# Semantic-aware LLM-Application Scheduling

Otto White

December 10, 2025

## Utilising LLM Applications

Easier than ever to write

Hard to productionize

## Utilising LLM Applications

Easier than ever to write

- LangChain, LlamaIndex, Pydantic AI, ...

Hard to productionize

# Utilising LLM Applications

Easier than ever to write

- LangChain, LlamaIndex, Pydantic AI, ...
- LLM Engines - vLLM, SGLang

Hard to productionize

## Utilising LLM Applications

Easier than ever to write

- LangChain, LlamaIndex, Pydantic AI, ...
- LLM Engines - vLLM, SGLang
- Public APIs

Hard to productionize

## Utilising LLM Applications

Easier than ever to write

- LangChain, LlamaIndex, Pydantic AI, ...
- LLM Engines - vLLM, SGLang
- Public APIs

Hard to productionize

- Reliability, Guardrails, Security

## Utilising LLM Applications

Easier than ever to write

- LangChain, LlamaIndex, Pydantic AI, ...
- LLM Engines - vLLM, SGLang
- Public APIs

Hard to productionize

- Reliability, Guardrails, Security
- **Performance & Efficiency** (our focus)

## Utilising LLM Applications

Easier than ever to write

- LangChain, LlamaIndex, Pydantic AI, ...
- LLM Engines - vLLM, SGLang
- Public APIs

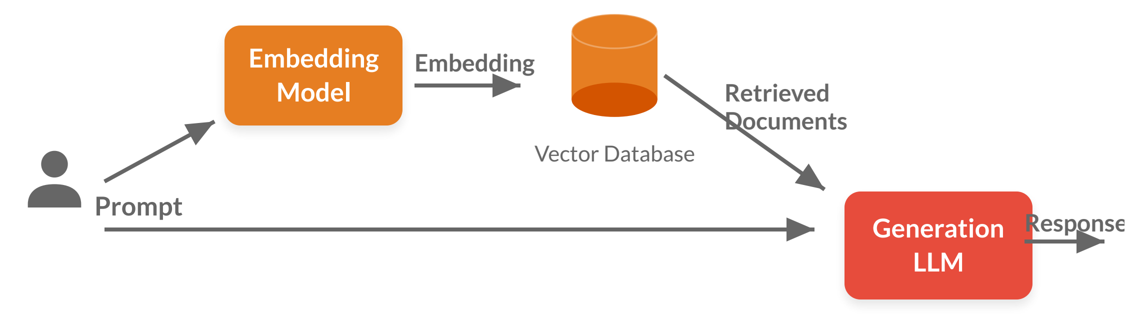
Hard to productionize

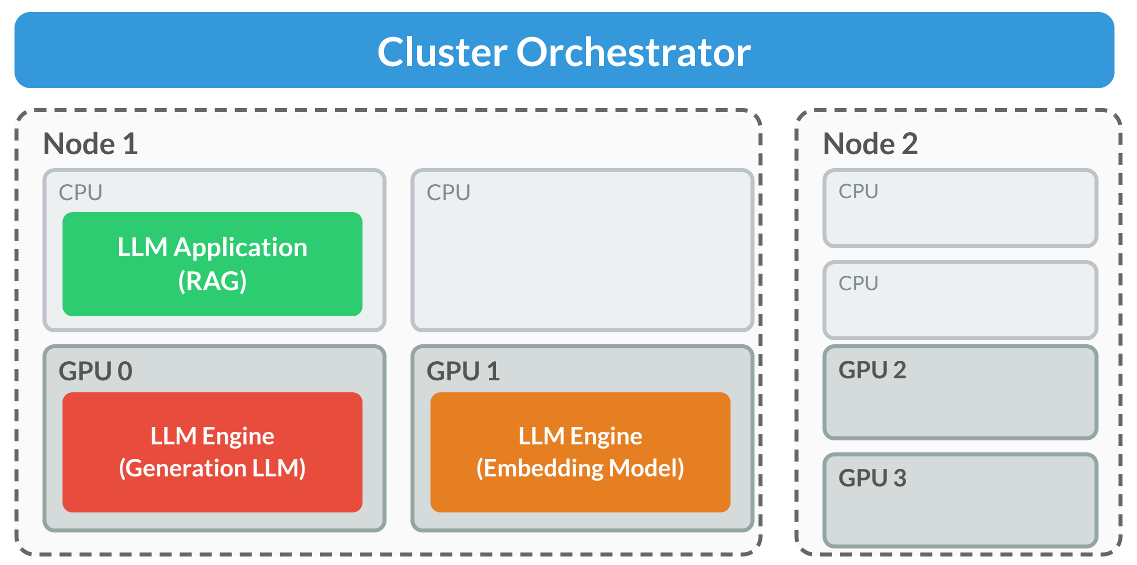
- Reliability, Guardrails, Security
- **Performance & Efficiency** (our focus)

Infeasible for companies at smaller scale to achieve efficient deployments applications. They need automated solutions.

# LLMs → LLM Applications

- LLM Invocations -> Graphs
- Can't optimise for end-to-end performance
- Lack of Critical Path Awareness
- Unfairness





## Related Work

| System | Level | Multi-Engine | Application-Aware | Scheduling Granularity /Co-location |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |

## Related Work

| System | Level | Multi-Engine | Application-Aware | Scheduling Granularity /Co-location |
|---|---|---|---|---|
| **Autellix** | LLM Engine | ✗ | ✓ | ✗ |
| | | | | |
| | | | | |

## Related Work

| System | Level | Multi-Engine | Application-Aware | Scheduling Granularity /Co-location |
|---|---|---|---|---|
| **Autellix** | LLM Engine | ✗ | ✓ | ✗ |
| **MuxServe** | Orchestration | ✓ | ✗ | ✓ |
| | | | | |

## Related Work

| System | Level | Multi-Engine | Application-Aware | Scheduling Granularity /Co-location |
|---|---|---|---|---|
| **Autellix** | LLM Engine | ✗ | ✓ | ✗ |
| **MuxServe** | Orchestration | ✓ | ✗ | ✓ |
| **Kubernetes** | Orchestration | ✓ | ✗ | ✗ |

# Related Work

| System | Level | Multi-Engine | Application-Aware | Scheduling Granularity /Co-location |
|---|---|---|---|---|
| **Autellix** | LLM Engine | ✗ | ✓ | ✗ |
| **MuxServe** | Orchestration | ✓ | ✗ | ✓ |
| **Kubernetes** | Orchestration | ✓ | ✗ | ✗ |
| **KServe** | Orchestration | ✓ | ✗ | ✗ |

## Critical-path Aware Parallelism

## Critical-path Aware Parallelism

- K8S default behavior:

## Critical-path Aware Parallelism

- K8S default behavior:
  - One model per GPU

## Critical-path Aware Parallelism

- K8S default behavior:
  - One model per GPU
  - Reactively scales based on queue sizes

# Critical-path Aware Parallelism

- K8S default behavior:
  - One model per GPU
  - Reactively scales based on queue sizes
  - Optimizes throughput, no latency awareness

