
KubeCon
CloudNativeCon
North America 2024

KubeCon CloudNativeCon
North America 2024

# Optimizing LLM Performance in Kubernetes with OpenTelemetry

Ashok Chandrasekar (Google), Liudmila Molkova (Microsoft)

# About us



**Ashok Chandrasekar**

@achandrasekar

- Senior Software Engineer at Google
- Leading AI Inference workloads' performance optimization in GKE
- Leading model server metrics standardization and benchmarking efforts in K8s Serving WG

**Liudmila Molkova**

@lmolkova

- Principal Software Engineer at Microsoft
- Working on Azure client libraries and developer experiences
- Member of the OTel Technical Committee and OTel SemConv maintainer

# TOC

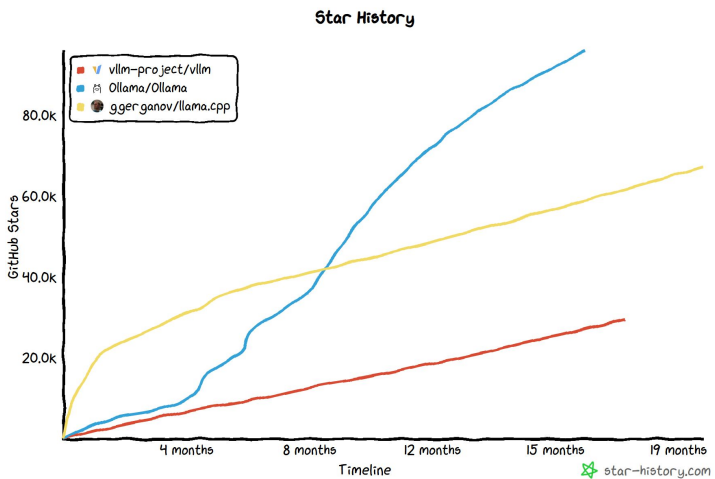


- Intro
- Client-side observability
- Model server performance analysis
  - Demo
  - Auto-scaling
- How to get involved

# LLMs in Kubernetes



- **Self-hosting LLMs** is becoming more prominent
  - HuggingFace has over a million models now
- **Kubernetes** is the preferred platform for serving LLMs
  - New AI/LM workload deployment and managements capabilities (LWS, DRA, etc)



# LLM deployments have new observability needs



- **New and fast-growing** technology
- New usage patterns with **high complexity**
- **Non-deterministic** responses that need to be evaluated
- Compute and data intensive operations that need **deeper insights**

