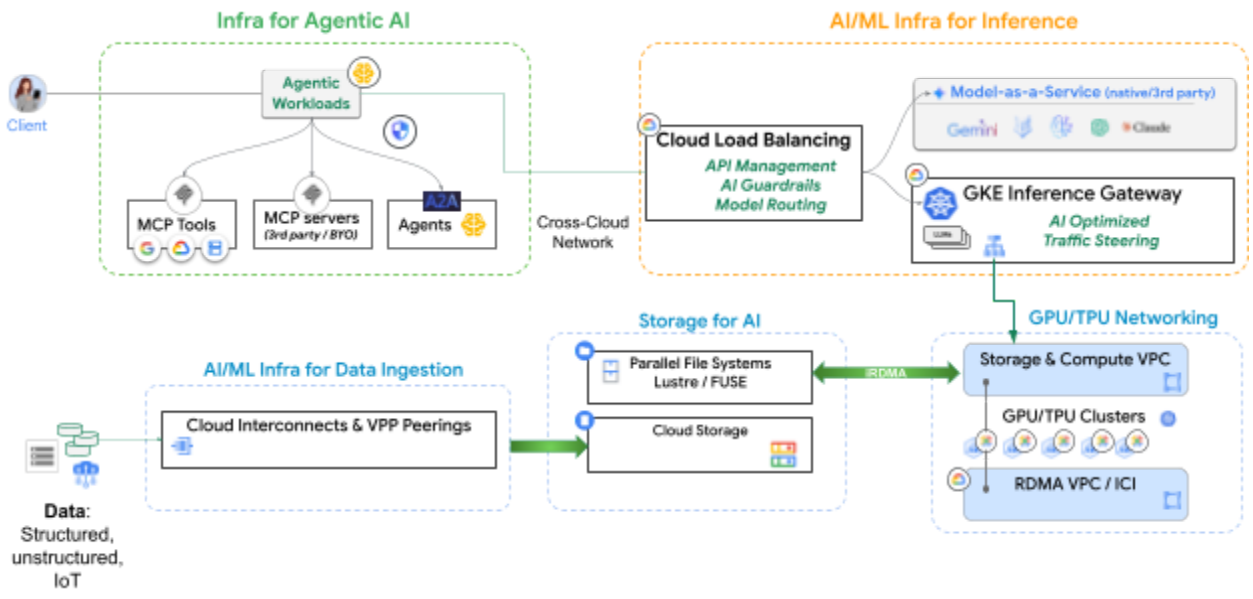
Google Cloud

Inference Model Serving with Cross-Cloud Network

Solution Overview



For more information visit **cloud.google.com**

Solution overview

Optimizing AI inference with Cross-Cloud Network

Organizations deploying AI models for inference seek the normalization of the infrastructure patterns by which models are deployed and scaled while maximizing model performance and minimizing costs. The following objectives are commonly pursued when serving models for inference:

