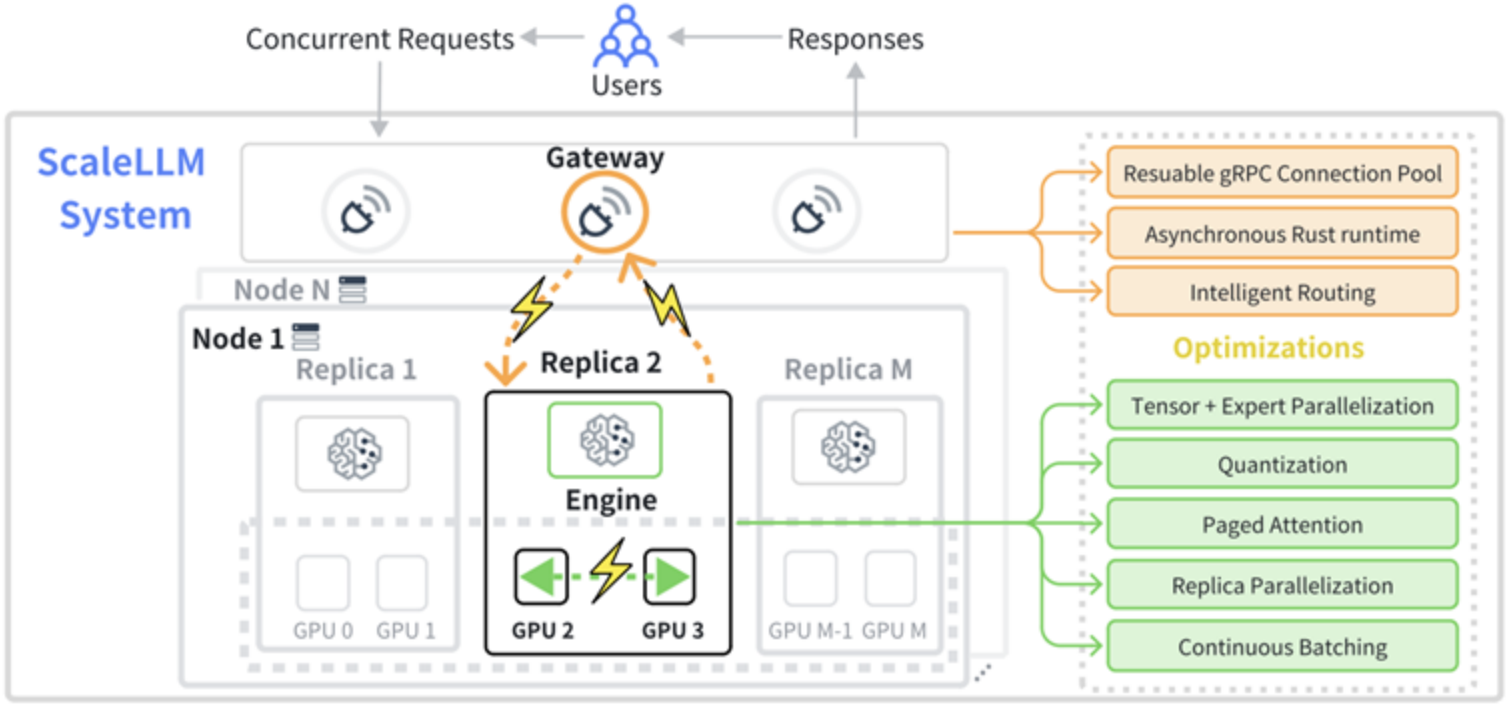
ScaleLLM: A Resource-Frugal LLM Serving Framework by Optimizing End-to-End Efficiency

Yuhang Yao, Han Jin, Alay Dilipbhai Shah, Shanshan Han, Zijian Hu, Yide Ran, Dimitris Stripelis, Zhaozhuo Xu, Salman Avestimehr, and Chaoyang He

