SCALE WITHOUT SURPRISE

Visual Companion

Understanding Cost, Capacity, and Capability

Timothy O'Brien

A shorter route into the book

The main ideas, selected explanations, practical decisions, and chapter diagrams. Follow the links when you need the full argument, worked examples, and qualifications.

Draft edition. Preliminary chapters remain preliminary here.

https://withoutsurprise.com/briefs

Copyright © 2026 Timothy O'Brien. All rights reserved.

Contents

What Computing Costs 3

Capacity Is a Decision 11

The Players and the Vocabulary 14

The Simplest Practice 17

Knowing Your Workloads 20

Built-In Tools 23

AI Services and Token Capacity 26

Measuring AI Workloads 34

Orchestration and Agentic Topologies 41

Specialized Hardware 44

Exotic Hardware: Quantum and Emerging Capacity 48

Databases and Data-Bound Technologies 59

Measuring Value 62

CHAPTER 1 / 4 MINUTE BRIEF

What Computing Costs

Read the full chapter → / Read this brief online →

The main idea

Go back to the case from the introduction: a demonstration has turned into a launch, the service can absorb the new work only while nothing fails, and the additional capacity won't arrive in time. One of Finance's first questions is why the resources we already pay for can't do more of the work. It's a fair question. Before we can answer it, or ask anyone to approve a purchase, we need to say what the organization pays for today, what it has already promised to keep paying, and what would actually change if we did something different.

Explore this section in the full chapter →

Knowing what you consume before you change it

Sometimes an application will stop using resources the organization has already committed to paying for. Releasing them may let another team use them or avoid a purchase later, but it does not cancel the payment. Every proposed change has to be followed through to the obligation before anyone calls it a reduction.

Explore this section in the full chapter →

Several dimensions of the same dollar

Fixed, variable, committed, metered: these are useful words, but trouble starts when we treat them as four separate boxes and assume that placing an expense in one tells us how to manage it. A single cloud product might charge one component as fixed overhead and another by usage. Consumption can be metered whether it is covered by a commitment or billed at an on-demand rate. A service whose usage varies can still carry a minimum payment. The label alone does not tell us how the charge will respond to a change in use.

Explore this section in the full chapter →

Then there is **attribution**, which connects an expense to the workload or decision that caused it. The easy version is to tag resources to the teams that use them and charge those teams, but many costs are shared. Some organizations spread them evenly across the company and call it a *peanut butter spread*; management accountants know it as peanut-butter costing and warn that it overcharges light users and undercharges heavy ones.[1] Other shared costs go to the groups that use a particular service, and some are simply overhead. Attribution also gets political, because who pays is a question about who is answerable. It tells us who is consuming resources and where to start a conversation, not what the enterprise would save if that consumer went away, and shifting a cost between teams is surprisingly easy to mistake for reducing it.

Explore this section in the full chapter →

[1] Charles T. Horngren, Srikant M. Datar, and Madhav V. Rajan, *Cost Accounting: A Managerial Emphasis*, 15th ed. (Pearson, 2015), chapter 5, "Activity-Based Costing and Activity-Based Management": broad averaging, or peanut-butter costing, spreads resource costs uniformly across products or services that use those resources non-uniformly, overcosting some and undercosting others. The application to shared cloud costs is the author's.

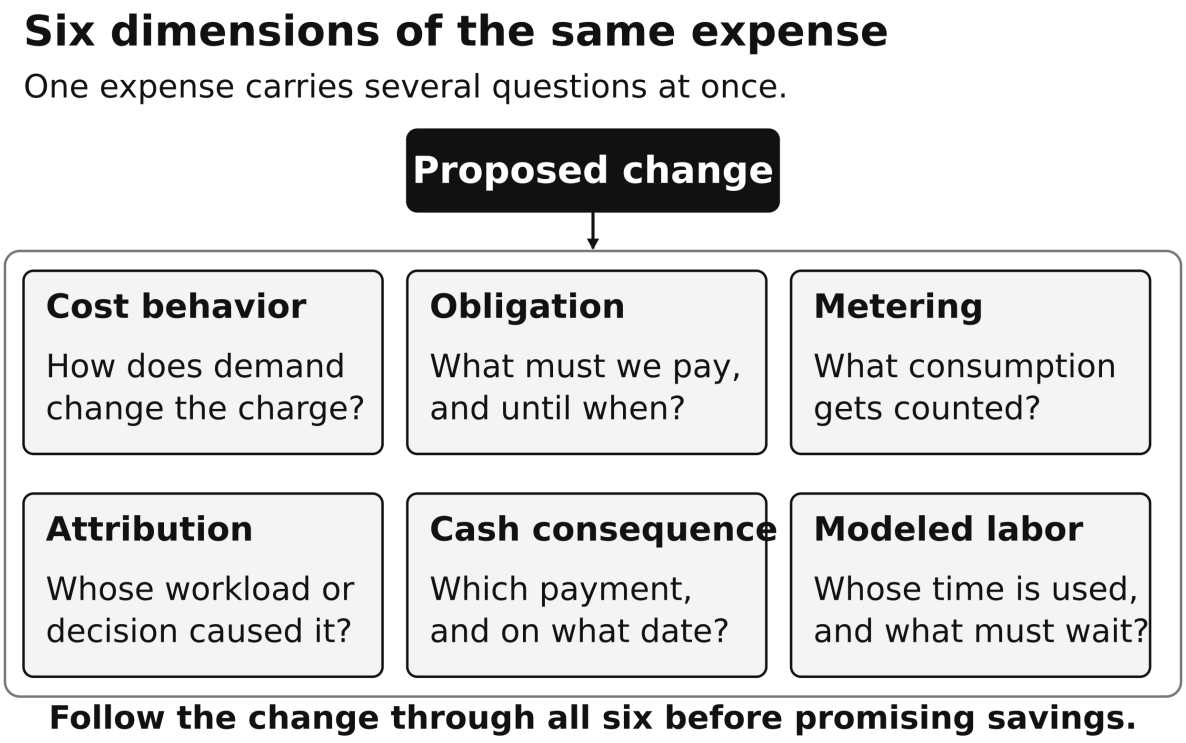


Diagram 1.1. Every expense carries overlapping dimensions: how its charge behaves, what is owed, what is counted, who caused it, which payment changes, and whose time is consumed. These questions must be answered together.

Explore this section in the full chapter →

A small number of teams dominate cost

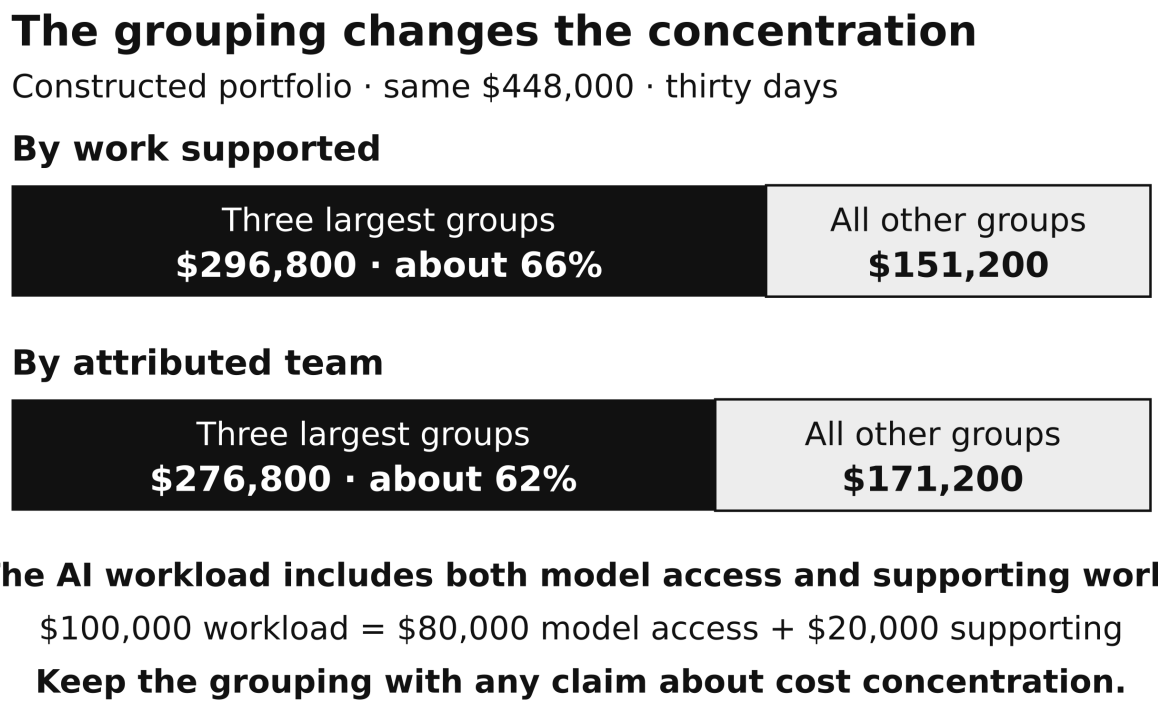


Diagram 1.2. The three largest workloads account for $296,800 of the $448,000 bill, about 66 percent; the three largest attributed teams account for $276,800, about 62 percent. The difference is the AI workload's supporting work, which belongs to other teams. Keep the grouping with the concentration percentage.

Explore this section in the full chapter →

How the charges behave

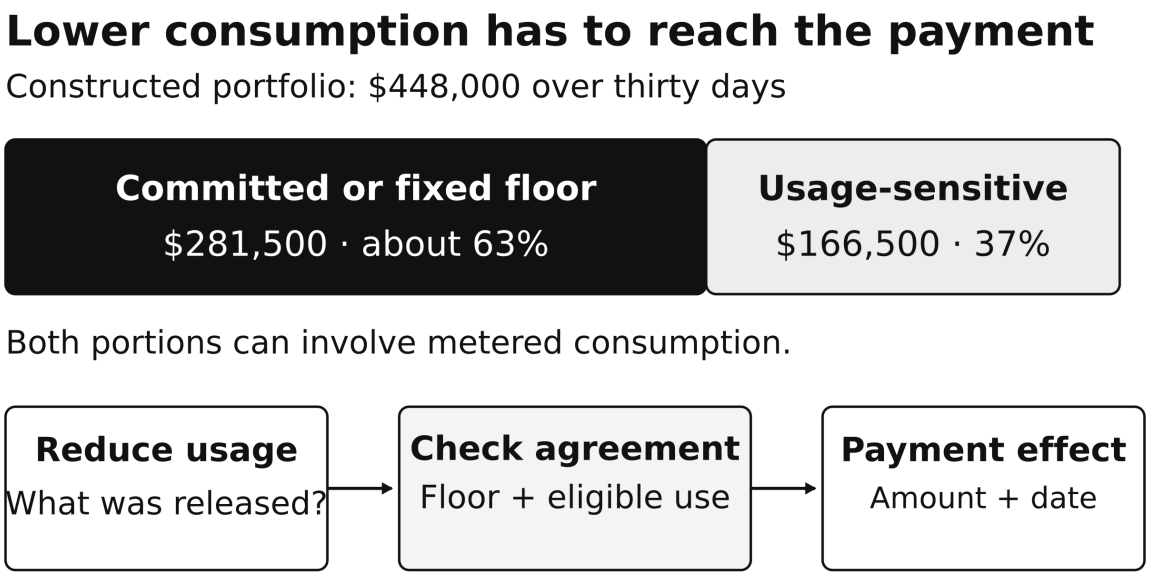


Diagram 1.3. In the constructed thirty-day portfolio, $281,500 is a committed or fixed floor and $166,500 is usage-sensitive. Reducing consumption must be followed through the agreement before it becomes a payment reduction; seasonality is a separate property of demand.

Explore this section in the full chapter →

Follow a change across the boundary

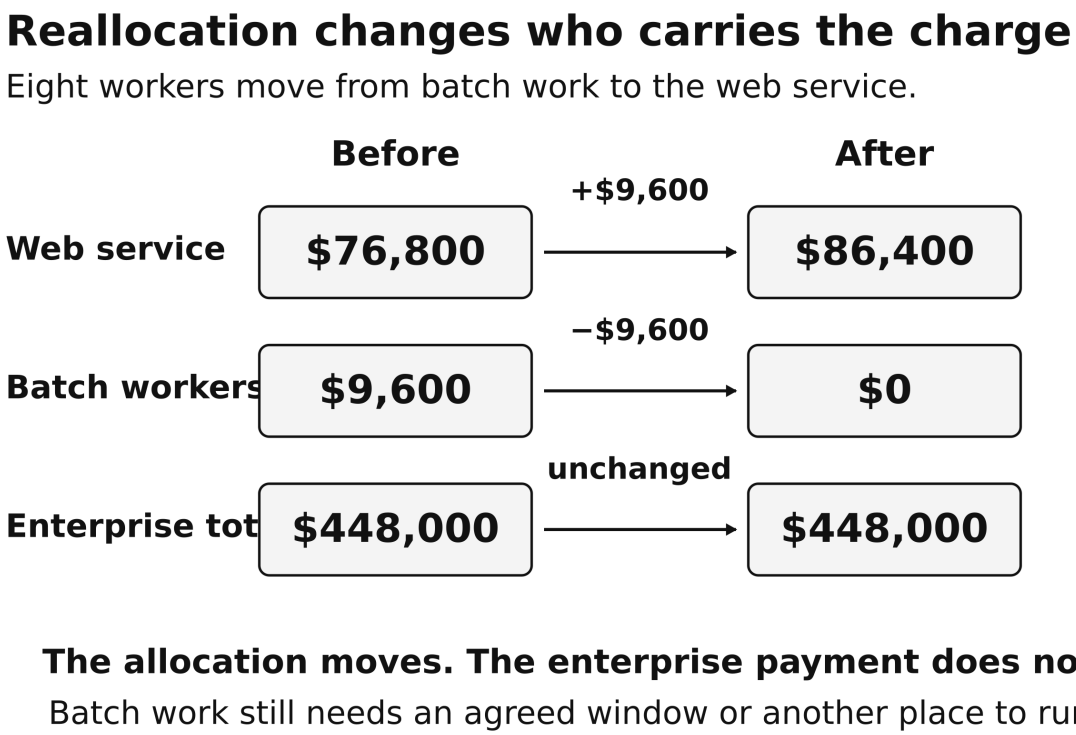


Diagram 1.4. Moving eight batch workers changes attribution: the web service rises by $9,600, the batch allocation falls by $9,600, and the enterprise total stays $448,000. The batch work and any funding transfer still need separate agreement.

Explore this section in the full chapter →

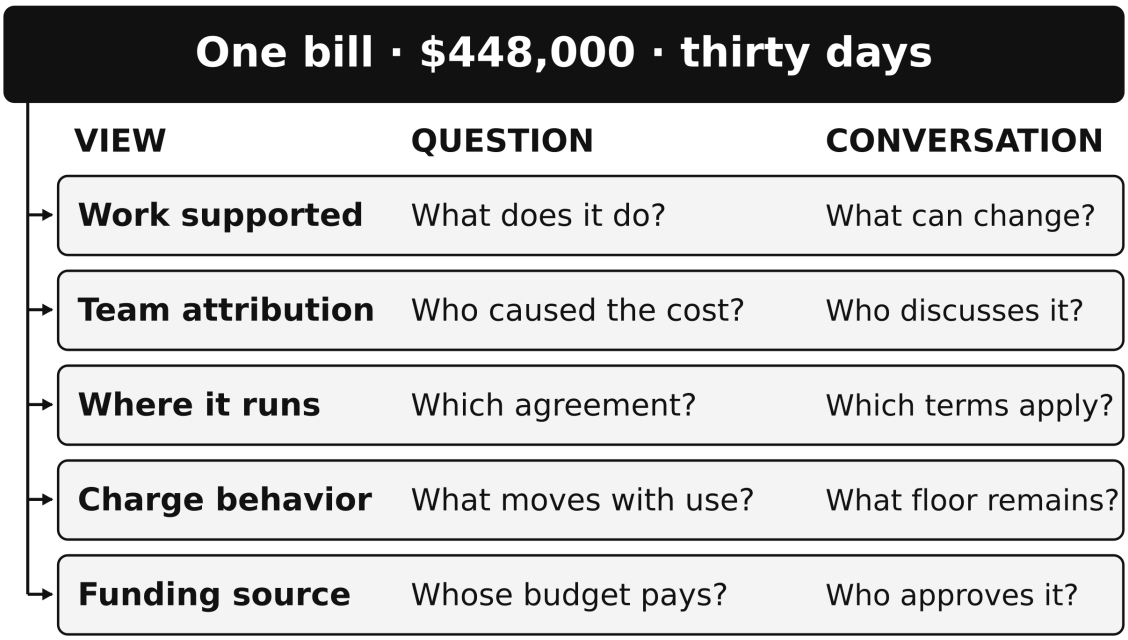
The same bill, five ways

The same $448,000 has now appeared as five tables, each because a different conversation needs it. The work view says what could change and what the change would disturb; the attribution view, whom to talk to; the location view, which agreement governs the change; the charge view, how much of the bill is floor; and the funding view, whose budget moves and whether its owner has agreed.

Explore this section in the full chapter →

One bill, five conversations

Each view groups the same money; never add these totals.



A proposal needs all five conversations about the same expense.

Diagram 1.5. The same $448,000 over thirty days grouped five ways: by work supported, team attribution, where it runs, charge type, and funding source. Each view answers a different question about the same money.

Explore this section in the full chapter →

Enough information to have the next conversation

I would begin the cost record with the work being supported, its owner, the period, the amount, the decision under consideration, and whether each charge is direct, shared, or still unresolved, then follow the change through every dimension. For this chapter's two changes, it would start like this:

Dimension	Eight borrowed batch workers	Sixteen new workers
Amount	$9,600 per thirty days moves to the web service; total unchanged	$19,200 per thirty days added, before any commercial offset
Cost behavior	Usage-sensitive; the charge continues because the workers stay in use	Usage-sensitive unless bought under a commitment
Obligation	None new; the agreement with the batch owner sets the return date	None until bought; a one- or three-year commitment would outlast the launch
Metering	Worker-days, unchanged	Worker-days from provisioning, which precedes qualified service
Attribution and funding	Web service; funding moves only if both budget owners agree	Web service; website product line, if it agrees
Cash	No additional payment	Set by the agreement, effective date, and invoice terms
Modeled labor	Qualifying the moved workers; the batch catch-up	Qualifying the new supply
Still open	The cost of the batch delay: unpriced, not zero	Lead time to qualified service

Keep the underlying records available, because the first version will contain questions that somebody else can answer better than we can.

Explore this section in the full chapter →

CHAPTER 2 / 3 MINUTE BRIEF

Capacity Is a Decision

Read the full chapter → / Read this brief online →

The main idea

Anytime I hear “yeah, we’ve got enough capacity,” I want to ask what they mean. Enough for which operating mode? A complex system runs under steady load, through a seasonal swing, and at the peak that arrives once a year. When someone who runs one breezily tells me everything is fine, no problems, I start wondering whether they have ever experienced a failure, because the honest answer to “do you have enough capacity” is a complex one. Something like: for the work in front of us right now, yes; here is the evidence for that; here is the traffic we expect over the next 30 to 90 days; and here is what we are watching so we can prepare before it becomes a problem. No measurement can give that answer by itself, because it depends on what the organization has decided to promise, who is entitled to rely on the promise, and who answers for keeping it.

Explore this section in the full chapter →

Headroom has a purpose

Headroom is often described as the difference between a measured limit and expected demand, but that leaves out the question of which limit we mean. In our original configuration, normal capacity of 16,000 minus expected demand of 10,000 leaves 6,000 requests per second. Under one-group loss, the calculation is 12,000 minus 10,000, leaving 2,000. These describe different states of the same service, not two reserves we can add together.

Some of that room is carrying more than one responsibility in the plan: growth while everything works, and service preserved when something fails. The healthy service's 6,000 looks like room for the rollout's 4,000 and also for the 4,000 that a lost group takes with it, but it cannot be both, because together they need 8,000. Test the combined requirement in the condition you intend to promise. Counting each responsibility separately and then celebrating the apparent total can make us feel considerably better prepared than we are.

Explore this section in the full chapter →

Autoscaling still needs someone to supply the capacity

I like automation that responds to changing demand. What worries me is the point at which the word autoscaling starts doing work that belongs in a supply plan. An autoscaler can ask for resources when a signal changes; it cannot, by that request alone, establish that the right resources will be available, ready, and connected to their dependencies in time. Azure's quota guidance makes the distinction explicit: quota is a subscription's permission to deploy, capacity is the compute actually available in a region or zone, and a deployment with enough quota can still fail for lack of capacity.[2] That leaves two questions to settle before the autoscaler needs an answer — will we be allowed, and will the supply be there — with options such as qualifying another size or region, or securing an eligible capacity reservation where the workload requires it.

Explore this section in the full chapter →

Make the promise understandable

The person approving a launch should be able to explain what the customer will receive and what has to remain true for us to deliver it. If that explanation requires reconstructing a month of technical discussions, we have more work to do. I would give them a short account of the useful work, quality, deadline, operating condition, and usable supply, with named owners and the underlying evidence available to whoever needs to challenge it.

[2]Microsoft Learn, "Check vCPU quotas", checked October 3, 2026. Defines quota as a subscription's permission to deploy and capacity as the compute available in a region or zone, says a deployment with sufficient quota can still fail for lack of capacity, and suggests a different size, zone, or region, or an on-demand capacity reservation.

For our web service, the account might run like this. At the tested request mix, the service carries the expected 10,000-request peak with 2,000 to spare after losing one of its four sixteen-worker groups. The full rollout, at 14,000, would leave it 2,000 short in that condition. A staged rollout reaches 12,000 and uses all of the margin, so it also needs eight batch workers borrowed for thirty days, if the batch owner accepts the delay and the workers pass qualification. Sixteen new workers would carry the full rollout once funded, but they are estimated to be six weeks from qualified readiness, and the borrowed workers are due back before then.

Explore this section in the full chapter →

CHAPTER 3 / 4 MINUTE BRIEF

The Players and the Vocabulary

Read the full chapter → / Read this brief online →

The main idea

As an engineer, I can learn the difference between a capital expense and an operating expense, and I can put together a comparison that explains why changing one into the other looks attractive. Then I talk to someone in Finance and discover how much of the question I have left out. The timing matters, the planning process matters, the person who can approve the commitment matters. "It depends" begins to sound less like an obstacle and more like the answer I should have expected.[3]

Explore this section in the full chapter →

The nouns that change a meeting

A **forecast** estimates future demand under stated assumptions. Give it an as-of date, because the information available at the time is part of what makes the estimate interpretable. Revising it after a customer changes plans may be exactly the right thing to do; the revision is a new forecast, not evidence that the old one was accurate. Check the unit, too: in a Finance meeting "the forecast" usually means expected spending for the year, and in a capacity meeting, demand in requests, cores, or tokens.

A **target** is something the organization intends to achieve. A sales target may be deliberately ambitious to encourage people to pursue an opportunity, while the most likely estimate of realized demand is lower. Capacity needs to examine the target as a scenario without silently converting ambition into a prediction.

A **budget** is an approved plan for spending: an amount, for a period, held by someone accountable for it, usually set months before the demand it funds. It doesn't approve a particular purchase. Who can approve that kind of spending, and whether the amount

[3] Tim O'Brien, "It's Not That Simple", Medium, June 21, 2026. The opening passage adapts the author's description of recurring conversations with Finance.

crosses a threshold that sends it through a different process, are separate questions, and often what Finance means by "it depends." Whether the money will support the service's obligations is an engineering and business question, and a discrepancy between the two needs a conversation rather than a revised label.

A **request** says what someone wants and starts an investigation or a negotiation. It doesn't establish that the resources exist, that funding has been approved, or that another customer will give up an allocation, which sounds obvious until a requested amount starts appearing in other people's plans as confirmed supply. Requests also lose their reasons on the way: the SRE book's account of Auxon, a Google planning tool, describes requests arriving as "X cores in cluster Y," their degrees of freedom "long lost."[4] In our launch, the degrees of freedom were most of the answer. A quota increase says no more about supply: it raises the limit on what we may request without putting any machines behind it.

A **reservation** needs to be understood in its particular context. It might be a provider product, an internal hold, or a position in a planning queue, and a provider's reservation may buy a discount, a hold on capacity, or both. I want to know what has been reserved, for whom, where, for which dates, and with what release rule before I use it to support a customer promise.