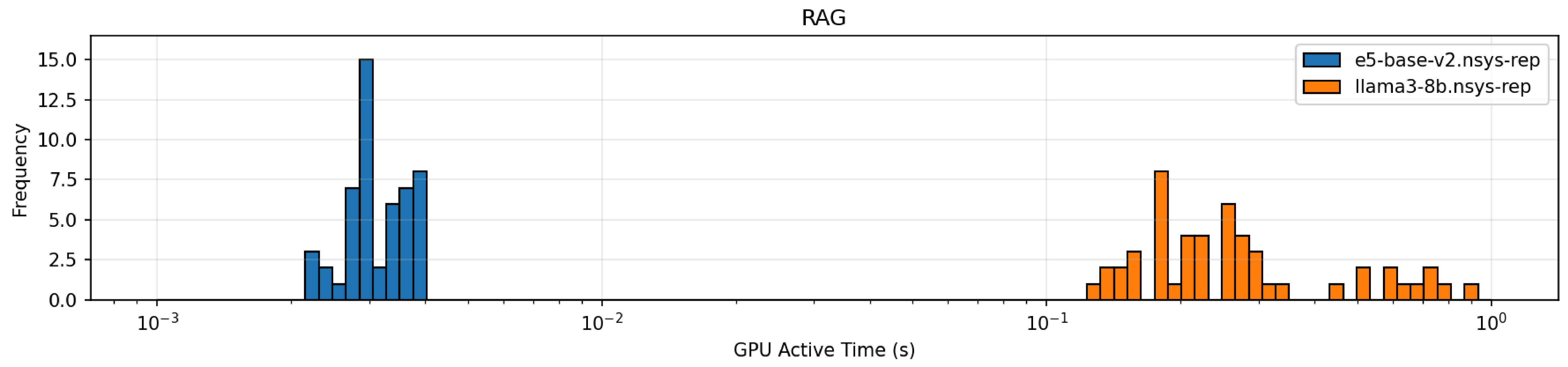
## Critical-path Aware Parallelism

- K8S default behavior:
  - One model per GPU
  - Reactively scales based on queue sizes
  - Optimizes throughput, no latency awareness
  - Only explores the data parallel dimension

## Critical-path Aware Parallelism

- K8S default behavior:
  - One model per GPU
  - Reactively scales based on queue sizes
  - Optimizes throughput, no latency awareness
  - Only explores the data parallel dimension
- We can do better

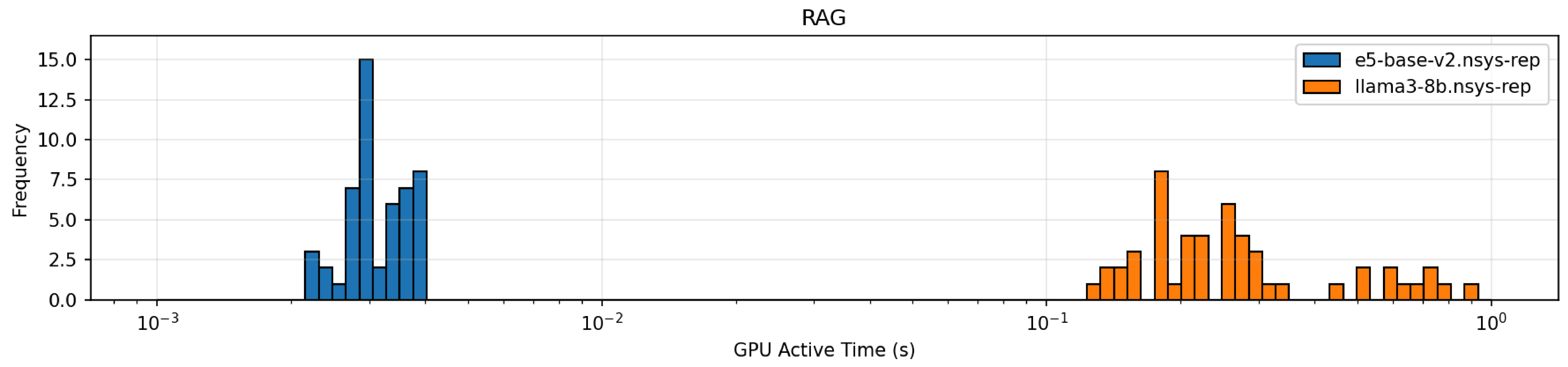
# Critical-path Aware Parallelism



- K8S default behavior:
  - One model per GPU
  - Reactively scales based on queue sizes
  - Optimizes throughput, no latency awareness
  - Only explores the data parallel dimension
- We can do better

# Critical-path Aware Parallelism

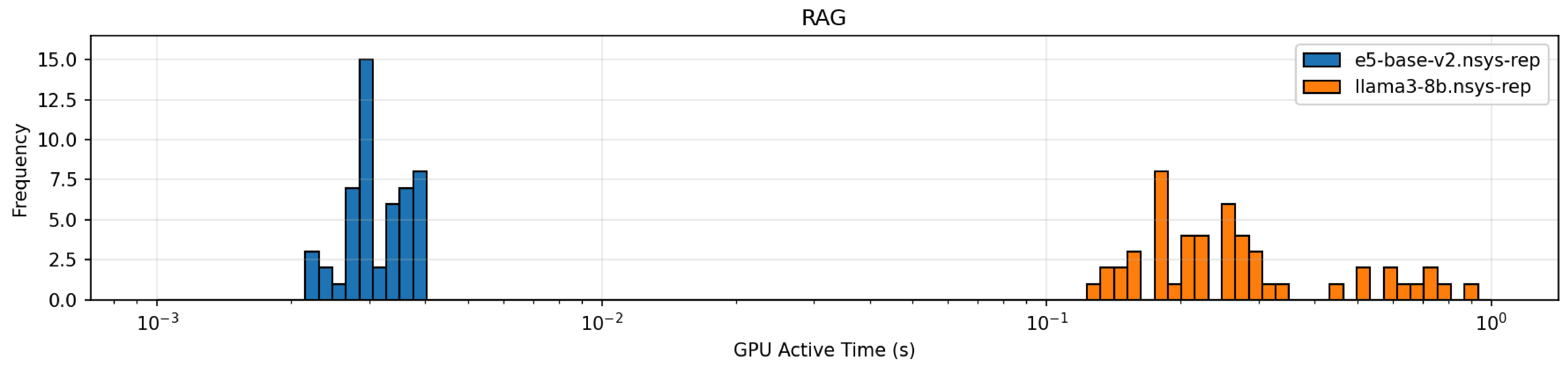
- K8S default behavior:
  - One model per GPU
  - Reactively scales based on queue sizes
  - Optimizes throughput, no latency awareness
  - Only explores the data parallel dimension
- We can do better



- LLM latency 100x embedding latency

# Critical-path Aware Parallelism

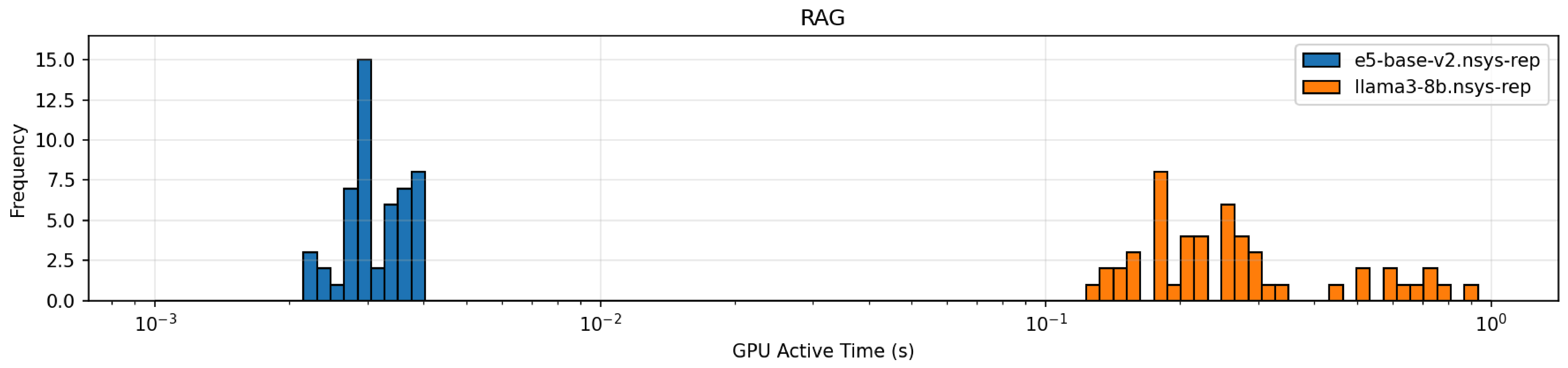
- K8S default behavior:
  - One model per GPU
  - Reactively scales based on queue sizes
  - Optimizes throughput, no latency awareness
  - Only explores the data parallel dimension
- We can do better



