The Founder’s Guide to Scaling AI

IO.NET

The Founder’s Guide To Scaling AI

Why orchestration, system design, and infrastructure strategy matter more than GPU price.

Most people building with AI today aren’t blocked by a lack of ideas or talent. They’re blocked by infrastructure. CIO reports estimate that 88% of AI proof-of-concepts never make it to production. Stanford’s HAI report states that 90% of AI startups fail with the most common reason being financial: they are too expensive to train, too slow to iterate, and too difficult to scale. The models get better. The infra gets in the way.

> *Teams stop experimenting. Founders burn runway without results. Critical sectors like healthcare, logistics, and finance delay adoption, not because the models aren’t ready, but because the infrastructure isn’t. Infrastructure has become the bottleneck of AI. It determines who moves forward and who stalls out.*

This isn’t a “nice to have” issue, it’s an existential risk. As model sizes double and performance demands surge, the infrastructure burden is accelerating faster than budgets can keep pace. Most cloud platforms are optimized for predictable workloads, such as web servers, databases, and scheduled tasks, not for GPU-intensive training or multi-node inference at scale. The result: quota shortfalls, nagging provisioning delays, hidden egress fees, and constrained workflows drive up costs and kill momentum.

The problem isn’t that teams are choosing the wrong model. The problem is that they’re stuck with infrastructure designed for static SaaS companies, not for distributed training, real-time inference, or multi-agent systems.

Here’s what that looks like in practice: A mid-sized AI team trying to fine-tune or benchmark a 13B-parameter model under cost-sensitive time pressure faces delays of hours or days while waiting for GPU availability. Every hour of idle provisioning racks up costs, compounds scheduling complexity, and erodes experimental velocity. In many cases, this hidden infrastructure friction consumes hundreds of thousands of dollars in burn, sometimes more than half of the total project budget.

io.net

Why Infrastructure Hasn’t Kept Up

There are more infrastructure options than ever, but most still assume compute is a fixed resource. The problem isn’t GPU access. It’s orchestration. A TechStrong survey found that 85% of AI teams have experienced project delays due to GPU scarcity, and 39% of those delays lasted between three and six months.

Teams aren’t only training models. They better iterate with fine-tuning, test reinforcement loops, deploy inference APIs, benchmark across architectures, all often in parallel. That complexity requires real-time coordination. Most platforms, however, treat servers like static endpoints. Engineers make manual decisions: selecting nodes, routing jobs, rerunning tasks after failures. That’s not orchestration, it’s operational drag.

And every day, these delays stack up. A study published by Alibaba found even modest GPU fragmentation (unused memory across nodes) can degrade efficiency by 10–20%. That overhead translates directly into longer runtimes and higher costs per experiment. Put it all together, capacity delays, downtime, fragmentation, and most teams are paying far more than the headline GPU rate.

How Orchestration Works & Why It Matters For Scaling AI

Most AI teams focus on model architecture, dataset quality, and evaluation metrics. However, once models become larger, more complex, or move into production, the real bottleneck is infrastructure coordination. This is where orchestration comes in.

Orchestration is the system-level process that matches jobs to compute. It determines how resources are allocated, how quickly a training job starts, and how reliably inference is served. In most environments, this coordination is either manual or rigid. Engineers submit jobs to fixed nodes, monitor provisioning queues, and reroute when something fails. As complexity increases, this process breaks down.

Modern AI development requires running many types of jobs in sequence or parallel. A team might run fine-tuning jobs across different model sizes, serve multiple inference endpoints with varying latency requirements, and spin up short-lived experiments to test new hyperparameters. These jobs require different amounts of VRAM, various GPU types, and varying levels of network throughput. Manually matching each one to hardware slows everything down.

Effective orchestration solves this by abstracting the matching layer. It observes job specs and system state, then automatically routes tasks to the proper hardware. It reduces idle time, eliminates scheduling bottlenecks, and improves cluster efficiency. It also enables faster iteration, since teams no longer wait for manual approvals, static quotas, or backlogged provisioning.

Without orchestration, even the best infrastructure underperforms. With it, teams can treat compute as a programmable resource. That shift unlocks faster development cycles, smoother deployment, and better control over cost.