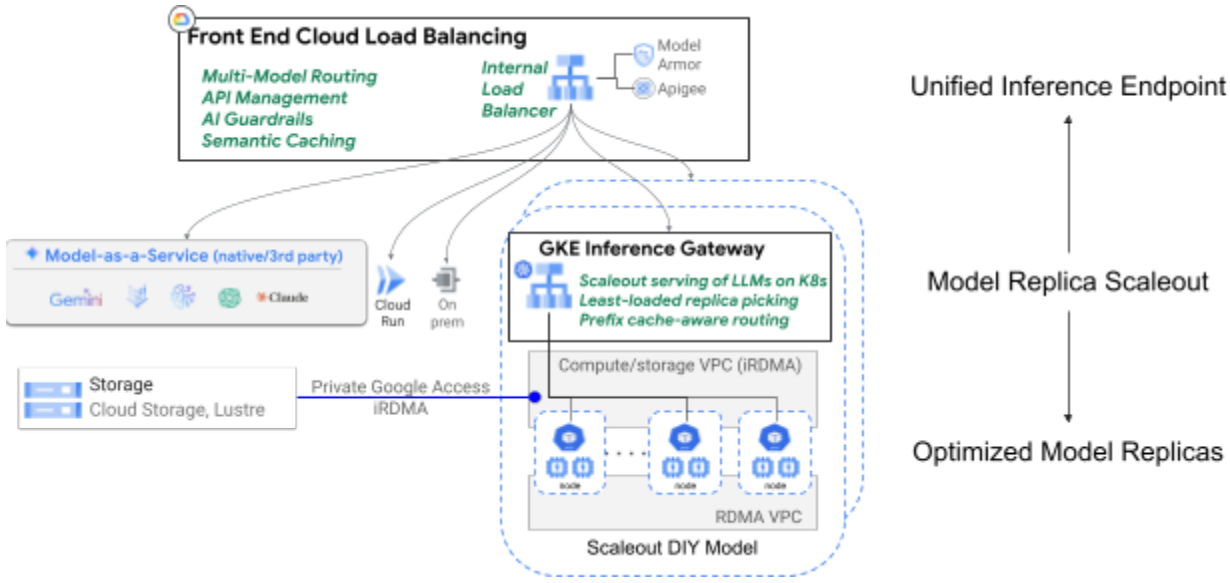
01	02	03	04	05
Minimize response latency	**Optimize GPU/TPU utilization**	**Unified API for all models**	**Consolidated API management**	**Consolidated AI guardrails & security**
Optimize user experience and application performance	Reduce GPU idle time and its associated costs; re-use computations (prefix-cache) and deployed GPUs (dynamic LoRA); and elastically adapt to resource demand	Accelerate deployment velocity, and enhance portability and modularity	Common API authentication, authorization, and management services and policies for all models	Safety and security governance and policy enforcement

Google Cloud offers a comprehensive network solution for AI inference model serving. A unified inference model endpoint enables access to all models with consolidated API management and AI guardrail services. The solution enables the use of the OpenAI API as the normalized API for all models and integrates the full variety of model runtime form factors: GKE, Cloud Run, on-prem/multicloud, Vertex AI, and SaaS. Form factors deployed in Google Cloud, such as GKE and Vertex AI, benefit from the AI optimized scaleout mechanisms delivered by the GKE Inference Gateway, where GPU specific metrics are leveraged to enable least loaded replica selection and prefix-cache routing. All runtime form factors benefit from the high performance connectivity delivered by Google's global network. GKE delivers differentiated performance for gen AI applications with the addition of new inference capabilities. These capabilities reduce serving costs by up to 30%, tail latency by up to 60% and

increases throughput by up to 40% compared to other managed and open source Kubernetes offerings based on our internal benchmarks.

The AI inference scaleout architecture is developed and specified in the Kubernetes Inference Gateway project and llm-d open source software communities with significant guidance and contributions from Google Cloud. This architecture is implemented and offered as a managed service through the GKE Inference Gateway. By offering these optimizations as a turnkey managed service accessible via the GKE Gateway API inference extensions, Google Cloud enables enterprises and model-as-a-service providers to focus their resources in their core competencies rather than the assembly of the multiple open source components required to achieve these optimizations.

The solution is illustrated in the following figure. The network infrastructure plays a critical role in optimizing the performance of each replica of a model, scaling out the service with multiple replicas, routing to different models with a unified API and streamlining the deployment of AI guardrails and API management functions common to all models. The solution proposed effectively creates a unified endpoint to access all models for inference. Let's explore the different aspects of this multi-model endpoint starting with the AI specific optimizations delivered by the GKE Inference Gateway to efficiently scale out multiple model replicas for maximum performance and optimal GPU/TPU utilization, enabling the creation of the best performing model replicas and providing a consolidated surface for routing, enforcement of guardrails, security, API management, and observability.



Model Replica Scaleout

An instance of a model server for inference is a service capable of processing a prompt to generate a response. An instance of a model server is commonly referred to as a model server replica and comprises an inference server, the model computation layers with its trained weights and one or more accelerators, such as GPUs or TPUs. A replica of a model server can be optimized to maximize its capacity. To address concurrent requests multiple replicas must be deployed and the distribution of requests across the multiple replicas must be properly orchestrated to offer a scaleout service that minimizes response latency while optimizing the utilization of costly GPU/TPU accelerator resources.

