

MASARYK
UNIVERSITY

ETH zürich



Engine-Agnostic Model Hot-Swapping for Cost-Effective LLM Inference

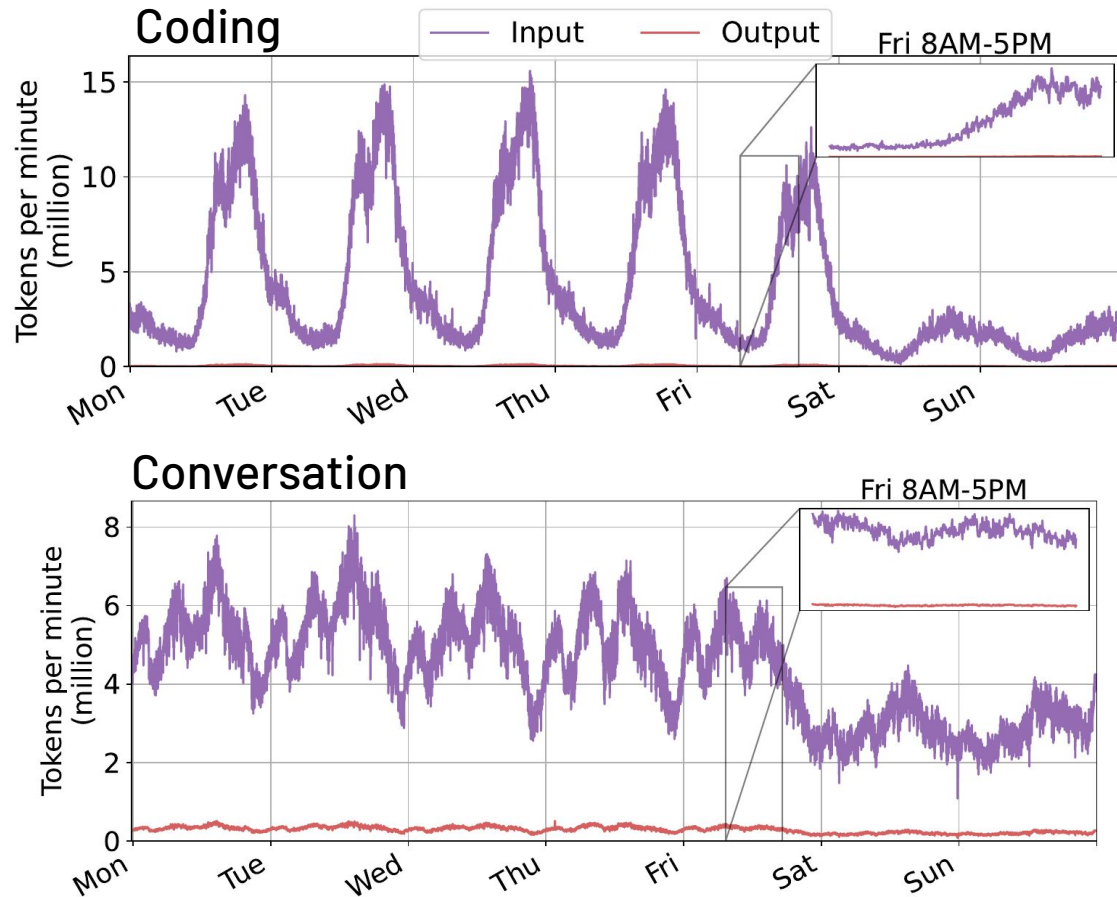
Radostin Stoyanov - PhD Student, Scientific Computing Group