Overview of ScaleLLM Serving System

TensorOpera AI

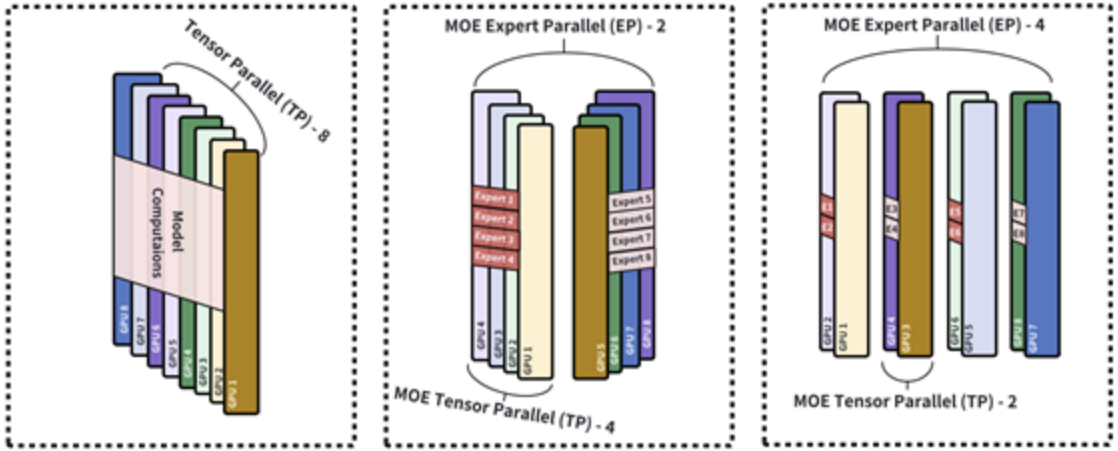


Gateway and serving engine are the key components.

Optimize LLM Serving Engine

TensorOpera AI

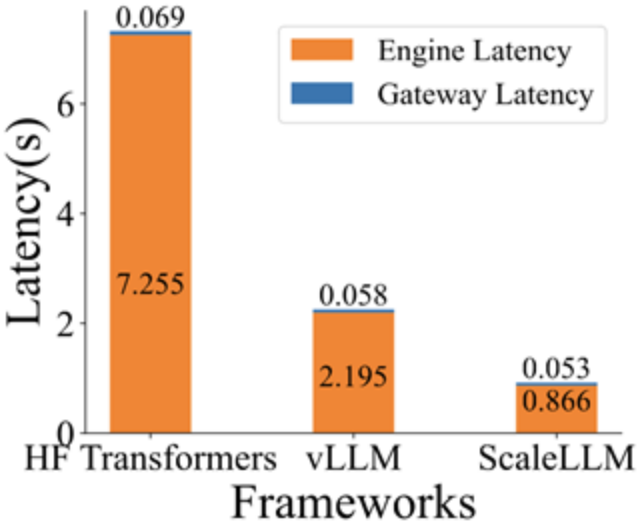
- Model Parallelization



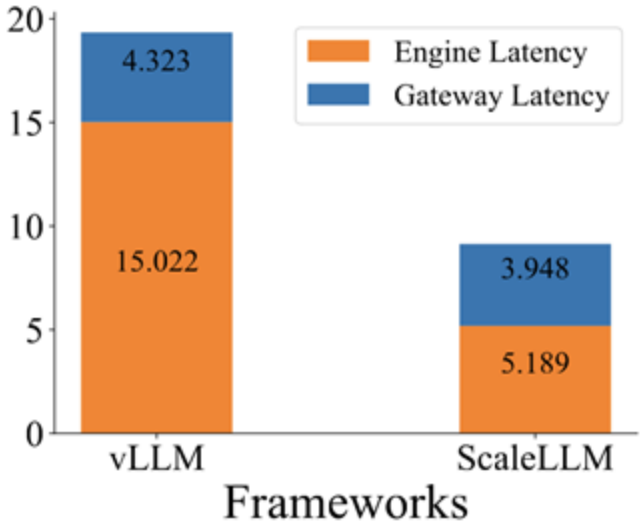
- Model Quantization: fp8
- Continuous Batching and Batch Scheduler
- FlashAttention and PagedAttention

Evaluation of LLM Serving Latency

TensorOpera AI



(a) Concurrency: 4.