- LLM latency 100x embedding latency
- LLM is the critical path

# Critical-path Aware Parallelism

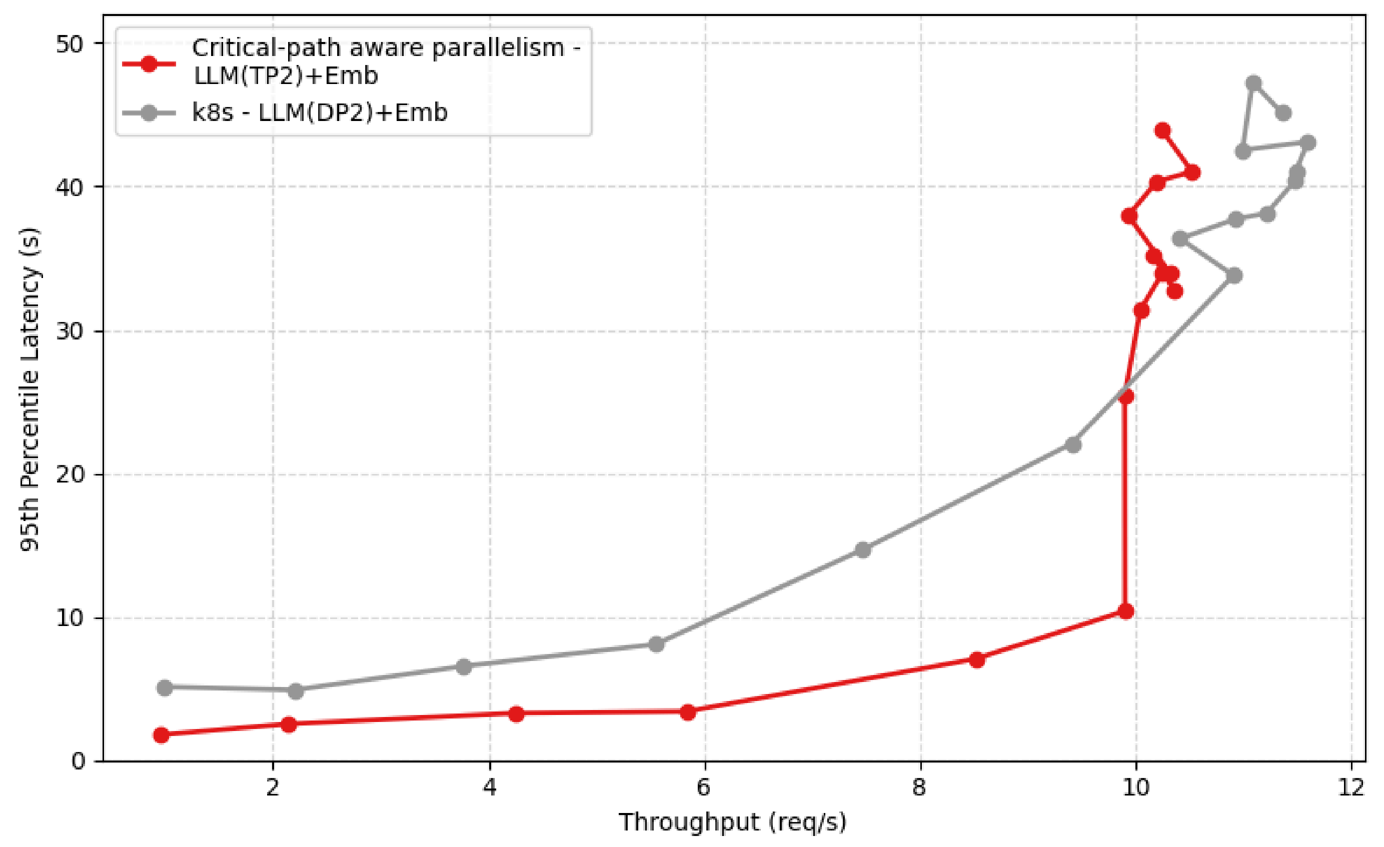
- K8S default behavior:
  - One model per GPU
  - Reactively scales based on queue sizes
  - Optimizes throughput, no latency awareness
  - Only explores the data parallel dimension
- We can do better



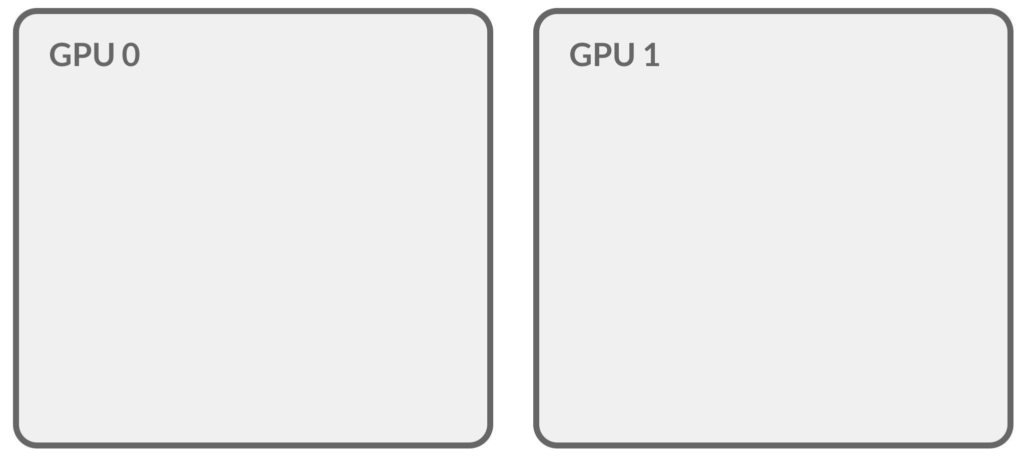
- LLM latency 100x embedding latency
- LLM is the critical path
- TP2 halves RAG application latency