A **commitment** is a promise with defined parties and conditions. In Volume II, Chapter 20 we will separate a promise to spend from a promise to supply usable capacity, because each can exist without the other. An optimistic forecast in a presentation establishes neither.

One more noun is not a planning word but a product word, and meetings trip over it as often as the others: the **SKU**. The term is stock-keeping unit, borrowed from retail, where it names one distinct item a store sells — one product, in one configuration, that someone can order by a code. The cloud providers borrowed it for their catalogs, and in this book a SKU is one sellable configuration of a service: one size, one feature set, one price. Every line on the bill in Chapter 1 names a SKU.

[4] Beyer et al., *Site Reliability Engineering*, Chapter 18, "Software Engineering in SRE", quoted from the Auxon case study. Auxon was an internal Google tool; the connection to this chapter's launch is the author's.

The name is worth reading slowly, because providers encode the machine into it. In Microsoft's convention, `Standard_D4s_v5` decodes as a general-purpose family (D), four vCPUs, premium-storage capable (the s), fifth version (v5), and the encoded extras run to local disks, network bandwidth, memory ratios, processors, and, for the accelerated families, the accelerator model itself: the GPU size Chapter 10 works with, `Standard_ND96isr_H200_v5`, announces the H200 in its own name.[5] Other providers publish naming conventions of their own, and the skill of reading them transfers even where the grammar differs.

Explore this section in the full chapter →

Who can change the promise?

For each condition the plan relies on, I want to know who supplied it, who can check it, who can change it, and who else needs to hear when it changes. For our constructed launch, a first version might look like this:

Condition	Supplied by	Checked by	Can change it	Must also hear
First stage adds 2,000 requests/s	Product lead	Engineering and operations	Product and the business owner	Operations, Finance
Batch work can wait while eight workers move	Batch owner	Batch owner, against a catch-up plan	Batch owner	Product, operations
Moved workers ready after two days' testing	Operations	Operations, by testing	Operations	Product
Sixteen new workers in about six weeks	Supplier, via Procurement	Operations, at qualification	Supplier and operations	Product, batch owner
Money for the new workers	Budget owner	Finance	Budget owner, within a delegated limit	Product, Procurement
Service survives loss of one failure group	Business owner	SRE, by testing	Responsible business executive	Everyone relying on it

Explore this section in the full chapter →

[5] Microsoft Learn, "Azure virtual machine sizes naming conventions", checked October 6, 2026. Field order (family, sub-family, vCPUs, additive features, accelerator, version), the additive-feature letters, and the rule that only specialized SKUs launched from the third quarter of 2020 carry the accelerator in the name. The `Standard_D4s_v5` reading follows the page's conventions and examples.

CHAPTER 4 / 3 MINUTE BRIEF

The Simplest Practice

Read the full chapter → / Read this brief online →

The main idea

If you're starting a capacity and cost practice, I would resist the temptation to begin by designing the process for the entire organization. You'll need to understand that organization eventually, and it will need more than one approach. But there's a more useful place to begin: find a decision somebody needs to make and stay involved long enough to see what it takes to make it well.

You might start with a service that's about to grow. The engineering team can explain what it runs today, somebody in the business has a plan that will change the demand, and Finance wants to understand the cost of supporting that plan. Getting those people into the same discussion sounds straightforward until you discover that the growth estimate assumes a launch date, the launch date assumes available capacity, and the capacity depends on a purchasing decision that hasn't been made. Now you have something worth working on.

Explore this section in the full chapter →

A portfolio small enough to follow

For the first decision, we'll concentrate on the web service and the measurement Chapter 2 gave it. Its operators have tested a particular request mix at the configurations we're considering: one worker can complete 250 requests per second while holding ninety-fifth-percentile (p95) latency to 200 milliseconds and errors to no more than 0.1 percent of offered requests. That evidence exists because we've assumed it. In a real discussion, I'd want the operators to explain how they qualified the rate and what change in request mix, configuration, or downstream behavior would make it unreliable.

The service has 64 workers arranged in four independent failure groups of 16, providing a tested 16,000 requests per second when everything is available. The business has

also agreed that the service must continue after losing any one group. That requirement changes the capacity we can promise:

$$(64 - 16) \times 250 = 12{,}000 \text{ requests/second.}$$

Twelve thousand is the limit I'd use for this decision; as Chapter 2 showed, sixteen thousand answers a question about normal operation. Even the lower number covers only the loss of one independent group; we haven't established what happens when an entire region fails.

Explore this section in the full chapter →

Borrowed capacity comes with a return date

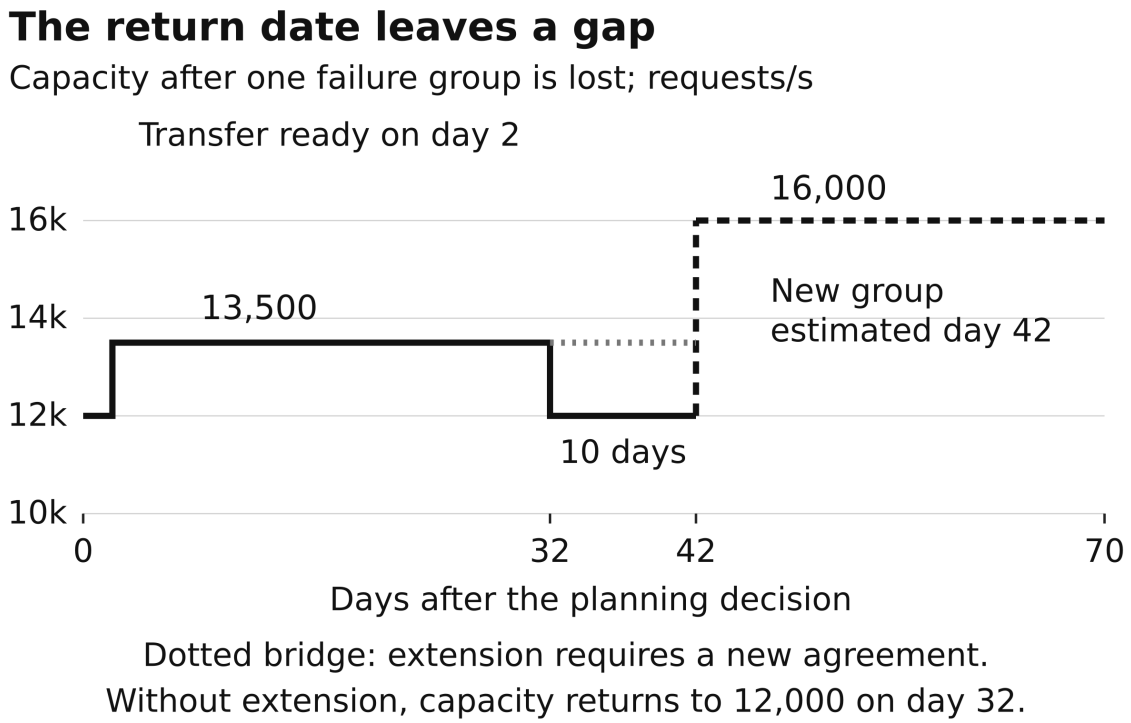


Diagram 4.1. The temporary transfer expires on day 32; the new group's qualified readiness is only estimated for day 42. The dotted bridge requires a further batch agreement.

Explore this section in the full chapter →

Make the decision legible

For this proposal to proceed, five of the people in Chapter 3's authority map have to commit to something:

Who	Commitment	Needed before
Product lead	Stage the rollout at the estimated 2,000 additional requests/s, and keep the ability to stop expansion	the rollout
Batch owner	Accept a bounded delay with a catch-up plan	the transfer
Web operations lead	Qualify the transferred workers, including their behavior when a group fails	the rollout
Budget owner	Authorize the $640-a-day purchase within a delegated limit, contingent on supply and the actual terms	the purchase
Responsible business executive	Accept the remaining exposure, or change the launch scope or date	the rollout

Several may be settled in one conversation. They're still five commitments, and the decision record should make a missing one obvious. If batch refuses the delay, we no longer have this option. If the business refuses staging, 14,000 still exceeds 13,500, and changing the failure requirement to make the arithmetic pass would mean offering a different service. We may need to discuss that possibility, but with the person entitled to change the obligation and with its consequence understood.

The agreement also needs a way to reopen the discussion before the next scheduled review. Demand running above the staged estimate, a changed request mix, a larger launch, an infeasible catch-up plan, or delayed delivery can each remove a condition we relied on. Waiting for the monthly meeting would only be sensible if we could afford to wait. Whoever learns about the change should know whom to contact and what decision it may affect.

Explore this section in the full chapter →

CHAPTER 5 / 3 MINUTE BRIEF

Knowing Your Workloads

Read the full chapter → / Read this brief online →

The main idea

Give me a report showing that a machine uses a fifth of its CPU and I can make a plausible argument for shrinking it. Tell me that the machine runs a critical database and I can make an equally plausible argument for leaving it alone. Neither argument requires much knowledge of the database, which is precisely what should make us uncomfortable.

The interesting work begins when we try to explain why that fifth of the CPU is all the system uses. Perhaps nobody needs the machine anymore. Perhaps the application is waiting on a lock, or the memory is keeping an expensive set of reads away from storage. Perhaps the person who called it critical knows something about next month's work that has never appeared in our measurements. I want to understand those possibilities before we turn either the low utilization or the word *critical* into an answer.

Explore this section in the full chapter →

Look at utilization, waiting, and errors

Gregg's USE method—utilization, saturation, and errors—gives resource investigation a disciplined start: for every resource, check all three. He defines utilization as the average time a resource was busy servicing work, saturation as the degree to which it has extra work it can't service, often queued, and errors as the count of error events.[6] The averaging matters. Gregg describes a customer whose CPUs were saturated although monitoring, reporting five-minute averages, never showed them above 80 percent; they were hitting 100 percent for seconds at a time. Relating busy, queued, and failing work tells us far more than sorting a spreadsheet of CPU percentages from smallest to largest.

Explore this section in the full chapter →

Test the arrivals you expect to receive

The workload generator deserves the same skepticism as the system under test. A reassuring latency result means little if the test quietly stopped sending work whenever the service became slow, and many generators do exactly that by default: each simulated client waits for a response before sending its next request.

Gil Tene calls the resulting measurement error coordinated omission: the generator stops sending the requests that would have suffered most.[7] In his `wrk2` documentation, a 30-second run aimed at 2,000 requests per second over 100 connections included one 1.4-second server pause. Measured from when each request should have been sent, the 99th percentile was 1.27 seconds; measured from when each was actually sent, 6.04 milliseconds. About 2,800 requests were due during the pause, 7 percent of the 39,500 recorded, but 100 waiting connections could record at most 100 slow responses, too few to reach the 99th percentile.

Explore this section in the full chapter →

[6] Brendan Gregg, *The USE Method*, https://www.brendangregg.com/usemethod.html, accessed October 4, 2026. Gregg also notes a capacity-based definition of utilization; the method sets no universal utilization target.

[7] Gil Tene, "How Not to Measure Latency," QCon San Francisco 2012, https://qconsf.com/sf2012/dl/qcon-sanfran-2012/slides/GilTene_HowNotToMeasureLatency.pdf, accessed October 3, 2026; `wrk2` README, https://github.com/giltene/wrk2, accessed October 4, 2026. The percentiles and the 39,500 requests recorded after a 10-second calibration period are from the README's second example; the other request counts are this chapter's arithmetic. `wrk2` is an HTTP tool.

A record someone can challenge

By now we can keep three kinds of knowledge apart instead of blending them into one confidence score: measurements from a known interval and configuration, experimental claims bounded by what was tested, and business judgments about demand, deadlines, and acceptable consequences. For the web service, a short evidence record might read:

Claim	Evidence	What would change it
250 requests/second per worker at p95 ≤ 200 ms, errors ≤ 0.1%	Experiment: tested mix, each configuration used	A different mix; a generator that could not hold its rate
Next peak of 10,000 requests/second	Measurement and forecast: 16 weeks of daily one-minute peaks of offered requests	Peak demand rejected or queued; sub-minute bursts
Staging adds 2,000 requests/second	Business judgment: product lead, dated, staged scope only	Changed scope or customers; the first stage's peak
13,500 requests/second after losing a group	Arithmetic: 54 × 250, linear within tested configurations	Per-request demand on the limit rising over 12.5%
The shared database is not the limit at 13,500	None yet	A database test at the new mix, before launch

A missing metric must not become zero during an export, and a successful experiment on one workload must not silently become permission to shrink every machine of the same type. Contrary runs and failed tests belong in the record too. It is working if a specialist or customer can point to the line they think we have misunderstood.

Explore this section in the full chapter →

CHAPTER 6 / 3 MINUTE BRIEF

Built-In Tools

Read the full chapter → / Read this brief online →

The main idea

It is possible to open three perfectly useful tools and receive what look like contradictory instructions. A machine-sizing report recommends a smaller instance, a budget alert says the project is spending too much, and the autoscaler wants to create more instances. Before we schedule a meeting to determine which tool has failed, it would help to know what each one was asked.

The sizing report summarizes at least a week of one machine's use, the alert compares charges that typically arrive within a day with a monthly threshold, and the autoscaler responds to the last few minutes of CPU. All three could be right. The difficult part is deciding whether today's load is temporary, whether the proposed smaller machine can handle the work we have promised, and whether somebody has already agreed to pay for it.

Explore this section in the full chapter →

Keep inventory, use, and cost distinguishable

When I look at a service, I want to move among three views without losing track of what each tells me. Inventory shows what exists and where; telemetry shows what it did during an interval; billing shows what was charged under the commercial arrangement. A resource that disappeared yesterday may be absent from today's inventory and still account for a substantial part of this month's bill. A workload that stopped running may have left billed resources behind, while an apparently unused resource may still be needed for recovery.

The distinction seems obvious until someone joins the three views, drops the records that do not match, and presents the result as a complete account. We need the unmatched records precisely because they may explain what we have failed to understand.

Explore this section in the full chapter →

A recommendation is the beginning of an experiment

Before I accepted or dismissed one, I would want to connect the recommendation and its evidence date to the workload's next important event. Has the application changed? Is a migration about to end? Does the proposed smaller configuration affect a recovery obligation? I would also ask Finance what the reduction means under the commercial arrangement. Freeing capacity inside an existing commitment can be worthwhile even when it does not lower this month's cash expenditure.

For a stable stateless service, a representative subset may give us a straightforward experiment: agree on completion and latency criteria, make the change through the operating team's process, compare with the baseline, and preserve a feasible route back. A data service may need more patience, because ordinary traffic does not exercise the maintenance or recovery work that justified the allocation.

Suppose a smaller configuration serves requests successfully but cannot complete a required rebuild. We have discovered that the proposed saving changes the service being supplied. The business may be willing to accept that change, but I would not present it as an equivalent service for less money.

This is also why I want a useful explanation when a recommendation is deferred. "Awaiting a representative peak" gives us a next step, as does "retained for this recovery obligation" or "superseded by the planned migration." "Engineering refused" mostly tells me that we have run out of patience with each other. The capacity team's insistence deserves no more authority than Engineering's refusal unless somebody can explain the work.

Explore this section in the full chapter →

Alerts, caps, and the person who receives them

An Azure budget notifies people when actual or forecast cost crosses a threshold; in Microsoft's words, "Resources aren't affected, and your consumption isn't stopped." At subscription and resource-group scopes, an action group can pass the notification to automa-

tion such as a function or webhook, which needs its own design: which work may stop, who authorizes it, and how the service resumes.[8]

Cost data typically arrives 8 to 24 hours after use, budgets are evaluated every 24 hours, and email follows within about an hour, so an overrun can be nearly two days old when someone reads about it. An application that needs an immediate limit has to enforce it in its own request or admission path, reconcile it with billing, and decide what happens when the limit is reached.

Explore this section in the full chapter →

[8] Microsoft Learn, “Tutorial - Create and manage budgets”, checked October 4, 2026. Notification without stopping consumption; data and evaluation timing; actual cost including purchases, with usage-only filters; action groups at subscription and resource-group scopes.

CHAPTER 7 / 4 MINUTE BRIEF

AI Services and Token Capacity

Read the full chapter → / Read this brief online →

Before any theory: the four ways to buy tokens

There is more than one way to buy the same token, and the differences matter to a capacity plan.

The most direct path is the provider's own API, usually priced by consumption: so much per million input tokens, so much per million output tokens, with separate treatment for caching and other features. Payment may be prepaid, invoiced, or governed by a contract; a usage-based price does not establish when the money changes hands.[9][10]

The second path is through a platform. Microsoft Foundry, Google's Gemini Enterprise Agent Platform (formerly Vertex AI) and OpenRouter provide model access under their own deployment, routing and billing arrangements.[11][12][13] Before comparing rates for a

[9] OpenRouter, FAQ, https://openrouter.ai/docs/faq, and "API Credit & Rate Limits," https://openrouter.ai/docs/api_reference/limits, checked October 4, 2026; Stripe, "Stripe agrees to acquire OpenRouter to help businesses optimize token routing and usage," August 19, 2026, https://stripe.com/newsroom/news/stripe-agrees-to-acquire-openrouter. Product terms and transaction announcement; no transaction price is inferred.

[10] OpenAI, "GPT-6 Astra: A new generation of intelligence," September 3, 2026, https://openai.com/index/gpt-6-astra/. Launch Standard rates $10/$50 per million input/output tokens; Fast mode twice the Standard price and advertised at up to twice the speed. On OpenRouter, which reports weekly route traffic, Astra was among the most-used routes through early October 2026 and processed about 98.6 billion tokens on October 9, 2026, moving down the popularity list.

[11] Microsoft, "Understanding deployment types in Microsoft Foundry Models," https://learn.microsoft.com/en-us/azure/foundry/foundry-models/concepts/deployment-types, checked October 3, 2026. Deployment categories, supported processing boundaries and model-specific availability; managed-compute deployments are a separate option.

[12] Google Cloud, "Use Provisioned Throughput," https://docs.cloud.google.com/gemini-enterprise-agent-platform/models/provisioned-throughput/use-provisioned-throughput, checked October 4, 2026. Default spillover, the dedicated-only 429, enforcement windows and spillover hidden by one-minute averages. Renamed from Vertex AI on April 22, 2026, per the release notes: https://docs.cloud.google.com/gemini-enterprise-agent-platform/release-notes.

[13] OpenRouter, FAQ, https://openrouter.ai/docs/faq, and "API Credit & Rate Limits," https://openrouter.ai/docs/api_reference/limits, checked October 4, 2026; Stripe, "Stripe agrees to acquire OpenRouter to help businesses optimize token routing and usage," August 19, 2026,

named model, establish which version and features each offering supports, who supplies the inference, where processing may occur and which agreement pays for it. An existing cloud agreement can make this path operationally convenient; the workload still needs its own comparison.

The third path is a negotiated agreement. Large consumers may negotiate discounts, usage minimums, and model-specific terms. Negotiation does not guarantee a discount or held supply, and the price of the same work can depend on which agreement it flows through.

The fourth path is provisioned capacity, because a workload can hit a throughput limit while its monthly spending is modest. Providers sell arrangements that reserve a defined level of processing capability, subject to the model, region, workload and service terms, and the charge continues while that capacity is idle. A spending commitment alone reserves no capacity.[14][15]

Explore this section in the full chapter →

https://stripe.com/newsroom/news/stripe-agrees-to-acquire-openrouter. Product terms and transaction announcement; no transaction price is inferred.

[14]Microsoft, "Provisioned throughput billing and cost management," https://learn.microsoft.com/en-us/azure/foundry/openai/concepts/provisioned-throughput-billing, checked October 4, 2026. Deployed throughput incurs hourly charges whether or not it is used; a reservation is a discount on those charges and does not guarantee capacity.

[15]Google Cloud, "Use Provisioned Throughput," https://docs.cloud.google.com/gemini-enterprise-agent-platform/models/provisioned-throughput/use-provisioned-throughput, checked October 4, 2026. Default spillover, the dedicated-only 429, enforcement windows and spillover hidden by one-minute averages. Renamed from Vertex AI on April 22, 2026, per the release notes: https://docs.cloud.google.com/gemini-enterprise-agent-platform/release-notes.

Token capacity is a different kind of capacity

What limits an AI service is a combination of three things that no single graph shows:

- **Scarcity.** Rate limits cap requests, tokens or concurrency within a window, and their scope varies by model and account. On shared service a limit is permission to send rather than capacity held for the sender, so a request within it can still be refused when the provider is busy. Queuing and retries are application decisions with latency and cost consequences.[16]
- **Pricing.** Per-token charges, subscriptions and provisioned capacity respond differently to demand. Compare the full cost of the service, including unused commitments, before drawing a conclusion from its token rate.
- **Budgets.** A prepaid balance can run out; a commitment can charge after demand disappears. Alerts, hard caps and advisory limits behave differently, so the plan needs to say which one is in force and what happens when it is reached.

Explore this section in the full chapter →

[16] OpenAI, "Rate limits," https://developers.openai.com/api/docs/guides/rate-limits, and "Error codes," https://developers.openai.com/api/docs/guides/error-codes; Google Cloud, "Standard PayGo," https://docs.cloud.google.com/gemini-enterprise-agent-platform/models/deploy/standard-paygo, and "Priority PayGo," https://docs.cloud.google.com/gemini-enterprise-agent-platform/models/deploy/priority-paygo; Microsoft, "Enable priority processing for Microsoft Foundry Models," https://learn.microsoft.com/en-us/azure/foundry/openai/concepts/priority-processing; AWS, "Service tiers for optimizing performance and cost," https://docs.aws.amazon.com/bedrock/latest/userguide/service-tiers-inference.html; checked October 3, 2026. Documented ramp, retry, spend-limit and downgrade behavior; shared pool sizes are not published.

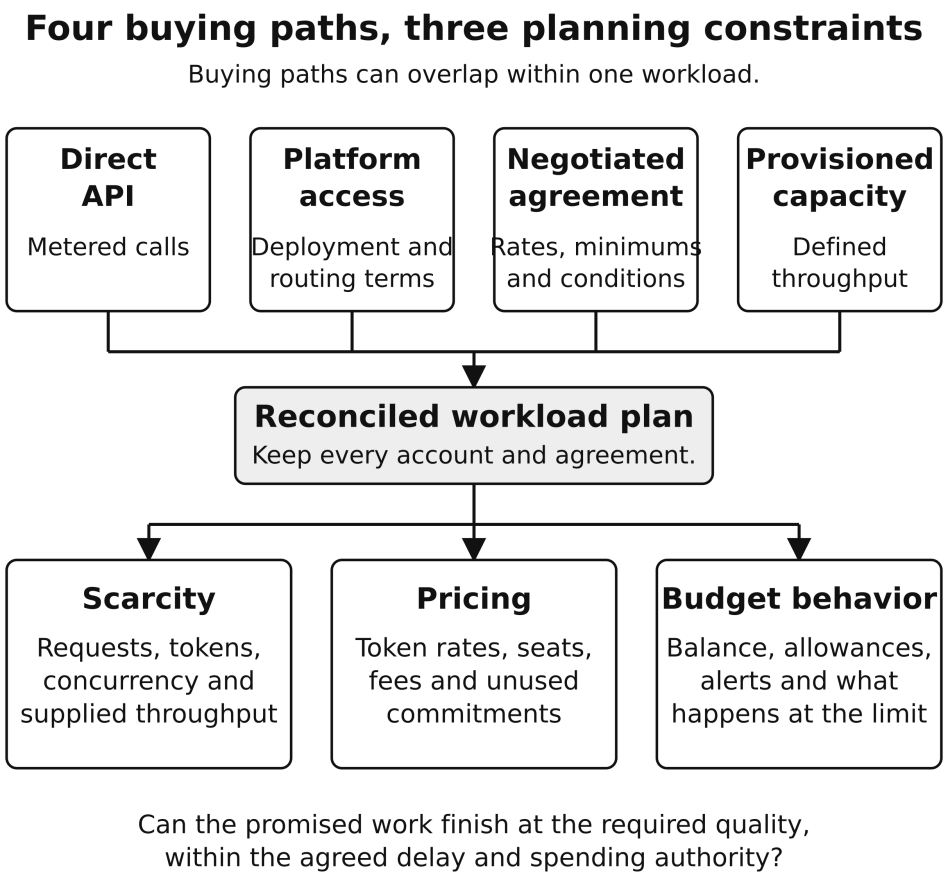


Diagram 7.1. Four buying paths feed one workload plan. Scarcity, pricing and budget behavior remain separate checks even when the organization buys through several paths.

