Google Cloud Summit
Jakarta

Optimizing AI Inference with GKE: Reducing Costs and Latency

Ananda Dwi Rahmawati
Google Developer Expert Cloud

Agenda

- Introduction to AI Inference on GKE
- Strategies for Reducing Inference Latency on GKE
- Cost Optimization for AI Inference Workloads
- Best Practices and What's Next to Do

Introduction - The Challenge of AI Inference

AI models are increasingly complex, demanding significant computational resources.

Inference Latency

Operational Costs

Scalability

Complexity

Resource Underutilization

Model Lifecycle Management

Dependency Management

Google Cloud AI Hypercomputer

Flexible Consumption

- Dynamic Workload Scheduler
- On Demand
- CUD
- Spot

Open Software

- JAX, TensorFlow, PyTorch
- Multislice Training, Multihost Inference, XLA
- Google Kubernetes Engine & Compute Engine

Performance-Optimized Hardware

- Compute (GPUs, TPUs, CPUs)
- Storage (Block, File, Object)
- Networking (OCS, Jupiter)

Optimizing system-level co-design streamlines the entire AI lifecycle—from training to tuning and serving

AI Hypercomputer Architecture

Google Vertex AI | **NVIDIA DGX Cloud** | **Ecosystem Platform & Solutions**

→ State-of-the-Art Models & Integrated Developer Platform

AI Hypercomputer

Flexible Consumption

- Dynamic Workload Scheduler
- On Demand
- CUD
- Spot

→ Intelligent Resource Optimization

Open Software

- JAX, TensorFlow, PyTorch
- Multislice Training, Multihost Inference, XLA
- Google Kubernetes Engine & Compute Engine

→ High Flexibility, Usability, and Optionality

Performance-Optimized Hardware

- Compute (GPUs, TPUs, CPUs)
- Storage (Block, File, Object)
- Networking (OCS, Jupiter)

→ Exceptional Performance and Efficiency

Why Google Kubernetes Engine for AI

1 Portability & Customizability

Choice of frameworks and ecosystem tools that are portable

2 Performance & Scalability

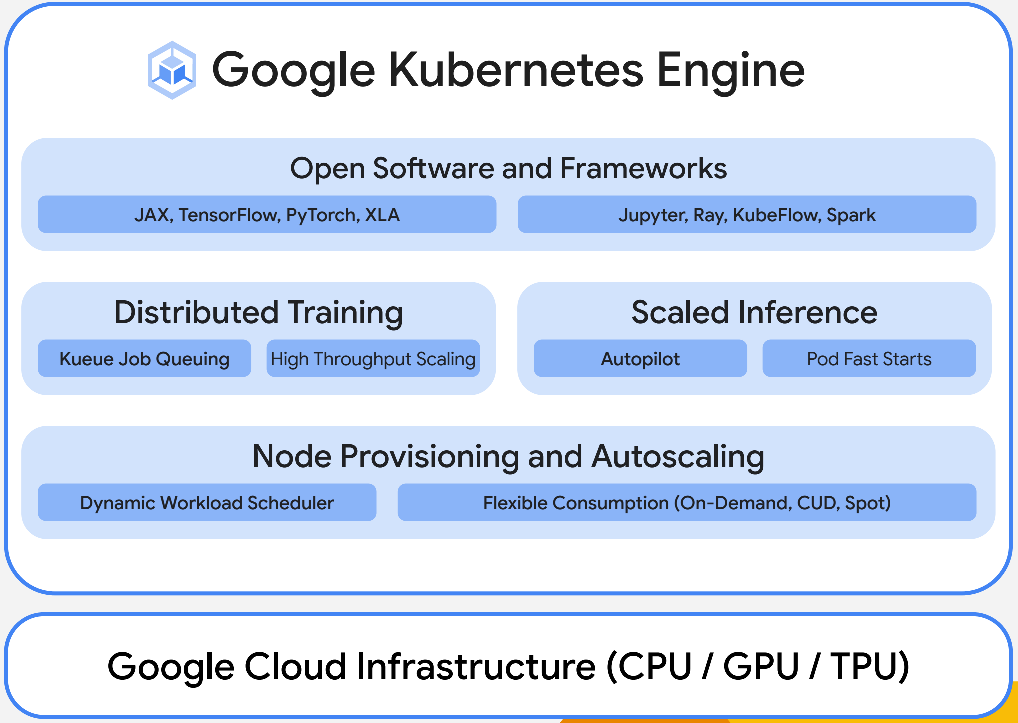
Scale the platform for supercomputer scale training and inference