What A Well-Orchestrated Compute System Looks Like

At a high level, orchestration is about decision-making. But underneath that, it is a set of systems working together to evaluate job requirements, monitor available hardware, and coordinate execution.

A modern orchestrator takes in real-time information about every job request, including the amount of memory required, the types of GPUs that are compatible, whether the job requires local storage or fast interconnects, and its tolerance to latency. It then matches that job against a constantly changing pool of compute resources.

This requires more than a scheduler. It involves a real-time inventory of nodes, performance history, bandwidth availability, and past failure rates. It also requires an execution environment that can handle containerized jobs, support isolated runtimes, and preserve reproducibility across hardware types.

A sound orchestration system does not just route tasks; it also coordinates them effectively. It maintains visibility over everything running in the system. It captures logs, performance data, and failure states.
It makes this information available to the user, rather than burying it inside abstract dashboards or hiding it behind enterprise support plans.

For founders, the impact is direct. When orchestration is handled at the system level, teams are not required to manage provisioning manually. They do not need to build retry systems or cluster schedulers from scratch. And they do not need to sacrifice flexibility in exchange for speed. The infrastructure adapts to the workload, not the other way around.

What Founders Miss About The Real Cost Of Scaling AI

It’s easy to believe compute costs amount to GPU-hour pricing alone. But in practice, several hidden factors significantly raise the actual total cost of ownership (TCO):

- AI workloads impose an “AI tax,” adding roughly 15% to TCO compared with general-purpose servers because of extra stress on storage systems and interconnects.
- Inefficient GPU scheduling in large-scale environments leads to as much as 50% of capacity going unused. Optimized orchestration systems can cut TCO by nearly 40%.
- Monolithic server setups waste around 30% of infrastructure costs. Disaggregated systems have achieved roughly 49% TCO reductions over three years.

These inefficiencies go beyond sticker shock:

- Idle provisioning time still burns budget, even when no work is being done
- Fragmented memory and/or hardware mismatches reduce effective throughput and productivity.
- Rigid server architectures force founders to over-provision compute or abandon innovative experiments.

The right question isn’t “What is the GPU rate?” It’s

- What is the cost per experiment?
- What percentage of capacity is wasted?
- How flexible is the system when workloads shift?

Why $/GPU-hour doesn’t tell the full story

	LLaMA 7B training (1.4T tokens)	QLoRA fine-tuning (13B tokens)	Inference (100K/day)
AWS	~$350K	~$25K	~$5.5K/mo
Lambda	~$190K	~$13K	~$2.3K/mo
io.net	~$120K	~$6–8K	~$1.1K/mo

Teams waste up to 50% of GPU capacity from idle time, fragmentation, or misrouting (Tan et al., 2021).

Where Orchestration Creates Leverage For Scaling AI

As AI teams move from experimentation to production, most infrastructure systems begin to show strain. Orchestration provides a means to manage this transition, enabling founders to maintain high velocity while retaining control over costs and complexity. Below are four areas worth investigating inside your organization:

Job throughput and scheduling bottlenecks

What to investigate: Are jobs regularly delayed due to unavailable compute, long queue times, or manual approvals?

Why it matters: These slowdowns compound. Delays in training or evaluation reduce the number of experiments you can run,
which directly limits how fast your models improve.

Founder tip: Track how many experiments your team completes each week, and how many were delayed or rerun due to infrastructure issues.

Reactive vs. proactive resource allocation

What to investigate: What to investigate: Does your system automatically adjust compute based on job type, urgency, or expected duration?

Why it matters: Static provisioning often leads to either overspending (idle capacity) or under-provisioning (dropped performance). Dynamic allocation aligns resource use with what matters.

Founder tip: Run a resource audit for the last 30 days. How often were large clusters running low-priority or incomplete jobs?

Scaling under variable demand

What to investigate: What to investigate: Are jobs regularly delayed due to unavailable compute, long queue times, or manual approvals?

Why it matters: Without orchestration, scaling tends to lag behind demand, resulting in failed jobs, increased latency, or unnecessary cost overruns.

Founder tip: Simulate a burst load. Measure how long it takes your system to add capacity and how efficiently it scales back down.

Visibility across teams and workloads

What to investigate: Can your team see where compute is being used, by whom, and for what purpose?

