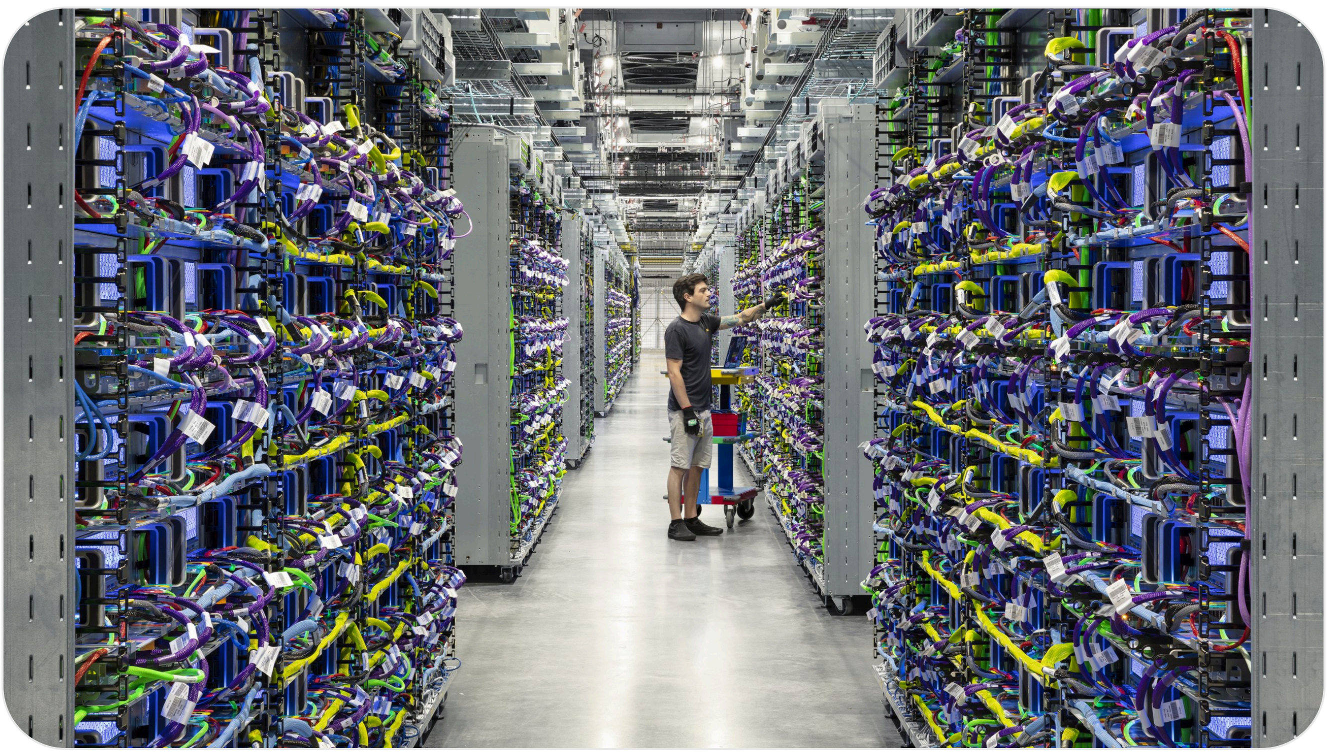
Compute

InstaDeep's scalable reinforcement learning on Cloud TPU

October 19, 2023



Armand Picard
Software Engineer, InstaDeep

Donal Byrne
Senior Research Engineer, InstaDeep

The long-term goal of artificial intelligence (AI) is to solve complex real-life problems. On the path to intelligence, reinforcement learning (RL) — a type of machine learning (ML) algorithm that learns optimal decision-making by interacting with an environment to maximize a reward — has made significant contributions.

Reinforcement learning's first through groundbreaking achievements came through the world of video games, exemplified by the defeat of world champions in games like Go, StarCraft 2 and Dota2 against RL-based systems. More recently, Reinforcement Learning has accomplished remarkable feats in the real-world such as navigating stratospheric balloons, controlling tokamak plasma of a nuclear fusion reactor, and even discovering novel algorithms with AlphaDev.

Despite notable successes, RL is not yet widely adopted for real-world applications. One major factor is the engineering challenge: high-performing RL models often require large-scale simulation and training of the decision-making process. For example, OpenAI Five was trained at an unprecedented scale, using thousands of GPUs over 10 months at a rate of 1 million frames per second [1]. This ultimately enabled its remarkable performance on Dota 2. These results echo the scaling laws recently discovered in Natural Language Processing and Computer Vision, behind the success of PaLM 2. Indeed, "scale" has become an essential component for achieving success in Artificial Intelligence.