Explore this section in the full chapter →

The model product list and how to read it

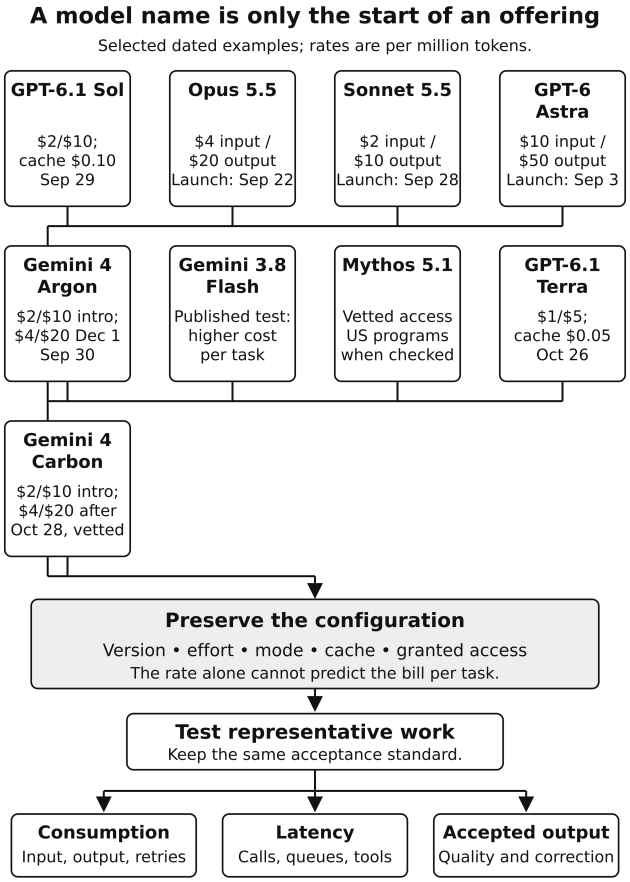


Diagram 7.2. A dated model offering feeds a configuration record, then a test on representative work. The selected examples illustrate different prices, consumption and access conditions; they are not a performance ranking.

Explore this section in the full chapter →

Sizing supply for the busy interval

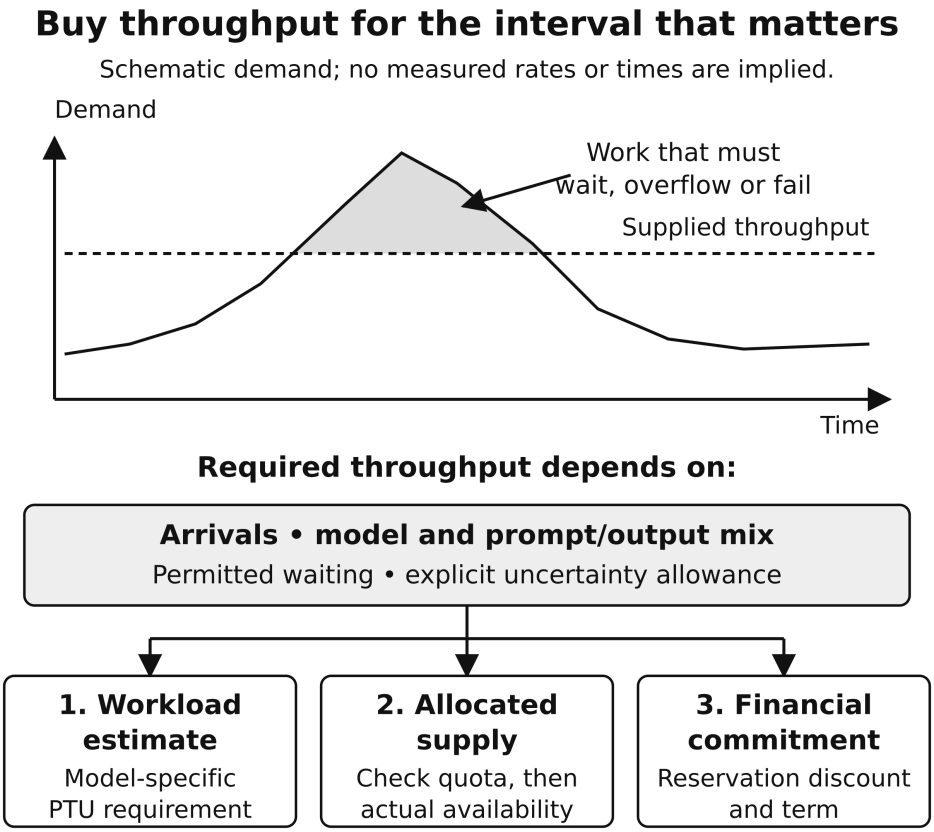


Diagram 7.3. A schematic busy interval exceeds supplied throughput. The amount to buy depends on arrivals, the model and prompt/output mix, permitted waiting and uncertainty; the workload estimate, allocated supply and financial commitment remain separate records.

Explore this section in the full chapter →

Measure the efficiency of the models, not just the spend

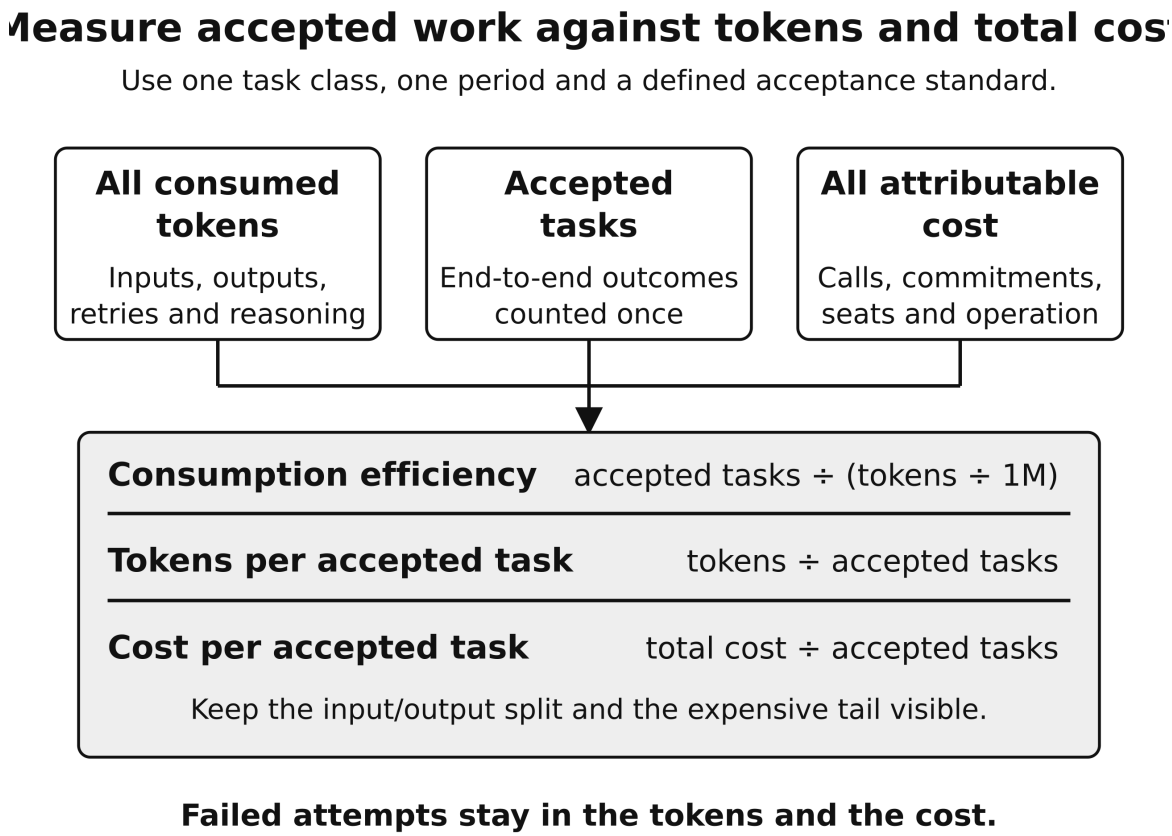


Diagram 7.4. Three measures share the same task class, time window and acceptance standard. Accepted tasks per million tokens and tokens per accepted task are reciprocals, read as efficiency and as intensity; total cost per accepted task adds price and fixed charges. Failed attempts remain in consumption and cost.

Explore this section in the full chapter →

Consumption efficiency counts accepted tasks per million tokens:

$$\frac{\text{accepted tasks}}{\text{tokens consumed}/1{,}000{,}000}.$$

Record it per model and per task class, measured monthly.

Accepted ties the count to the business's outcome standard; preserve whether the result needed correction and what that correction cost. *Per task class* keeps unlike work from being averaged together. Model-level comparisons are useful when a model performs the complete task. For a workflow spanning several models, report the end-to-end outcome once and use step-level costs and acceptance checks to explain each model's contribution. Crediting every model with the same outcome and then summing would count the work repeatedly.

Explore this section in the full chapter →

Recommended practices

Size supply for the work that must finish on time. Use measured demand over the relevant interval, the permitted delay and an explicit uncertainty allowance. Decide which work may queue, which may use overflow and which needs reserved throughput, and test any reservation on price as well as supply.

Run the efficiency measurement every month, with an explanation. Report accepted tasks and token consumption by model and task class, alongside total cost and retries. Investigate changes rather than assuming they represent improvement or waste.

Explore this section in the full chapter →

Keep the budget policy usable across models and providers. State who may spend, what is counted and what happens at the limit separately from the feature implementing it. A provider change may require different controls: a balance, gateway quota, spend cap and commitment make different promises. Re-test their behavior rather than transferring the old configuration by name.

Explore this section in the full chapter →

CHAPTER 8 / 3 MINUTE BRIEF

Measuring AI Workloads

Read the full chapter → / Read this brief online →

The main idea

Ten million tokens sent through a highly rated model do not establish a known amount or quality of completed work. A smaller model may be sufficient for the task; a larger one may succeed where it cannot. Only the application's acceptance results can settle the comparison.

This chapter also proposes **benchmark-weighted consumption**: an unvalidated, descriptive index that joins token counts to model benchmark scores. It can show a change in model mix that a raw token total hides. It cannot turn tokens into a measured quantity of intelligence or substitute for counting accepted work.

Explore this section in the full chapter →

What the workload actually consumes

For an AI workload, the unit of consumption is not the user request. It is the **model call**. One task — a support question answered, a report drafted — typically triggers several calls: a cheap model to interpret the question, a capable model to produce the answer, another model to check the result, occasionally a retry when something failed. Each call consumes input tokens and output tokens on a particular model.

Explore this section in the full chapter →

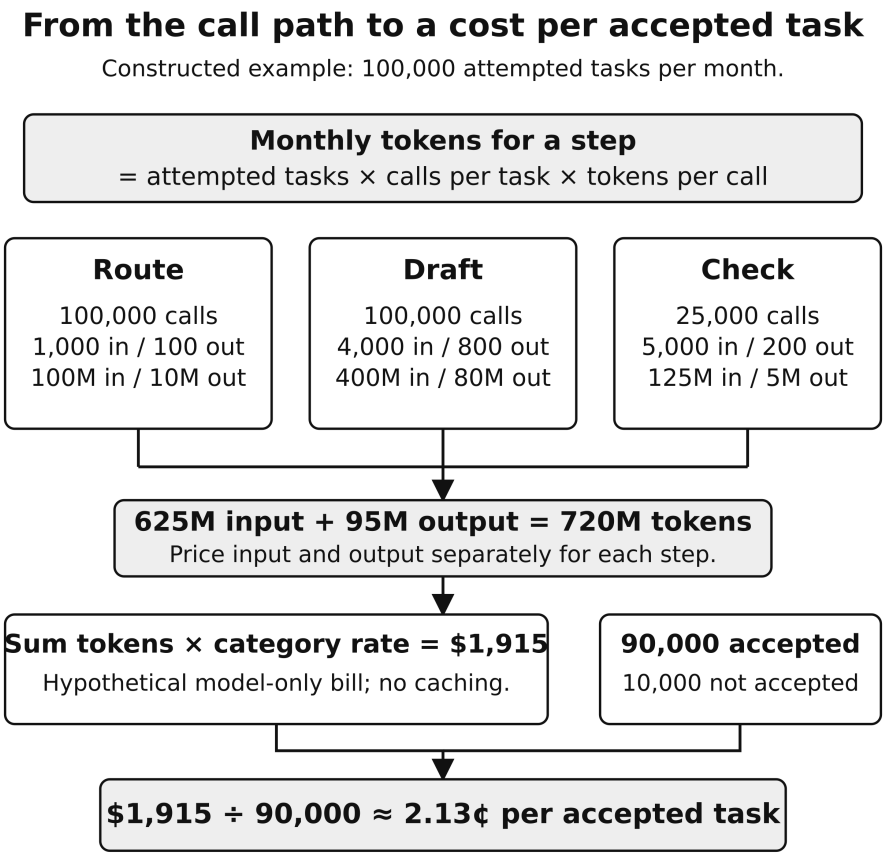


Diagram 8.1. The constructed call path expands attempted tasks into route, draft and check calls, then into token totals and the model-only bill. Dividing by accepted tasks retains the cost of rejected attempts.

Explore this section in the full chapter →

Describing the model mix

Start with a single benchmark version run under comparable conditions. For illustration, suppose two models score 0.85 and 0.45, where the score is a fraction of test cases passed. These are hypothetical results on one test, not universal capability ratings.

For each model, multiply its token count by that score, then add the products. If model k consumes T_k tokens and scores b_k, the index is:

$$B = \sum_{k} T_k \times b_k.$$

One million tokens at 0.85 contribute 850,000 benchmark-weighted tokens; one million at 0.45 contribute 450,000. This is an accounting convention proposed here, not an established measure of effective capability purchased.

An increase from ten million to twelve million weighted tokens is $(12 - 10) / 10 = \mathbf{20}$ **percent** growth in the index, not evidence of 20 percent more useful work. Repeating a prompt unnecessarily would raise the index too. A change in tokenization, benchmark version or task mix can also break the comparison. Publish those conditions alongside it.

Suppose routing moves work to a model with a lower benchmark score. The index falls for the same token count, making that model-mix change visible. It does not establish lost capability on this workload: the cheaper model may pass every task that matters here. Compare acceptance rates, correction effort and total cost before calling the routing change a saving.

Explore this section in the full chapter →

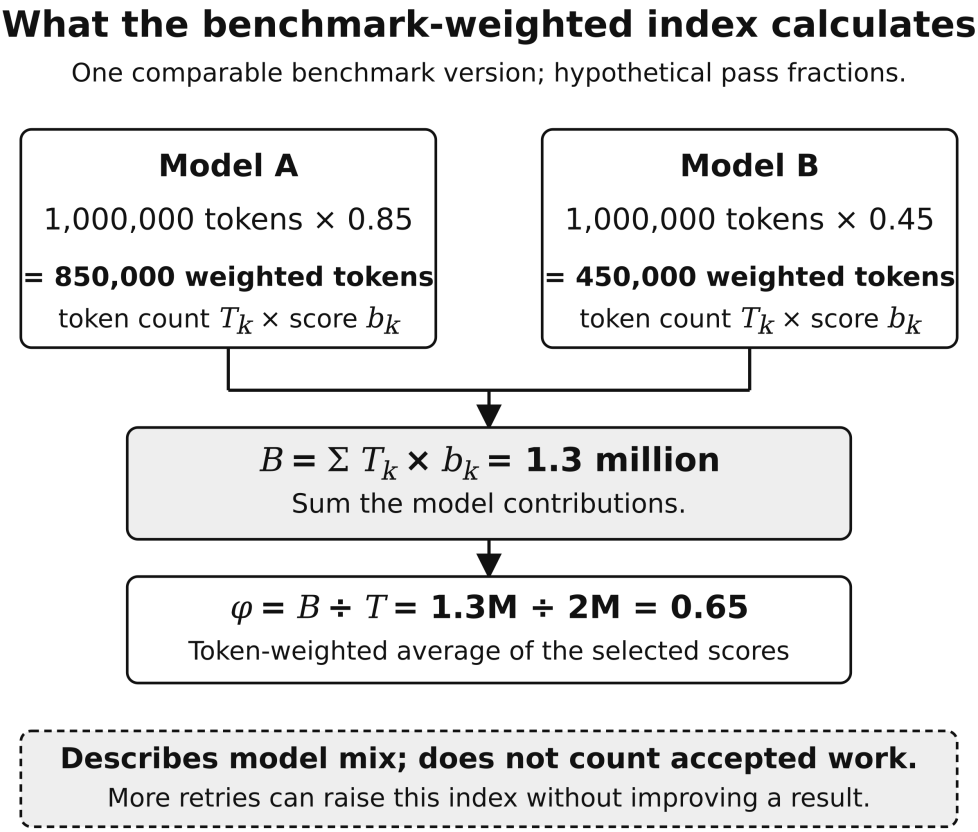


Diagram 8.2. The proposed benchmark-weighted index multiplies each model's tokens by a comparable benchmark score, sums the contributions and divides by total tokens to describe the model mix. All scores here are hypothetical; the calculation does not measure accepted work.

Explore this section in the full chapter →

Choosing the benchmarks

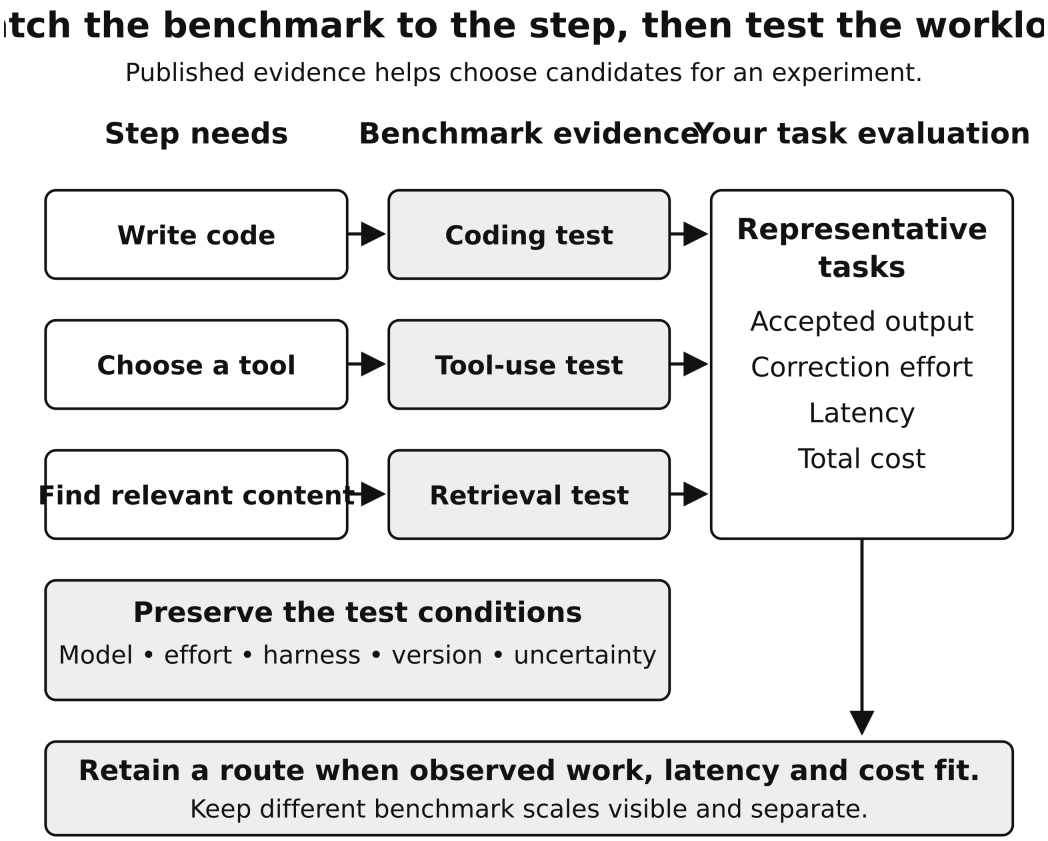


Diagram 8.3. Match the evidence to the skill a step needs, preserve its test conditions and evaluate the candidate on representative work. The routing decision depends on observed acceptance, correction, latency and total cost.

Explore this section in the full chapter →

The purpose question: what no consumption measure can say

So the measurement design needs a second mechanism: an audit question set, asked per task class, by people who can judge the answers:

- Does this step need a model at all? Formatting, validation and deterministic transformations may be better served by ordinary code, whose compute and maintenance still cost money.
- Does it need this class of model? Compare a smaller model on representative examples, including difficult cases and changes in the label set.
- Does it duplicate another workload? Two teams may have good reasons for separate systems, but the reasons should survive being asked.
- Is the architecture appropriate? Repeated retrieval, excessive context and unnecessary agent loops are testable design choices.

Explore this section in the full chapter →

Right-sizing: the cheaper ways to do the work

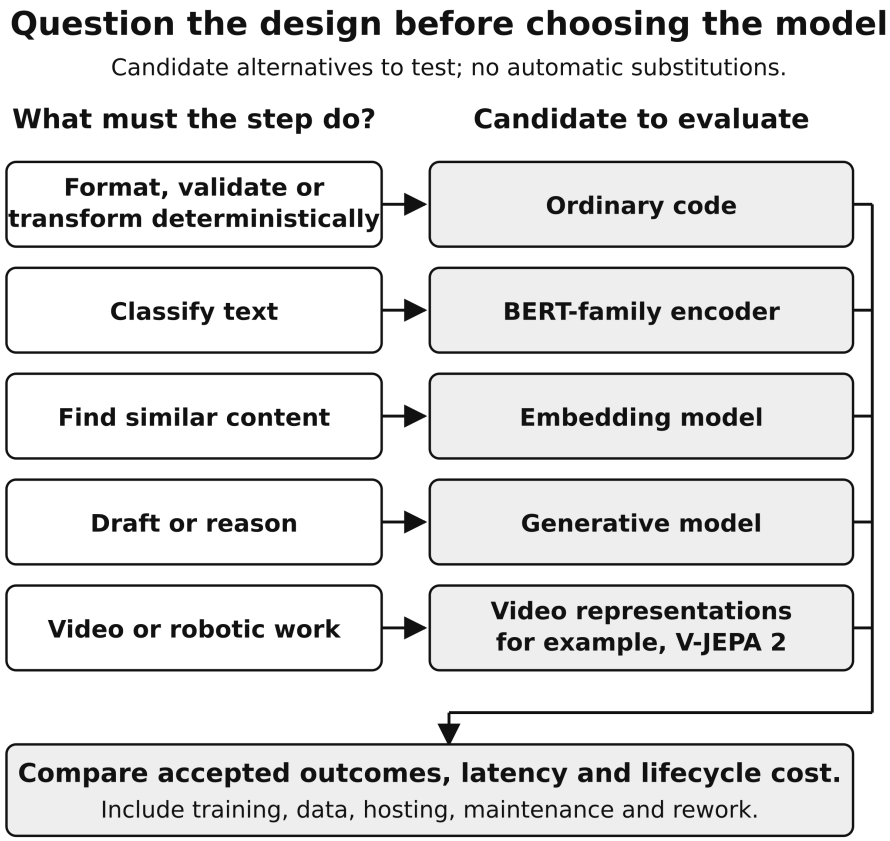


Diagram 8.4. Task requirements suggest candidates to evaluate, from ordinary code and encoders to embeddings, generative models and video representations. Compare accepted outcomes, latency and lifecycle costs before changing the design.

Explore this section in the full chapter →

CHAPTER 9 / 3 MINUTE BRIEF

Orchestration and Agentic Topologies

Read the full chapter → / Read this brief online →

The main idea

A forecast built on model tokens cannot explain an agent's cost: the memory, tools and runtime around each call carry their own bills and limits, and the agent's loops change the shape of consumption itself. Consider a hypothetical support assistant that looks up a customer's account, drafts a reply, checks a refund rule, updates its memory and writes a trace. The customer sees one answer, but several systems did work.

A memory store, a tool server and a gateway may begin as small additions, then acquire their own bills and scaling constraints. A loop rereads its growing history every turn, a lead agent fans work out to parallel workers, retries stack across layers, and a task waiting for a person can hold a sandbox for hours. Some failures interrupt the task; others leave it running with incomplete context or stale data. The capacity team needs to follow both the cost and the behavior back to the work these components support, then ask whether that work was worth what it cost.