3 Cost-Efficiency

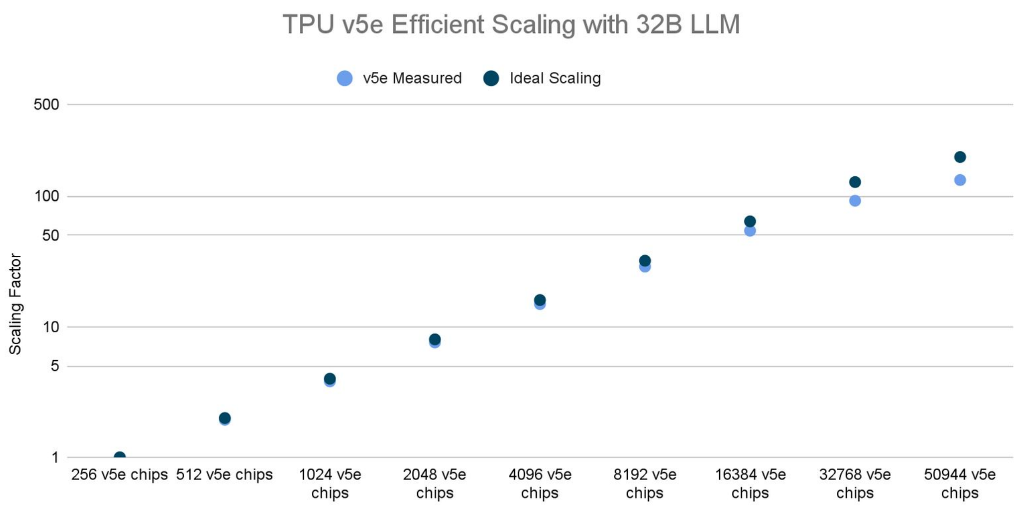
Increase utilization of valuable resources while reducing operational overhead

Google Kubernetes Engine
cloud native infrastructure for AI training and inference

- **Limitless Scale:** Deploy AI at industry-leading scale, supporting thousands of TPUs and nodes.
- **Cost-Efficient Performance:** Maximize price-performance with smart GPU/TPU use, job queuing, and fast provisioning.
- **Effortless Ops:** Focus on models, not infra, with GKE Autopilot's managed, optimized Kubernetes.
- **Enterprise Reliability:** Trust AI workloads to the leading Kubernetes contributor's cloud-native infra.



World’s largest distributed training job on GKE with TPU Multislice Training



Scaled to

50,000+

TPU v5e chips

*Google Internal data for TPU v5e As of November, 2023: All numbers normalized per chip. seq-len=2048 for 32 billion parameter decoder only language model implemented using MaxText. *2*

Strategies for Reducing Inference Latency

Understanding the Bottlenecks

- **Model Size & Complexity:** Larger models inherently take longer to process.
- **Compute Resources:** Insufficient CPUs/GPUs or outdated hardware.
- **Network Latency:** Data transfer between client, GKE, and data sources.
- **Software Stack Overhead:** Inefficient inference servers, redundant data processing.
- **Batching Strategies:** Incorrect batch sizes can lead to underutilization or increased latency.