Why it matters: Without clear visibility, teams often duplicate effort, over-request resources, or run blind to inefficiencies.

Founder tip: Build a simple dashboard that groups jobs by team and workload. You don’t need perfect tracking; just enough to identify where bottlenecks and overspending occur.

When these patterns are understood and addressed, AI projects scale more predictably. The goal isn’t just automation, it’s clarity. Founders who treat orchestration as part of their scaling strategy unlock faster iteration, better resource efficiency, and stronger alignment across teams

“When compute is programmable, you don’t just reduce cost—you recover time.”

What Scalable Infrastructure Actually Looks Like For AI Teams

Founders often focus on scaling compute volume. More GPUs, faster clusters, bigger training runs. But what matters just as much is how that infrastructure is structured. The difference between a team that scales fast and one that stalls is often architectural, not budgetary.

What to investigate: Is your infrastructure tightly coupled to your application logic? Can you add or remove resources without rewriting pipelines or redeploying models?

Why it matters: Rigid systems make iteration slow and risky. When infrastructure changes require product or model changes, flexible, modular systems let your team evolve how it builds and ships without infrastructure getting in the way.

Founder tip: Map your current stack across training, inference, and experimentation. Ask where changes are blocked by configuration, interconnects, or environment constraints—not by the model itself.

What to investigate: Are you using the same setup for both prototyping and production?

Why it matters: Early-stage experimentation environments tend to be overbuilt or under-optimized for production use. As your team grows, the lack of separation between the two leads to instability and wasted resources.

Founder tip: If your experimentation and production systems are identical, that's a red flag. Introduce at least one layer of abstraction that lets you scale production without breaking workflows upstream.

What to investigate: Do you have observability across your stack?

Why it matters: Without visibility into job performance, resource allocation, and utilization, you're making blind bets. Teams often discover they're overpaying for underperforming infrastructure, months after the fact.

Founder tip: Set a baseline for cost per training hour and latency per inference request. Use that baseline to flag when new jobs deviate by more than 20%.

Scalable infrastructure isn't just about how much compute you can access. It's about whether your system enables you to adapt quickly, safely, and with sufficient context to make informed decisions that keep your product and team moving forward.

Scaling AI Isn't About More Compute. It's About Better Systems

For most founders, the challenge isn't access. It's alignment. The infrastructure you start with often can't support the pace, complexity, or scale of what your team grows into. Jobs run slower. Experiments back up. Teams duplicate work or wait for each other. And the more ambitious your models become, the more this friction compounds.

That's why scaling AI is as much about systems thinking as it is about hardware. The way you coordinate jobs, route resources, track performance, and design workflows will determine how far your team can go.

The good news is, most of these constraints are visible and fixable. When founders pay attention to orchestration, cost per experiment, and architectural flexibility, they stop treating infrastructure as a sunk cost. It becomes a strategic input. Something you can shape. Something you can learn from.

That shift, from managing compute to building systems, is what separates teams that scale confidently from those that stall out early.

What Io.Net Has Built And How It Supports Scaling

Every AI company is solving a different problem. However, the infrastructure challenges tend to remain the same: slow provisioning, unpredictable pricing, delayed feedback loops, and increasing operational drag as models scale.

io.net is focused on solving those problems by building an orchestration-first infrastructure that developers can access directly. Through its global GPU network and developer tools, io.net is helping AI teams eliminate capacity lock-in, reduce experimentation delays, and scale without needing to build their infra from scratch.

Here’s what that looks like today:

- 300,000+ GPUs and CPUs available globally with 5,000+ active GPUs, including A100s, H100s, B200s, and 4090s
- Up to 70% lower cost than traditional providers, with no contract lock-in
- 500,000 free tokens per day for developers testing and running inference on 30+ open models through io.net Intelligence
- Self-serve access through io.cloud, including options for both individuals and teams
- Whether you’re running multi-model training, live inference endpoints, or agent workloads, the system is already built to support that scale
- Teams like Krea, Wondera, and Leonardo.Ai are already deploying on it and have moved faster, with fewer constraints, by shifting orchestration out of their critical path

Teams building on io.net today

wondera

Leonardo.Ai

KREA AI

Start building at io.net