# Critical-path Aware Parallelism

- **2.4x** improvement in latency
- Minor degradation in throughput

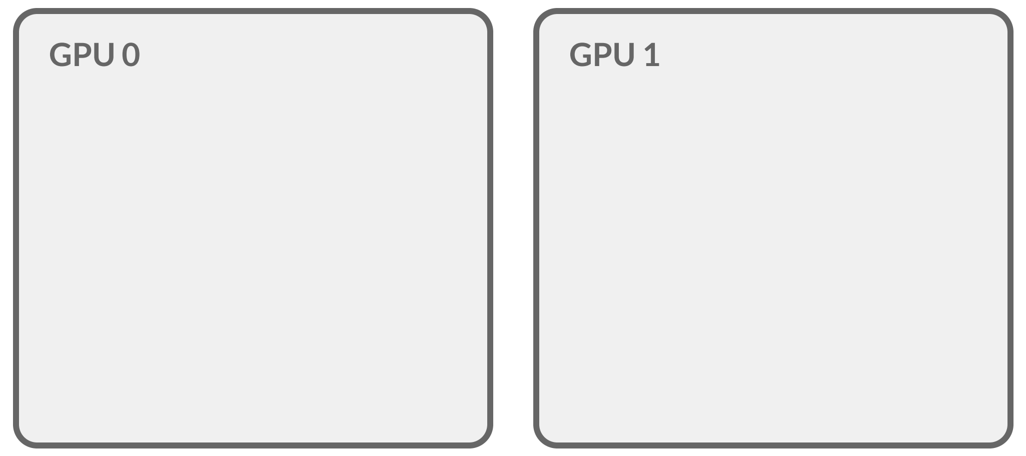


## Critical-path Aware Co-location



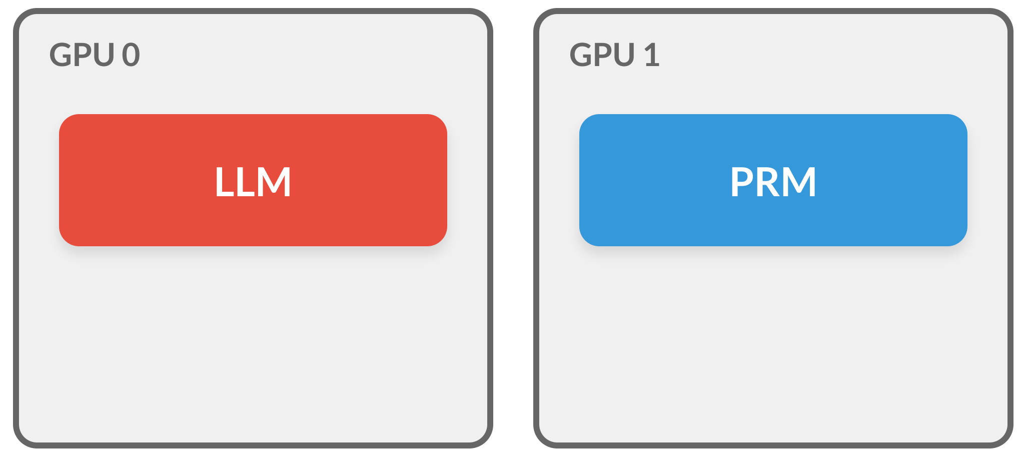
# Critical-path Aware Co-location

- How should we deploy this application?



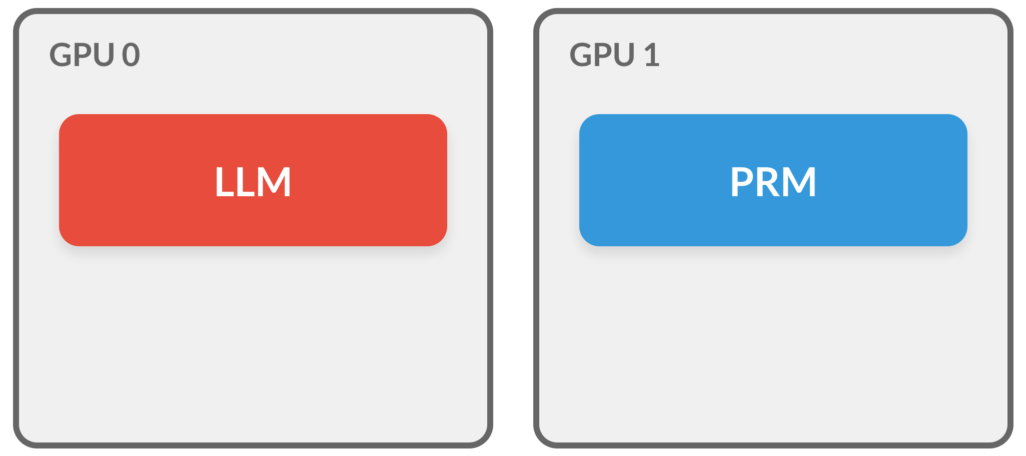
# Critical-path Aware Co-location

- How should we deploy this application?

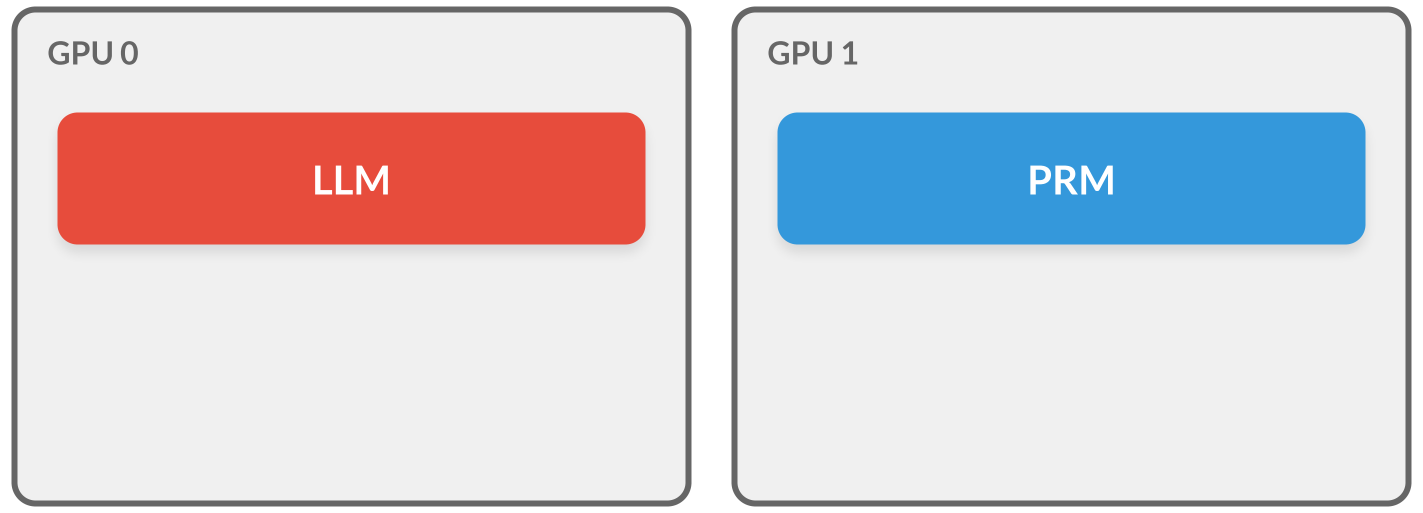


## Critical-path Aware Co-location

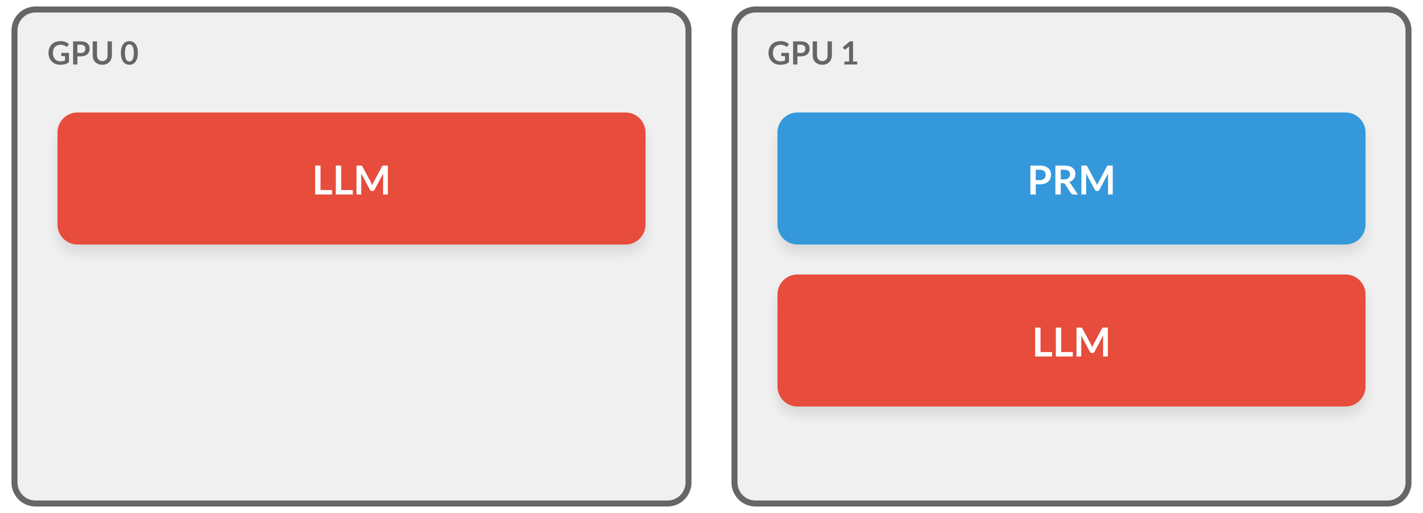
- How should we deploy this application?
- Severe underutilization