Explore this section in the full chapter →

Topologies

An agent may work alone with tools, pass work through a pipeline, or delegate to parallel workers. Each arrangement changes context reuse, latency and coordination. None is reliably the cheapest: isolating a subtask can avoid replaying irrelevant context, while unnecessary delegation can duplicate it. Even a single agent can accumulate substantial input consumption by repeatedly sending its growing history.

Explore this section in the full chapter →

The operating inventory

This is the inventory I would keep for the layer; check your own stack against it for lines nobody owns.

Component	What it is	Consumption scales with	Bill shape
Model gateway	Routing, retries and fallbacks	Traffic × retries at the gateway	Gateway tier, request charges or self-run compute; model charges remain separate
Prompt cache	Reuse of identical prompt prefixes	Repeated prefixes within the cache lifetime	Premium on writes, discount on reads; reads may not count toward rate limits[17]
Session store	Conversation state with time-to-live expiry	Concurrent sessions × retention	Stored state and infrastructure; expiry is a policy choice
Memory service	Extracted facts, episodes and instructions	Writes × model calls per write; retained history	Extraction and embedding calls, retrieval queries, storage under the retention policy
Vector database	Similarity search over embeddings	Vectors × dimensions × replicas, plus query rate	RAM-heavy nodes or hosted tiers
Graph database	Entities and relationships, sometimes with validity intervals	Entities × traversal rate; superseded facts if kept	Licensing varies by engine; self-hosting is an operations obligation
Tool/MCP servers	Structured access to external systems	Agent activity × tools per step × retries	Per-seat or per-call hosted, or self-run processes; downstream charges separate
Sandbox runners	Isolated execution for generated code	Arrival rate × holding time, including waits	Compute time or session charges; startup and expiry affect cost
Observability	Traces of agent steps and tool calls	Agent activity × sampling rate	Ingestion plus storage; sampling is a cost decision

Explore this section in the full chapter →

[17] Anthropic, “Prompt caching” and “Rate limits”, checked October 4, 2026. Five-minute writes at 1.25 times the base input price, one-hour writes at 2 times, reads at 0.1 times (less on some newer models); hits need identical prefixes; on most models, cached reads do not count toward input-tokens-per-minute limits. Provider-specific and subject to change.

What the review inherits

Agent spend has the shape of Volume II, Chapter 15's long tail — thousands of small autonomous decisions, no single owner watching each one — inside the shape of Volume II, Chapter 17's relationships, because the workflow owners are the people who know what the agents should be doing. I would have the capacity team build an attribution record first, tying each event on each system to an agent, workflow, customer and task. With it, the monthly review in Volume II, Chapter 22 can ask what the agents completed and what that work cost across every layer, and the workflow owner can say what it was worth.

Questions I would bring to that review:

- Which events, on which systems, carry no agent, workflow or customer?
- What did an accepted task cost across all layers, counting failed and abandoned attempts, and how wide is the spread?
- Which resources are held rather than used, and what releases them?
- How many attempts can one failure produce, and where is the retry budget enforced?
- What does the memory store keep, and does deletion reach its indexes, history and backups?

Explore this section in the full chapter →

CHAPTER 10 / 4 MINUTE BRIEF

Specialized Hardware

Read the full chapter → / Read this brief online →

The main idea

A hosted model is not the only way to serve an AI workload. Data-placement requirements, latency or sustained volume may justify running a model on rented or owned GPUs, but none of them by itself requires buying hardware: a permitted regional service may solve the placement problem, and a negotiated API arrangement may solve the price problem. Compare those options before taking on a device fleet.

GPU capacity can be metered by instance time, reserved under contract or owned, and each choice changes lead times and operating obligations. The plan needs to know where the devices can run, how much power and cooling they need, which failures remove a whole group at once, and the supplier's current lead time; "measured in quarters" is not a property of every purchase.

Explore this section in the full chapter →

The arithmetic a planner owns

The raw weight payload is parameters times bytes per parameter. A seventy-billion-parameter model therefore needs about 140 GB at 16 bits, 70 GB at 8 bits or 35 GB at 4 bits, using decimal gigabytes. Actual allocation also includes scales, metadata and runtime state. Quantization can reduce memory and alter quality or speed; evaluate the chosen method rather than assuming a fixed quality loss.[18]

For a conventional transformer attention cache, the bytes per retained token are:

$$\begin{aligned}\text{cache bytes per token} = 2 \times \text{layers} \times \text{key-value heads} \\ \times \text{head dimension} \times \text{bytes per value}.\end{aligned}$$

[18] vLLM, "Quantization", checked October 3, 2026. Methods and supported hardware differ by release. Raw weight arithmetic excludes scales, metadata and runtime allocations.

The factor of two counts keys and values. Grouped-query attention can use fewer key-value heads than query heads, and other attention designs need different calculations. PagedAttention addresses cache allocation and sharing; it does not imply that every seventy-billion-parameter model has the same cache size.[19] The memory-only concurrency bound is:

$$\text{memory-only concurrency bound} = \frac{\text{device memory} - \text{weights} - \text{reserve}}{\text{KV cache per request}}.$$

Take a hypothetical configuration with 80 layers, 8 key-value heads, head dimension 128 and two bytes per cached value. The cache per token is $2 \times 80 \times 8 \times 128 \times 2 =$ **327,680 bytes**. At 8,000 retained tokens, a request needs 2,621,440,000 bytes, or about **2.62 GB**, before allocation overhead. Suppose 70 GB of weights fit in 141 GB and we reserve 14.2 GB for other use: $141 - 70 - 14.2 =$ **56.8 GB** remains. The memory bound is $\left\lfloor \frac{56.8}{2.62144} \right\rfloor =$ **21 requests**. With 64 key-value heads instead of eight, each cache would be eight times larger and the bound would fall to two. These invented configurations show why head count belongs in the calculation, not whether either workload meets its latency target.

Explore this section in the full chapter →

What must be measured

There is no universal factor that turns a month's token count into the number of GPUs to buy. The conversion requires a measured configuration, task distribution, latency requirement, availability arrangement and demand pattern. A serving team may know that long contexts change the bottleneck; a platform team may know that the required devices cannot be placed together by launch. Establish those conditions before negotiating a price.

Explore this section in the full chapter →

[19] Woosuk Kwon and colleagues, "Efficient Memory Management for Large Language Model Serving with PagedAttention," SOSP 2023, https://arxiv.org/abs/2309.06180. Describes the cache memory lost to reservation and fragmentation under contiguous allocation, and paged allocation as the remedy. Its OPT-13B example applies the same per-token formula: $2 \times 5{,}120 \times 40$ layers $\times$ 2 bytes, about 800 KB.

An experiment on shared hardware, before the fleet

The chapter opens on the arithmetic a purchased fleet needs, and that arithmetic starts with a serving experiment. The experiment itself does not need purchased or reserved hardware. Three shared services cover the step: an interpretability service that runs intervention code on models already deployed on shared GPUs, and two rented options — a GPU pod billed by the second, and a function platform that runs custom code on a GPU it selects. Both rented options charge for the time the work uses.

Explore this section in the full chapter →

The account and the meter teach the same lesson Chapter 2 teaches about elasticity: per-second or per-hour capacity keeps an experiment's cost proportional to the experiment. The fleet decision still needs everything this chapter sizes — goodput, failure groups, usable fraction — but the evidence arrives first, on someone else's capacity.

Explore this section in the full chapter →

The sizing process

Measure, model, decide, and come back afterward: the loop Chapter 4 ran for web workers. The serving experiment establishes real rates at the real configuration. The model connects demand to devices: concurrency, cache budget, latency targets, growth. The decision compares purchasing arrangements, failure conditions and lead times before committing to supply.

Explore this section in the full chapter →

The process should leave a record that someone else can check and rerun:

- **Workload:** peak arrival rate, prompt and output lengths, TTFT and TPOT objectives and the share that must meet them.
- **Configuration:** model, precision, engine version, device, VM size, devices per replica.
- **Evidence:** benchmark command and date, goodput per device, memory bound, bandwidth floor.
- **Failure and supply:** purchase unit, failure allowance, maintenance plan, supply arrangement, lead time, fallback.
- **Revisit when:** the model, precision or prompt lengths change, goodput falls, or a contract date nears.

Explore this section in the full chapter →

CHAPTER 11 / 5 MINUTE BRIEF

Exotic Hardware: Quantum and Emerging Capacity

Read the full chapter → / Read this brief online →

The main idea

"Do we have a plan for quantum?" is an awkward question to receive in a meeting about next quarter's capacity. It may mean that a research team needs access to a particular machine, that Security needs help with a cryptographic migration, or that somebody read an impressive announcement. Those are three different requests. Answering all of them with "we've got it taken care of" would be efficient only until somebody asked what we meant.

I want a capacity team to take the question seriously without making every promising technology a purchasing requirement. The useful response names what exists, what the organization might do with it, the evidence that would justify a commitment, and the work we should already be doing on ordinary infrastructure: preparing for post-quantum cryptography does not require a quantum computer.

Explore this section in the full chapter →

Quantum, honestly assessed

Quantum error correction is making measurable progress. In Google's Willow experiment, logical error rates fell as the surface code grew from distance three to five to seven: adding physical qubits made the encoded memory more reliable.[20] IBM's published roadmap targets a fault-tolerant Starling system in 2029 with 200 logical qubits and a circuit capacity

[20] Google Quantum AI and collaborators, "Quantum error correction below the surface code threshold," *Nature*, 2024, https://www.nature.com/articles/s41586-024-08449-y. Accessible author account: https://blog.google/innovation-and-ai/technology/research/google-willow-quantum-chip/. The result concerns the demonstrated error-correction experiment, not enterprise workload economics.

of 100 million gates.[21] The first is an experimental result and the second a supplier's target; both matter, in different columns of the evidence record.

Two results deepened the experimental column in October 2026. An error-corrected machine runs some operations cheaply and consumes a prepared state, called a **magic state**, to perform the rest; Google's post notes estimates that producing such states can occupy up to 95 percent of a fault-tolerant machine's capacity. On the same Willow processor, a filtering method called magic state cultivation — repeated error checks that keep and grow only the cleanest attempts — cut errors in the prepared states roughly fortyfold at a reported 99.99 percent fidelity, with the authors' own note that demonstration at larger scale remains to be done.[22] Quantinuum, whose systems are available through Azure Quantum, reported a 99.9995 percent logical fidelity on its commercial Helios machine — five errors in a million operations, ten times better than its own result the month before — and ran two error-correction codes on the same hardware at different physical-qubit costs, ten versus up to thirty-seven per logical qubit.[23]

[21] IBM, "How IBM will build the world's first large-scale, fault-tolerant quantum computer," June 10, 2025, https://www.ibm.com/quantum/blog/large-scale-ftqc. Supplier roadmap and explanation of logical versus physical qubits; dates and delivered capabilities must be checked when procuring.

[22] Google Quantum AI, "Bringing the magic: Google breaks open a bottleneck in quantum computing," October 15, 2026, https://blog.google/innovation-and-ai/models-and-research/quantum-computing/magic-state-cultivation-quantum-computing/; the paper, "Magic state cultivation on a superconducting quantum processor," is in *Science*, https://www.science.org/doi/10.1126/science.aee7961. First hardware demonstration of the technique: about a fortyfold error reduction and a reported 99.99 percent T-state logical fidelity on the Willow processor. The post notes estimates that up to 95 percent of a machine's capacity can go to producing magic states and states that demonstration at larger scale remains. A vendor's account of its own result.

[23] Quantinuum, "Quantinuum Breaks Error-Correction World Record by 10x, Validating its Leading Path to Fault-Tolerant Quantum Computing by 2029," October 19, 2026, https://www.quantinuum.com/press-releases/quantinuum-breaks-error-correction-world-record-by-10x-validating-its-leading-path-to-fault-tolerant-quantum-computing-by-2029. Vendor announcement: a 99.9995 percent logical fidelity on the commercial Helios system using a color code, ten times the company's own September 2026 result on the same hardware; the announcement states its Helix code used 10 physical qubits per logical qubit against up to 37 for the color code, and links the result to its planned Apollo system. Quantinuum appears among Azure Quantum's priced providers; the record and the roadmap claims are the company's own.

A **physical qubit** is part of the hardware. A **logical qubit** is encoded information protected using a particular error-correction scheme. The number of physical qubits needed for a useful logical computation depends on error rates, code, connectivity and the operations being performed. Dividing one headline by another, such as a million-qubit resource estimate by the latest logical-qubit count, therefore does not measure the remaining distance.[24]

Explore this section in the full chapter →

For an enterprise capacity decision, qubit count is only a beginning. I would ask which algorithm the workload needs, what accuracy and confidence the result requires, how many repetitions it takes, and what classical preprocessing and verification remain. Then compare the complete run with the best suitable classical approach. A benchmark designed to expose a quantum advantage can be an important scientific result without establishing a cheaper way to complete the company's work.

That is a reason to keep qualified people engaged, not to dismiss the field or to buy a machine on a roadmap's date.

Explore this section in the full chapter →

[24] IBM, "How IBM will build the world's first large-scale, fault-tolerant quantum computer," June 10, 2025, https://www.ibm.com/quantum/blog/large-scale-ftqc. Supplier roadmap and explanation of logical versus physical qubits; dates and delivered capabilities must be checked when procuring.

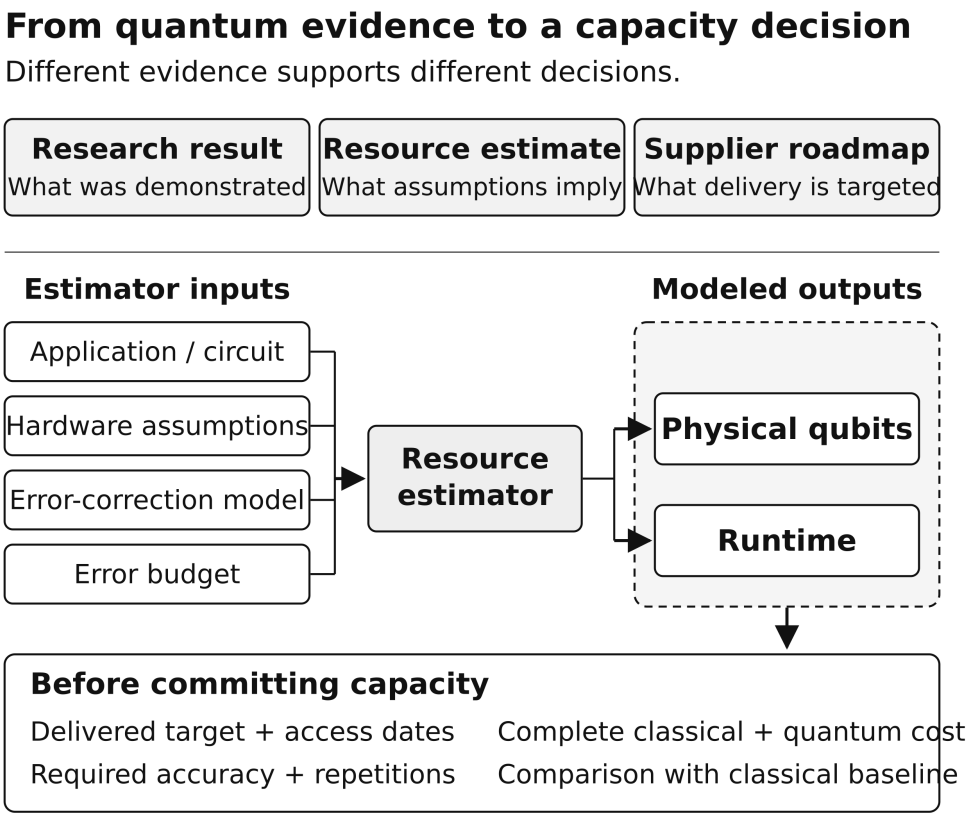


Diagram 11.1. A demonstrated result, a resource estimate and a supplier roadmap support different decisions. The estimator maps application and hardware assumptions, error correction and error budget to physical qubits and runtime. Access, complete cost and comparison with a classical baseline remain separate capacity checks.

Explore this section in the full chapter →

Post-quantum cryptography is a different project

Post-quantum cryptography replaces vulnerable cryptographic mechanisms with ones designed to resist quantum attacks. It includes key establishment and digital signatures, not just encryption. NIST finalized ML-KEM, ML-DSA and SLH-DSA in August 2024; these algorithms run on classical computers.[25] An organization can therefore have a substantial quantum-related migration while purchasing no quantum hardware.

Explore this section in the full chapter →

[25] NIST, FIPS 203, https://csrc.nist.gov/pubs/fips/203/final; FIPS 204, https://csrc.nist.gov/pubs/fips/204/final; FIPS 205, https://csrc.nist.gov/pubs/fips/205/final. Finalized August 13, 2024; key encapsulation and signature standards for classical implementation.

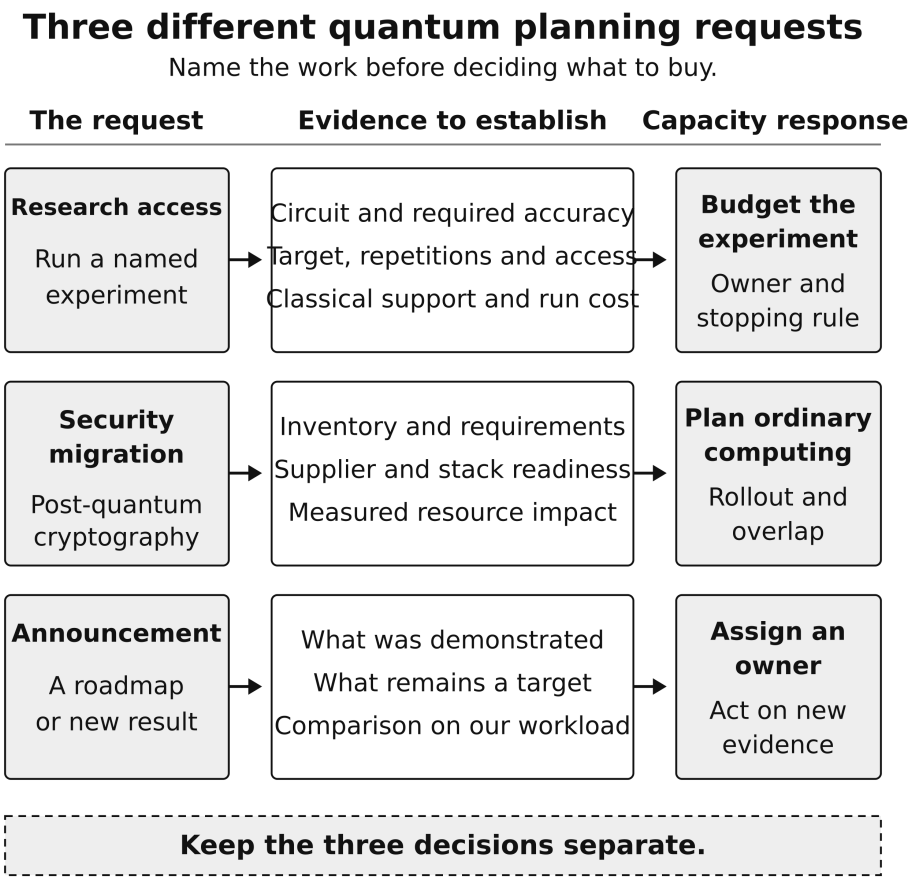


Diagram 11.2. Research access, cryptographic migration and a supplier announcement ask for different evidence and capacity responses. An experiment needs a budget and stopping rule; post-quantum migration uses ordinary computing; a roadmap needs an owner who can recognize when the evidence changes.

Explore this section in the full chapter →

The evidence standard for emerging architectures

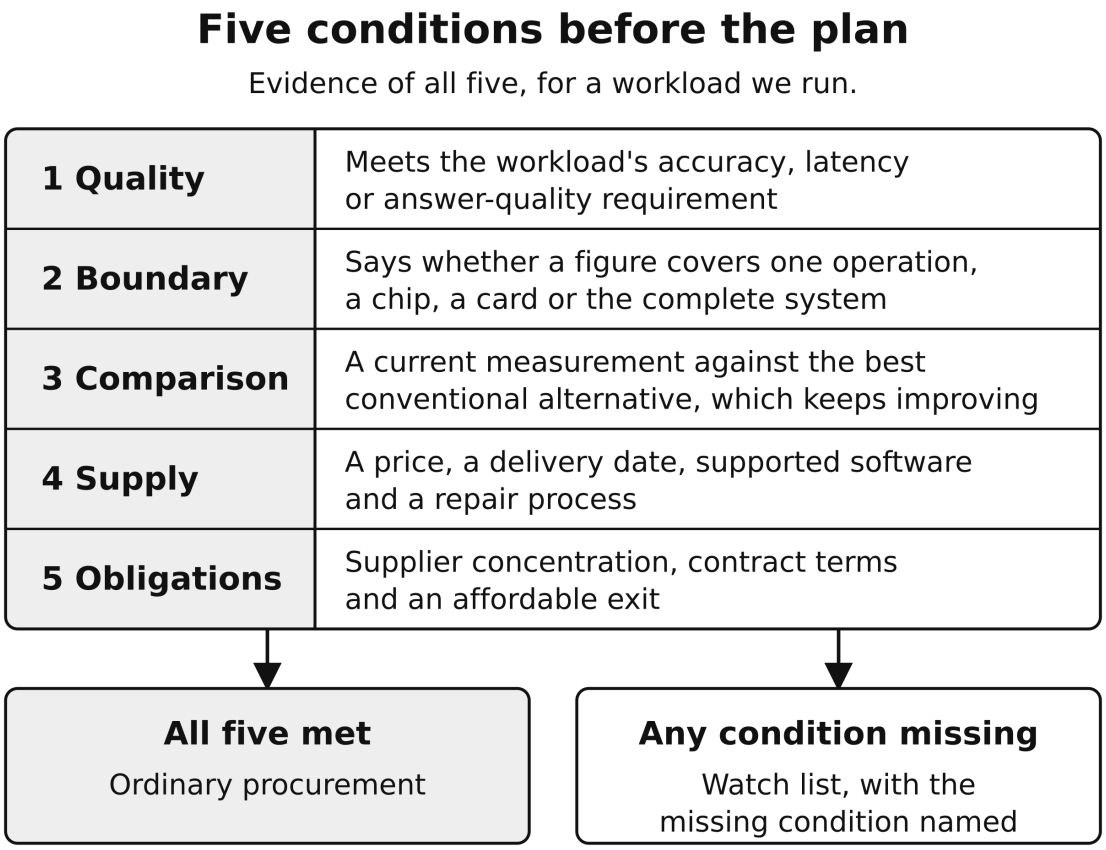


Diagram 11.3. An architecture enters the plan with evidence of all five conditions for a workload we run: quality, boundary, comparison, supply and obligations. A product that meets them all is ordinary procurement; anything short of that goes on the watch list, with the missing condition named.

