Infrastructure

Capacity planning, from Hyper-V clusters to model routing.

We spent years making sure a cluster could lose a host on its busiest day and carry on. AI systems need exactly the same thinking, applied to model quotas, latency and cost.

Toltec engineering | 25 September 2026 | Updated 28 October 2026 | 3 minute read

The short version

- Plan for the failure on the busiest day, not the average day.
- Headroom is a design decision, not wasted capacity.
- Measure percentiles, not averages.
- Decide in advance what degrades first, and how.
- Treat cost as a capacity limit, because it is one.

N+1 was never about the average

A Hyper-V failover cluster is sized so that when one host fails, the virtual machines it was running restart on the others. That only works if the survivors have room. Clusters were planned around the worst credible moment: a host down, at month end, during the backup window.

AI services have their own version of a failed host. A model provider throttles requests, a region slows down, or a quota runs out halfway through the afternoon. If the design assumes the average day, the busiest day becomes an outage.

Headroom, then and now

On a cluster, headroom meant spare memory, processor capacity and storage throughput, kept free on purpose. For AI systems it means spare quota, in requests and tokens per minute, and a latency budget with room in it. Running a model endpoint at its rate limit is the equivalent of running a cluster with no spare host.

Percentiles, not averages

Storage taught us that averages hide the pain. People feel the slowest requests, not the typical one.[1] Agents make this sharper, because one task can chain several model calls, retrieval queries and tool calls. If ten independent steps each have a one-in-twenty chance of being slow, roughly four tasks in ten will hit at least one slow step. Real dependencies and correlated delays need measurement.

Decide what degrades first

A well-run cluster had an agreed order for switching things off under pressure. AI systems need the same plan: route routine requests to a smaller, faster model, serve cached answers where they are still valid, queue work that can wait, and keep the most capable model for the tasks that need it. Model routing balances task quality, cost, latency, capacity and permitted data routes. A fallback must still meet the task and data-handling requirements.[2]

Cost is a capacity limit

Every model call has a price, and a runaway loop can spend a month's budget in an afternoon. Set limits per task and per day, alert before they are reached, and review the cost of each completed task rather than each call. It is the same discipline as storage quotas, applied to a meter that moves far faster.

What we would do differently

Load test the failure, not just the traffic. Throttle the model endpoint on purpose during a test and watch what the system does. It is the AI equivalent of pulling a cluster node, and it tells you just as much.

References

1. The Tail at Scale Dean and Barroso, Communications of the ACM, 2013

2. Handling Overload, in Site Reliability Engineering Google

Read the current edition: toltec.io/note-capacity-planning.html