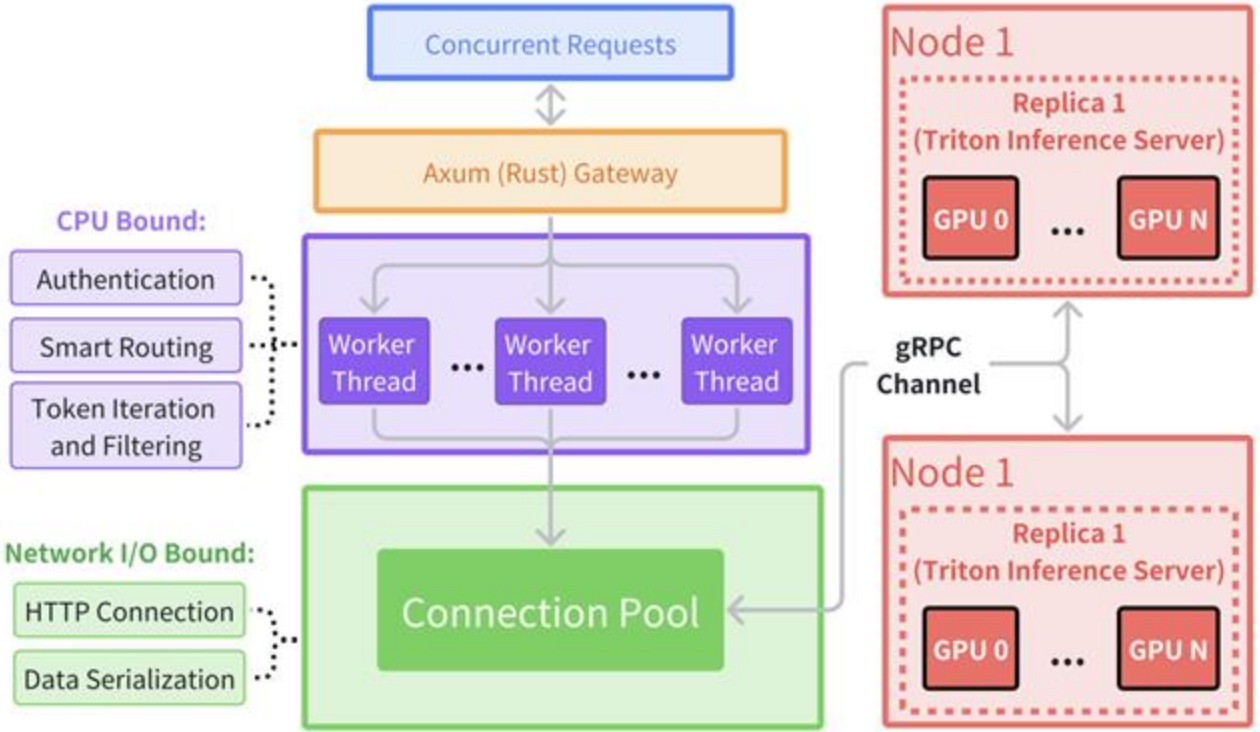
(b) Concurrency: 256.

Comparisons with the two baseline solutions. ScaleLLM is applied without gateway optimization.