Explore this section in the full chapter →

Photonic

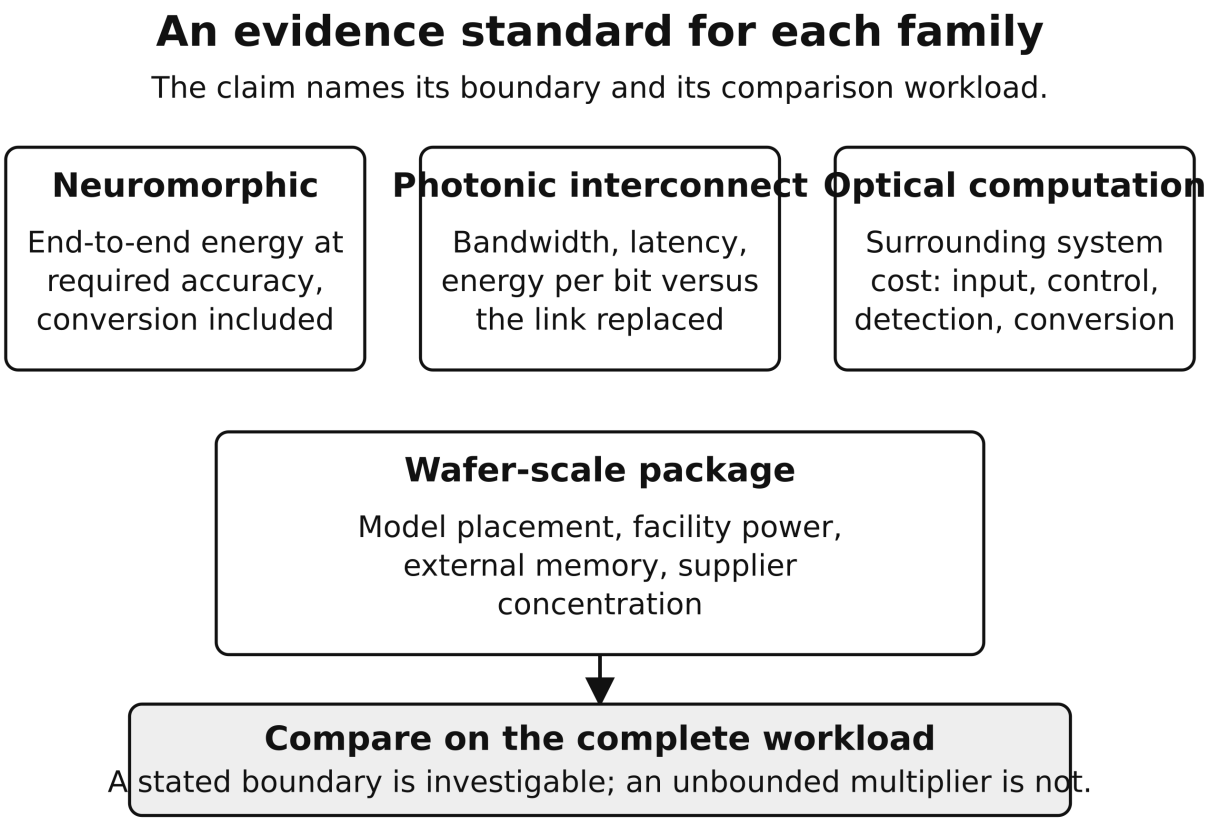


Diagram 11.4. One rule applies to every family: a claim names its boundary and its comparison workload. That means energy at the required accuracy for neuromorphic chips, energy per bit against the replaced link for optical interconnect, the surrounding system for optical computation, and placement, facility power and supplier concentration for wafer-scale systems.

Explore this section in the full chapter →

Quantum annealing

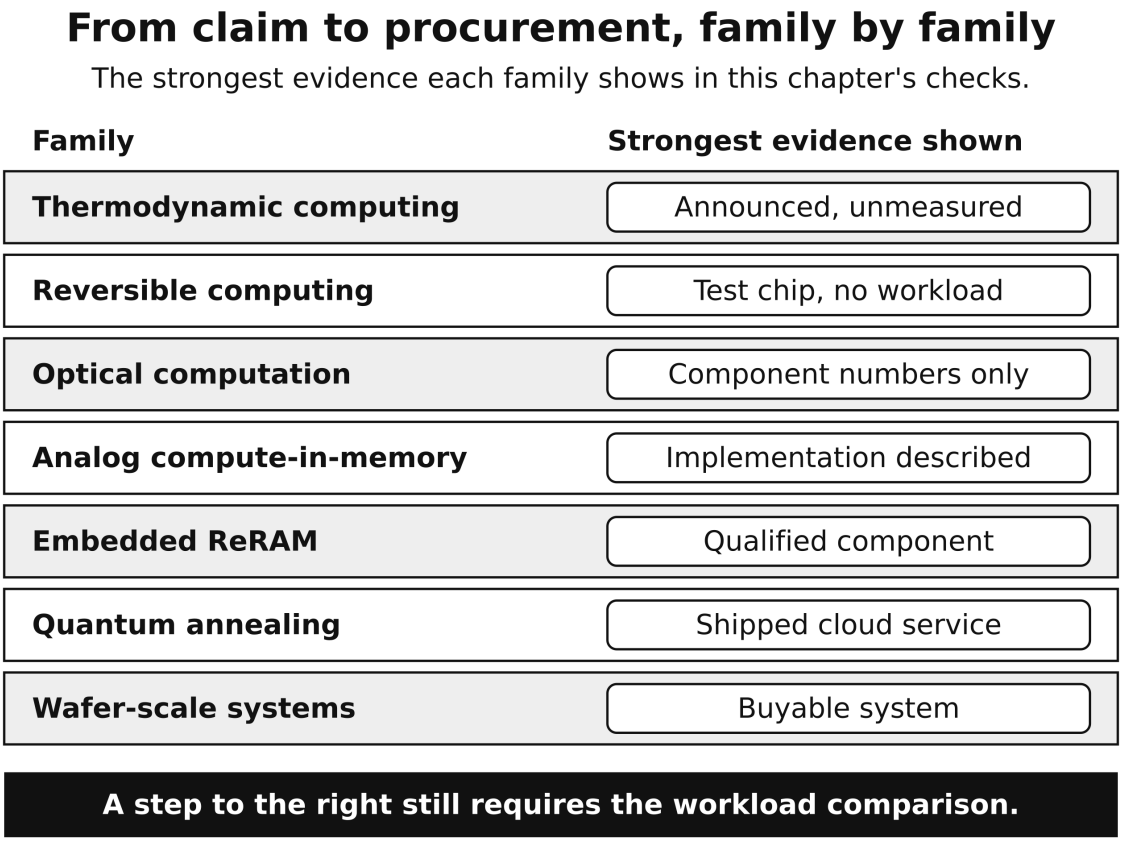


Diagram 11.5. Each hardware family in this section sits at a different evidence stage: announced designs, measured components, a qualified component you can source, or systems you can buy today. Every stage still faces the same workload comparison before it enters a capacity plan.

Explore this section in the full chapter →

Wafer-scale

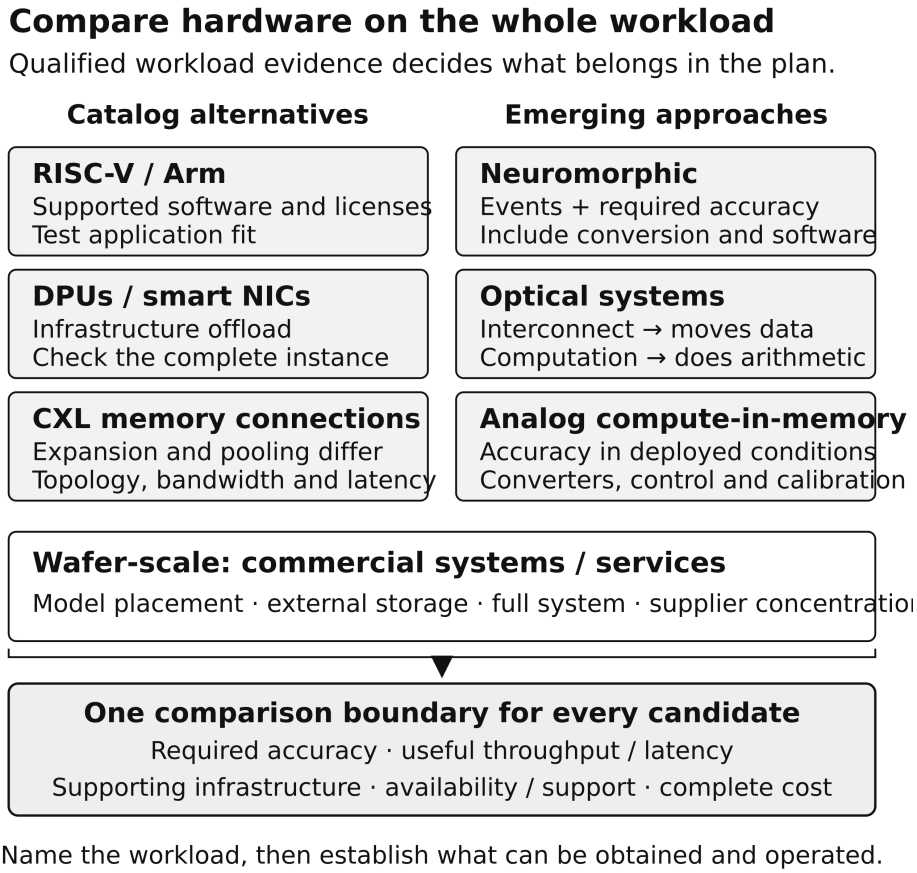


Diagram 11.6. Catalog alternatives, emerging approaches and commercial wafer-scale systems change different parts of a computing system. Every candidate is compared on the complete workload, including support, surrounding infrastructure and cost.

Explore this section in the full chapter →

The watch list

The watch list should say what would change a decision and who will recognize the change. “Quantum is interesting” is too vague to review. “Run a funded comparison when a supplier can execute this circuit at this accuracy within this experimental budget” gives the specialist something to evaluate.

Entry and owner	Evidence to keep	Decision it could change
Quantum experiment (research lead)	Circuit, accuracy, repetitions, classical baseline, access and complete run cost	Whether to fund the next research stage
Emerging accelerator or priced specialist service (platform owner)	Quality and latency on a representative workload, full-system power, price and terms, supported software	Whether to pay for a comparison or procure for a named workload
New memory or interconnect (hardware specialists)	Supported configuration, measured bottleneck, placement and failure behavior	Whether the next pool should use it
Cryptographic migration (Security, with capacity)	Required algorithm and stack, interoperability results, measured resource impact, deadline	When to fund rollout capacity

The last entry belongs to an active migration program when requirements are established; it should not be left waiting for a quantum breakthrough. The others can remain conditional without becoming forgotten.

Explore this section in the full chapter →

CHAPTER 12 / 3 MINUTE BRIEF

Databases and Data-Bound Technologies

Read the full chapter → / Read this brief online →

Plan the work as well as the space

I find it helpful to follow four questions through an investigation. What has to exist, including physical representations, copies, and retained history? What has to happen to it—reads and writes, maintenance, transfer, or recovery? Where can that work happen, given the nodes, partitions, volumes, and failure domains? Finally, when must it finish, and what service has to remain available while it does?

The questions resist being collapsed into one metric because an answer to one may tell us almost nothing about another. Knowing the retained footprint does not establish recovery time; a good aggregate throughput result may conceal the busiest partition. Free disk space cannot be spent as network throughput, however attractively a report combines their percentages.

Explore this section in the full chapter →

Cassandra: the data model is part of the capacity plan

Cassandra's partition key determines how data is grouped into partitions. Rows sharing that key stay together on their replica nodes, so adding nodes does not subdivide the partition. The project's modeling guidance starts with the queries the application needs to support.[26] That makes the distribution of partition sizes and activity much more interesting to me than the result of dividing total bytes by node count.

Explore this section in the full chapter →

[26] Apache Cassandra documentation, "RDBMS design" and "Data Definition", checked October 3, 2026. Query-first modeling and partitions as undivided units of storage; the customer example is hypothetical.

Kafka: retention is a promise to a future reader

A Kafka consumer tracks its position in a partition's log and can return to earlier positions to read retained records again. Consuming a record does not instruct Kafka to delete it. Under the delete policy, retention limits govern deletion, while log compaction preserves the latest value for a key rather than a complete event history.[27] Before estimating storage, I want to know which of those things the customer expects to have available when they return.

Explore this section in the full chapter →

A deletion does not settle the storage question

PostgreSQL updates and deletes leave old row versions for later reclamation. Routine vacuuming makes that space reusable, and it also freezes old transaction IDs; if freezing falls far enough behind, PostgreSQL stops assigning new ones and accepts only read-only transactions until vacuum catches up. Ordinary `VACUUM` generally does not return space to the operating system. `VACUUM FULL` rewrites the table, so it needs room for a complete new copy and an exclusive lock that blocks all other use of the table until it finishes.[28]

Explore this section in the full chapter →

Build a data-capacity decision record

We need a way to carry this work into the forecast without asking every reviewer to become an expert in the technology. The following record preserves the technical differences while exposing the obligations and choices that other teams need to understand.

[27] Apache Kafka 4.3 documentation, "Design", checked October 3, 2026; the downloads page still listed 4.3.1 (June 25, 2026) as the newest release when rechecked October 4, 2026. Replay of retained records; consumption versus deletion and key compaction.

[28] PostgreSQL 18 documentation, "Routine Vacuuming", checked October 3, 2026. Reuse versus release, the steady-state goal, `VACUUM FULL`'s copy and `ACCESS EXCLUSIVE` lock, and read-only operation near wraparound.

Record	What the decision needs
Business obligation	Customer, useful outcome, retention purpose, latency requirement, recovery deadline, and who may change each commitment
Data footprint	Logical versus physical measurements, indexes and logs included, replica scope, growth and deletion assumptions, and placement distribution
Work and limits	Foreground demand, tested bottleneck, maintenance backlog and its deadline, effective transfer or replay rate, and competing work
Operating scenarios	Normal service, maintenance, expansion, a specified failure, recovery, and migration overlap; identify which coincide
Evidence boundary	Engine version, configuration, schema/query assumptions, measurement window, test conditions, and unresolved questions
Decision	Action, cost, resource location, lead time, accountable owner, review trigger, and the consequence if the assumption is wrong

I would not turn the record into a universal utilization score. Cassandra repair, Kafka replay, and PostgreSQL vacuuming need different measurements; what we can share is a way to explain how their results affect a decision. The specialists retain their methods, and the business can see which commitments a proposed change preserves or puts at risk.

Explore this section in the full chapter →

CHAPTER 13 / 3 MINUTE BRIEF

Measuring Value

Read the full chapter → / Read this brief online →

The main idea

One recurring part of this job is being shown an attractive savings number and then making myself less popular by asking how it was calculated. I usually start with staffing, licensing, and the prices used for the alternative: were they list prices or our negotiated rates, and would the supplier recognize them? Those questions are often where an enthusiastic conversation becomes a little more restrained.[29]

I understand the frustration. Someone has found a worthwhile improvement and wants to proceed, and my questions can easily become a referendum on whether I support the team, when I only want to know whether we can rely on the estimate. Often the staffing is covered and the alternative really is more expensive, and saying so strengthens the proposal. The capacity team's own proposals get the same questions.

A savings number is a claim about two states of an organization, and it is only as good as what it counts: a unit that means something, the cost of the route as well as the destination, the kind of benefit and the date it reaches the bill, and the people who will run the result. The comparisons that go most wrong usually leave out the people.

Explore this section in the full chapter →

Choose a unit that means something

The FinOps Foundation describes unit economics as bringing together what an organization spends on technology and the value that spending creates, and it separates resource-efficiency metrics, such as cost per gigabyte stored or per token, from business metrics such

[29] Tim O'Brien, "What's the Secret to FinOps? I'll Tell You Mine—Now Share Yours!", May 11, 2025. The source of the opening's questions about staffing, licensing, and list versus negotiated prices. It describes a recurring experience, not a particular proposal or realized saving.

as cost per transaction or cost to serve.[30] That connection is worth building. The difficulty comes when a view of the whole system becomes the only view permitted in a decision about a particular customer, migration, or recovery obligation.

We need to say when a unit of work is complete and what quality it must meet. A completed customer order, a gigabyte of data kept recoverable for a month, and an accepted AI task are possible starting points, each needing its own definition. Requests, retries, copied records, and generated tokens explain activity within those services, but more activity doesn't mean the business received more value.

Explore this section in the full chapter →

Name the benefit that actually occurred

I keep the names of benefits specific, as the FinOps Foundation does in asking practitioners to separate avoidance, savings, and cost reduction.[31] **Reduced expenditure** means an actual obligation or charge fell within the stated boundary; it is the only one of these that lowers a bill already in the baseline, so it is the only one I would call a saving without qualification. An **avoided purchase**, often reported as cost avoidance, needs an identifiable plan and a credible reason the purchase would otherwise have happened; it changes a forecast rather than a bill. **Released capacity** is available for another use, **shifted cost** is carried by another workload or organization, and **improved service** is a useful change in quality, capability, or timing even where no bill fell.

One action may produce several of these effects; they are not separate savings to add together. Reducing an allocation inside an unchanged commitment can create headroom without lowering expenditure.

Explore this section in the full chapter →

A number worth carrying forward

A useful value estimate states the expected outcome, the costs included, today's spend, and the proposed system's cost during and after the transition. It names who supplied

[30] FinOps Foundation, "Unit Economics", FinOps Framework capability, checked October 4, 2026. Relates technology spending to the value it creates and divides unit metrics into resource-efficiency and business categories, with the examples quoted here. The website and migration examples are hypothetical.

[31] FinOps Foundation, "Unit Economics", checked October 4, 2026. Lists "Separate avoidance, savings, and cost reduction measurements" among the FinOps practitioner's activities. The five benefit names and their definitions here are the author's.

the inputs and keeps uncertain conditions, especially future volume, visible enough to test. For the website migration:

Item	Website migration (hypothetical)
Outcome and unit	Completed transactions, counted once: 25 million a month
Spend today	$2.5 million a month for computing, amortized view
Proposed steady state	$2 million a month for the same outcomes and obligations
Cost of getting there	(4 × $2 million) + $1 million = $9 million in cash; Finance confirms any capitalized share
Benefit	Reduced expenditure of $0.5 million a month at run rate from cutover, realized as old charges and commitments end
Payback	18 months after cutover, 22 from the start; each month of slip moves breakeven five months
Assumptions to test	Constant demand, the replacement's cost at production volume, and the length of the overlap
Owners and reviews	Application, platform, data, and Finance owners; review at cutover and six months later

Explore this section in the full chapter →

SCALE WITHOUT SURPRISE

Visual Companion

Managing Demand, Risk, and Scale

Timothy O’Brien

A shorter route into the book

The main ideas, selected explanations, practical decisions, and chapter diagrams. Follow the links when you need the full argument, worked examples, and qualifications.

Draft edition. Preliminary chapters remain preliminary here.

https://withoutsurprise.com/briefs

Copyright © 2026 Timothy O’Brien. All rights reserved.

Contents

Seasonal Variation 3

The Long Tail and the Spike 7

A Portfolio of Customers 10

The Relationship Discipline 13

Reserve Capacity 16

Managing Risks and Anomalies 19

Commitments Between Organizations 27

Multiple Markets 31

The Monthly Capacity Review 37

What Scale Brings 39

Markets for Compute 42

CHAPTER 14 / 3 MINUTE BRIEF

Seasonal Variation

Read the full chapter → / Read this brief online →

The main idea

When someone asks for a better capacity forecast, I'd first want to understand what's wrong with the one they have. Perhaps the model handles seasonality poorly, or perhaps the organization keeps changing the conditions the model was supposed to describe. If the launch moved, the promotion expanded, or a new feature changed the kind of work each customer performs, those are different problems from failing to recognize a recurring pattern in the measurements.

I want a good model, and I want to know how well it performs, but I also want access to the people who can explain what the business intends to do. Otherwise we can spend a lot of time improving our estimate of what would have happened if the plan had stayed the same. The model may be doing an excellent job within its assumptions while the capacity team prepares for the wrong workload.

There is real mathematical work here, and we'll do some of it. A forecast earns trust when it is scored against what was known on the day it was made, in the unit of the decision it served, with the business's information kept beside the model.

Explore this section in the full chapter →

Start with a comparison we can understand

A **seasonal-naive forecast** gives us a useful starting point: it repeats the latest observation from the matching season, so next Monday's forecast is the most recently observed Monday. Hyndman and Athanasopoulos treat it as a standard benchmark in *Forecasting: Principles and Practice*, with a blunt rule: a new method that can't beat these simple alternatives isn't worth considering.[1]

Explore this section in the full chapter →

Recreate what you could have known

The evaluation has to recreate what we knew when each forecast would have been made: choose a date in the past, forecast using only the information available then, compare the result with what happened, then move the date forward and repeat. Each starting date is a **forecast origin**; the repeated comparison, which can test tomorrow's forecast or one made weeks ahead, is **rolling-origin evaluation**.[2]

Our companion starts after four weeks of history, advances the origin weekly, and compares both methods at one, seven, eight, fourteen, and forty-two days. Eight days corresponds to the constructed launch decision; forty-two days corresponds to the initial supply estimate. I'd want to see the latter before relying on the model for purchasing: predicting tomorrow well says little about a decision whose supply consequence arrives six weeks later. That horizon also has the fewest completed cases, seven, and their overlapping six-week windows make them far from independent.

Explore this section in the full chapter →

[1] Rob J. Hyndman and George Athanasopoulos, *Forecasting: Principles and Practice*, third edition, section 5.2, "Some simple forecasting methods," https://otexts.com/fpp3/simple-methods.html, accessed October 3, 2026. Defines the seasonal-naive method, including its general form for horizons beyond one season, and says a new method that does not beat these benchmarks is not worth considering. The four-week matching-weekday mean and numerical fixture are constructed for this book.

[2] Hyndman and Athanasopoulos, *Forecasting: Principles and Practice*, section 5.10, "Time series cross-validation," https://otexts.com/fpp3/tscv.html, accessed October 3, 2026. Describes evaluation on a rolling forecasting origin, including multi-step horizons. The companion's weekly origins and evidence-availability dates are this book's adaptation.

Put the conversation next to the model

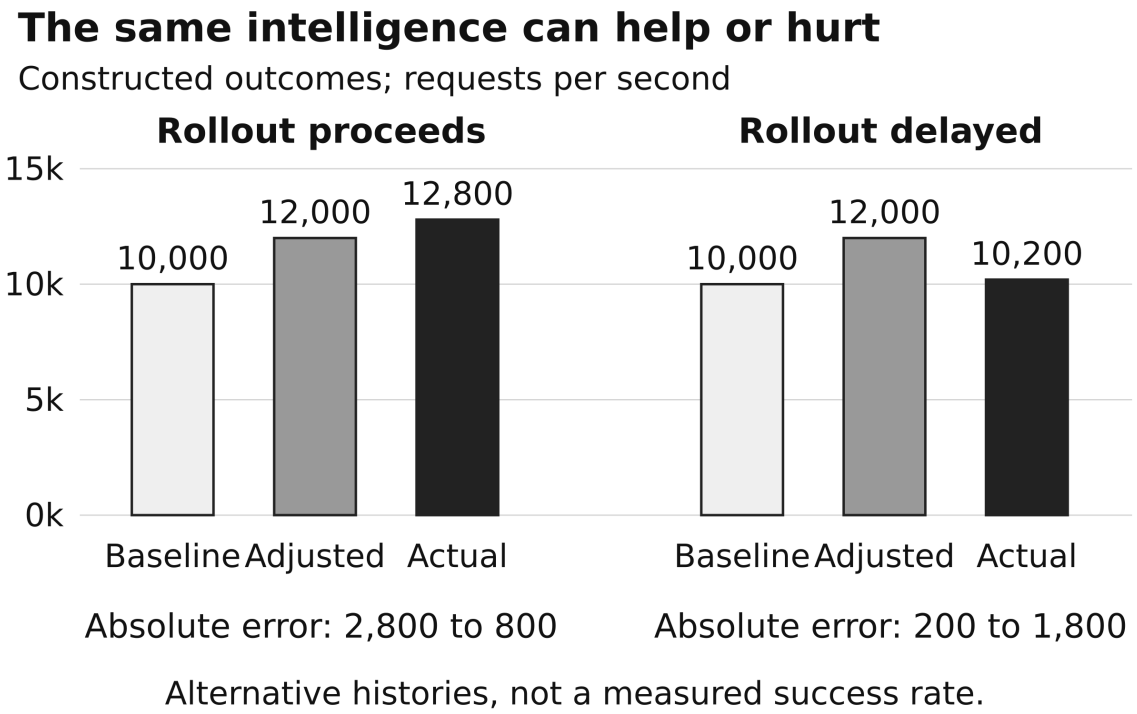


Diagram 14.1. The same event adjustment reduces error when the rollout proceeds and increases error when it is delayed. The bars show two alternative constructed outcomes.

Explore this section in the full chapter →

What the uncertainty changes

A **prediction interval** gives a range with a stated probability under a model and its assumptions. We can check whether a claimed 90 percent interval contains roughly 90 percent of later outcomes at the horizon we care about. A low-demand and a high-demand business scenario give us a different kind of range.[3] Both can help a decision, but our staged 12,000 and full-rollout 14,000 cases don't become the 50th and 95th percentiles merely because those labels would make the chart easier to present. We haven't established those probabilities.

Explore this section in the full chapter →

The discussion becomes useful when we connect those mechanisms to options. A delayed rollout may let us defer the purchase. Delayed supply may mean staging longer, if the business and batch owner can accept the consequences. A changed request mix requires someone qualified to reassess the service rate. Where we have defensible probability estimates, they can inform those choices. Where we don't, the alternatives and their conditions still deserve to be examined.

I sometimes call forecasting magic, partly because of how much judgment can disappear behind the final number. I'd rather let the reader see that judgment: why we chose the historical comparison, what we learned from the business, and where we still have to make an assumption before the decision deadline. That doesn't eliminate guessing. It gives us something specific to revisit when the world behaves differently.