**We just don't know how to use, serve or observe LLMs yet**

# Client side observability

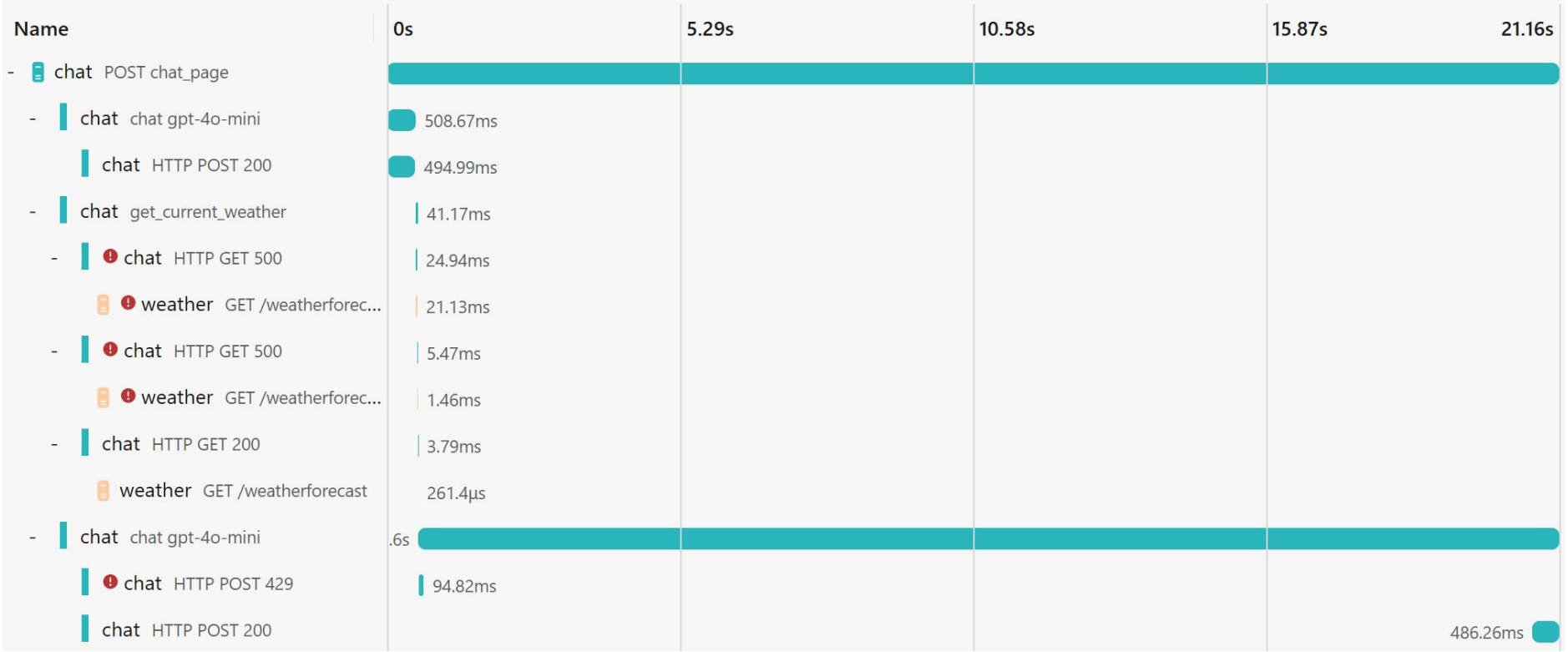


- **Reusable** client instrumentations following common **semantics**
- Covers **common observability needs**
- Provides **details into GenAI** operations:
  - GenAI context: parameters, token usage
  - Prompts and completions
  - Evaluations are coming

# Client side: distributed tracing



## Applications that use GenAI need **general-purpose observability**



# Client side: GenAI-specific context




**chat: chat gpt-4o-mini** 75dab11

Resource **chat** Duration **390.15ms** Start time **504.17ms** View logs Filter...

| Name | Value |
|---|---|
| gen_ai.operation.name | chat |
| gen_ai.request.max_tokens | 100 |
| gen_ai.request.model | gpt-4o-mini |
| gen_ai.response.finish_reasons | ["stop"] |
| gen_ai.response.id | chatcmpl-AOrqRhURDq3CeGjegpI64QIQZ5BGH |
| gen_ai.response.model | gpt-4o-mini-2024-07-18 |
| gen_ai.system | openai |
| gen_ai.usage.input_tokens | 55 |
| gen_ai.usage.output_tokens | 12 |

# Client side: prompts and completions (opt-in)



| Resource | Level | Timestamp | Message | Trace |
|---|---|---|---|---|
| chat | None | 12:44:10.109 PM | {"content":"You are helpful assistant. Keep your answers short."} | e621ed4 |
| chat | None | 12:44:10.111 PM | {"content":"weather in seattle?"} | e621ed4 |
| chat | None | 12:44:10.551 PM | {"index":0,"finish_reason":"tool_calls","message":{"role":"assistant","tool_calls":[{"id":"call_RRXFkrwZH7fG9zqBSvcheyHr","type":"functi... | e621ed4 |
| weather | Error | 12:44:10.556 PM | An unhandled exception has occurred while executing the request. | e621ed4 |
| chat | None | 12:44:10.571 PM | {"content":"You are helpful assistant. Keep your answers short."} | e621ed4 |
| chat | None | 12:44:10.573 PM | {"content":"weather in seattle?"} | e621ed4 |
| chat | None | 12:44:10.574 PM | {"tool_calls":[{"id":"call_RRXFkrwZH7fG9zqBSvcheyHr","type":"function","function":{"name":"get_current_weather","arguments":"{\u002... | e621ed4 |
| chat | None | 12:44:10.575 PM | {"content":"42 and rainy","id":"call_RRXFkrwZH7fG9zqBSvcheyHr"} | e621ed4 |
| chat | None | 12:44:31.151 PM | {"index":0,"finish_reason":"stop","message":{"role":"assistant","content":"The current weather in Seattle is 42\u00B0F and rainy."}} | e621ed4 |