Gateway becomes the new bottleneck after optimizing the Seving Engine

Optimize Gateway Latency

TensorOpera AI

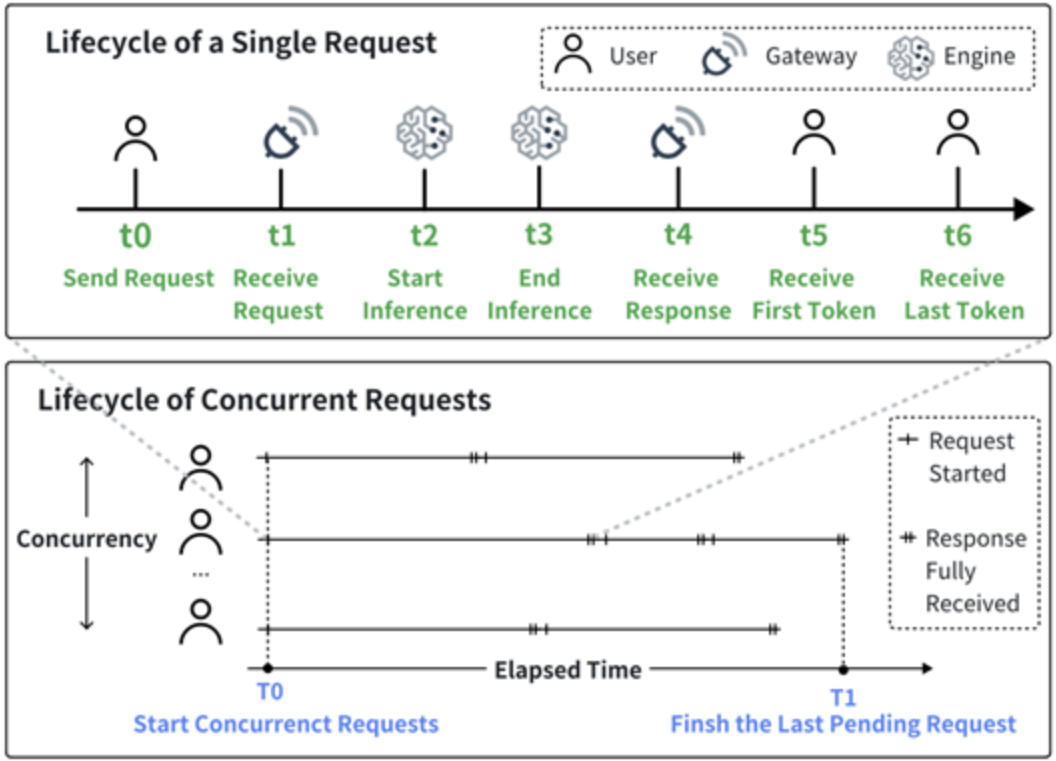


Key Features:

- CPU Bound Job Optimization
- Network I/O Bound Job Optimization

Evaluation of LLM Serving Latency

TensorOpera AI



Send X concurrent requests and record the latency

Latency with Streaming Output

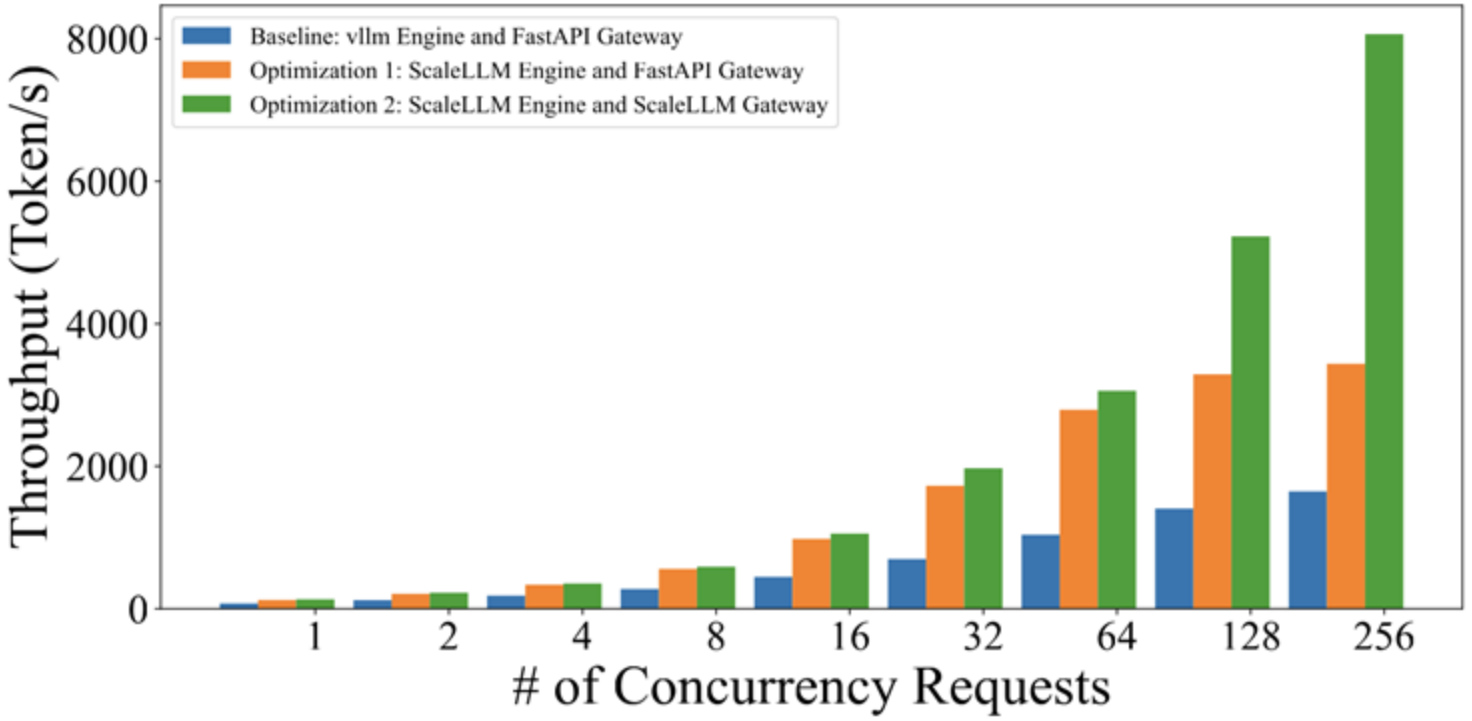
TensorOpera AI

Concurrent Requests	Huggingface Endpoint		vLLM Endpoint		ScaleLLM	
	TTFT/ms	TBT/ms	TTFT/ms	TBT/ms	TTFT/ms	TBT/ms
1	315.6	83.4	48.4	16.5	25.0 (1.9x)	8.5 (1.9x)
2	637.2	218.3	51.9	16.7	25.3 (2.1x)	8.7 (1.9x)
4	1157.8	506.4	55.1	21.1	25.5 (2.2x)	10.4 (2.0x)
8	Timeout	Timeout	70.2	30.1	25.9 (2.7x)	12.2 (2.5x)
16	Timeout	Timeout	93.1	38.3	26.7 (3.5x)	13.4 (2.9x)
32	Timeout	Timeout	135.8	50.1	29.8 (4.5x)	14.6 (3.4x)
64	Timeout	Timeout	285.4	70.8	99.4 (2.9x)	16.5 (4.3x)

Smaller TTFT means faster response for the first token and smaller TBT means faster generation of tokens.
Timeout: 90% of the users' requests cannot complete in 60s.

TensorOpera AI

Throughput vs Number of Concurrent Requests.



Optimizing gateway is as important as optimizing engine

ScaleLLM on Mixtral-8x7B

TensorOpera AI

https://tensoropera.ai/prod/model/mistralai/ScaleLLM-Mixtral-8x7B

A Bad Inference Optimization Strategy

TensorOpera AI

1. **Find something in the paper/arxiv/blog**

   May not suitable of the current system

1. **Spend time understand and integrate**

   Waste time in research/development

1. **Measure the speedup**

   Maybe the gain is less than 10%

A Good Inference Optimization Strategy

TensorOpera AI

1. **Apply the current infrastructure**

   Start with the current solution.

1. **Profile the efficiency bottleneck**

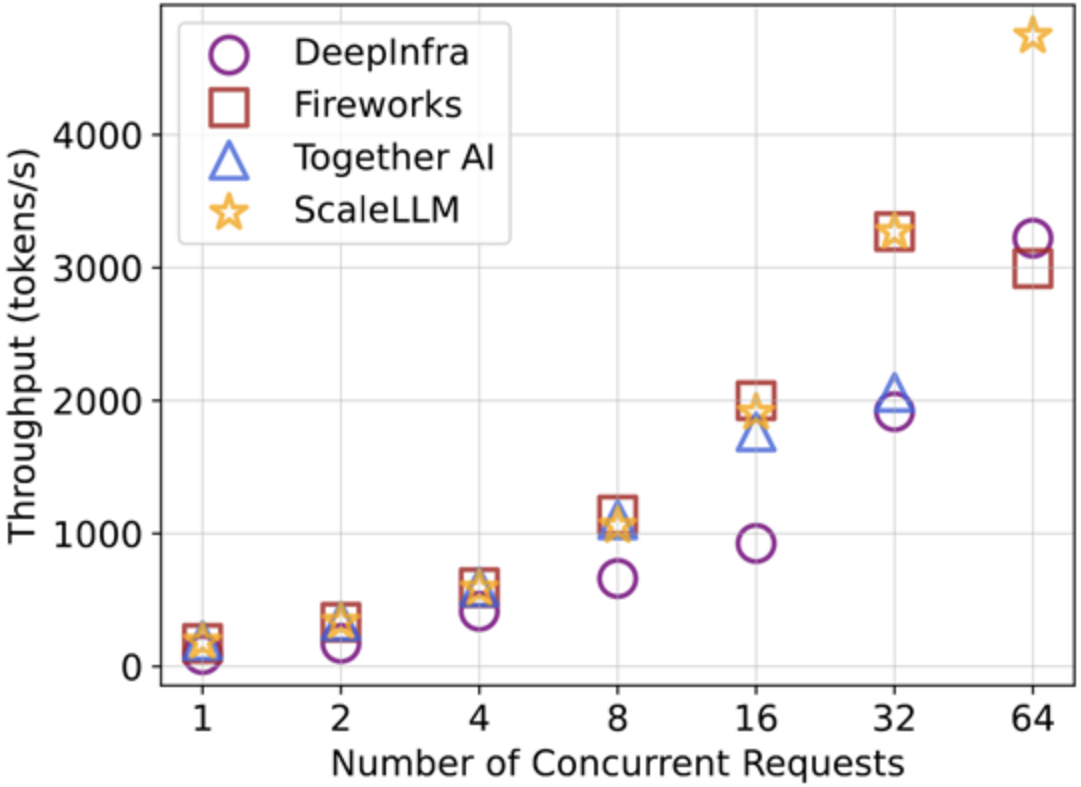
   Quantify the impact of each part in the endpoint