Explore this section in the full chapter →

[3] Hyndman and Athanasopoulos, *Forecasting: Principles and Practice*, section 5.5, "Distributional forecasts and prediction intervals," https://otexts.com/fpp3/prediction-intervals.html, accessed October 3, 2026. Supports the distinction between point forecasts, forecast distributions, and prediction intervals; this book's rollout scenarios are not calibrated intervals.

CHAPTER 15 / 3 MINUTE BRIEF

The Long Tail and the Spike

Read the full chapter → / Read this brief online →

The main idea

If the largest twenty projects account for most of your capacity, you're going to spend a lot of time with those teams: their plans, the technologies they've chosen, and which assumptions are likely to change. That is a sensible use of a capacity team's attention, particularly when a decision in any one of those projects could affect everyone sharing the same constrained resources.

The difficulty is what that understanding allows you to say about the rest. Suppose the twenty have individual forecasts and regular conversations with the capacity team, while another thousand projects appear under "Other" in the review. We might know those smaller projects' current consumption quite well without knowing what could make them change together, and that gap matters even when the aggregate number looks reassuringly small.

Closing that gap doesn't take a thousand more customer meetings. Small projects rarely threaten the plan one at a time; they threaten it when something they share, such as a platform default, a common rollout, or a business event, moves many of them at once. That is the level at which to give the remainder an owner and an investigation threshold.

Explore this section in the full chapter →

Measure concentration before naming its distribution

If we want to make a power-law claim, there's more statistical work to do. One concentration figure can't settle it: some power-law exponent will reproduce any single point on the curve, and so will some log-normal. Clauset, Shalizi and Newman showed that the familiar test, a straight line on a log-log plot, can badly misestimate the exponent and says nothing about whether the data follow a power law at all. Their procedure estimates where the power law starts and its exponent, tests the fit, and compares alternatives; the log-normal often can't be ruled out.[4] I'd keep that work separate from ranking the portfolio, which has to hold up whatever the distribution turns out to be.

Explore this section in the full chapter →

A small change can spend the remaining headroom

Assume we have 1,030 units of usable supply under the scenario's operating conditions. Against baseline demand of 1,000, that leaves 30 units for forecast error and change. The small projects average 200 / 1,000 = 0.2 units each, so an additional 0.02 per project could look unremarkable to anyone reviewing individual consumers.

Now suppose that increase comes from a shared configuration change reaching all thousand small projects. Its aggregate effect is 1,000 × 0.02 = 20 units: 10 percent of tail demand, 2 percent of total demand, and two-thirds of our available headroom.

Scenario	Large projects	Small projects	Total demand	Supply minus demand
Baseline	800	200	1,000	30
Small projects each add 0.02	800	220	1,020	10
Large projects also grow 2%	816	220	1,036	-6

[4]Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman, "Power-law distributions in empirical data," *SIAM Review* 51(4), 661–703 (2009), https://doi.org/10.1137/070710111; preprint https://arxiv.org/abs/0706.1062, checked October 3, 2026. Supports the weakness of least-squares fits on logarithmic plots, the maximum-likelihood, goodness-of-fit and likelihood-ratio procedure, the difficulty of ruling out the log-normal, and exponents typically between 2 and 3. It says nothing about the distribution of any computing workload.

A close forecast for the large projects doesn't protect this plan: 2 percent growth in the head, on top of the tail change, exhausts the margin. Failures and bursts beyond these stated conditions need their own test even when the last column is positive.

Explore this section in the full chapter →

Put an investigation threshold next to the headroom

Response time should influence the threshold. Quickly obtainable, qualified capacity may give us room to learn from a small change. A supply process taking months requires earlier action, and the test should then count everything that could land within the lead time, from scheduled shared changes and expected arrivals to the cohort's recent bias. That's why a low spending rank doesn't settle whether a workload needs investigation: its contribution may be small in the budget and large relative to the room we have left before we can respond.

Once we understand the proposed mechanism, we can discuss an action with someone able to carry it out: stage the common change, obtain supply, qualify an optimization, or accept the exposure through the appropriate business owner. If the application teams can't influence the shared change, telling a thousand of them to use less would leave the relevant decision untouched.

Explore this section in the full chapter →

CHAPTER 16 / 3 MINUTE BRIEF

A Portfolio of Customers

Read the full chapter → / Read this brief online →

The main idea

One of the choices a capacity team has to make is how much work to ask of its customers. A major project that consumes a substantial part of a constrained pool may need recurring conversations with both the engineering team and the people who understand the business, and asking those questions is part of supporting it. Asking the same of every application on a shared platform would spend everyone's time, theirs and ours, on conversations that change nothing.

For a thousand small consumers, I may get much further with a platform owner who understands adoption, can change common defaults, and knows how to reach the members when something unusual happens. Those customers don't need to become capacity planners; the team providing the service may be best placed to understand what they will ask of it.

An experiment needs something else again: close attention for a while, as the organization learns what the system does and decides what it might become. I wouldn't describe that as less mature than a stable forecasting relationship; it suits a different kind of uncertainty. Specialist attention is the scarce resource in all three cases, and this chapter's process spends it deliberately, deciding who gets an individual forecast, who shares a group forecast, and what moves a consumer between them.

Explore this section in the full chapter →

3. Assign a planning treatment

Treatment	What the team maintains	Relationship needed
Individually managed projects	Workload model, business drivers, scenarios, commitments, and forecast revisions	Named technical and business owners
Cohorts of smaller projects	Membership, aggregate baseline, variation, shared drivers, and project-level exceptions	A cohort owner who can reach member teams
Temporary exceptions	A time-bounded investigation of a new, changing, or consequential workload	The person who can resolve the specific uncertainty

Explore this section in the full chapter →

4. Gather intelligence at the appropriate level

- **Record:** event `E-ROLLOUT`, revision 1, known June 25, three days before the forecast it informs.
- **Change and source:** product proposes wider customer access; the product lead supplied the plan.
- **Boundary:** additional demand on the web worker pool, measured as requests per second at the existing request mix.
- **Alternatives:** a full rollout adds an assumed 4,000; a staged rollout adds 2,000. Timing and scope remain negotiable.
- **Overlap:** zero of this increment is assumed to be present in the historical baseline used for this decision.
- **Dependencies:** the rollout proceeds on the stated schedule, the measured request mix remains representative, and the transferred capacity is qualified before expansion.
- **Review and expiry:** review by June 30; no forecast made after July 7 may use this revision.

Explore this section in the full chapter →

6. Promote, investigate, and release

A project or cohort needs closer attention when its growth, forecast error, contribution, or consequence crosses the threshold we've agreed to use. A small workload can qualify before its spend is large, by growing quickly, supporting a critical transaction, or needing a resource with a long lead time, and a suspected shared driver can qualify a cohort before any member is large. Sometimes, in the monthly meetings, a very small project that nobody thinks is a big deal starts giving signals that it is bigger than even its own team thinks. It may not be asking for capacity today, but it is going to need it. In each case I'd want a person responsible for resolving the uncertainty and a deadline tied to the decision, rather than an indefinite designation that the project is being watched.

Explore this section in the full chapter →

There also needs to be a way out of the special treatment. A system can stabilize enough to join an existing cohort, or an investigation can resolve the uncertainty that justified individual attention. We should agree what would bring it back, then release the extra effort. I'd set the release condition below the promotion trigger and require it to hold for two consecutive reviews, so a consumer near the line doesn't change treatment every month, and I'd return its history to the cohort baseline and its open events to the cohort owner. Otherwise every temporary exception becomes permanent, and the central team's limited attention goes to maintaining arrangements nobody has reconsidered.

Explore this section in the full chapter →

CHAPTER 17 / 3 MINUTE BRIEF

The Relationship Discipline

Read the full chapter → / Read this brief online →

The main idea

Whoever can warn us about a change in demand is probably doing something other than waiting for the capacity team's questionnaire. They may be trying to close a sale, decide which customers get a feature first, or work out whether a migration is going to finish at all. Your deadline is another deadline in that person's day, and before answering, they have a perfectly reasonable concern: what are you going to do with the answer?

That concern usually has a history. If what they told us last time was later held against them, or reporting a change earned criticism instead of help, we have taught them that keeping quiet is easier. We can call this a communication problem, but we helped create it.

Most of these people do not report to the capacity team, and nobody can be ordered to mention a cancellation they are still arguing about. What we control is what happens to the information once it reaches us, and that decides whether we hear about the next change in time.

Explore this section in the full chapter →

Make early information worth sharing

I would begin by asking what decision the customer is trying to make, then work backward to the information that could improve our answer. An earliest plausible date, the work likely to change, a credible range, and somebody who can update it may be enough to preserve an option. A central template can help us remember these questions; it should not turn every empty field into a reason to send the request back. If I cannot explain how a field affects this decision, I should be careful about making someone else spend an afternoon filling it in.

The return trip matters just as much. Show the customer the supply range or conditional cost estimate their information made possible, including the part we still cannot answer. After the event, come back with what happened and what we learned, even if their adjustment made the forecast worse. Otherwise, from their perspective, information disappears into a process that sends another questionnaire next month.

Explore this section in the full chapter →

Work through the specialists

A capacity team needs access to specialists who have lived with their systems, although hiring one person who has suffered through a difficult database does not somehow give us mastery of all databases. The tools in the earlier chapters can make it tempting to imagine a central team measuring everything independently. At enterprise scale, I would build partnerships with the people who understand the streaming platforms, databases, AI infrastructure, and applications, then hire central staff who know enough to ask them worthwhile questions. Those are already demanding jobs.

I want the specialists to disagree with me when my question is incomplete. If I ask how much throughput a database can support, the useful answer may begin with a discussion of maintenance that I had not included. A test with maintenance paused can tell us something about foreground throughput, but if the customer needs both activities to happen, we have another experiment to discuss. Similarly, an AI team may have measured how many requests meet a latency target without yet evaluating answer quality. I would rather learn that distinction from the team than discover, after making a commitment, that I had combined two results that never held at the same time.

Explore this section in the full chapter →

When the answer has a sponsor

Claim	Question that makes it inspectable
“Efficiency improved.”	Is the useful outcome unchanged? Were workload mix, latency, quality, durability, maintenance, and recovery conditions comparable?
“This is the demand forecast.”	Is it a forecast, sales ambition, funding request, or commitment? Who supplied the assumptions, and are contrary scenarios preserved?
“We saved money.”	Was cost removed, deferred, reallocated, or avoided in a hypothetical future? Are migration, parallel running, remaining commitments, and shared costs included?

Explore this section in the full chapter →

CHAPTER 18 / 3 MINUTE BRIEF

Reserve Capacity

Read the full chapter → / Read this brief online →

The main idea

Show me a large amount of idle capacity and I want to understand it. I may also want to leave it alone, at least until someone can explain what happens if we take it away. That can be an uncomfortable answer in a cost review, especially when the chart appears to have done the investigation for us: here are the resources, here is the low utilization, and here is the proposed saving.

Perhaps the resources are waiting for a recovery, a customer deadline, or maintenance whose timing is difficult to predict. Perhaps the event that justified them has already passed and nobody has revisited the hold. Both explanations are possible, which is why I begin with the work the reserve protects and the conditions under which it becomes useful. Calling it insurance does not establish that it is adequate; calling it waste does not establish that we can release it.

Explore this section in the full chapter →

Available for what?

Return to the constructed portfolio's web service, whose generic failure groups do not specify a provider's regional or zonal layout. It has 64 qualified workers in four groups of sixteen, at a synthetic 250 requests per second per worker for the specified workload mix and quality. That gives us 16,000 in normal operation and 12,000 after losing one group, against a baseline peak of 10,000.

As Volume I, Chapter 2 showed, a report comparing only normal capacity with demand shows 6,000 requests per second available, but 4,000 of that room disappears with the failed group, leaving 2,000 under the condition we have promised to support. If we offer the whole 6,000 to a new customer, we have spent part of the allowance for failure without deciding to change that promise.

Explore this section in the full chapter →

Borrowed capacity changes the failure calculation

Configuration	Normal (requests/s)	After losing the largest group (requests/s)	Peak-use ceiling
64 workers, four groups of 16	16,000	12,000	75%
72 workers, four groups of 18	18,000	13,500	75%
72 workers, one group of 24 and three of 16	18,000	12,000	66.7%
80 workers, four groups of 20	20,000	15,000	75%
80 workers, five groups of 16	20,000	16,000	80%

The constructed portfolio assumes these rates were tested in each configuration and that a lost group's traffic spreads evenly over the survivors; adding workers to a real system does not establish a proportional increase in throughput. The eighty-worker rows are future configurations that need new supply and the batch workers' return; the last also needs a qualified fifth group.

Explore this section in the full chapter →

Borrowing is an agreement with an end

Volume I, Chapter 4's proposal borrows the workers for thirty days, beginning when they are ready on day two. That covers day 2 up to, but not including, day 32. New supply is estimated to be qualified and ready on day 42, leaving ten days the launch calculation has not covered.

Until day 32 the service survives a group loss at 13,500 requests per second. When the workers go back, it returns to 12,000: exactly the staged demand, with nothing left for forecast error, and 800 short if demand reaches 12,800, the peak in Volume I, Chapter 4's first constructed outcome. That exposure lasts ten days, or thirty-eight if supply slips to day 70. Keeping the workers would close it by changing somebody else's commitment.

Explore this section in the full chapter →

I want the next decision scheduled while alternatives remain available: renew the agreement, qualify another source, constrain demand, or bring the unmet obligation to someone who can change it. Work back from day 32 by the lead time of the slowest alternative we still want. Reminding everybody on day 32 leaves them solving the problem on the day the solution is required.

Explore this section in the full chapter →

Decide what would permit release

A reserve can outlive its original purpose remarkably easily if nobody defined an ending. A launch finishes, a migration changes, a customer departs, or the recovery design improves, while the allocation looks exactly as it did before. When authorizing a hold, agree what evidence would permit its release so that the next review can ask a concrete question.

Explore this section in the full chapter →

CHAPTER 19 / 4 MINUTE BRIEF

Managing Risks and Anomalies

Read the full chapter → / Read this brief online →

The main idea

Earlier in my career, I had a small business building a search engine. We had a primary system and a backup, which was considerably less reassuring after lightning struck the building and took out both. That was effectively the end of the business. I went back to school.[5]

The story takes a few sentences now. What mattered then was that we had prepared for a narrower problem than the one that occurred: having a backup had not given us a recovery we could perform after losing the place that contained it.

That experience is behind the way I use the word **anomaly**: an event outside the ordinary operating assumptions of the business, one that changes what the capacity plan can promise. No capacity team can predict these events, and I would not ask one to try. I would ask it to know, before one arrives, which decisions will be its own, what each would cost, and how long each would take.

Explore this section in the full chapter →

[5]Tim O'Brien, *Leaving the Cloud Isn't for Everyone*, October 31, 2025, "The Risk Premium You're Paying Cloud to Carry." The passage is the author's recollection.

Force: the kinds of events that hit the business

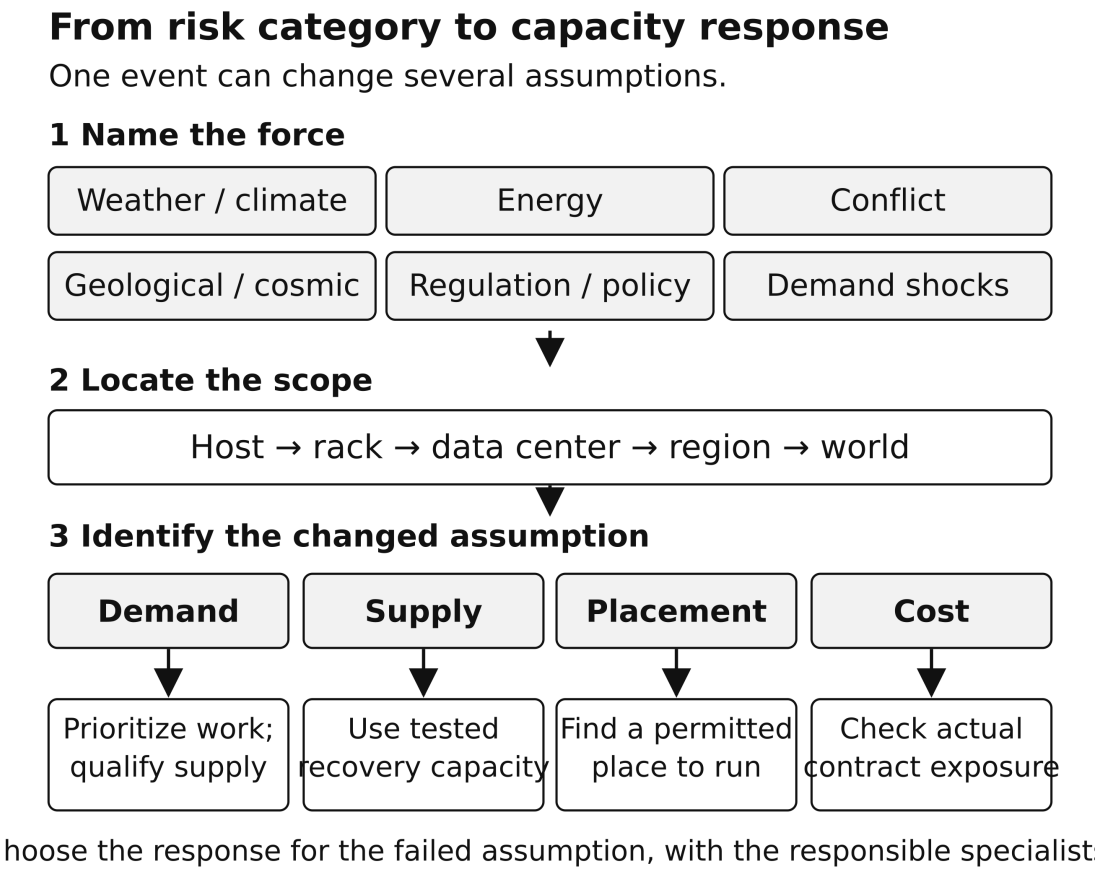


Diagram 19.1. Classify an event by force and scope, then identify which capacity assumptions it changes. The same event can affect demand, supply, permitted placement, and cost; each calls for a different response with the responsible specialists.

Explore this section in the full chapter →

Weather, climate, and the map you inherit

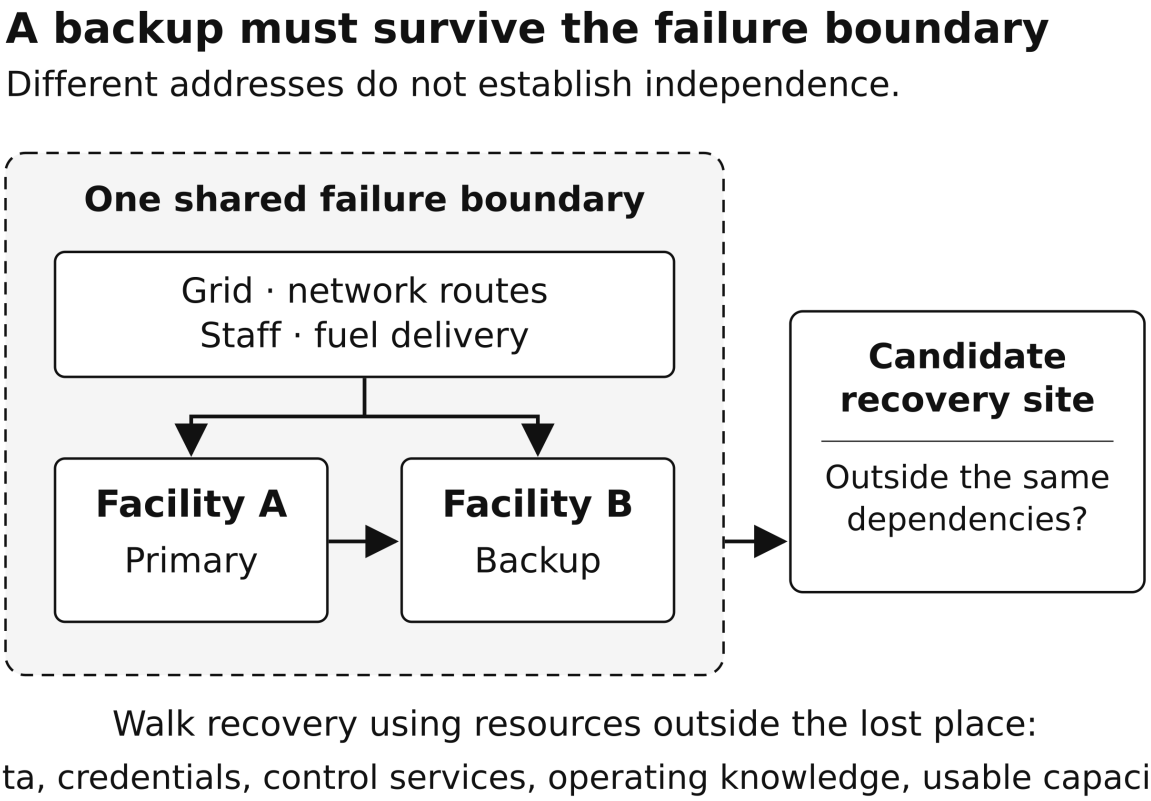


Diagram 19.2. Different addresses do not establish independence. This schematic places two facilities inside one shared failure boundary; a candidate recovery site still needs its dependencies checked and its recovery resources qualified.

Explore this section in the full chapter →

Solar storms: warning and preparation

Solar flares and coronal mass ejections are different phenomena; for the power grid, the one that matters is an Earth-directed coronal mass ejection, or CME. Forecasters can watch a CME leave the Sun. The fastest reach Earth in 15 to 18 hours, and slower ones take several days. Spacecraft near the L1 point, about 1.5 million kilometers toward the Sun, measure the disturbance as it passes, which often gives 15 to 60 minutes of warning.[6] That last interval is far too short to provision and qualify replacement infrastructure. Compare the warning with the actual recovery time, and hold in advance whatever the promised response needs.

Explore this section in the full chapter →

[6]NOAA SWPC, Coronal Mass Ejections, checked October 4, 2026: the fastest Earth-directed CMEs arrive "in as little as 15-18 hours," slower ones in several days; DSCOVR at L1 "can often provide 15 to 60 minutes advanced warning." Typical intervals, not guarantees.

Meteors and the loss of a place

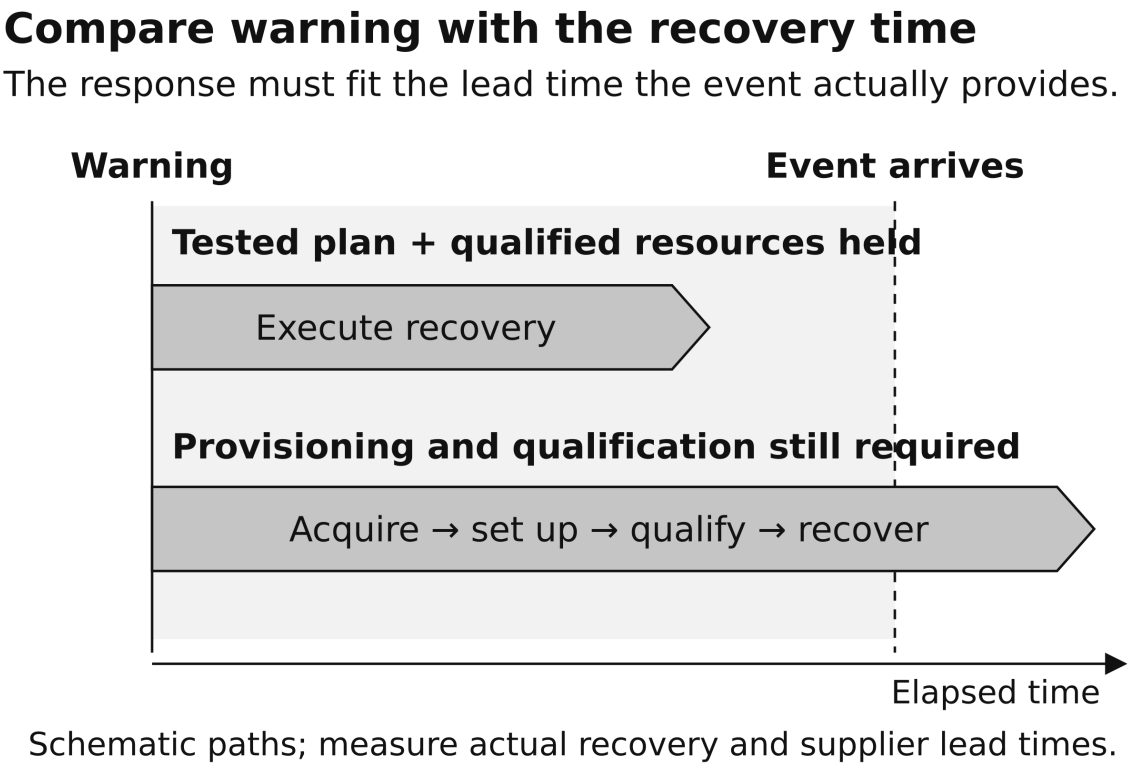


Diagram 19.3. Compare the warning available with the work still needed to recover. These schematic paths illustrate why resources that still need acquisition and qualification cannot be treated as ready capacity; actual response and supplier lead times must be established for the promised recovery.