## Critical-path Aware Co-location

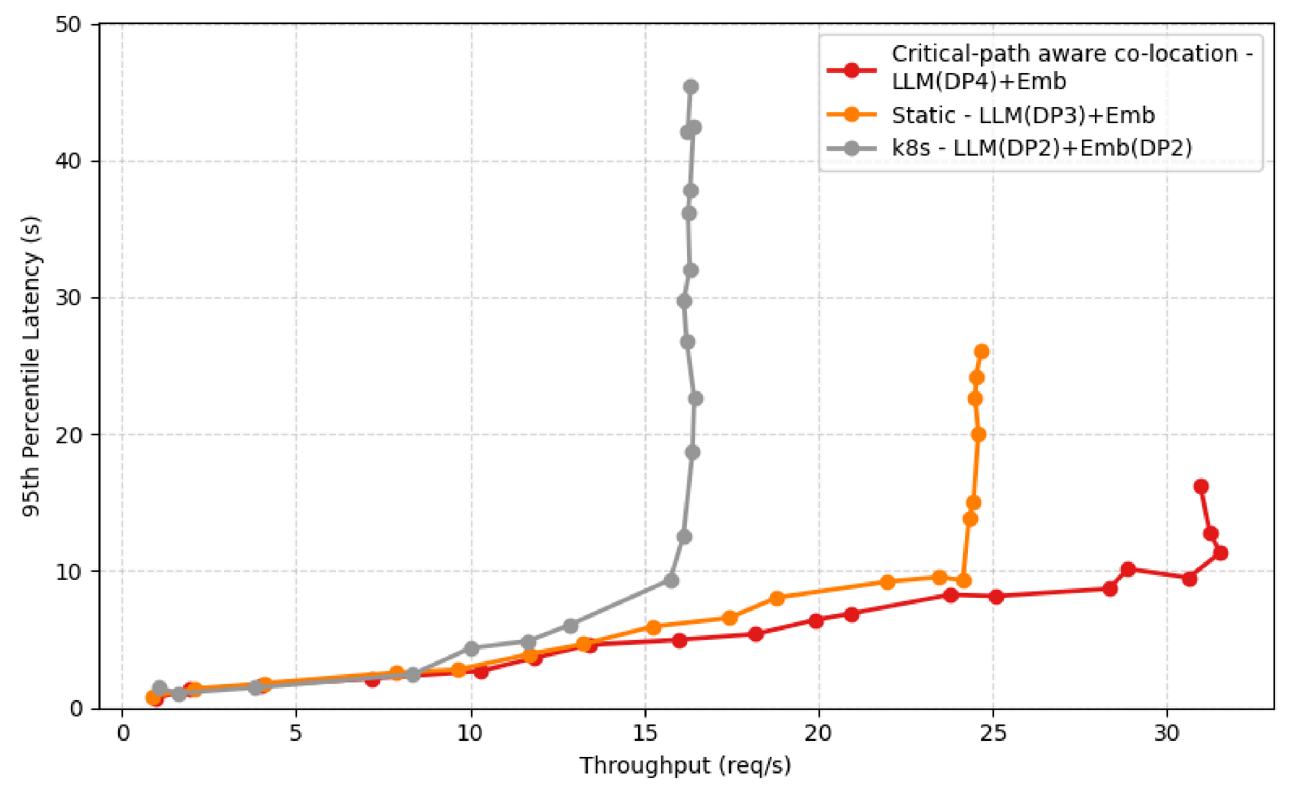


# Critical-path Aware Co-location

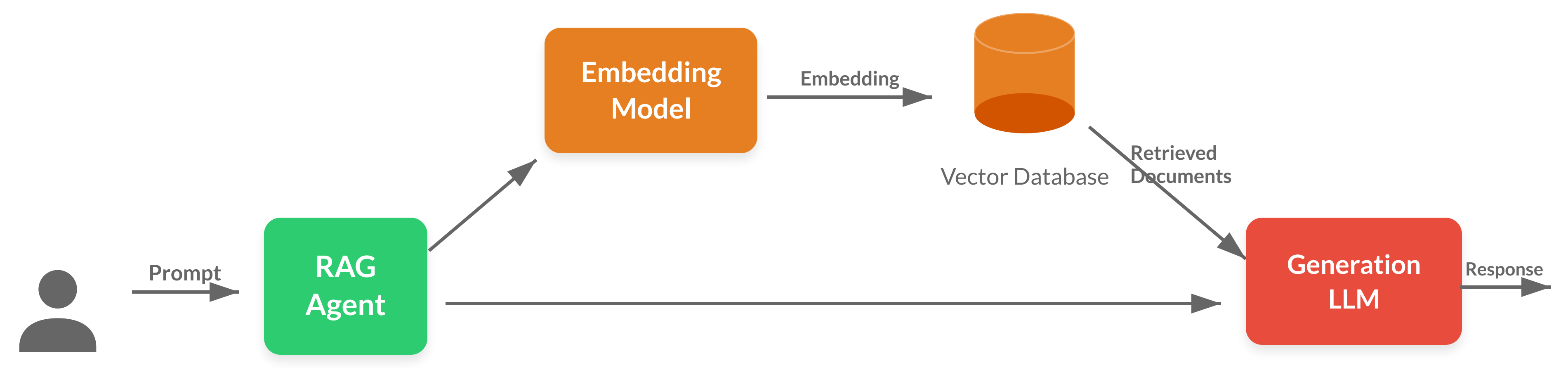


# Critical-path Aware Co-location

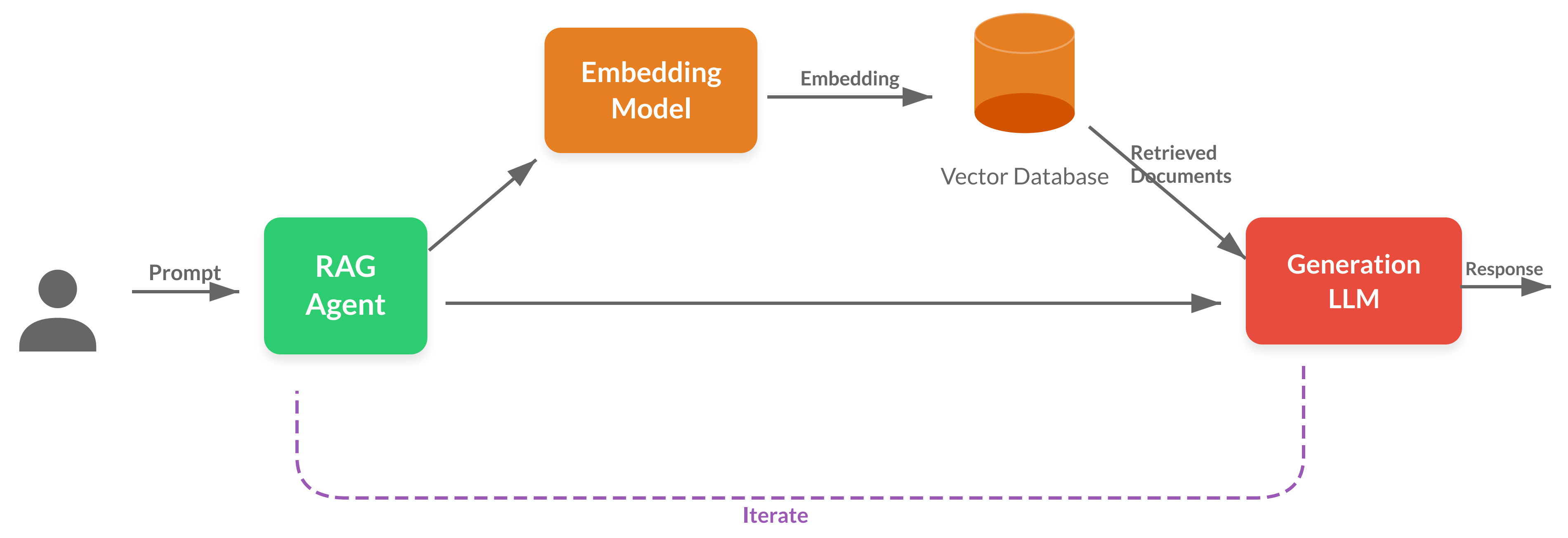
- **2x** throughput over K8S
- **50%** over best manual K8S config
- K8S: data parallelism only