In this article, we dive into the scaling capabilities of Cloud TPUs and their transformative impact on Reinforcement Learning workloads for both research and industry. Notably, they played a pivotal role in improving the AI agent behind DeepPCB, InstaDeep's AI-driven Printed Circuit Board (PCB) design product. Thanks to TPUs, it achieved a 235x boost in throughput, slashing training expenses by nearly 90%. In addition to being more cost efficient, TPUs also improved the agent's long-term performance, translating into better quality of the routings generated by DeepPCB, elevating the overall experience of its users.

Scaling reinforcement learning

RL agents learn by acting in a simulated environment to discover how to solve a problem without external examples or guidance. This process of trial-and-error means RL training infrastructure needs to both simulate a vast number of agent-environment interactions, and then to update the agent — often a neural network — based on the gathered experiences.

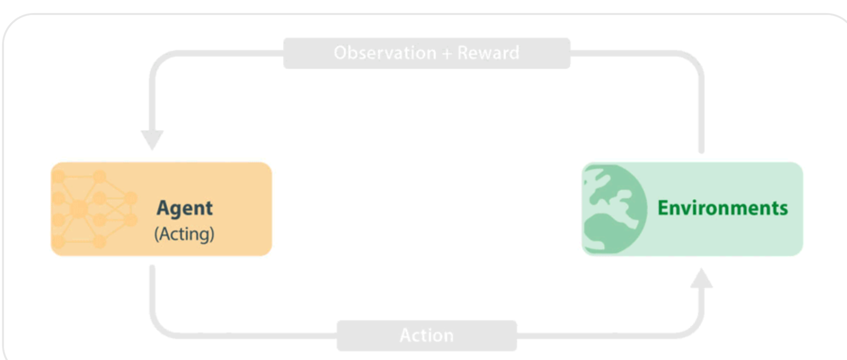


Figure 1: Standard RL training loop. An agent takes in the latest information from the environment (observation and reward) and uses this to choose the next action to take. The action is passed to the environment, which then carries out the next step in the simulation. New information is passed back to the agent which then learns from observations and rewards.

Multiple techniques have been employed to enhance data collection throughput. A popular method involves decomposing the agent into an "actor," responsible for generating and collecting data through an environment, and a "learner," responsible for using the actors data to update the agent. This architecture has its variants, each with distinct pros and cons. For instance, SEED-RL focuses on a centralized inference design, sharing a hardware accelerator for batch inference and learning. This allows SEED-RL to reduce the communication overhead of model retrieval and accelerate acting inference by utilizing hardware accelerators. Another architecture, Menger, takes an alternative approach, focusing on localized inference. Actors are replicated across a pool of cpus to generate data that is then provided to the learner through a sharded replay buffer. This removes the overhead of sending large batches of observations to a centralized server, at the cost of sending model updates to the cpu-based actors.

Overall, the efficiency of an RL system is inseparable from its hardware deployment. An effective distributed architecture must consider multiple factors such as the available hardware (e.g. accelerator type, connectivity, processor clockspeed, shared memory access), the environment complexity (e.g. simulation time), as well as the algorithm itself (e.g. on/off-policy). In order to maximize computational efficiency, the system must address the challenging task of balancing data communication, resource utilization and algorithmic constraints.

What are Google Cloud TPUs?

Tensor Processing Units (TPUs) are purpose-built AI accelerators designed for large-scale, high-performance computing applications, such as training large machine learning models. TPUs stand out due to their optimized domain-specific architecture, designed to accelerate tensor operations underpinning modern neural network computations. This includes high-memory bandwidth, dedicated Matrix Processing Units (MXU) for dense linear algebra, and specialized cores to speed up sparse models.

TPU pods are clusters of interconnected TPU chips. These leverage high-speed interconnects allowing smooth communication and data sharing between chips, thereby creating a system that offers immense parallelism and computational power. Google's TPUv4 pods can combine up to 4,096 of these chips, delivering a staggering peak performance of 1.1 ExaFLOPS [2]. TPU v4 also entails optical circuit switch (OCS) to dynamically configure this 4096 chip cluster to provide smaller TPU slices.