Explore this section in the full chapter →

COVID, and the anomaly that changes demand

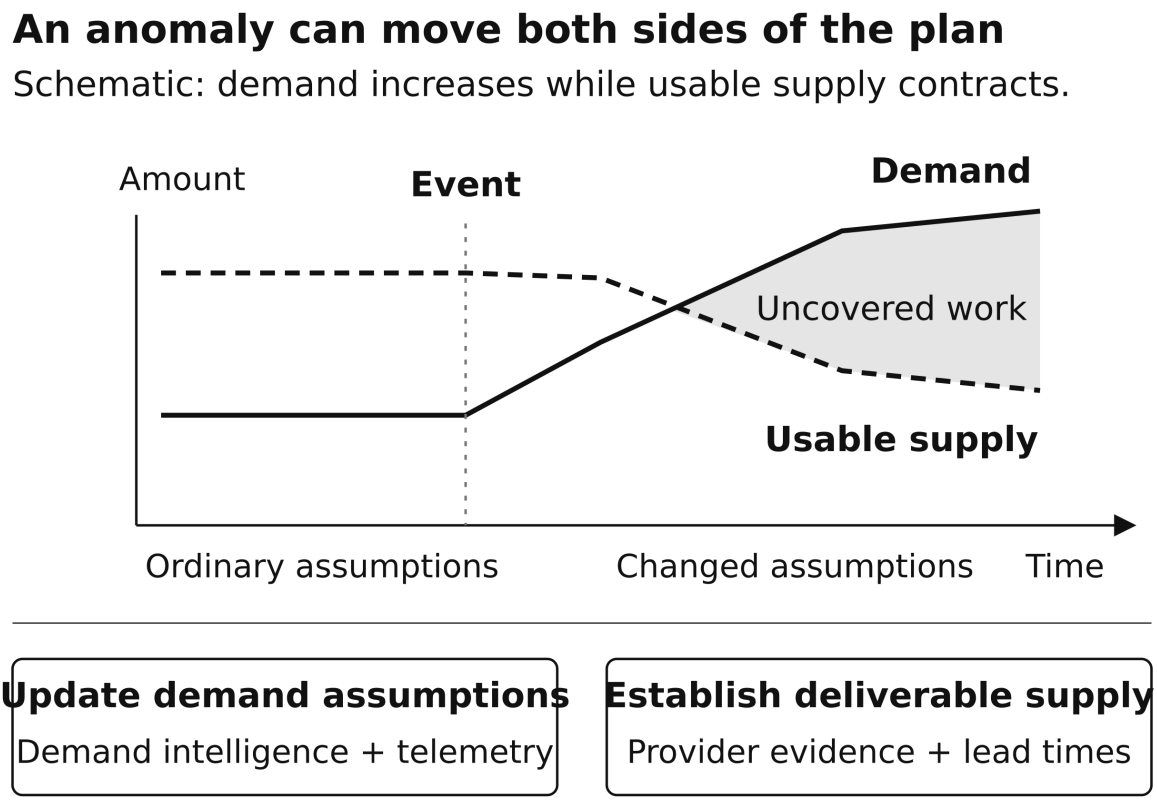


Diagram 19.4. An anomaly can change demand and supply together. This schematic, rather than a measured historical series, shows why updated demand assumptions and evidence of deliverable supply are separate parts of the response.

Explore this section in the full chapter →

The headroom you hold is the failover you can do

This is Chapter 18's failure-group arithmetic with regions as the groups. Take N equal regions sharing the work evenly, each able to sustain a qualified rate C, and let $u_{\max}$ be the highest fraction of C a survivor may carry after one region is lost. Normal utilization u must then satisfy

$$u \le \frac{N-1}{N}\, u_{\max}.$$

With two regions and $u_{\max} = 0.8$, the ceiling is 40 percent: two regions at $0.4C$ send $0.8C$ to the survivor, leaving 20 percent of its qualified capacity free. Letting the survivor run to its full tested limit raises the ceiling to 50 percent. Three regions with the same margin can each run at about 53.3 percent ($3 \times 0.533C \div 2 \approx 0.8C$); at two-thirds each, a loss leaves the survivors no headroom. These are upper bounds under stated assumptions, not utilization targets. Uneven routing, recovery work, a changed request mix, and shared failures each need their own calculation, and no percentage guarantees zero downtime.

Explore this section in the full chapter →

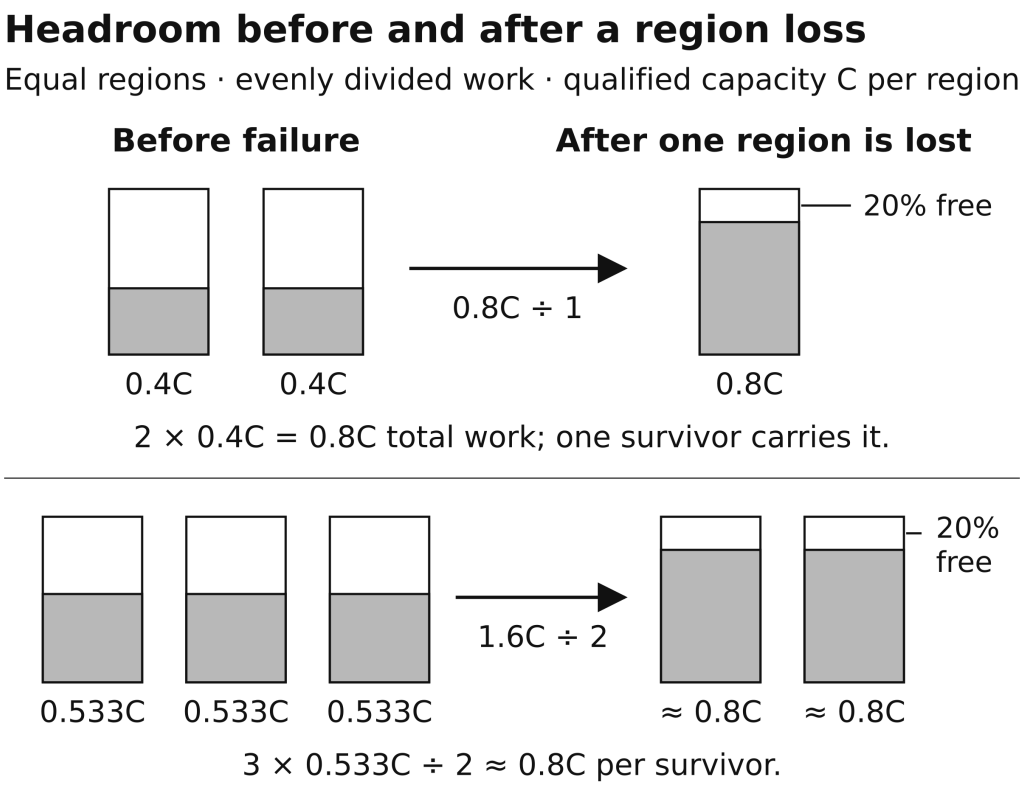


Diagram 19.5. Each outlined bar is a region's qualified capacity C; its gray portion is the work it carries. Under the stated equal-region assumptions, both arrangements leave about 20 percent free after one region is lost. Routing, state, recovery work, and shared failures still need qualification.

Explore this section in the full chapter →

What the capacity team is for, when it happens

The organization should assign ownership of business continuity and disaster recovery explicitly. In the arrangement described here, the capacity team contributes without owning the whole plan: when someone asks “can we handle this?”, we answer alongside the service operators, risk specialists, and business owners whose requirements define what handling it means.

That answer cannot be assembled during an event. Keep a short record for each scenario the organization has decided to plan for:

- **Event and scale**, with the date of the evidence.
- **Affected workloads**, their criticality, and the dependencies they share (Volume I, Chapter 1).
- **Surviving capacity**, and what it is qualified to do (Chapter 18).
- **Recovery deadline**, and the lead time for anything not already held.
- **Permitted destinations**, including legal eligibility (Chapter 21).
- **Provider commitments**, and what they entitle us to in a shortage (Chapter 20).
- **Decision owner and trigger**, and the evidence that would change the response.
- **Accepted gaps**: who accepted them, and when they will be reviewed.

I would write the first record for the scenario the organization is most confident it has covered, because confidence is what stops people asking what two sites share. Its findings go to the next monthly review (Chapter 22) and the next provider conversation.

Explore this section in the full chapter →

CHAPTER 20 / 3 MINUTE BRIEF

Commitments Between Organizations

Read the full chapter → / Read this brief online →

The main idea

"We expect to have it in six weeks" is useful information. It becomes a different kind of information when a project plan turns it into a date and a customer announcement turns that date into a promise. By the time the estimate returns to the capacity team, several people may be relying on a level of certainty that the supplier never offered.

Nobody needs to have intended that transformation. Each person used the best information available to move their own work forward while the original answer's conditions became harder to see. We need those conditions to survive the journey, whether the arrangement is between companies or between teams whose shared company logo conceals different customers, budgets, and obligations.

Explore this section in the full chapter →

A discount and a resource are different things

Azure and AWS arrangements, as of October 2026:

Arrangement	What we owe	What we have when supply is short
Spend commitment: Azure savings plan, AWS Savings Plans[7][8]	A fixed amount per hour for one or three years	No capacity; unused hours expire but are still paid for

[7] Microsoft Learn, "What are savings plans?" and "Automatically renew your savings plan", checked October 4, 2026. Hourly commitment for one or three years; unused hours expire; no cancellation or refund; current savings plan prices apply; renewal turned off by default, with pay-as-you-go rates after expiry. The $10/$7 illustration is hypothetical, not an Azure offer.

[8] Amazon Web Services, Savings Plans User Guide, "What are Savings Plans?", "Compute Savings Plans and Reserved Instances", "Understanding how Savings Plans apply to your usage", and "Returning a purchased Savings Plan", checked October 3, 2026 (return conditions checked October 4, 2026). Hourly commitment with prices fixed for the term; no capacity reservation; no cancellation; unused hours not carried over; the seven-day return conditions.

Arrangement	What we owe	What we have when supply is short
Consumption commitment, such as Microsoft's MACC[9]	A total over the term, perhaps with milestones; any shortfall is charged	No capacity; only eligible spending reduces the balance
Instance commitment: Azure Reserved VM Instance, AWS regional Reserved Instance[10][11]	The reservation price for one or three years, upfront or monthly	No capacity; the discount covers only what we can deploy
Capacity reservation: Azure on-demand capacity reservation, AWS On-Demand Capacity Reservation[12][13]	The on-demand rate for the reserved quantity, used or not; discounts can apply	The reserved size and quantity in one region or zone; Azure's SLA remedy is a service credit

[9]Microsoft Learn, "Track your Microsoft Azure Consumption Commitment (MACC)", checked October 4, 2026. Commitment, milestones, shortfall charged as Azure prepayment credit, and alerts 90, 60, and 30 days before expiry. The page does not address capacity availability; each agreement's terms govern.

[10]Microsoft Learn, "Prepay for Azure virtual machines to save money" and "Transition guide for retired Azure Reserved VM Instances", checked October 3, 2026 (transition guide checked October 4, 2026). Reserved VM Instances are a billing discount, not a capacity guarantee, and can be combined with capacity reservations; capacity priority requires single-subscription scope; purchases and renewals for several older series ended July 1, 2026, with existing reservations running to term. Vendor documentation; terms change.

[11]Amazon Web Services, Amazon EC2 User Guide, "Reserve compute capacity with EC2 On-Demand Capacity Reservations", "Concepts for Amazon EC2 Capacity Reservations", "View the state of a Capacity Reservation", "Capacity Reservation pricing and billing", and "Regional and zonal Reserved Instances (scope)", checked October 3, 2026 (future-dated terms and eligible families checked October 4, 2026). Billing and discounts for Capacity Reservations; zonal versus regional Reserved Instances; future-dated lead times, minimum commitment, eligible families, and states. Product limits change.

[12]Microsoft Learn, "On-demand capacity reservation in Azure" and "Create a capacity reservation", checked October 3, 2026 (overview checked October 4, 2026). Region or zone holds billed at the VM rate whether used or not, with discounts applicable; no SLA for capacity priority; the SLA's worked service-credit example and customer-calculated claims; ND-series and NCads H100 v5 unsupported. Microsoft's SLA text governs, and supported sizes change.

[13]Amazon Web Services, Amazon EC2 User Guide, "Reserve compute capacity with EC2 On-Demand Capacity Reservations", "Concepts for Amazon EC2 Capacity Reservations", "View the state of a Capacity Reservation", "Capacity Reservation pricing and billing", and "Regional and zonal Reserved Instances (scope)", checked October 3, 2026 (future-dated terms and eligible families checked October 4, 2026). Billing and discounts for Capacity Reservations; zonal versus regional Reserved Instances; future-dated lead times, minimum commitment, eligible families, and states. Product limits change.

[14]Amazon Web Services, Amazon EC2 User Guide, "Capacity Blocks for ML" and "Capacity Blocks pricing and billing", checked October 4, 2026. Up to eight weeks ahead and 64 instances per block; no cancellation; price set at purchase by supply and demand, charged upfront; no Savings Plans or Reserved Instance discounts. Supported types and Regions change.

[15]Amazon Web Services, Amazon EC2 User Guide, "Reserve compute capacity with EC2 On-Demand Capacity Reservations", "Concepts for Amazon EC2 Capacity Reservations", "View the state of a Capacity Reservation", "Capacity Reservation pricing and billing", and "Regional and zonal Reserved Instances (scope)", checked October 3, 2026 (future-dated terms and eligible families checked October 4, 2026). Billing and discounts for Capacity Reservations; zonal versus regional Reserved Instances; future-dated lead times, minimum commitment, eligible families, and states. Product limits change.

Dated capacity: AWS Capacity Blocks for ML, future-dated Capacity Reservations[14][15]	Blocks: the full price upfront, no cancellation. Future-dated: the on-demand rate for a minimum term, usually 14 days	The reserved instances for the booked period; a future-dated request can be delayed or declined before it starts

Explore this section in the full chapter →

The internal promise is still a promise

I would keep the agreement short enough that the people relying on it can read it, but specific enough to establish the resource or service, quantity and unit, placement, readiness date, duration, quality or failure condition, and payer. It also needs the authority to change the arrangement and the demand assumption behind it. If the plan works only for a staged rollout, that condition is doing more work than another list of administrative approvals.

Explore this section in the full chapter →

Evaluate the price of certainty

In a simplified pricing illustration, variable purchase costs $10 per unit-period. A take-or-pay agreement offers $7 per unit-period for 100 units over twelve periods, with no resale, cancellation, or other charges. Assume use never exceeds 100 units in any period, so there are no overage charges. We would commit to $100 \times 12 \times \$7 = \$8{,}400$, the same amount that 840 unit-periods would cost at the variable price.

There are 1,200 possible unit-periods, so break-even use is $840 \div 1{,}200 = 70$ percent, which is simply $\$7 \div \$10$. At 600 unit-periods, variable purchase would cost $6,000, so the agreement costs $2,400 more; at all 1,200, variable purchase costs $12,000 and the agreement costs $3,600 less. That $3,600 is the most it can gain, while demand that disappears entirely would lose the whole $8,400. The $7 rate has not changed between the cases; the amount of useful demand has changed its value to us.

Explore this section in the full chapter →

Carry the conditions into the review

Agreement gives us a set of conditions to watch. Product changes the rollout, a new request mix changes the service rate, a supplier revises delivery, or qualification reveals less

throughput than expected, while the borrowed workers approach their return date regardless. Any one of these changes can reopen the combined decision; Chapter 22 turns them into triggers agreed in advance.

Someone has to own those connections, not just the inputs. In the constructed portfolio, the capacity lead maintains the joined view and brings the choice back before the available options expire; the people with authority over the promises still decide which ones can change.

Explore this section in the full chapter →

CHAPTER 21 / 4 MINUTE BRIEF

Multiple Markets

Read the full chapter → / Read this brief online →

The main idea

Once an organization serves several markets, some of its placement rules become conditions that no commercial agreement can change, and two locations with the same machines stop being interchangeable. The earlier examples kept those privacy, sector, and infrastructure rules in the background. Scale makes them harder to coordinate, but a small business using an overseas provider meets them too. Before moving work between locations, the capacity team needs to know which ones its specialists have cleared, and on what conditions.

The example organization from this book has run a European operation for years, under its own budget and its own agreement with Provider B, which is why Volume I, Chapter 1's ledger leaves it out. That bill put the web workers on Provider A, the serving layers behind the data platforms and AI assistant on Provider B, and the customer and order records in the organization's own data center. European customers are served from Provider B's European regions, chosen partly because its data-residence terms satisfied legal review, while Provider A keeps serving the Americas. The same React application, the same tier structure, the same databases behind it, and a topology that looks arbitrary until somebody asks why the European workers cannot simply fail over to Virginia. The duplicated stack shows up in the bill long before anyone mentions the law.

Explore this section in the full chapter →

Location is a capacity property

Once placement is constrained, geography becomes part of what "capacity" means. The tested worker capacity is unchanged, but the locations are no longer interchangeable: a region that could absorb the web tier's failover traffic may be unable to hold the data the tier serves. Chapter 18 asked what a reserve is available for, and each candidate location an-

swers differently. Capacity counts toward a workload only where it is permitted, available, and tested.

Then price the permitted arrangements, including quoted rates, chargeable data transfers, and the standby or duplicated services each topology requires. Volume I, Chapter 1's "where it runs" view was drawn with two providers and a data center; I would add permitted locations and their conditions to it before comparing costs, so that the cheapest row is also one we could operate.

Explore this section in the full chapter →

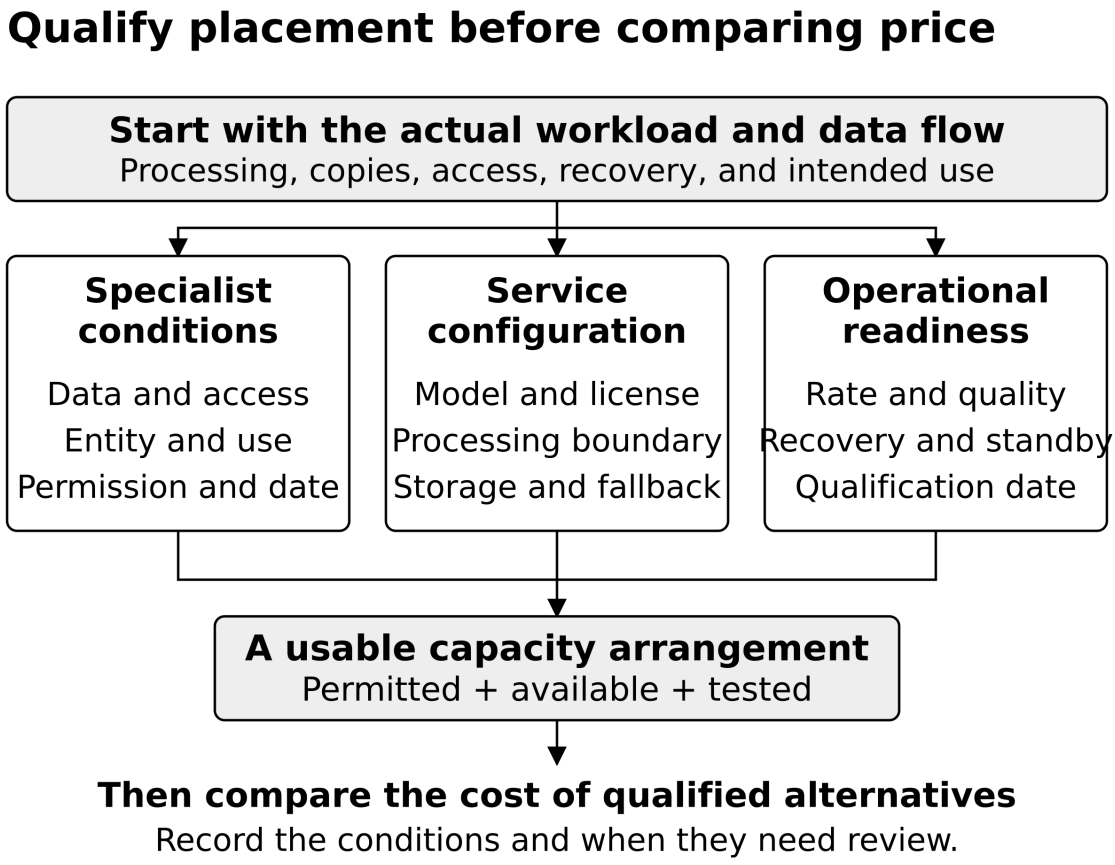


Diagram 21.1. Placement becomes usable capacity only when specialist conditions, service configuration, and operational readiness agree. Compare prices among those qualified arrangements, and retain each condition's review date.

Explore this section in the full chapter →

Consider segmentation by market

Segmentation is an option to evaluate, not a requirement to infer from a country boundary. I would identify which workloads need separate placement or operations, then look for what still connects them: identity, telemetry, support tooling, backups, and the deployment pipeline can each carry data or access across the line. Separation can limit the effect of a local rule change; it cannot guarantee independence. Where lawful sharing is possible, weigh its savings against the cost of separating later.

Explore this section in the full chapter →

Segmentation costs utilization

Separated systems will usually not reach the efficiency of one pooled system. That is a fact of life, and it does not contradict the earlier chapters: deliberately unused capacity can protect a service or satisfy a constraint.

The mechanism is the minimum configuration that availability and recovery requirements impose: replicas, standby capacity, test environments, and the people and systems to operate them. If two deployments must stay separate and each needs its own standby, idle capacity in one cannot meet the other's obligation. Show those duplicated minimums and why they exist. Judging each market against an unconstrained global pool would mistake a placement requirement for an efficiency failure.

Explore this section in the full chapter →

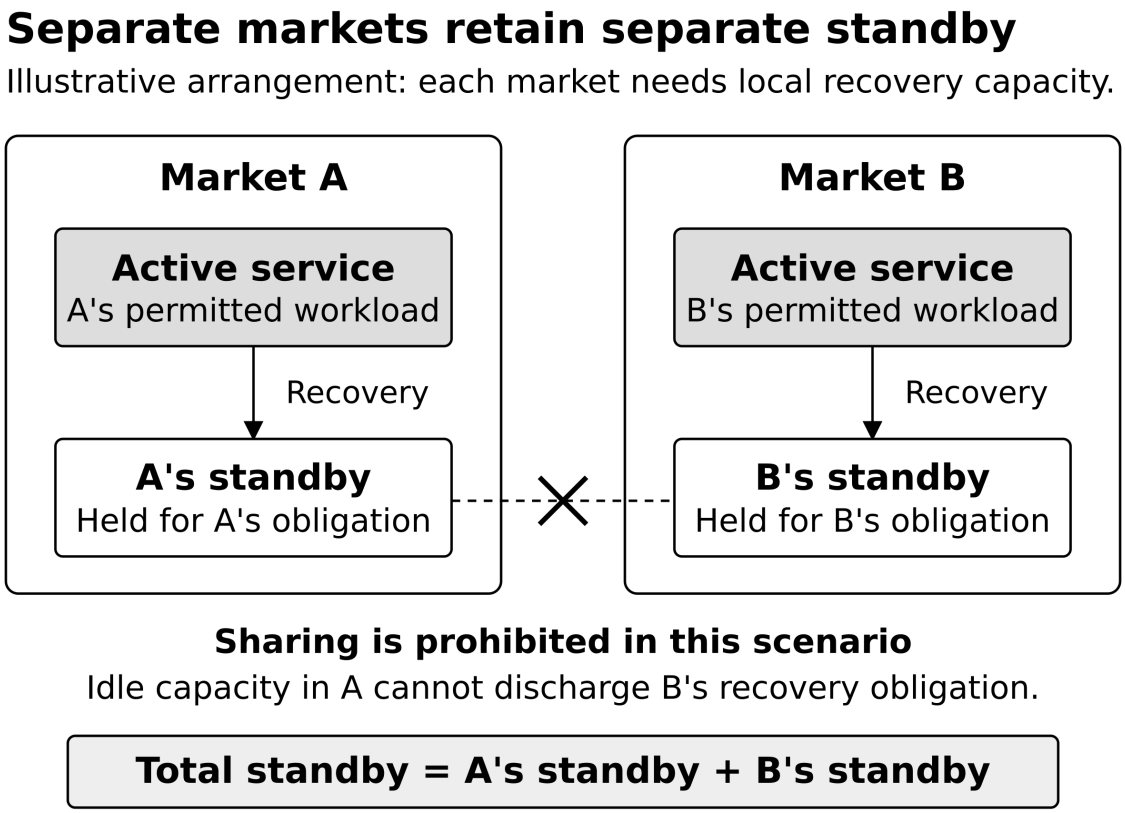


Diagram 21.2. In this illustrative arrangement, sharing is prohibited and each market needs its own standby. The unused capacity protects a local recovery obligation; it cannot be counted again as the other market's reserve.