Once model server replicas are optimized to operate to their full potential, they need to be scaled out to service multiple concurrent requests of varying size and complexity. This is achieved by horizontally scaling multiple replicas of a model to service requests in parallel and avoid conditions such as head of line blocking that would result in slow response times or, worse yet, application deadlocks. To extract the best performance out of these horizontally scaled replica sets, the GKE Inference Gateway in the Cross-Cloud Network offers key functionality that is designed specifically to balance AI inference loads. The GKE Inference Gateway combines implementations of the Kubernetes open source inference gateway architecture with the Cloud Load Balancing infrastructure to offer a managed service that meaningfully enhances the GKE load balancing layer to route traffic based on AI inference specific application processing behavior.

AI inference workloads present a resource utilization profile that may be easily misinterpreted when using traditional server utilization metrics for load balancing and traffic routing. GPU specific metrics such as kv_cache_utilization and model server queue depth provide a better data point on the utilization level of GPUs in an inference replica and its ability to accept and service new jobs. Through the delivery of the GKE Inference Gateway, Google Cloud has enhanced the load balancing infrastructure in the Cross-Cloud Network to leverage these metrics in custom load balancing algorithms to optimize the overall response of inference replica sets by being able to pick the least loaded replica in a given set. Benchmark tests show up to a 60% tail latency improvement when using the GKE Inference Gateway's least-loaded-replica picking algorithms.

An inference scheduler, integrated with the load balancing infrastructure as part of the GKE Inference Gateway implementation, allows for an extensible set of functions that also include the ability to evaluate a prompt and make routing decisions based on the length of the prompt and whether a similar prompt has been processed previously. With prefix cache routing, the inference scheduler can match a prompt with a model replica that may have processed a similar request earlier and send it to that replica. This functionality referred to as prefix cache-based routing allows requests to be serviced without having to run them through a full prefill stage, but, in combination with paged-attention memory management in the inference server, allows the prefill stage to be shortened. The result of combining least loaded replica picking and prefix cache-based routing are faster and more numerous successful responses with optimal resource utilization. Benchmark tests show an improvement of up to 40% in throughput when using the GKE Inference Gateway prefix-cache aware routing.

One important function delivered by the GKE Inference Gateway implementation of the inference scheduler is the ability to extract information from the payload of the request to influence the routing of the request. This is known as body-based routing and is used primarily to extract the model identifier from the body of requests that use the OpenAI API. Routing on model identifiers is fundamental to serving dynamic LoRA adapters. When using dynamic LoRA adapters, a replica of the model will have multiple LoRA adapters active in memory (and some on standby on local disk or external storage). Each LoRA adapter has its own model identifier. Any requests must be routed to a replica that hosts the LoRA adapter specified in the model identifier. More importantly, the inference scheduler has knowledge of which specific LoRA adapters are active in memory on the different model replicas to make a more efficient routing decision that considers load metrics, prefix caches as well as the active or standby status of the target LoRA adapter on the different replicas.

The inference scheduler can evaluate the length of a prompt that isn't cached and determine if it should be routed to a dedicated set of prefill nodes to minimize resource contention by keeping the computationally intensive prefill tasks separate from the memory I/O demanding decode tasks of the inference process. This optimization is known as prefill/decode (P/D) disaggregation and its

effectiveness depends on the ability of the network (by means of an inference scheduler) to identify prompts that are candidates for P/D disaggregation.

The inference scheduler's implementation, including its algorithms, heuristics, and interactions with load balancing, adheres to the architecture of the open-source Kubernetes Inference Gateway project. This functionality is available as a managed service through Google Cloud's GKE Inference Gateway.