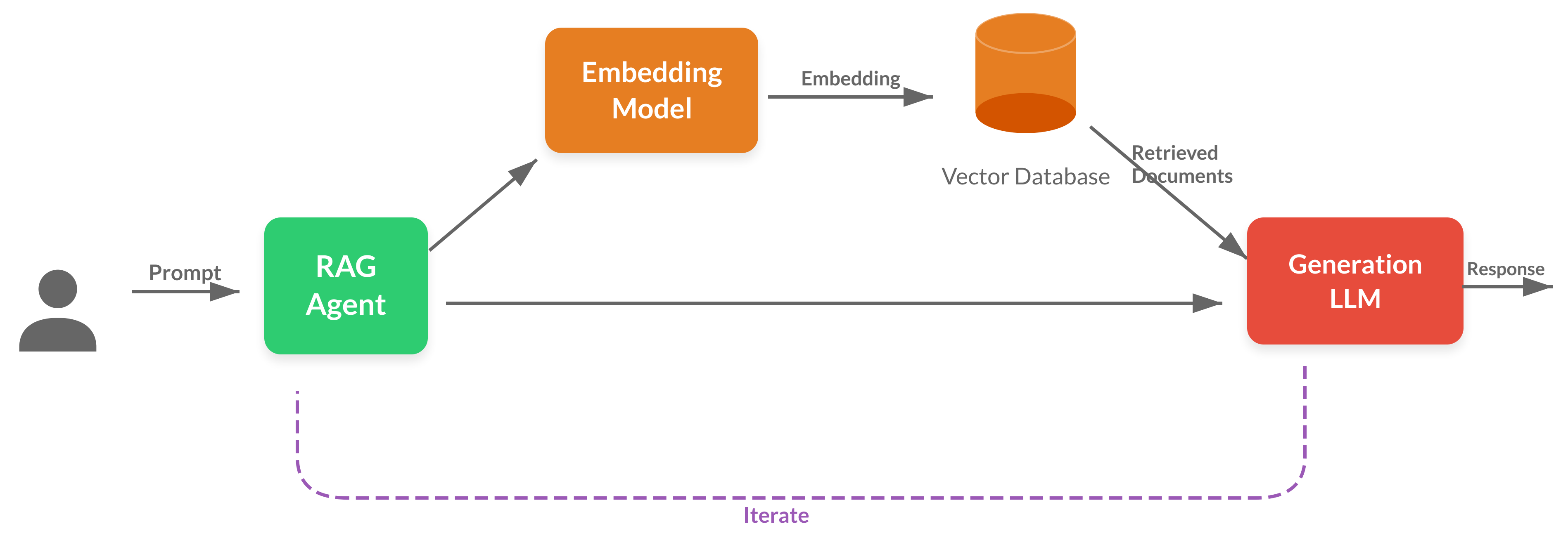
# Future Work: Multi-Engine Fairness



# Future Work: Multi-Engine Fairness

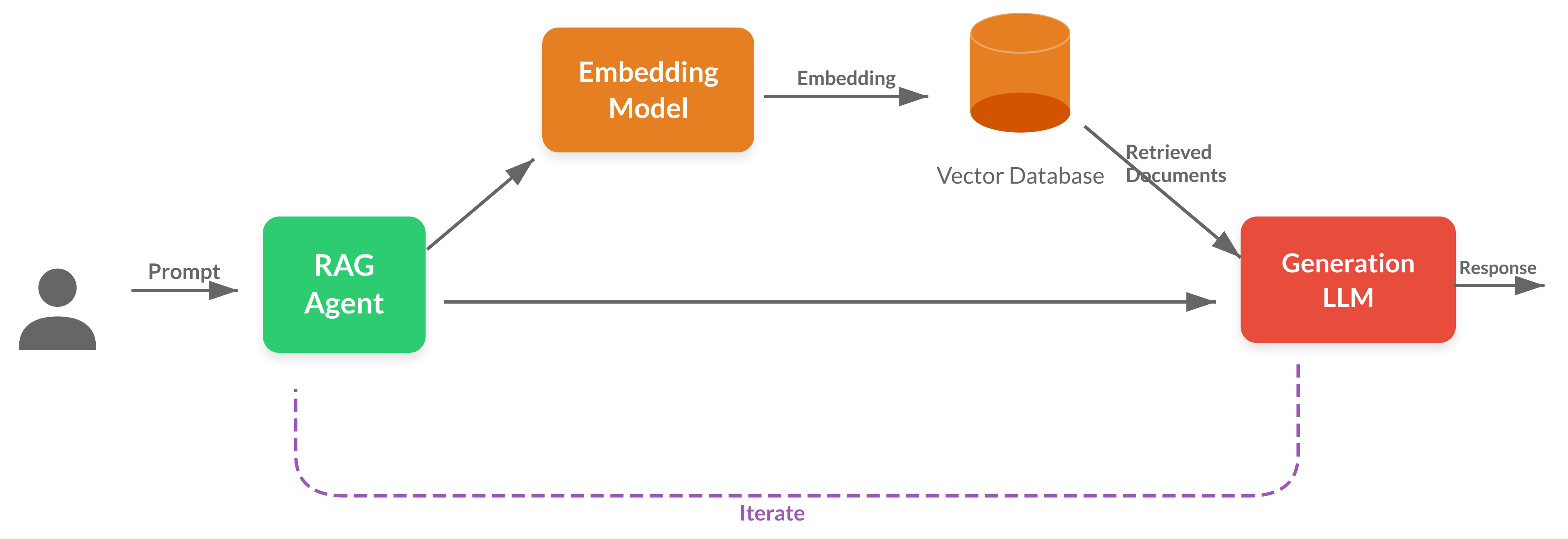


# Future Work: Multi-Engine Fairness



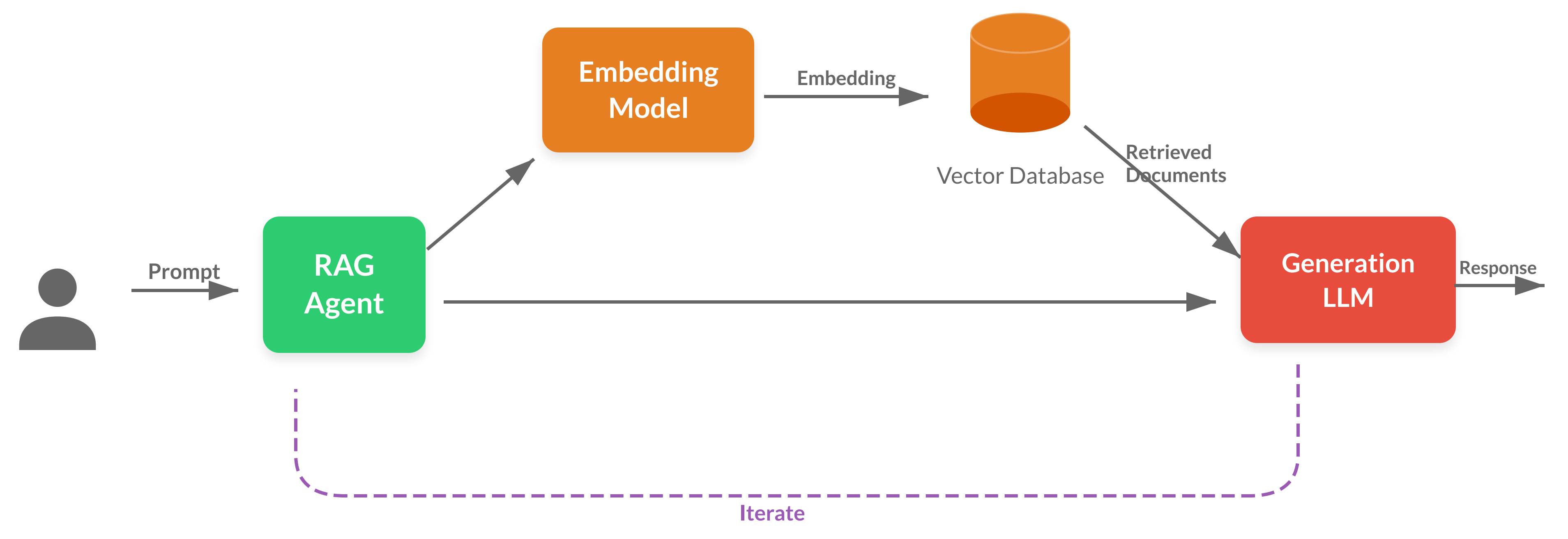
- Common pattern in agentic applications