1. **Always solve the most inefficient bottleneck**

   Search for the right techniques and apply

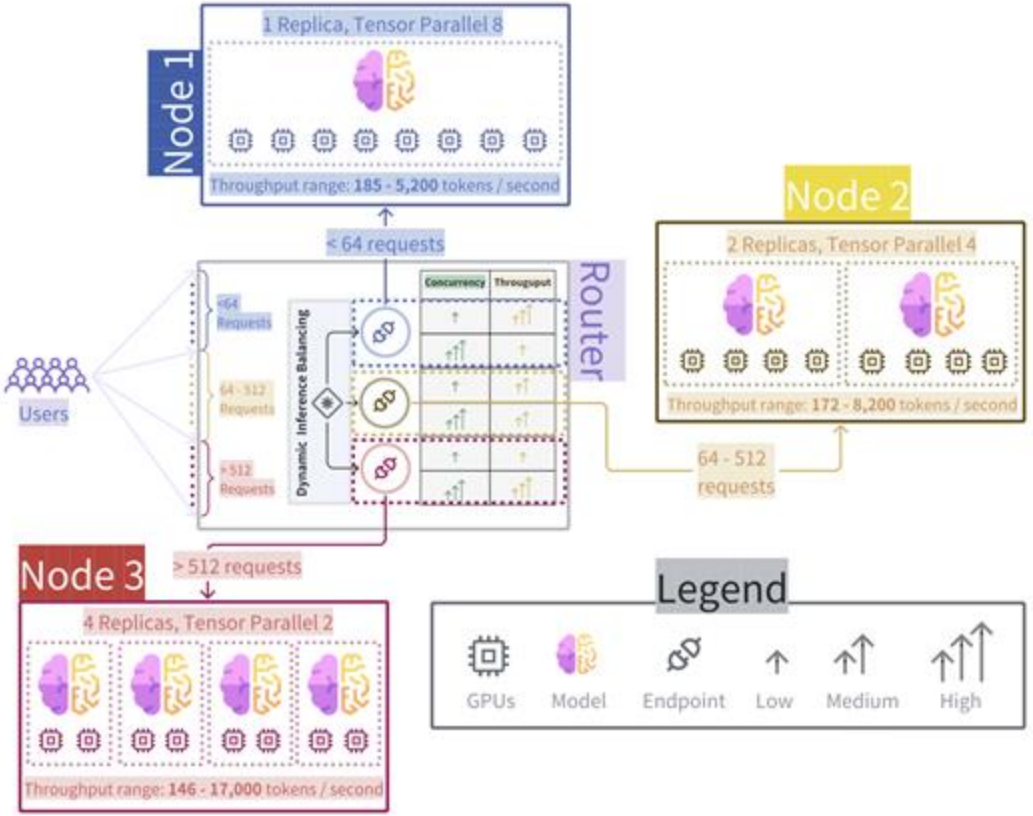
Endpoints Throughput Evaluation

TensorOpera AI



- Comparable with State-of-The-Art endpoints
- 1.5X faster when sending 64 concurrent requests

Blueprint: Dynamic Inference Load Balancing System



Low concurrency (< 64 requests)

Fewer replicas but higher tensor parallelism to optimize resource utilization for smaller batch computations.

High concurrency (≥ 64 requests)

More replicas but lower tensor parallelism effectively distributing the workload to squeeze everything out of available compute.

TensorOpera AI