**gen_ai.user.message** opentelemetry.instrumentation.openai_v2

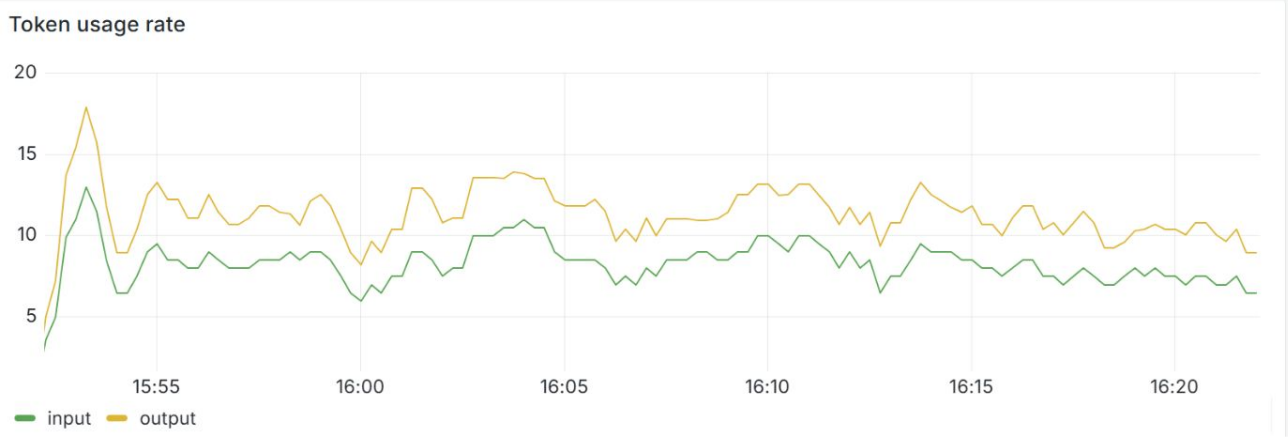
Resource **chat** Timestamp **12:44:10.573 PM**

| Name | Value |
|---|---|
| event.name | gen_ai.user.message |
| TraceId | e621ed425ccf3c284089aff13ee5bbad |
| SpanId | 8445855cc62a3cba |