# Future Work: Multi-Engine Fairness



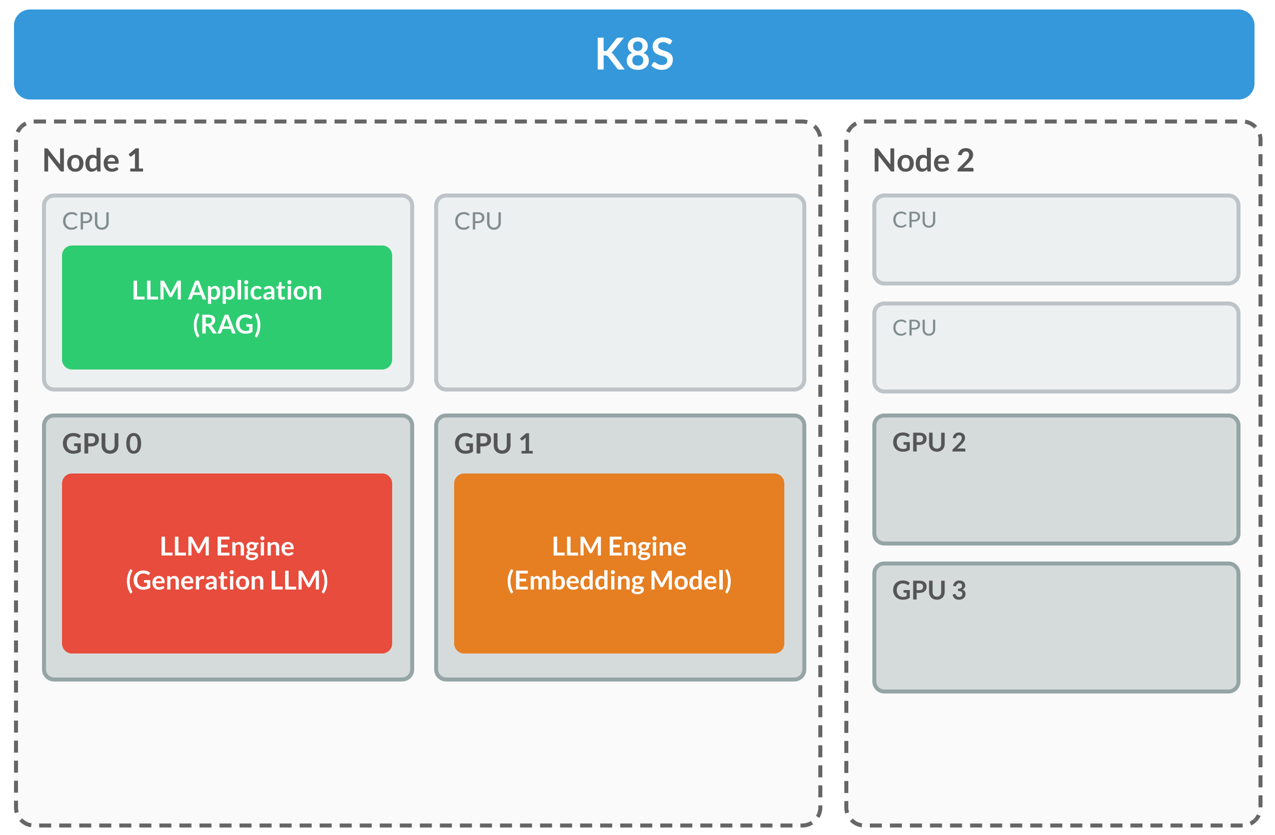
- Common pattern in agentic applications
- Some requests far more expensive than others

# Future Work: Multi-Engine Fairness



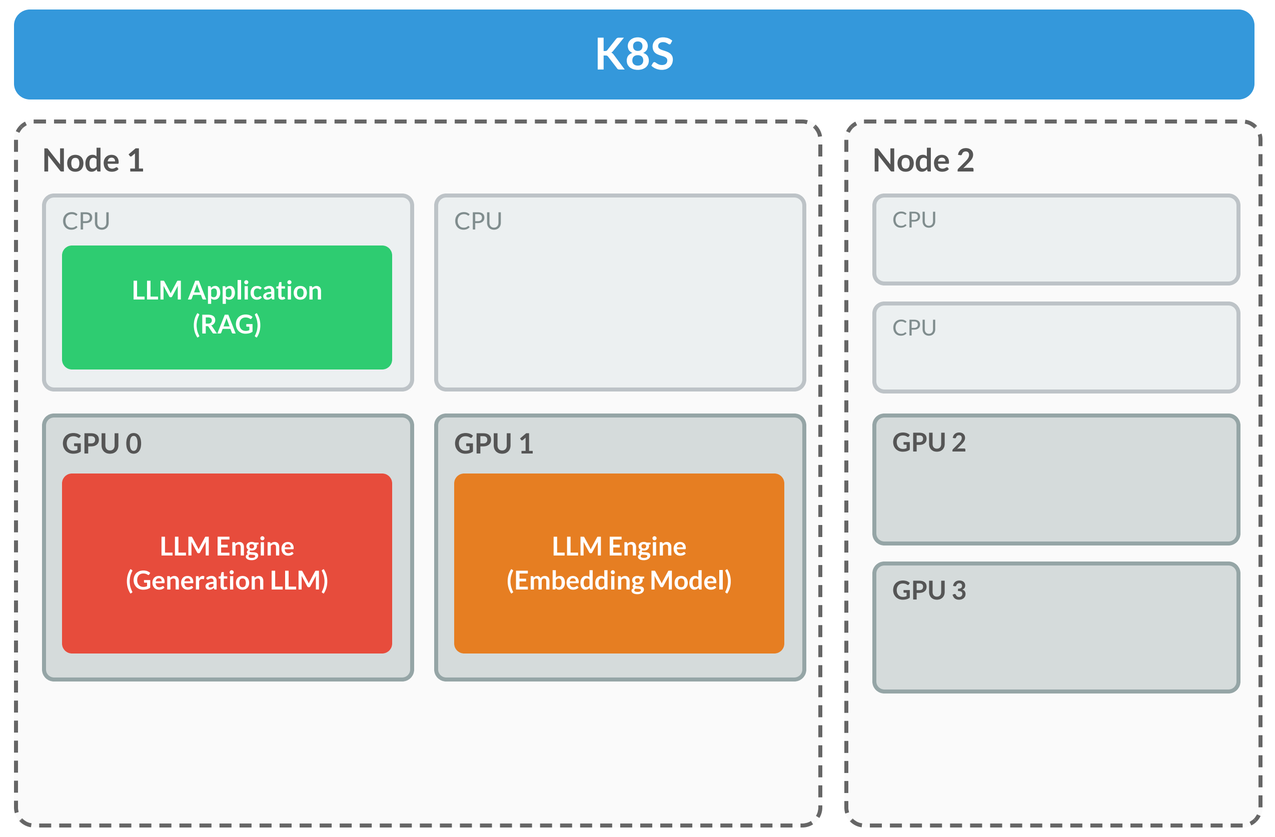
- Common pattern in agentic applications
- Some requests far more expensive than others
- HoL blocking for the many cheaper requests that iterate fewer times