Strategies for Reducing Inference Latency

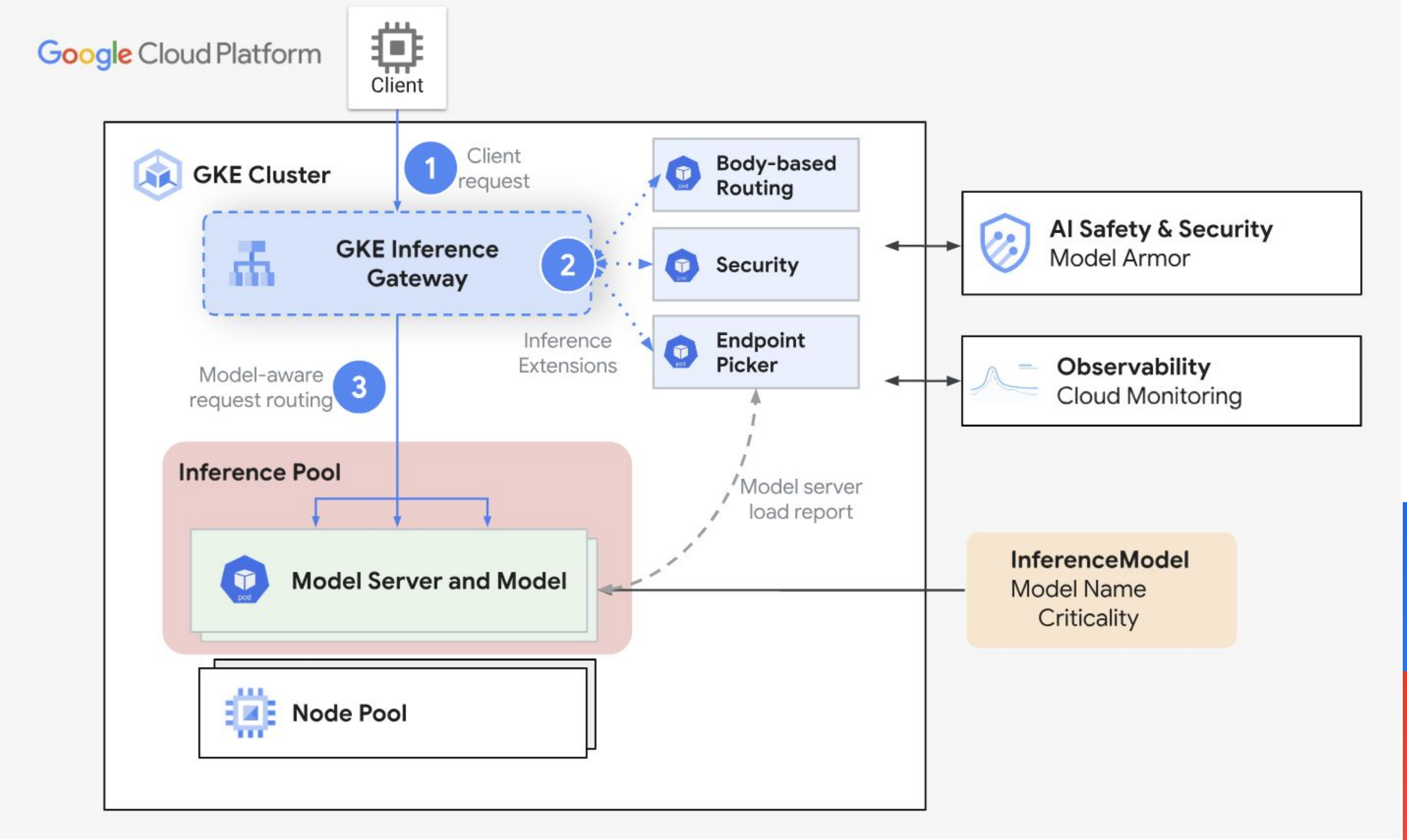
Software & Deployment Optimizations

- **Model Quantization & Pruning:** Reduce model size and complexity without significant accuracy loss.
- **Optimized Inference Frameworks:** Use frameworks like TensorFlow Serving, TorchServe, Triton Inference Server.
- **GKE Inference Gateway**
- **KubeFlow Serving (KServe)**