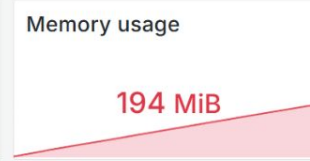
# Client side: metrics

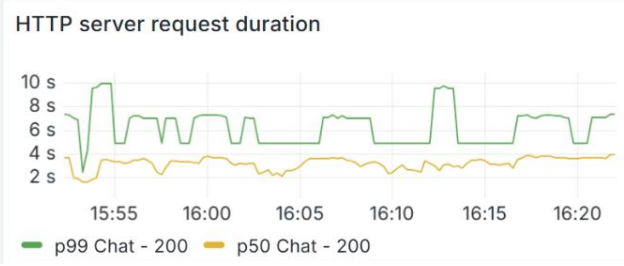










# Client telemetry is not enough



Effective self-hosting of models depends on insights into

- Deployment
- Resource management
- Scaling

# Model server telemetry: LLM performance

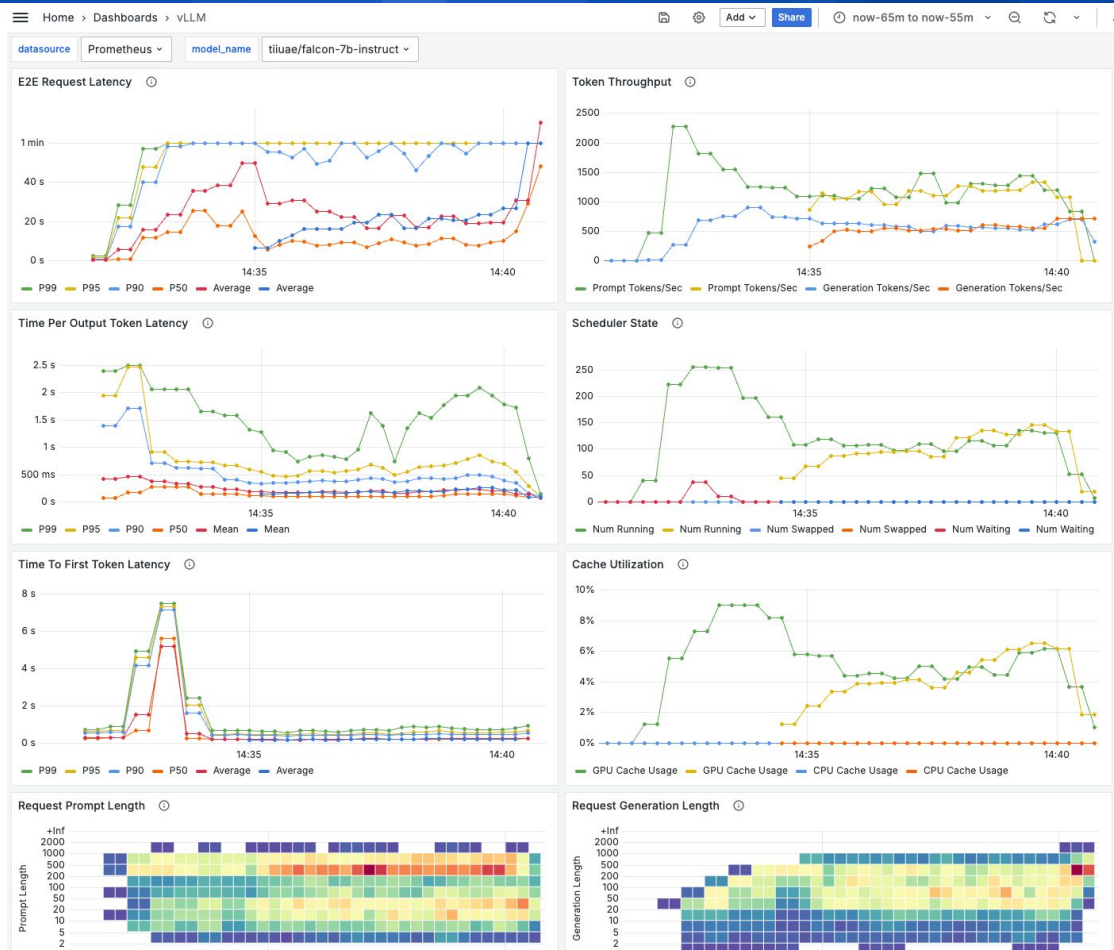


- Common metrics to measure performance
  - Throughput
    - Output tokens / second
    - Input tokens / second
    - Requests / second
  - Latency
    - Time to first token (TTFT)
    - Time per output token (TPOT)
    - Time per request
  - Price / Perf
    - $ per million output tokens
    - $ per million input tokens
    - Throughput / $

# Model server telemetry: metrics



- Detailed metrics for
  - Observability
  - Debugging
  - Performance optimization
- Various insights on
  - Load
  - Capacity
  - Latency
    - Admission
    - Inference