# Future work: Multi-Engine Fairness



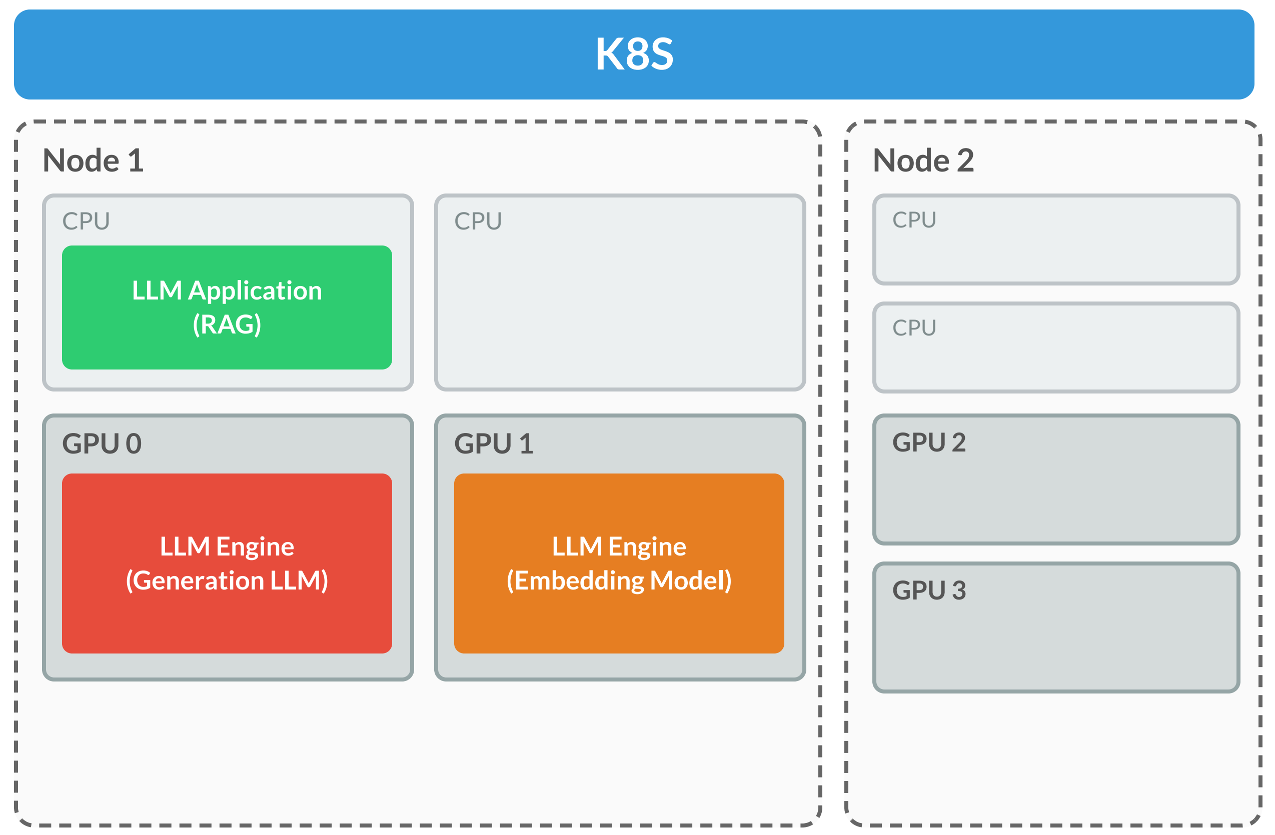
# Future work: Multi-Engine Fairness

- Current approaches only mitigate HoL blocking for single engine



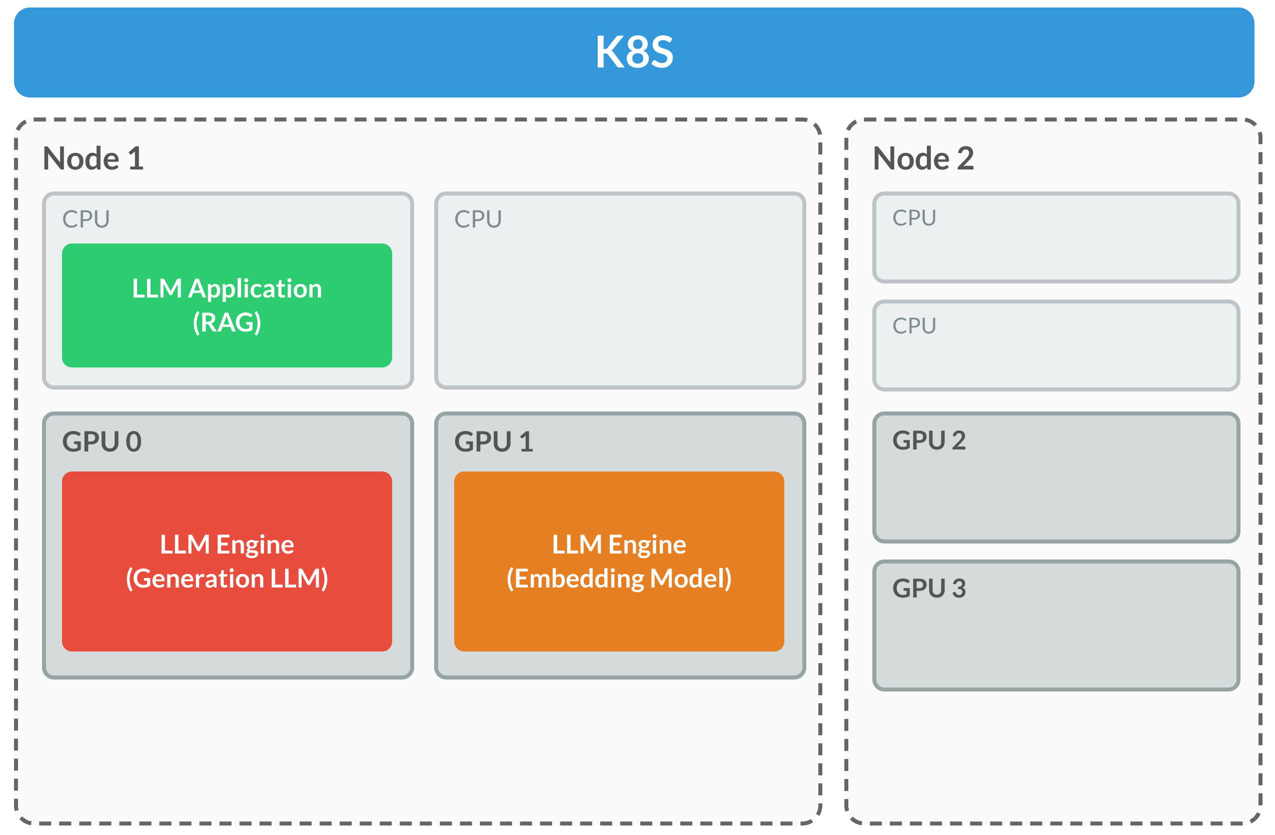
# Future work: Multi-Engine Fairness

- Current approaches only mitigate HoL blocking for single engine
- Applications can span multiple engines



# Future work: Multi-Engine Fairness

- Current approaches only mitigate HoL blocking for single engine
- Applications can span multiple engines
- Can also have HoL blocking between workflows


K8S
Node 1
CPU
LLM Application (RAG)
CPU
GPU 0
LLM Engine (Generation LLM)
GPU 1
LLM Engine (Embedding Model)
Node 2
CPU
CPU
GPU 2
GPU 3