose the cause, and rising prices make that default more expensive every quarter.

Replicas remain the right tool for read-after-write isolation, geographic distribution, and high-availability failover. The question is whether the next replica is solving a capacity problem or paying to re-run the same reads on a more expensive base without anyone knowing.

The same pressure hits any memory-heavy layer sitting alongside the database. Teams running Redis or another in-memory cache in front of their database are provisioning two memory-intensive systems, each sized for the moment the other one is unavailable. The database must handle full traffic when the cache cold-starts or evicts under memory pressure; the cache must hold the working set plus headroom for spikes. Both are drawing from the same constrained DRAM supply, and both are repricing at the same time. Teams that already have a caching layer and are watching its bill rise are hitting a version of the same problem this briefing describes, one instance deeper into the stack.

Reasonable objections

The counterarguments are real. Hyperscalers may engineer their way through part of this pressure, additional fab capacity is expected in 2027-2028, and AI demand could moderate. None of those possibilities changes the near-term calculus. Every action this briefing recommends is justified by current prices alone. If the pressure eases, you've cut costs. If it doesn't, you've cut costs and hedged the increase.

Three more objections are worth taking seriously.

> **"Prices have spiked before and come back down."**
>
> True, and worth weighing, but the difference this time is the driver. Past fluctuations were largely cyclical; the current pressure traces to a structural reallocation of manufacturing capacity toward AI and a buildout the largest buyers have committed to for years.[6] Prices may soften, but planning on a return to steady annual deflation is betting against the structural signals.

> **"We are on reserved instances, so we are insulated."**
>
> Reserved instances help with the compute they cover, but they do not cover everything. The RDS Extended Support surcharge is billed independently of reservations,[11] and enterprise discount agreements are calculated against list prices, so a list-price increase flows through.[3]

> **"The database is a small part of our spend."**
>
> For some teams that is accurate. For many it is the opposite: the database is one of the largest infrastructure line items, precisely because it is memory-heavy, runs continuously, and scales by adding more full instances. The teams that know what their data layer costs, line by line, are positioned to decide whether it is worth attention; the ones who assume tend to find out from the bill.

05 The hedge: engineer for efficiency now

Finding a cheaper provider will not help, because the major providers all buy from the same constrained suppliers. The response to a multi-year input-cost shock is to need less of the constrained resource per unit of business.

Move one: reclaim the compute you already pay for

Most organizations run on a decade of "provision for peak, round up to be safe" habits that were invisible when compute was cheap. Those habits haven't changed but the cost of holding them has. In its 2026 State of Kubernetes Optimization Report, Cast AI reported that across tens of thousands of production Kubernetes clusters, average CPU utilization was just 8 percent and memory utilization 20 percent, and both fell year over year, while CPU overprovisioning had climbed to 69 percent and memory overprovisioning to 79 percent.[12]

Right-sizing, bin-packing, and autoscaling down are standard practice. They reclaim the overprovisioned capacity most organizations already pay for.

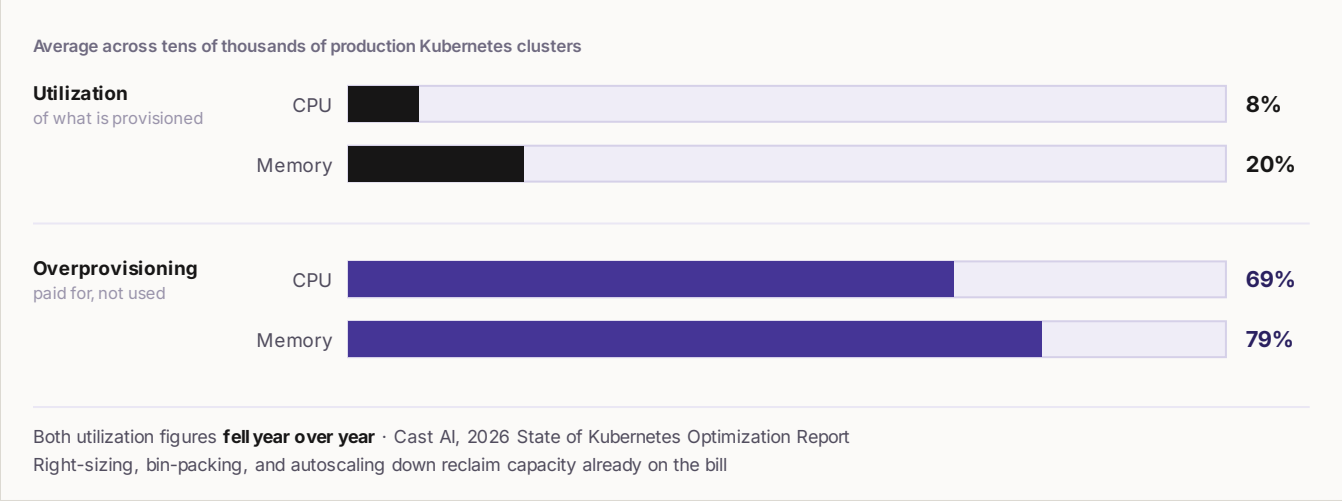


Figure 6. Average production Kubernetes clusters use a small fraction of the CPU and memory they provision. In an era of falling prices this was a rounding error; at current prices it is the largest single cost-reduction opportunity most companies have.