Certain optimizations such as P/D disaggregation imply the distribution of inference tasks (prefill and decode) across separate nodes. The transfer of the KV cache contents from the prefill stage to dedicated decode nodes demands very high performance lossless connectivity across nodes to be truly effective. To complement the sophisticated AI-specific routing necessary to scale out multiple model replicas into a service, high speed networking between accelerator nodes and to storage is a fundamental aspect of a complete infrastructure stack to support high performance AI inference. Let's explore the optimization of connectivity for the model replicas.

Optimized Model Replicas

The compute, storage, and network infrastructure must support the optimization of each individual model replica so that the model can perform to its maximum potential. Each model replica requires optimal connectivity to other services, to storage, and between accelerators.

Google Cloud offers powerful compute form factors with optimized network connectivity to maximize throughput and minimize the latency for lossless GPU communications across nodes, storage access and connectivity to external networks. The Cross-Cloud Network supports remote direct memory access (RDMA) for lossless, high bandwidth and low-latency GPU communications. In conjunction with kernel bypass techniques, such as GPUDirect, the RDMA-enabled network allows GPU communications to happen without involving the CPU or the operating system kernel, effectively reducing communications latency and accelerating the inference process. With the CPU no longer burdened by network processing tasks, it can dedicate its cycles to other essential operations, such as preparing new inference requests. This enhanced efficiency allows the entire system to handle a greater volume of requests simultaneously, thereby increasing the overall throughput in terms of tokens per second.

RDMA connectivity between GPUs and storage, in standard VPCs, accelerates operations dependent on efficient storage access such as: checkpointing, model weight uploads, retrieval augmentation, and storage-enabled key value caching.

The RDMA VPC enables access to a massive rail-aligned physical network with RDMA over converged ethernet (RoCEv2) designed for lossless communications with ultra-low latency and high bandwidth. The RDMA VPC accelerates GPU to GPU communications necessary for the synchronization of inference computation, the transfer of activations and Key Value cache synchronization between GPUs in models that are parallelised across multiple nodes. The benefits of RDMA are most pronounced in tensor parallelism. Tensor parallelism is a model parallelization strategy that splits individual layers of the model across multiple GPUs. This approach is communication-heavy, as it requires frequent and high-bandwidth data exchanges between GPUs to synchronize the results of the split computations. In such scenarios, the lossless and low latency communications enabled by RDMA can lead to substantial improvements in throughput and reduced inference times. This optimized GPU to GPU communication also plays a critical role in enabling optimizations such as prefill-decode disaggregation, which require the fast transfer of large Key Value cache tensors between prefill nodes and decode nodes.

TPU to TPU communications use the inter-chip interconnect (ICI) and optical circuit switching to offer adaptable topologies that enable meaningful training and inference speedup for different types of model parallelism.

To reap the benefits of this network infrastructure, workloads must be properly orchestrated. GKE offers a series of functions that are key to the successful deployment of model replicas and replica sets. GKE Horizontal Pod Autoscaling (HPA) enables the elastic provisioning of model replicas within a replica set. GKE HPA monitors GPU utilization metrics to trigger autoscaling based on the effective utilization of the accelerator resources. GKE leader worker sets are instrumental in the deployment of very large models that span multiple nodes. GKE multi-networking enables pods to access the aggregate bandwidth of multiple GPU interfaces connecting an inference compute node to an RDMA VPC. With 8 x 400 Gbps GPU NICs per server, this capability translates into 3.2 Tbps of connectivity to a rail aligned network. For serverless deployments, Cloud Run offers fully managed autoscaling for single node models.