Additionally, thanks to Google's integrated approach to data center infrastructure, TPUs can be 2-6 times more energy-efficient than other Domain Specific Architectures (DSA's) run in conventional facilities — cutting carbon emissions by up to 20 times [2].

Using Cloud TPU to scale reinforcement learning

TPU pods offer an innovative solution to the unique challenges of scaling RL systems. Their highly interconnected architecture and specialized cores allows for rapid data transfer and parallel processing, without the additional overhead present in traditional RL architectures. One such approach is the Sebulba Architecture introduced in Podracer, which utilizes an actor-learner decomposition to efficiently generate experience. The Sebulba architecture is designed to support arbitrary environments and co-locates acting and learning on a single TPU machine, maximizing the utilization of TPU resources.

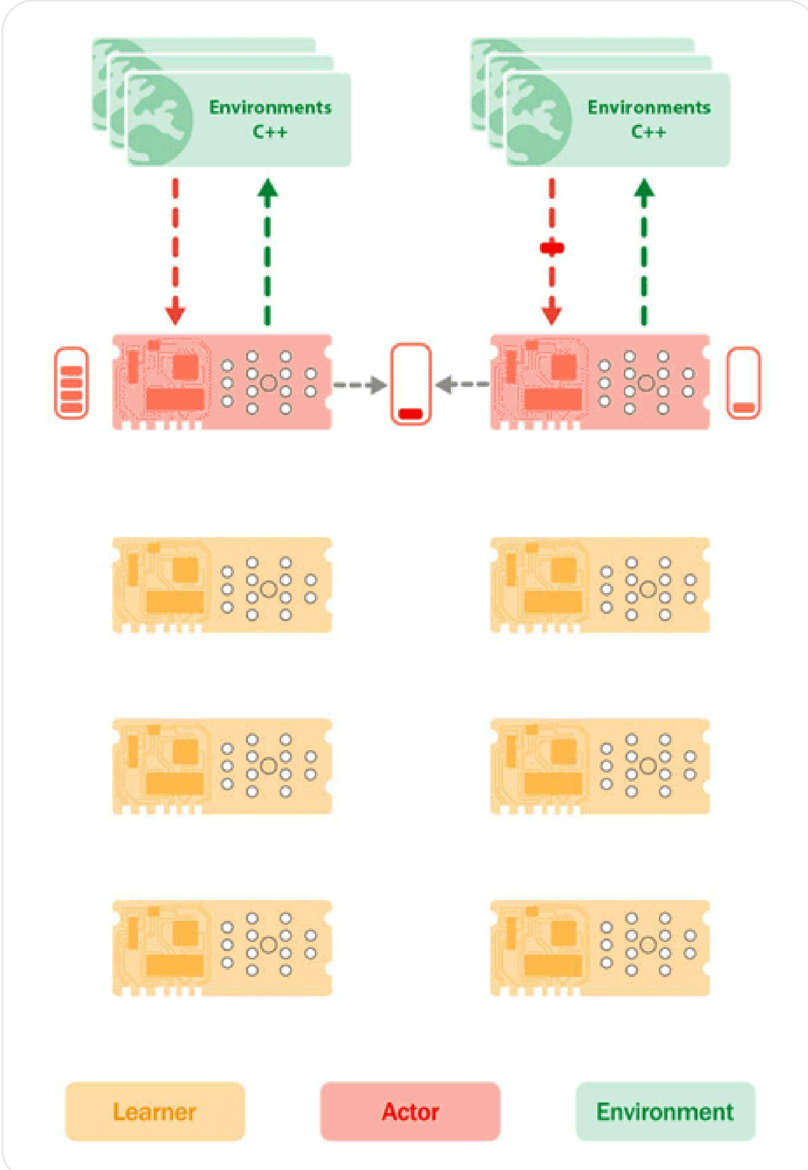


Figure 2: Sebulba architecture depicting the placement of the Learner Cores (Yellow) and Actor Cores (Red) on the TPU, as well as the environments on host CPU's (Green). The environments send their observations and rewards to the Actor cores that then send in response the next actions to take. While doing so, the actors build up a buffer of trajectories. Concurrently, the learner pulls batches of trajectories from the actors buffer and carries out learning updates, pushing new model params to the actor devices.

Sebulba divides the eight available TPU cores of a single machine into two groups: actors and learners. Acting involves interacting with batches of environments in order to generate experiences for the learning process. Sebulba uses multiple Python threads for this purpose, with each thread responsible for a batch of environments. These threads run in parallel, sending batches of observations to a TPU core, which then selects the next actions. The environments are stepped using a shared pool of C++ threads to minimize the impact of Python's Global Interpreter Lock (GIL). To ensure the TPU cores remain active while environments are being stepped, multiple Python threads are used for each actor core.