Move two: stop scaling the database, cache in front of it

If memory-intensive services are where cloud inflation bites hardest, the database is where most companies are most exposed, because the default response to a slow or overloaded database is the most expensive possible move in this environment: scale up to a larger, memory-heavier instance, or add another read replica. The direct alternative is to reduce the load that reaches the primary in the first place. Most application database traffic is reads, and a large share of those reads are repetitive: the same dashboard rollups, the same list endpoints, the same aggregate counts, executed thousands of times against data that changes far less often than it is read.

Serving those repeating reads from a cache layer instead of the primary breaks the scale-up reflex. It absorbs read traffic on cheaper infrastructure, flattens load spikes, and often eliminates the need to buy a bigger instance or another replica. The distinction from a replica is precise: a replica answers a read by running the query again somewhere else, while a cache answers it by returning a result it has already computed. For the queries that dominate read load, the second is both faster and dramatically cheaper.

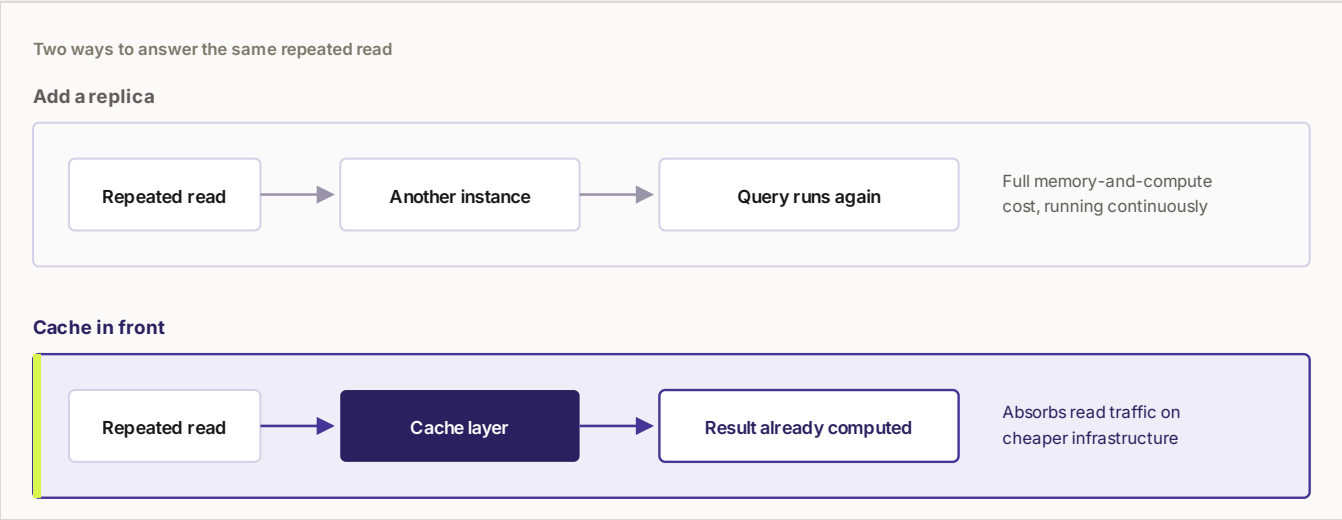


Figure 7. A replica answers a repeated read by running the query again. A cache answers it by returning a result it has already computed.

A caching layer worth having decides on its own which queries are worth accelerating and how fresh each one needs to be. It also protects the working set from one-off traffic that would evict useful entries, and puts memory where it buys the most savings. Those are the same decisions a team would otherwise make by hand, quarter after quarter which is what turns caching from an operational tax into a cost lever.

Those decisions also require two views that live below the application layer. One is a picture of query traffic across every application sharing the cache, so the layer can weigh which reads are hot in aggregate rather than per-service. The other is a picture of what the underlying database is doing, so the layer can tune freshness and eviction against actual load rather than against a guess. A caching layer sitting in the database's query path has both views by default, while an application-level cache (e.g. Redis or Valkey) is working from neither.

Teams historically chose replicas over caching because application-level caches required manual invalidation and ongoing maintenance. Modern caches that stay current from the database's own replication stream remove much of that burden.