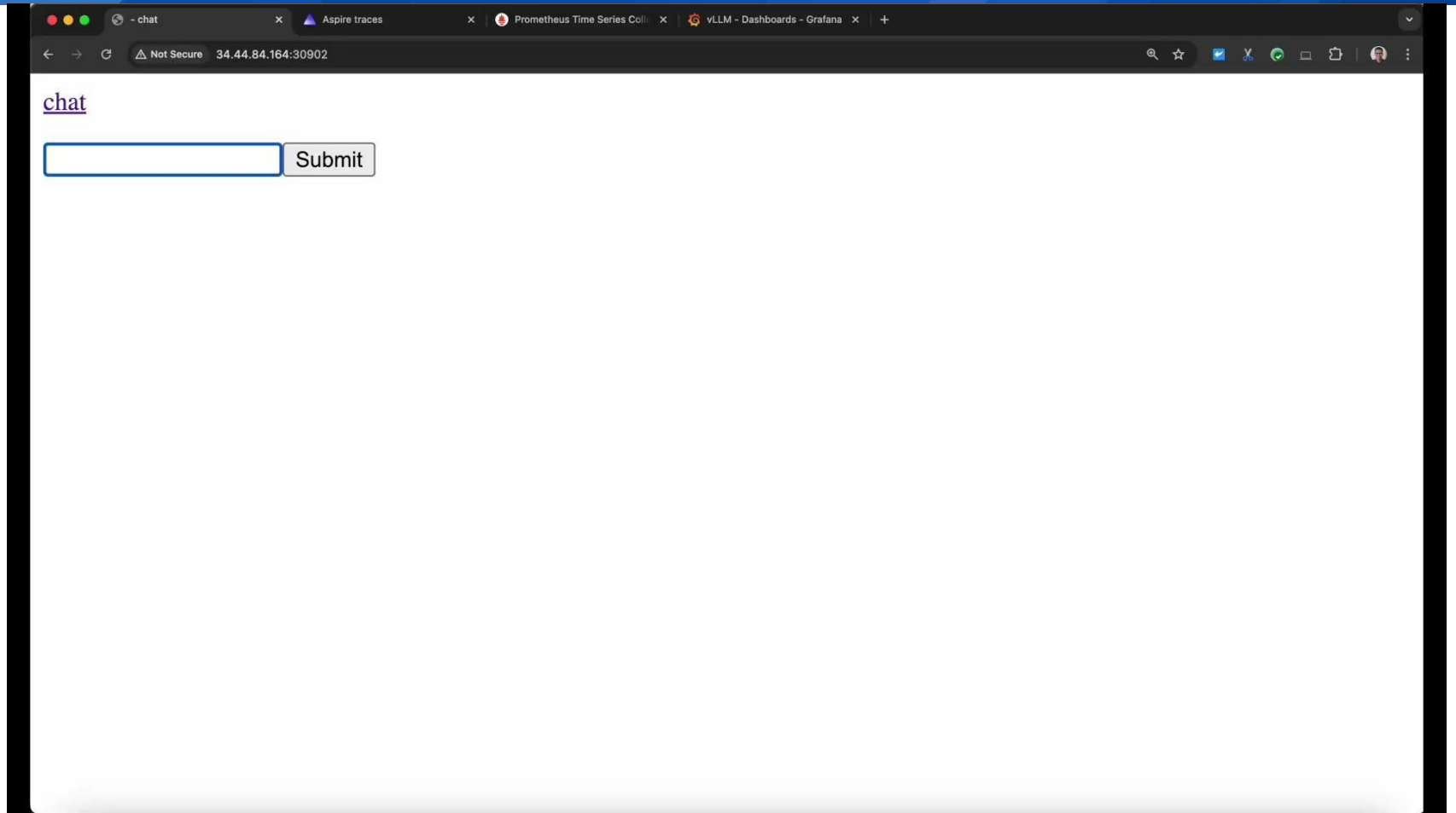
# Model server telemetry: More than just observability



- **Performance profiles** of different models on different accelerators
- Intelligent **load balancing**
- **Autoscaling** to address latency and throughput goals
- Priority or fairness based **scheduling**