Unified Inference Endpoint

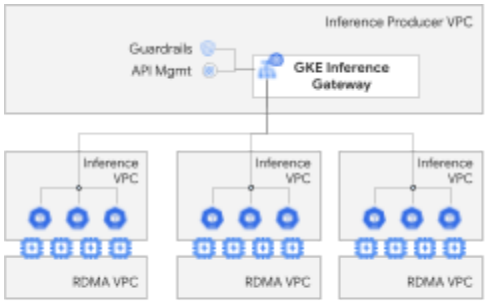
Organizations are keen to simplify their API structure and presenting a single endpoint for the consumption of all the different models is compelling to many organizations. A variety of replica sets hosting different models can be organized as backend services to a unified load balancer that can enforce routing to different models using a unified API for all models. This is the basis for offering a variety of models using a single service endpoint. The load balancer at the front end of such a service endpoint must be able to route traffic to different models based on model identifier metadata present in the request payload. We refer to the ability to extract model information from the request body and route on it as body based routing. Body based routing enables the normalization of the API calls to one scheme across all models by using a combination of url/path translations and service extension plug-ins for the extraction of the model identifiers. One viable API for the normalization is the OpenAI API, which includes the model identifier information in the request body/payload. Alternatively, Apigee can be inserted as a callout from the load balancer and both the body based routing and URL/path translation can be executed by Apigee. This has the advantage of also including all the API management functionality that Apigee provides.

With the access to the unified model endpoint under a common load balancer for the different model types, the enforcement of AI guardrails can be done in the network as a Service Extensions callout from the load balancer to the AI guardrails service. The network insertion of the guardrails allows organizations to ensure that access to the diverse models is governed by a consolidated guardrail policy that is common to all applications consuming the models. This solution supports a growing ecosystem of guardrails inclusive of Google Cloud's Model Armor and NVIDIA NEMO Guardrails.

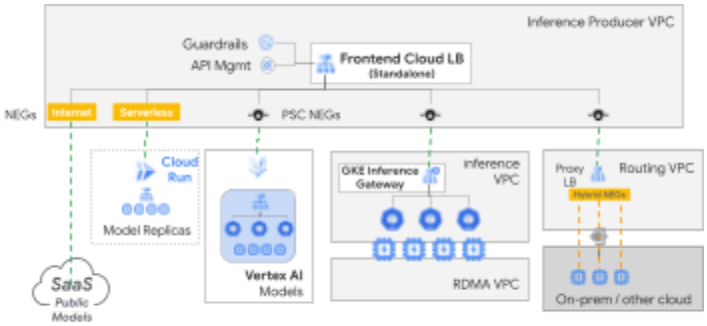
API management and authentication can also be delivered as a Service Extensions callout from the load balancer. The Apigee extension processor allows callouts to Apigee to fulfill required API authentication, authorization, and management. One important function of this API management layer is semantic caching. With semantic caching, previously generated responses are cached in the context of the prompts they address. The semantic cache function identifies whether a prompt is semantically similar to one of the cached prompts and if there is a match it will respond with the stored response. This functionality can offload a large number of requests from the model inference servers, saving expensive GPU cycles, and dramatically reducing response latency.

Network Deployment Patterns

Google Cloud collaborates with a broad ecosystem of partners like NVIDIA, Anthropic, OpenAI, and open source communities such as vLLM and Kubernetes. Many efforts are driven through these partnerships to converge on architectures, APIs, protocols, and the implementation of relevant metrics and memory handling mechanisms in leading inference server implementations. The architecture for a scaleout multi-model endpoint such as the one described in this document is the product of consensus in the Kubernetes open source community, specifically the gateway API inference extension project and the llm-d.ai community that focuses on the aspects of disaggregated serving. All components of this architecture are implemented and offered as a managed service in the GKE Inference Gateway. This includes body based routing for model identifier routing, replica picking, and disaggregation heuristics optimized for AI inference. Cloud Load Balancing enhances this architecture with Service Extensions that enable the insertion of API management functions, semantic caching, security enforcement, and AI guardrails. These functions are ideally deployed at the load balancer where the model routing is being enforced. For deployments in which all models are deployed on GKE runtimes, the load balancing can be implemented in a single tier that is fully managed through the GKE Gateway API (inclusive of the inference extensions). When other runtimes, such as Vertex AI, serverless, or replicas outside of Google Cloud, need to be integrated as part of the model inference endpoint, a front-end standalone load balancer is used to implement the body based routing function along with the Service Extensions. Both of these patterns are illustrated in the following figure. Replica picking optimizations are always deployed at the load balancing layer immediately above the model replicas, which results in a two tier load balancing pattern.