Questions worth asking of any caching approach

Any caching layer will demo well. The questions below separate approaches that reduce database load from approaches that add a maintenance tax on the same team already managing the database.

1. **Can it see the full picture, across applications and inside the database?** A layer with visibility into both which reads are hot across the whole application footprint and what the underlying database is doing can make caching decisions on evidence rather than on guesses. A layer without that view is running each application in isolation and each cache decision without a load signal.
2. **Does it require application code changes?** Every line of cache-management code the team writes is a line the team maintains forever.
3. **Who owns invalidation, you or the system?** Keeping the cache correct is where most projects run into trouble.
4. **Does it understand queries, or just store values?** A layer that speaks SQL can cache and refresh results intelligently; a key-value store hands the hard part back to the team.
5. **Do results stay correct as the underlying data changes?** Caching that silently goes stale is worse than no caching, because it looks like it is working.
6. **Does it speak the database's wire protocol?** If it does, adoption is a connection-string change. If not, it is a migration.
7. **Does it fail open?** When the cache has a problem, do queries fall through to the database, or fail? The right answer makes the database the floor.

The test across all six is whether the layer removes work from the team or just relocates it.

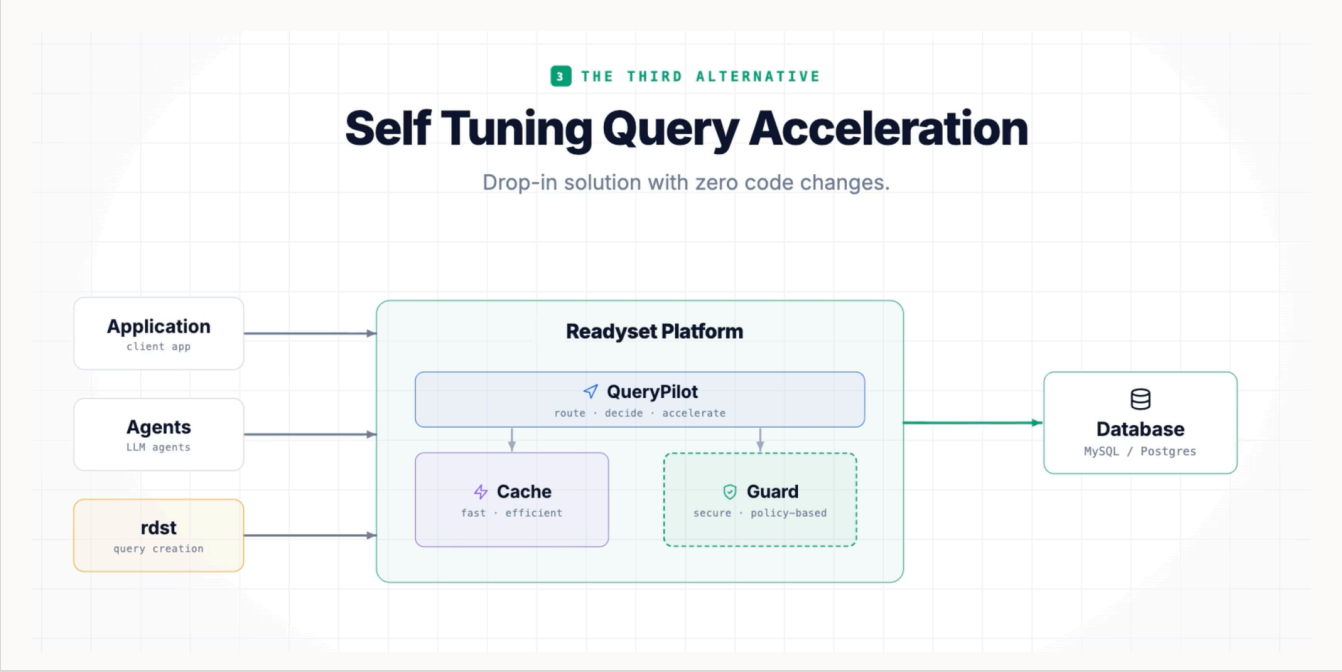


Figure 8. Readyset Platform sits in the read path. Application, agent, and rdst traffic arrives at QueryPilot, which routes each query to cache or through to the database.