# Demo: Debugging a performance issue





# LLM autoscaling: challenges



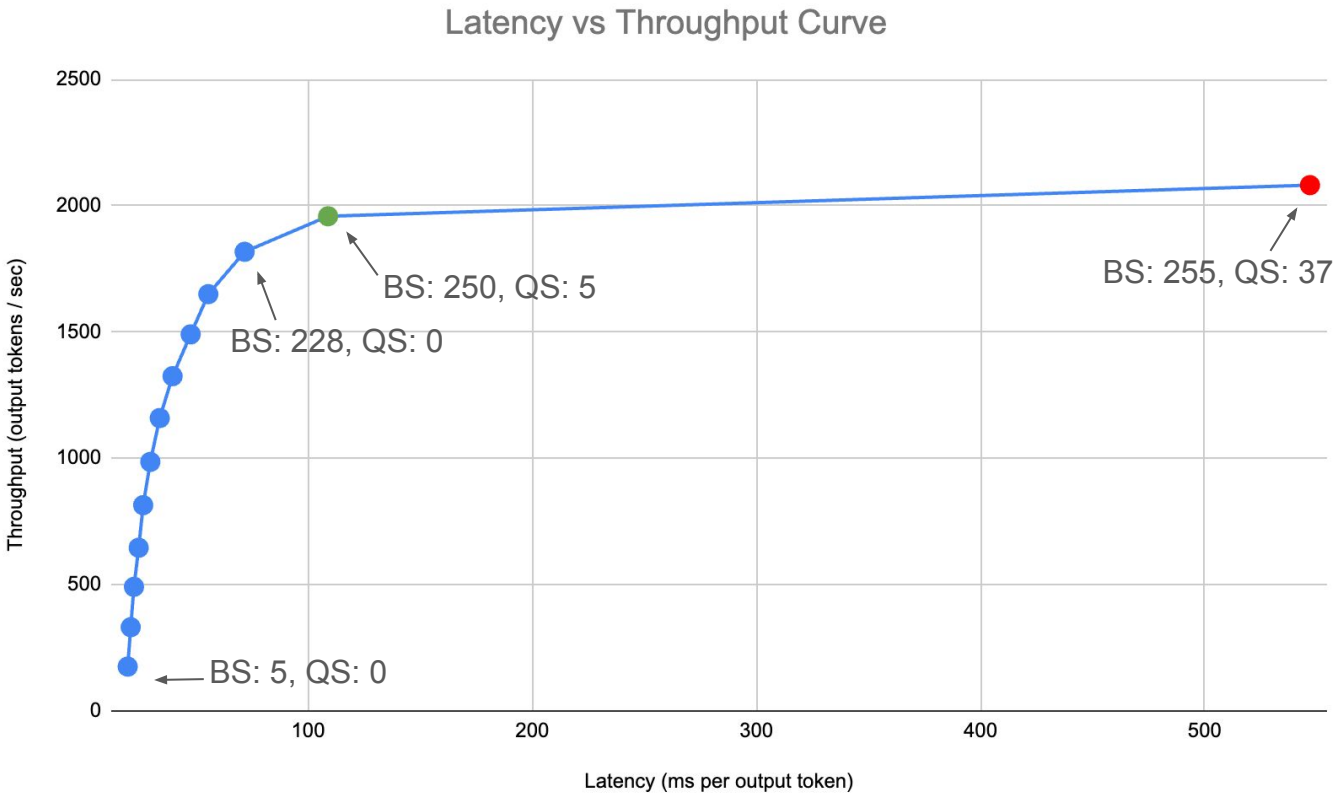
- Not as simple as web server autoscaling
- GPU utilization (duty cycle) is not a good indicator of model server load
- Longer pod start-up time because of model loading
- Different use cases require different approaches
  - Latency sensitive
  - Throughput optimized

# LLM autoscaling: hardware vs server metrics



- Lot of GPU utilization metrics available
  - Duty cycle
  - Power usage
  - Memory bandwidth usage
- LLM inference bottlenecks make it not so viable for autoscaling
  - Memory
  - Memory bandwidth
  - Compute
- Model server metrics provide a workload centric approach
  - Load and capacity are easily understandable

# LLM autoscaling: latency profile




Legend:

BS: Batch Size

QS: Queue Size

Blue dot: Low latency zone

Green dot: Inflection point

Red dot: High latency zone

Above chart generated using the [ai-on-gke benchmarking tool]

# LLM autoscaling: recommendations



**Lower Latency**

- Use **batch size or KV cache usage percentage**
  - Proactive approach
  - Might not utilize the accelerator fully
  - Might not hit peak throughput
- Batch size can vary based on context length
- KV cache usage is a percentage and more predictable

**Maximum Throughput**

- Use **queue size**
  - Reactive approach
  - Requests start to queue when model server is full
  - Easier to hit peak throughput and utilization
  - Latency can suffer