Explore this section in the full chapter →

The monthly review across markets

Suppose the organization starts measuring the reuse of AI infrastructure across markets, and maximizing reuse becomes a success criterion. Capacity that looks idle on one market's chart may be unavailable to another, and a criterion written as if the deployment were unified will mark correctly segmented systems as failures. So the review keeps the constraint register beside the efficiency numbers, with specialists confirming the conditions behind any cross-market comparison. When a rule or interpretation changes, note its effective date, the options it removes or adds, and the time needed to respond, so nobody discovers an expired condition during a failed recovery or a purchase review.

Explore this section in the full chapter →

One practice or two

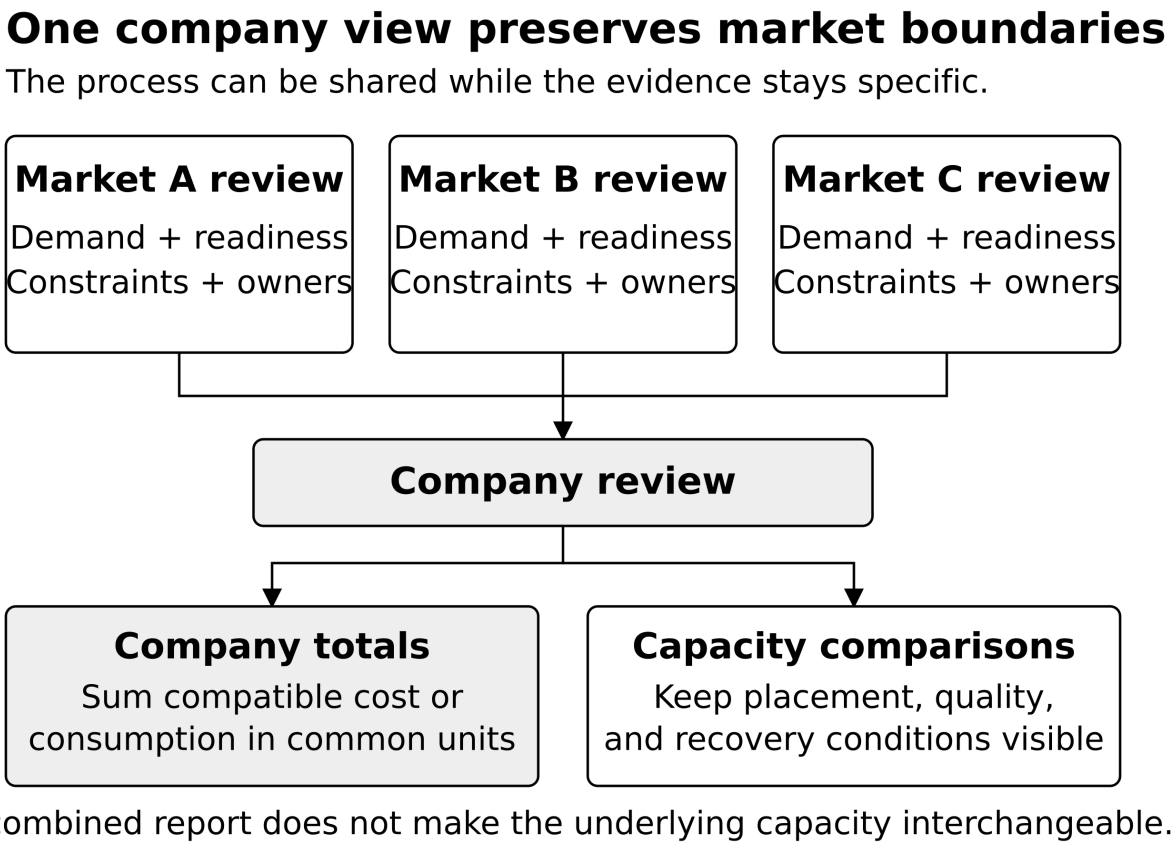


Diagram 21.3. Separate market reviews can feed one company view. Compatible cost or consumption units can be summed; capacity comparisons must preserve the placement, quality, and recovery conditions that make each market's numbers meaningful.

Explore this section in the full chapter →

CHAPTER 22 / 3 MINUTE BRIEF

The Monthly Capacity Review

Read the full chapter → / Read this brief online →

The main idea

The staged launch happened, and the peak in our constructed example was 12,800 requests per second. We had a historical baseline of 10,000 and a business-adjusted forecast of 12,000, so both estimates were low. Assuming the tested request mix and quality conditions still apply, the temporary arrangement supports 13,500 after losing one modeled failure group. That leaves 700 requests per second of degraded headroom, less than half the 1,500 we planned.

We could make this a meeting about a successful launch or a meeting about a forecast that missed. I would rather begin with what the result means for the next choice. The room left is smaller than the amount by which the forecast just missed, another rollout stage is still planned, and the borrowed workers have a return date that our satisfaction with the launch will not move.

Explore this section in the full chapter →

Keep the original forecast

To learn anything from the comparison, we need the baseline as it existed before the conversation, the adjustment, the dates when each became available, and the later observation. Chapter 14 showed what happens otherwise: Product's July 8 revision, which raised the first stage's increment to 2,800 and the full rollout's to 4,500, would make the June 28 forecast look perfect if it leaked in. The revision belongs in the next decision instead.

Explore this section in the full chapter →

Review the decision, not only the prediction

Forecast error tells us something important, but it cannot carry the entire assessment. A conservative estimate might be numerically worse while preserving an option the business values. An accurate average can conceal an unacceptable peak, and an improved demand forecast does not make supply in the wrong location useful. We can even satisfy one customer's resource requirement while imposing a delay on another customer that nobody in the review has examined.

Explore this section in the full chapter →

The agenda

The pack and the meeting share one order, and the time goes where the deadlines are:

1. **Corrections**, first and briefly, starting with the capacity team's own, so that nobody argues a decision on a number we have already withdrawn.
2. **Decisions due before the next review**, in deadline order, each with its record. In an hour, this should take at least half. Each ends with an owner's decision, a stated refusal, or a date by which one will exist.
3. **Exceptions that crossed a threshold agreed in advance**: forecast bias at the decision horizons (Chapter 14); room under the stated failure condition and its trend, where Chapter 19's slow drift in failover headroom appears; and arrangements expiring inside their lead time (Chapters 18 and 20).
4. **The portfolio view**, a few minutes of totals and trends, keeping attribution, enterprise cash, and committed spend apart (Volume I, Chapters 1 and 13), with AI consumption beside the accepted work it produced (Volume I, Chapters 7 and 9).
5. **Carried items**: investigations past their date, and any measurement we keep requesting and never obtain, with what is preventing it and which decision the delay now blocks.
6. **Read back the record**: decisions, conditions, refusals, triggers, and the next update date for each estimate.

Explore this section in the full chapter →

CHAPTER 23 / 3 MINUTE BRIEF

What Scale Brings

Read the full chapter → / Read this brief online →

The main idea

Near its end, Volume I, Chapter 1 multiplied its bill by ten: the same five views, $4,480,000 over thirty days, roughly $54.5 million a year, and a promise that the conversations around those views change at that size. The change I would watch most closely is where a decision lands. Even the smaller ledger has agreements that carry several owners' consumption, but at that size a capacity lead and a finance partner can keep the connections in their heads. At ten times the bill, the same agreements can carry minimum spending terms with shortfall charges in the millions, and a sensible choice by one owner can create an obligation in a budget that owner never sees.

Explore this section in the full chapter →

The portfolio at ten times the bill

Take Volume I, Chapter 1's ledger and multiply every line by ten, holding the workload categories, locations, unit rates, and funding arrangements fixed. This is a bookkeeping assumption for the comparison. It does not predict that a real service can grow tenfold without changing its architecture, performance, or commercial terms. The thirty-day table reads:

Work supported	Thirty-day modeled cost
Data platforms	$1,200,000
AI support assistant	$1,000,000
Transaction-facing web workers	$768,000
Shared platform hosting many small applications	$600,000
Shared network, observation, and support	$396,000
Legacy system being retired	$240,000
New system replacing it	$180,000
Deferrable batch workers	$96,000
Total	**$4,480,000**

On a 365-day basis, $4,480,000 ÷ 30 × 365 = $54,506,667, or about $54.5 million a year. Twelve identical thirty-day periods would instead total $53.76 million. The web pool keeps its shape at ten times the size: 640 workers at the same $40 per worker-day, now 640 × $40 × 30 = $768,000, still arranged in failure groups and still reading from serving layers whose own lines have grown tenfold. The eight compatible batch workers become eighty, and the transfer that moved $9,600 of attribution in Volume I, Chapter 1 moves $96,000 here. The larger transfer deserves the same explanation of displaced work and responsibility.

Explore this section in the full chapter →

What the number buys, and what it costs

A larger budget can improve our negotiating position, but annual spending is not the same as an annual commitment. We should decide how much to promise after examining the stable portion of demand. Volume I, Chapter 1's constructed bill was about 63 percent committed or fixed; multiplying its lines by ten preserves that proportion. It does not prove that larger organizations necessarily commit a larger share. If we choose a higher minimum to obtain a better rate, show the reduced freedom to shrink alongside the saving.

Explore this section in the full chapter →

The obligations grow their own management

Volume I, Chapter 1 described obligations as promises that outlast the reasons we made them. At the small scale, a capable finance partner and a diligent engineer can enumerate the commitments, subscriptions, minimums, and refresh cycles together. At $54.5 million, the enumeration itself is a job: an obligation register, a renewal calendar, usage-versus-commitment tracking, and a reconciliation between what the agreements promise and what the workloads actually consume. Organizations that skip this do not avoid the obligations; they discover them at renewal time, one at a time, in the order the providers raise them.

Explore this section in the full chapter →

Where I would start

The five views still answer the same questions at either size: what the money does, who answers for it, where it runs, how its charges behave, and who funds it. What changes is that each view needs someone to maintain it. The work view needs owners for its largest lines; attribution needs an allocation process with someone accountable for whatever stays unallocated; the location view needs an owner for each agreement; the charge view becomes the register; and the funding view needs program end dates that Finance can see coming. A reader with a smaller budget can use the same questions without adopting an organization built for this chapter's spending level.

Explore this section in the full chapter →

CHAPTER 24 / 3 MINUTE BRIEF

Markets for Compute

Read the full chapter → / Read this brief online →

The main idea

This book has treated capacity as decisions made inside one organization: what we measure, promise, reserve, and commit to pay. This closing chapter looks outward. Compute already has markets; what 2026 added was an attempt to list futures on the rental price of a GPU, so that buyers and sellers could pass price risk to someone else. As of October 4, 2026, none of those contracts was trading on a U.S. exchange. The work they ask of a buyer, defining exactly what is priced and comparing our own price with an outside one, can start now.

Treat the chapter as advanced reading and a direction of travel; nothing here changes the practice of the previous twenty-three chapters.

Explore this section in the full chapter →

A market appears

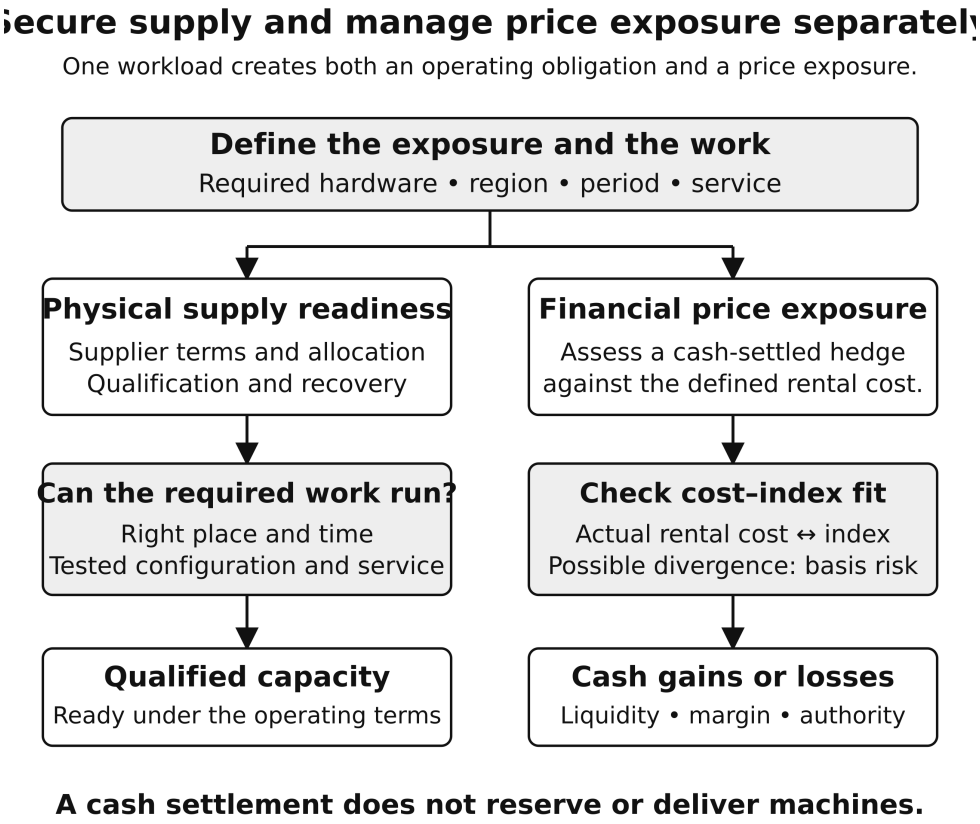


Diagram 24.1. The operating plan secures and qualifies physical supply; the financial plan evaluates price exposure and cost–index fit. A cash-settled hedge can leave basis risk and does not reserve or deliver machines.

Explore this section in the full chapter →

Why exchange trading has been difficult

The first obstacle to exchange trading is heterogeneity. An H100 and a B200 are different resources, and the same chip in two regions or on two networks may support different work. A token buyer is buying a service, not a GPU-hour. A standard contract has to say which differences it includes, which it adjusts for, and which stay with the buyer; Architect's proposed delivery network, for example, would publish basis tables by SKU, memory configuration, and location.[16]

Explore this section in the full chapter →

[16]Architect Financial Technologies, "Architect Financial Technologies Partners with Compute Desk to Launch Exchange-for-Physical Market for GPU Compute", July 8, 2026: the American Innovation Exchange is "forthcoming" and "pending regulatory review"; ComputeConnect would publish "standard basis tables for different SKUs, memory configurations, and locations." Company announcement, checked October 4, 2026.

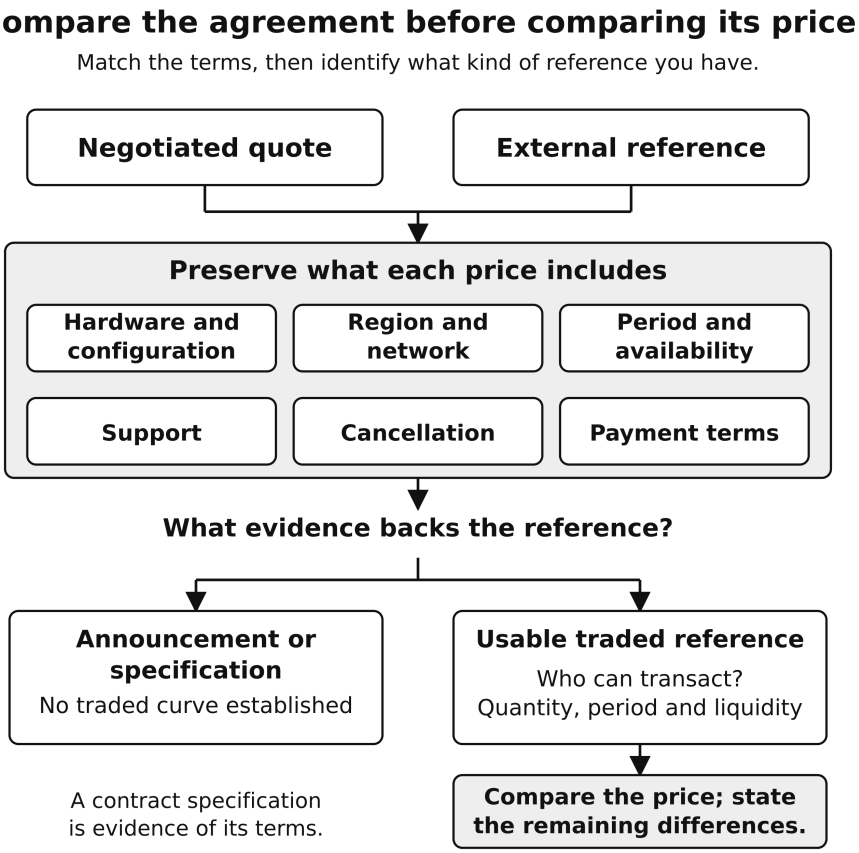


Diagram 24.2. A quote and an outside reference become comparable only after their hardware, configuration, location, period and commercial terms are stated. An announced specification describes terms; a usable traded reference also needs evidence of quantity, period and liquidity.

Explore this section in the full chapter →

What is available and what is announced

Status as of October 4, 2026:

Instrument	What is bought	Price set by	Example (status)
Provider spot	Interruptible capacity now	Provider, without bidding	EC2 Spot (operating)[17]
Prepaid capacity block	Named instances for a future window, prepaid	Provider, from supply and demand at purchase	AWS Capacity Blocks (operating)[18]
Commitment	Lower rates for a spending or usage obligation	Published or negotiated terms	Chapter 20 (operating)
Procurement marketplace	Competing quotes, then a direct supplier contract	Supplier offers	Compute Exchange (operating)[19]
Physical forward	Delivery in a future period at a price fixed now	The two parties	Compute Exchange (offered); OneChronos–Auctionomics (announced)[20]
Resale rights	Permission to sublease contracted time	Other buyers	SF Compute (operating)[21]
Cash-settled index future	Exposure to a monthly average rental index	Exchange trading	CME, ICE, Nodal, Architect (none trading)[22]
Perpetual future	H100 price exposure, no expiry	Crypto-venue trading	Seven venues (trading, thin)[23]

Explore this section in the full chapter →

[17]AWS, "Amazon EC2 Update — Streamlined Access to Spot Capacity, Smooth Price Changes, Instance Hibernation", November 28, 2017, on the change from bidding to smoother provider-set prices.

[18]AWS, Capacity Blocks and pricing and billing documentation, checked October 3, 2026, for the discount and billing rules described.

[19]Compute Exchange, marketplace site, checked October 3, 2026: quotes through a "live auction across verified providers," a services agreement signed directly with the chosen provider, and forward contracts on bilateral terms. Vendor descriptions.

[20]Auctionomics and OCX Group, "Auctionomics and OneChronos Partner on First Tradable Financial Market for GPU Compute", Business Wire, July 29, 2025: price discovery "through bi-lateral forwards"; "first" is the companies' claim. OneChronos, "Compute Markets", checked October 4, 2026: "Coming soon."

[21]San Francisco Compute Company, "About", checked October 3, 2026: clusters leased "on contracts that let our customers sublease." Vendor description.

[22]CFTC, product record 62544, "Approval Pending (90)," checked October 4, 2026; Division of Market Oversight, letter to CME Group, September 21, 2026, extending review of NYMEX Submission No. 26-370 to November 9, 2026, for "novel or complex issues." An extension is not a disapproval.

[23]Loris Tools, "H100 Perpetual Futures", checked October 4, 2026, 21:41 UTC: seven venues, $1.51 million of 24-hour volume, $80,800 of open interest, a $2.647 reference price, and 84.0% of volume on one venue. A third-party, unaudited aggregation of crypto-venue data; an order of magnitude only.

If suitable contracts start trading, I would take four questions to Finance and Treasury first. How closely has the index tracked our actual rate, month by month? How much trades in the months we would hedge? Who has authority to trade? Who funds the margin when the position moves against us? A hedged shortage is still a shortage.[24]

Explore this section in the full chapter →

[24] CME Group, "Learn about Basis: Grains", checked October 3, 2026, and *Self-Study Guide to Hedging with Grain and Oilseed Futures and Options*, on basis (cash minus futures), long hedges, and margin. Applying grain mechanics to compute still requires checking the actual contract.

If we all want the same GPUs

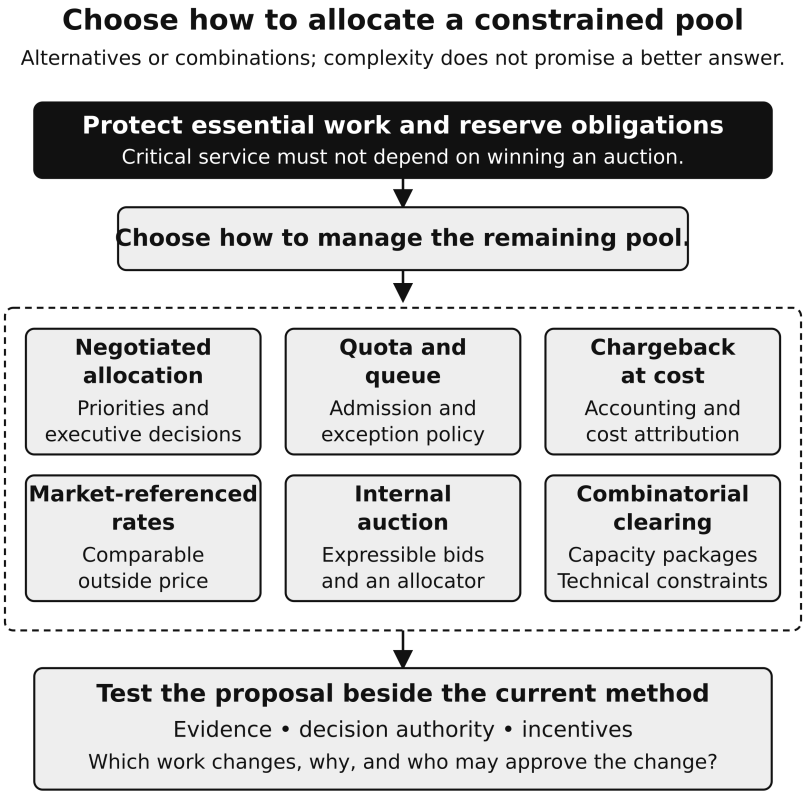


Diagram 24.3. Protect essential work, then choose among allocation and pricing mechanisms for the remaining constrained pool. These are alternatives or combinations; test the proposal against the current method with evidence, decision authority and incentives visible.

Explore this section in the full chapter →

The question the curve answers

None of this has to wait for an exchange. I would take the next GPU commitment due for renewal and write it as one row: hardware and configuration, region, term, price per GPU-hour, minimum spend, cancellation and transfer rights, and payment timing. Beside it go the nearest outside prices for the same period (the on-demand rate, a Capacity Block quote, a published index, a marketplace quote), each with what differs and roughly what the difference is worth. That page goes to procurement and Finance before the negotiation, with one question for Treasury: if a usable contract existed, could we hedge this exposure, and who would fund the margin? After signing, I would record our actual rate beside the index every month. That record is the basis estimate a future hedge will need, and it strengthens the next negotiation whether or not these exchanges ever become deep markets.

Explore this section in the full chapter →