What this can look like in practice

In one field test, routing real GitLab traffic through Readyset Platform on Azure Database for PostgreSQL let the cacheable reads run 5x to 26x faster from cache. With those reads served from memory, the Postgres tier dropped from eight vCores to two at the same throughput, taking monthly compute from $593 to $226, a 62 percent reduction, or about $4,400 a year including the cost of the caching instance itself.[13]

The same pattern holds at the top of the scale. The Brazilian credit intelligence company Lemit had exhausted vertical scaling, read replicas, and in-memory tables on heavily tuned MySQL hardware and still plateaued at roughly 13,000 reads per second, with reads competing with writes for the same resources. Routing the hot reads through Readyset pushed the stack past 109,000 queries per second, an 8x increase in read throughput, with cached reads served in under a millisecond and no application code changes.[14]

The bottom line

For two decades the rational strategy was to assume costs would keep falling and optimize for velocity over efficiency. That era is on pause, for at least the next two to three years, until semiconductor supply stabilizes around 2027 or 2028. The biggest near-term cost of the AI boom is what the buildout is doing to the price of everything else the business runs.

This is a joint mandate for the CTO and the CFO, and it does not require betting the roadmap. Measure the exposure. Reclaim the easy headroom, right-sizing, bin-packing, and real downscaling are weeks of work and pay back at current prices, let alone next year's. Before the next “we need a bigger database” decision, ask whether a caching layer removes the need entirely.

What to revisit in the next budget cycle

- Look for double-provisioning at the data layer. Teams running Redis or another in-memory cache in front of the database are usually sizing both for peak, which means the memory bill on both sides is now moving with the same pressure.
- Check for RDS Extended Support charges today. If any database is running MySQL 5.7, PostgreSQL 11, or another version past standard support, the year-three surcharge may already be on the bill.
- Re-run the upgrade math at current rates. An upgrade that looked too expensive last year may now have a much shorter payback period.
- Inventory replicas by actual utilization, not by count. Each one sized for a peak that lasts an hour or two is a full instance carrying continuous, rising cost.
- Pull utilization on the general fleet. If Kubernetes clusters look like the industry average, the team is paying for roughly five times the compute it uses, and right-sizing is weeks of work rather than quarters.
- Find the queries driving the read load. Until the team can name them, the default answer to a performance problem will keep being more capacity, which is the most expensive answer under rising prices.
- Re-examine any plan built on falling unit costs. Multi-year models that assumed this year's capacity would be cheaper than last year's are pointed the wrong way.

> The teams positioned to handle rising prices are the ones already measuring where the spend goes, questioning which work needs to keep running, and cutting the parts that do not.

Sources

> *Every figure in this briefing is drawn from a primary or named source, listed below. Forward-looking figures, capex forecasts, DRAM prices, and supply-growth estimates are drawn from sources that themselves vary; ranges are noted where estimates differ, and the direction of travel is what matters here. The timing of cloud pass-through is necessarily an estimate.*

1. Microsoft, Fiscal Year 2026 Third Quarter Earnings Conference Call transcript (April 29, 2026); CNBC, "Microsoft calls for $190 billion in 2026 capital spending on soaring memory prices" (April 30, 2026).
2. TrendForce, "Memory Price Outlook for 1Q26 Sharply Upgraded" (Feb 2, 2026); "Rapid Contract Price Surge Drives 1Q26 DRAM Industry Up 81% QoQ" (Jun 1, 2026).
3. The Register, "AWS raises GPU prices 15% on a Saturday" (January 5, 2026).
4. The Information, "Uber CTO Shows How Claude Code Can Blow Up AI Budgets" (April 2026); Forbes, "Uber Burns Its 2026 AI Budget In Four Months On Claude Code" (May 17, 2026).
5. Fortune, "Uber burned through its entire 2026 AI budget in four months" (May 26, 2026), citing Gartner.
6. Yahoo Finance / Financial Times / Tom's Hardware, "Google, Microsoft, Meta, and Amazon capex spending to hit $725 billion in 2026" (April 30, 2026).
7. IDC and Counterpoint Research forecasts as reported by SoftwareSeni, "When Will DRAM Prices Normalise?" (January 2026).
8. Wall Street Journal, interview with Apple CEO Tim Cook (June 17, 2026), as reported by MacRumors, ABC News, CNBC.
9. Microsoft / Xbox Wire, "Next 100 Days: Xbox Reset" memo, Asha Sharma and Matt Booty (June 10, 2026), as reported by VideoCardz and Game Developer.
10. Reuters and Sony public disclosures (2026); Valve Steam Deck price increase, hardware unchanged (May 2026).
11. Amazon Web Services, Amazon RDS for MySQL pricing and Amazon RDS for PostgreSQL pricing.
12. Cast AI, 2026 State of Kubernetes Optimization Report (April 2026).
13. Readyset, "Readyset on Azure Postgres: Real GitLab CE Field Test."
14. Readyset, "Lemit Reaches 100K QPS on MySQL with Readyset."

This brief synthesizes public reporting and analyst data available as of publication. Intended as strategic analysis, not financial, investment, or procurement advice. © 2026 Readyset, Inc. All third-party marks belong to their respective owners.