Planning Compute Capacity for Growing AI Workloads

As artificial intelligence projects move from small experiments to production-scale deployments, the question of compute capacity becomes central. A model that trains in a few hours on a single GPU may take weeks when data volumes multiply, and inference latency that was acceptable for a prototype can cripple a customer-facing application. Planning compute capacity for AI workloads means matching hardware resources to the actual demands of training and inference, while leaving room for growth without overcommitting capital.

This article walks through the main steps in that planning process, from understanding the workload profile to making procurement decisions that balance performance, power, and cost. Each step includes what to watch for, because the wrong assumption at any stage can lead to underutilised hardware or a system that stalls under load.

Step 1: Understand the workload profile

The first task is to characterise what the AI workload actually does. Not all AI workloads are the same. A large language model training run is compute-bound and memory-hungry, often requiring hundreds of GPUs working in parallel for days. A real-time image classification service, by contrast, is inference-bound and latency-sensitive, needing fast single-sample throughput rather than massive parallel throughput. A recommendation engine that updates models nightly sits somewhere between the two.

Ask three questions: What is the dominant operation (training, inference, or both)? What is the typical batch size and sequence length? And what precision does the model require? Many training runs use mixed precision (FP16 with FP32 master weights) to reduce memory and speed up computation. Inference can often use INT8 quantisation, which cuts memory bandwidth requirements sharply. Knowing these details tells you whether the bottleneck is floating-point operations per second (FLOPS), memory capacity, memory bandwidth, or inter-node communication speed.

Watch out for workloads that shift over time. A team might start with a small model for proof of concept, then scale to a larger model or add multiple models in production. The compute plan should accommodate a range of model sizes, not just the current one.



Step 2: Estimate the throughput and latency targets

Once you know what the workload does, define the performance targets that matter to the business. For training, the key metric is time to convergence: how many hours or days until the model reaches acceptable accuracy. For inference, it is queries per second and tail latency (the 95th or 99th percentile response time). These targets drive the number and type of accelerators needed.

A simple back-of-envelope calculation helps. Suppose inference on one GPU processes 100 samples per second, and the application needs 1,000 samples per second. That suggests at least ten GPUs, but the real number may be higher because of overhead from data loading, pre-processing, and communication between nodes. It is wise to add a safety margin of 20-30 percent to account for these real-world inefficiencies.

Watch out for peak vs. average demand. Many AI applications experience bursty traffic: a retail recommendation system sees spikes during holiday sales, and a fraud detection model sees surges when a new attack pattern emerges. Planning for average load leaves the system vulnerable during peaks. A good practice is to model the load distribution and plan for the 95th percentile, then use scaling or queuing to handle the remaining tail.

Step 3: Choose the right accelerator technology

The choice of AI accelerator technology shapes the entire compute plan. GPUs are the most common choice for deep learning because they offer high parallel throughput and mature

software ecosystems. CPUs are often sufficient for lightweight inference or for workloads that are more logic-heavy than matrix-heavy. Specialised AI accelerators (NPUs, TPUs, or custom ASICs) can deliver better performance per watt for specific models, but they come with narrower software support and less flexibility if the workload changes.

Key factors to evaluate: memory capacity and bandwidth, interconnect bandwidth between accelerators, and the software toolchain. A GPU with 40 GB of memory may fit a 13-billion-parameter model, but a 70-billion-parameter model needs either a single accelerator with 80 GB or model parallelism across multiple accelerators. Memory bandwidth determines how fast the model weights can be loaded for each inference, which directly affects latency. Interconnect bandwidth (such as NVLink or PCIe lanes) matters when data must be shared between accelerators during training or large-batch inference.

Watch out for vendor lock-in. The fastest accelerator is useless if the AI framework does not support it well or if the team lacks the skills to optimise for it. It is often better to choose a well-supported platform even if its peak FLOPS are slightly lower, because real-world throughput depends heavily on software maturity.

Step 4: Plan the server rack and cluster topology

With the accelerator type and count decided, the next step is to arrange them into server nodes and racks. Each server node typically holds multiple accelerators, and clusters combine many nodes. The topology determines how efficiently the workload can scale.

For training, the biggest risk is communication overhead. If the model is split across many GPUs, those GPUs must synchronise gradients after each batch. A high-speed interconnect like InfiniBand or a fast Ethernet fabric with RDMA reduces this overhead. For inference, the topology can be simpler: a load balancer distributes requests across independent nodes, and each node runs the full model. This scales almost linearly with the number of nodes, but it requires enough memory on each node to hold the entire model.



Watch out for power and cooling constraints. A rack of high-end GPUs can draw 10-20 kW or more. Standard data centre racks are often rated for 5-7 kW. The compute plan must include power distribution upgrades, liquid cooling, or both. Ignoring thermal limits leads to throttled performance and reduced hardware lifespan.

Another consideration is total cost of ownership. The purchase price of accelerators is only part of the expense. Power consumption, cooling, maintenance, and the cost of the network fabric add up over a three-year period. A cluster that uses slightly slower but more power-efficient accelerators may have a lower total cost of ownership than one that uses the fastest, hottest chips. Run a TCO model for at least two different configurations before finalising the topology.

Readers exploring vendor options can find more about AMD's approach to this space at amd.com, which details CPU, GPU, and adaptive computing solutions for AI workloads.

Step 5: Build in headroom and a scaling plan

Capacity planning is not a one-time exercise. AI workloads grow in unexpected ways: a model that works well today may need to be retrained with more data tomorrow, or the team may add a new model that has different requirements. The compute plan should include headroom for growth, both in compute capacity and in memory.

A common strategy is to start with a cluster that covers the current peak load plus 30-40 percent headroom, then design for horizontal scaling: adding more nodes to the cluster

without disrupting running workloads. This requires that the software stack supports dynamic scaling, which most modern AI frameworks do. The network and storage infrastructure should be over-provisioned from the start so that adding nodes does not create a bottleneck.

Watch out for software licensing or support costs that scale with the number of accelerators or nodes. Some AI platforms charge per GPU or per server. The cost of scaling can surprise a team that only looked at hardware prices. Include these in the TCO model.

Step 6: Monitor and adjust

Once the system is running, monitoring reveals whether the planning assumptions were correct. Track GPU utilisation, memory usage, network bandwidth, and request latency over time. A utilisation rate below 50 percent suggests over-provisioning; above 90 percent for sustained periods suggests the system is near its limit and may need expansion soon.

Many teams set up automated alerts for utilisation thresholds and use historical data to forecast when capacity will be exhausted. That forecast feeds back into the procurement cycle, turning capacity planning from a one-off project into a continuous process.



Watch out for silent bottlenecks. A high GPU utilisation with high request latency may indicate a memory-bandwidth bottleneck rather than a compute bottleneck. In that case, adding more GPUs may not help as much as upgrading to accelerators with faster memory. Similarly, if GPU utilisation is low but latency is high, the bottleneck is likely elsewhere: data loading, pre-

processing on the CPU, or network I/O. Fixing those bottlenecks often costs less than buying more accelerators.

For further guidance on building AI infrastructure, resources such as amd.com offer technical documentation and case studies on deploying AI accelerator technology in production environments.

Planning compute capacity for AI workloads is a balancing act. Understand the workload, set clear performance targets, choose the right accelerator technology, design the cluster topology for efficient scaling, build in headroom, and keep monitoring. Each step has its traps, but a methodical approach reduces the risk of expensive mistakes and keeps the system ready for whatever the next model brings.