The learning part of Sebulba involves processing the experience generated by the actors to improve the agent's decision-making. Each actor thread accumulates a batch of fixed-length trajectories on the TPU core, divides it into smaller shards, and sends them to the learner cores through a fast device-to-device communication channel. A single learner thread on the host then processes the data, which is already distributed across the learner cores. Using the JAX "pmap" operation, each learner core applies the update function to its shard of experience. The parameter updates are averaged across all learner cores using JAX's "pmean/psum" primitives, keeping the learner cores in sync. After each update, the new parameters are sent to the actor cores, allowing the actor threads to use the latest parameters for the next inference step.

The Sebulba architecture is ideal for large-scale RL as it addresses many of the engineering bottlenecks we face when scaling.

1. **Reduced communication overhead:** By co-locating, acting, and learning on the same TPU, coupled with fast device-to-device communication, Sebulba minimizes common bottlenecks associated with data transfer such as parameter updates and sending collected data to the learner.
2. **High parallelization:** The Sebulba architecture leverages the parallel processing capabilities of TPUs, complemented by the pool of C++ environments. This allows Sebulba to efficiently handle multiple environments at once. The concurrent processing of both trajectory data from the environments and learning steps significantly accelerates the overall reinforcement learning process.
3. **Scalability:** The Sebulba architecture is designed to scale. As RL tasks become more demanding, Sebulba can easily adapt to larger TPU configurations, paving the way for real-world applications.

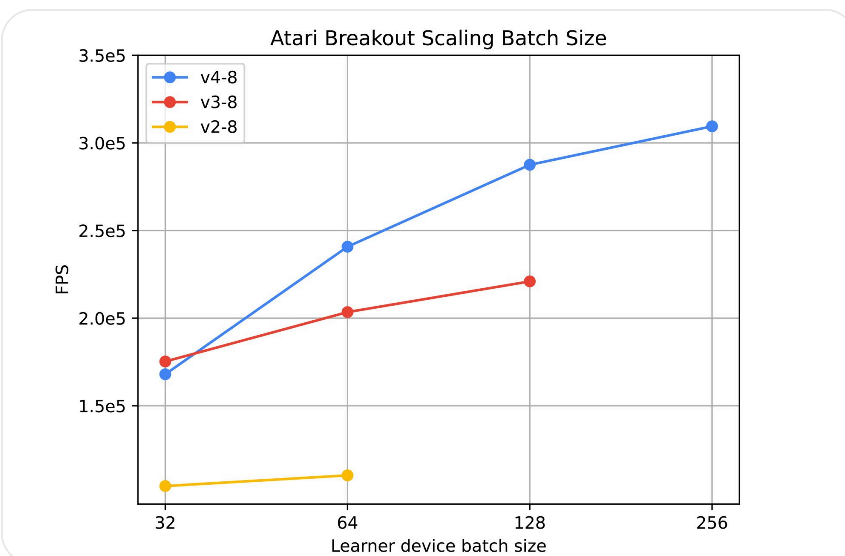


Figure 3: The effects of scaling batch size ranging from 32-256 across TPU v2 (yellow), v3 (red) and v4 (blue).

(Source: InstaDeep benchmarks: 7th of August, 2023)

We tested the architecture on the classic [Atari Benchmark](), using the industry-favored [PPO]() algorithm. Figure 3 shows the Frames Per Second (FPS) Sebulba reaches across different Cloud TPU generations, peaking at over 300k FPS on a single TPUv4. Increasing the batch size consistently improves agent's throughput but produces diminishing returns as we scale. The available on-chip memory can become a limiting factor. Earlier TPU generations are unable to support large batch sizes, i.e. 256 per TPU device.

The Sebulba architecture simplicity enables smooth scalability by replicating the single node configuration multiple times across an entire TPU pod. As before, each replica steps its own pool of environments, using its actor cores for inference and its learner cores to process the trajectories generated on its slice of the pod. The only difference being that the gradients computed to update the model parameters are synchronized, using JAX's collective operations, across all the learner cores of the entire TPU pod rather than just within the individual TPU.