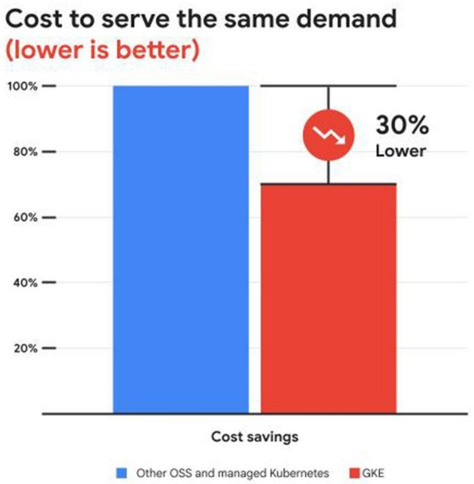
Strategies for Reducing Inference Latency

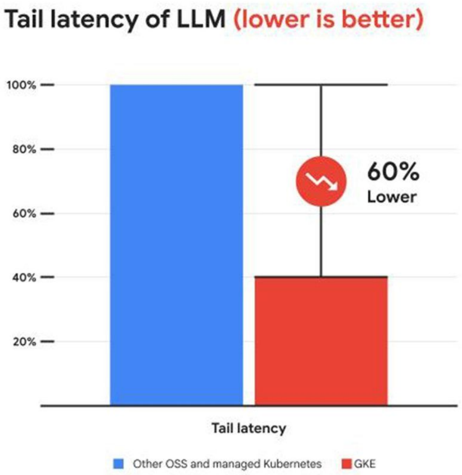
Smart Scaling and Batching

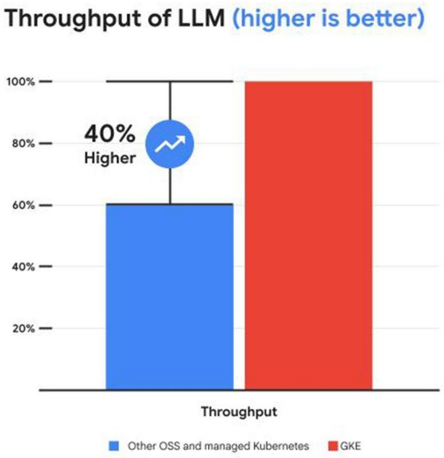
- **Horizontal Pod Autoscaler (HPA):** Scale pods based on CPU/memory utilization or custom metrics (e.g., QPS, latency).
- **Vertical Pod Autoscaler (VPA):** Automatically adjust CPU and memory requests/limits for pods.
- **Cluster Autoscaler:** Automatically adjust the number of nodes in your GKE cluster based on pod resource requests.
- **Dynamic Batching (with Triton):** Grouping multiple inference requests into a single batch for efficient GPU utilization.


Google Cloud Platform
Client
GKE Cluster
1
Client request
Body-based Routing
GKE Inference Gateway
2
Security
AI Safety & Security
Model Armor
Endpoint Picker
Inference Extensions
Model-aware request routing
3
Observability
Cloud Monitoring
Inference Pool
Model server load report
Model Server and Model
InferenceModel
Model Name
Criticality
Node Pool

GKE Inference Performance







Solution mapping to customer journey

	01 Evaluation	02 Onboarding	03 Production ramp (day 2 operations)
Typical needs	Learn about all the accelerator options and choose one Procure capacity for inference POC (reservations, on-demand, spot, or DWS)	Establish performance baseline for common open model (Llama, Mixtral, ...) and model server (vLLM) via basic benchmarking Expect parity with GPUs on-prem or in other clouds	Load balance traffic price-performance at scale with real traffic Balance availability and cost efficiency Monitor performance metrics
GKE Gen AI features	**GKE Inference Quickstart** simplifies accelerator qualification from months to days	**GKE Inference Quickstart** suggests scaling thresholds vLLM/TPU stack for price-perf	**GKE Inference Gateway** for LLM aware routing Built-in vLLM observability metrics

Thank you

@misskecupbung