A detailed comparison is available in this blog post

# Ongoing work in the Kubernetes community

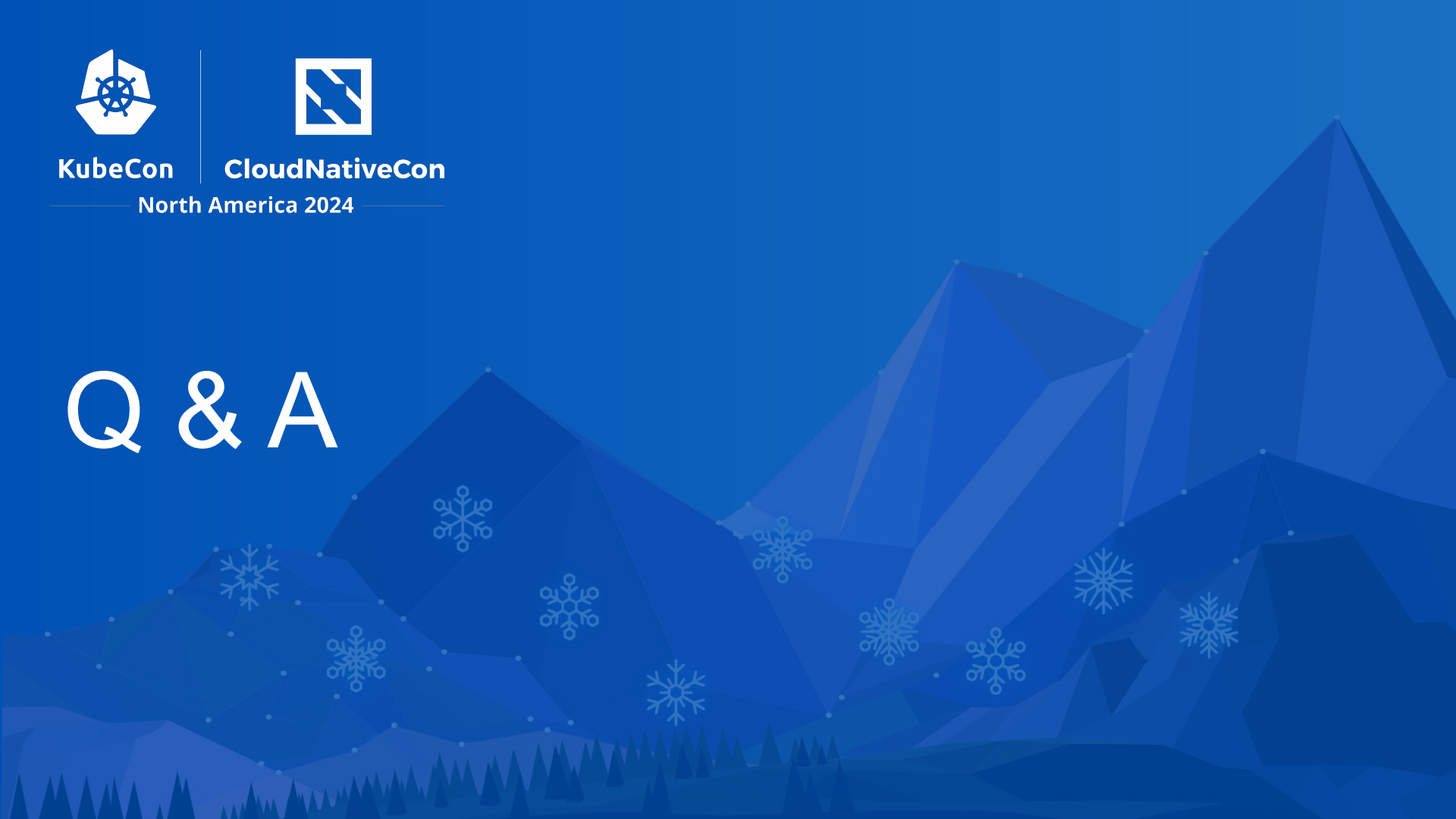


- Benchmarking of LLMs in a model server agnostic way easier
- Improved LLM scheduling and autoscaling support via instance gateway

# Call to action



- Participate in K8s Serving WG
  - Help standardize more metrics across more model servers
  - Help make benchmarking better
  - Improve tracing support
- Participate in OTel GenAI Semantic Conventions and Instrumentations SIG
  - Help us define conventions for evaluations, images, RAG, or anything else
  - Instrument your libraries and frameworks
  - Add usage and performance-related metrics


KubeCon
CloudNativeCon
North America 2024
Q & A

KubeCon CloudNativeCon
North America 2024

# Thank You!

Tell us how we did