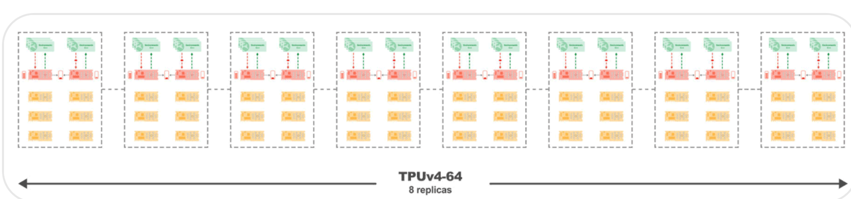


Figure 4: Sebulba on a TPUv4-68 where the cores of each of the 8 replicas are connected with high speed interconnects that allows for linear scaling.

The experimental results confirm the scaling properties of the Sebulba architecture and the communication efficiency of the TPUs, achieving a near-perfect scaling factor of 0.998. As we increase the number of TPU cores, it reaches an impressive 3.43 million FPS on the Atari benchmark when using a TPUv4-128, as illustrated in Figure 4.

Sebulba's ability to scale not only speeds up convergence but also increases the capacity to generate and process data, enabling a larger effective batch size for the system. Figure 5 shows the effect of this benefit in action: as we scale the number of replicas, the time to convergence (reaching score of 400 for Breakout) drops to a few minutes and the final training score improves.

These results echo the level of scaling observed in flagship large scale RL projects such as AlphaStar and OpenAI 5. To surpass human level performance, an increasingly greater amount of data is needed. By simplifying your ability to scale agents, Sebulba can accelerate your research and development, by reducing experimentation time and boosting overall performance.

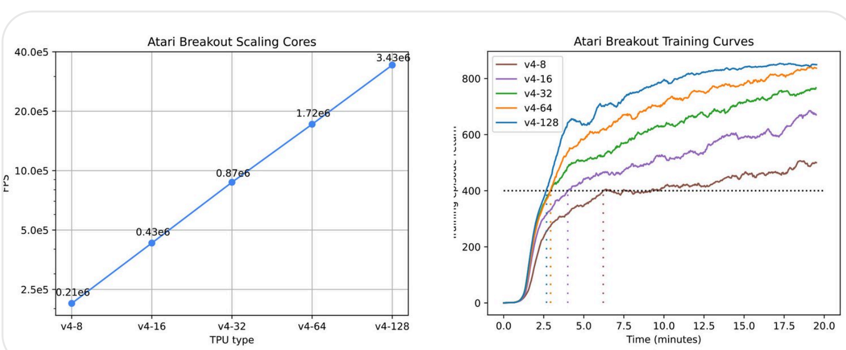


Figure 5 (Left): Linear scaling (Factor of 0.995) of the FPS when training PPO on Atari Breakout over multiple

TPU hosts. Figure 5 (Right): Learning curves of PPO on Atari Breakout, replicated over multiple TPU hosts and the effect on convergence time. (Source InstaDeep benchmarks: 30th of July, 2023)

Solving PCB routing with large-scale reinforcement learning

To further evaluate Sebulba and its scaling efficiency on Cloud TPUs, we've applied it to the Place & Route problem for Printed Circuit Boards (PCBs). This problem can be framed as a Reinforcement Learning task whose objective is to optimally wire the components of a PCB while meeting manufacturing standards and passing Design Rules Checks (DRC). InstaDeep has developed DeepPCB, an AI-based product enabling electrical engineers to have their PCBs optimally routed without human-intervention. To accomplish this, its product team developed a high-speed simulation engine to efficiently train reinforcement learning agents. However, due to the complexity and size of the problem, commonly used RL libraries fall short in terms of training throughput. When connecting the DeepPCB simulator to Sebulba, we observed a 15x-235x acceleration compared to running with its previous legacy distributed architecture.