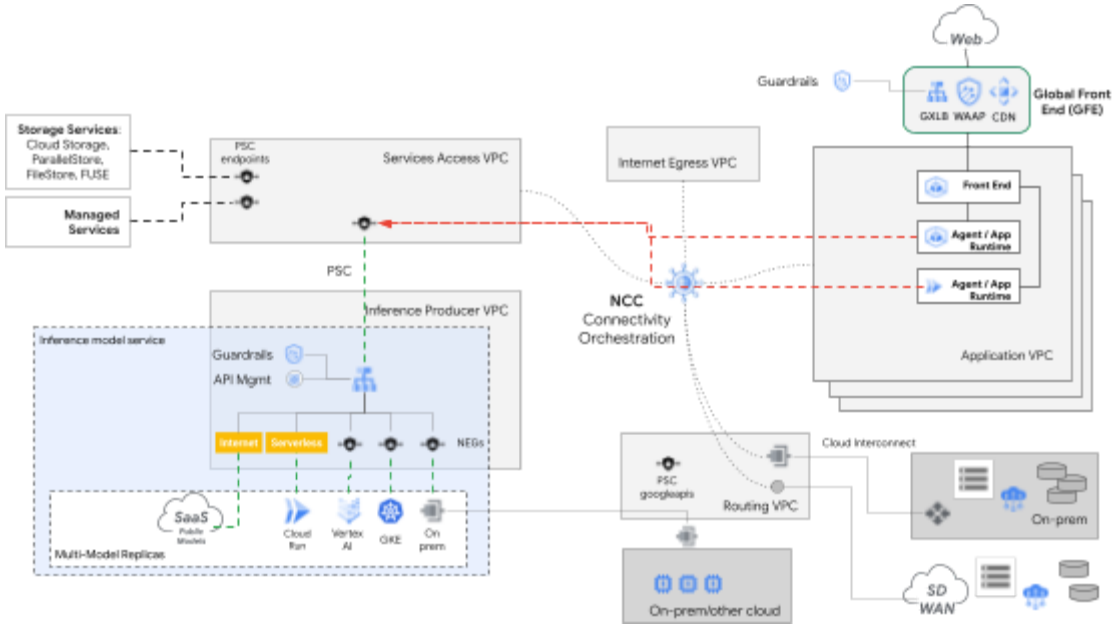


Consolidated single tier pattern for GKE runtimes



Two tier pattern for heterogeneous runtimes

The resulting inference model endpoint integrates seamlessly into the service centric structure utilized to build distributed applications in Google Cloud's Cross-Cloud Network. The following diagram shows the multi-model inference endpoint integrated in the Cross-Cloud Network as a published service.



This architecture makes the model endpoint available for applications to call upon it efficiently, privately and securely via the Cross-Cloud Network. Services calling upon the model endpoint include simple AI application runtimes and agents in an agentic application that require access to different models. A single published endpoint can be accessed from applications structured across a variety of VPCs and projects to fulfill any inference requirements without the need for application specific custom model deployments. The applications in turn benefit from the comprehensive set of services, security and optimized connectivity enabled by the Cross-Cloud Network for distributed applications.

One key use case that the Cross-Cloud Network enables is the Global Front End for scalable and secure web access to the AI application. Service Extensions in the Global Front End provide an insertion point for additional AI guardrails that would catch non-compliant queries early on, before even burdening the AI application runtimes.

Summary

This solution proposes a unified model endpoint using the OpenAI API to serve any model with AI specific scale out algorithms for model performance optimization. The model endpoint fits seamlessly in the paradigm of the Cross-Cloud Network distributed application to provide a modular model service for the development of agentic applications. The network plays a critical role in delivering models for inference at scale. Enhancements to replica selection algorithms introduced by the GKE Inference Gateway are key to preventing head of line blocking events that impact model response latency and efficient GPU saturation. The load balancing infrastructure enables model based routing and service extensions for API management and enforcement of AI guardrails to provide a complete solution for the delivery of models for inference with an API normalized on the OpenAI API.