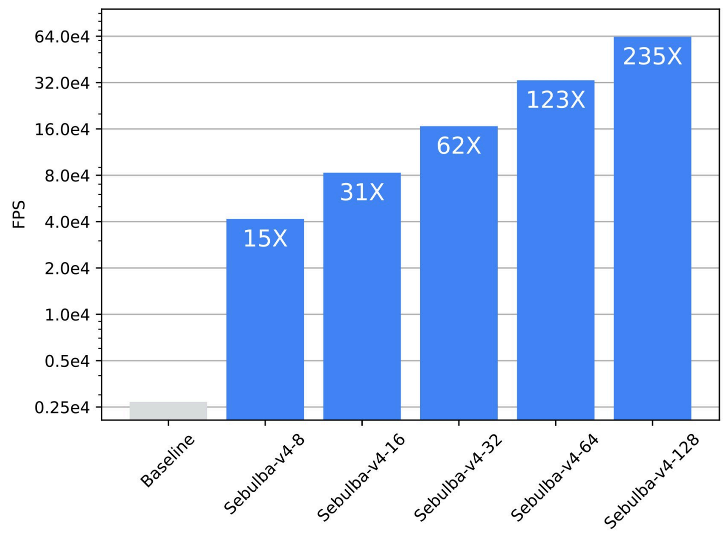


Figure 6: Benchmark comparison of the Sebulba architecture against the legacy system previously used for DeepPCB. (Source InstaDeep benchmarks: 1st of August, 2023)

Figure 6 highlights the dramatic speedup Sebulba offers over the baseline legacy system used for DeepPCB. The baseline system takes ~24 hours for a complete training and costs approximately $260, when using a high-end GPU on Google Cloud Platform. Switching to the Sebulba architecture on Google Cloud TPUs slashes both cost and time. The best configuration cuts training time to just six minutes at a mere $20, resulting in an impressive 13x cost drop.

In addition, thanks to Sebulba's linear scaling on Cloud TPUs and the fixed price-per-chip, the training cost remains constant as we scale up to larger TPU pods, all while significantly reducing the time to convergence. Indeed, although doubling the system size doubles the price per hour, this is offset by cutting the time to convergence in half.

With DeepPCB as a case study, we've seen how Cloud TPUs can offer cost-effective solutions to real-world decision-making problems. By harnessing the full potential of TPU, we're boosting the team's ability to speed up experiments and enhance system performances. This can be critical for research and engineering teams, enabling them to deliver new products, services, and research breakthroughs that were previously out of reach.

Alongside this post, we are pleased to open-source the codebase that was used to generate these results. This helps provide a great starting point for researchers and industry practitioners eager to integrate Reinforcement Learning into practical applications.

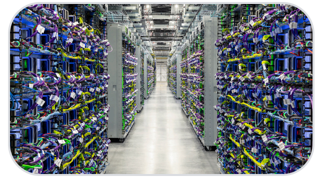
References

1. Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d.O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, Susan Zhang (2019). Dota 2 with large scale deep reinforcement learning. arXiv preprint arXiv:1912.06680.

2. Norman P. Jouppi, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Suvinay Subramanian, Andy Swing, Brian Towles, Cliff Young, Xiang Zhou, Zongwei Zhou, and David Patterson. (2023). "TPU v4: An Optically Reconfigurable Supercomputer for Machine Learning with Hardware Support for Embeddings". arXiv preprint arXiv:2304.01433.

Compute

Expanding our AI-optimized infrastructure portfolio: Introducing Cloud TPU v5e and announcing A3 GA



The new Cloud TPU v5e is the most cost-efficient, versatile, and scalable Cloud TPU to date, and the A3 Supercomputer is now generally available to power your large-scale AI models.

By Amin Vahdat • 7-minute read

Posted in [Compute](#)—[AI & Machine Learning](#)—[Customers](#)—[Developers & Practitioners](#)

Related articles

Compute

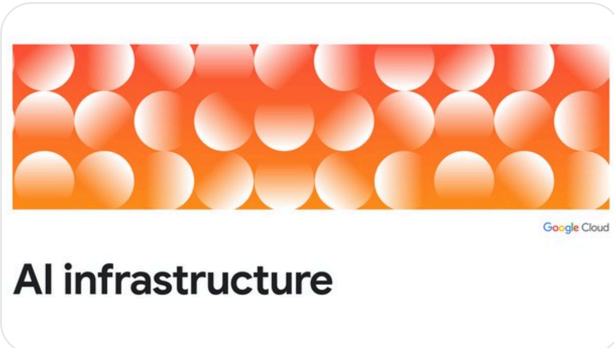
Compute

Three lessons in accelerating foundation model upgrades

By Radhika Mani • 4-minute read

C4N, now GA: Delivering cloud's highest per vCPU network and block storage I/O for x86 workloads

By Parinda Gandhi • 10-minute read



AI infrastructure

Google Cloud named Leader in the 2026 Gartner® Magic Quadrant™ for AI Infrastructure

By Mark Lohmeyer • 5-minute read

Compute

Report: 83% of organizations need to upgrade their infrastructure to support agentic AI

By Drew Bradstock • 6-minute read