Collaboration with Viktória Spišaková, Marcin Copik, and Adrian Reber

Supervisors: Prof. Rodrigo Bruno, Prof. Wes Armour



Real-World LLM Deployments



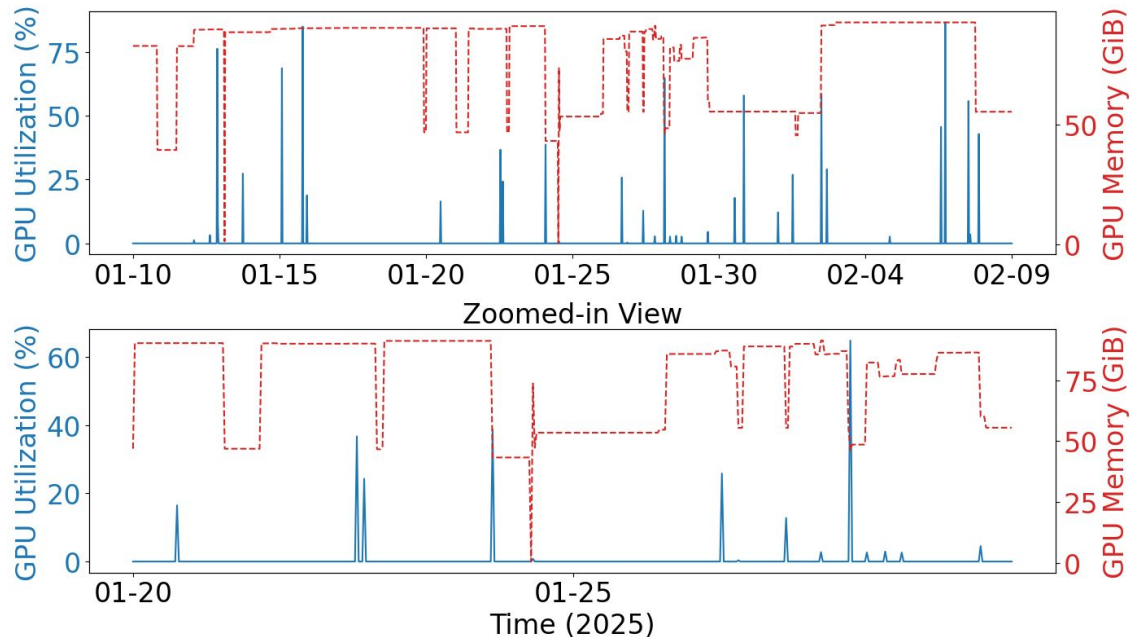
LLM Inference Workloads Characteristics

- Large number of queries with strict SLOs
- Response length is unpredictable
- Run on expensive, energy-intensive GPUs
- GPU resources are often underutilized

Microsoft Azure LLM Inference Dataset Trace: <https://github.com/Azure/AzurePublicDataset>

GPU Utilization of LLM Inference

e-INFRA CZ Kubernetes Cluster



H100 NVL (94GB) serving LLaMA 3.3, Qwen 2.5-Coder, Command R7B, Phi-4, QWQ, and MXBai-Embed-Large

Challenges

- Always-on LLMs keep GPUs reserved
- Low utilization due to sporadic requests
- Cold-start times of inference engines causing unacceptable TTFT

LLM Inference Optimizations

Key Performance Metrics

- **Latency per request:** *time-to-first-token (TTFT)* or *time-to-last-token (TTLT)*
- **Throughput:** *tokens/sec* (model speed) or *requests/sec* (request-handling)

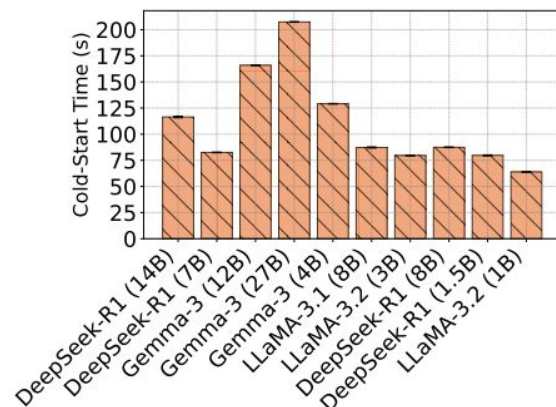
Optimization Strategies

- Efficient KV cache management with distributed inference
- Iteration-level scheduling with continuous batching
- Smaller (specialized) models are faster & less expensive to run

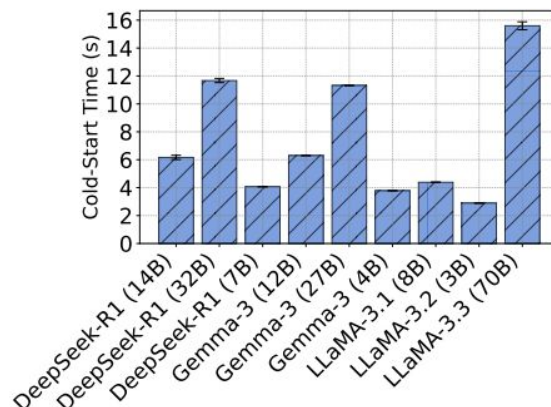
*These optimizations improve runtime performance & throughput
but **cold-start time remains a challenge.***

Cold-Start Latency

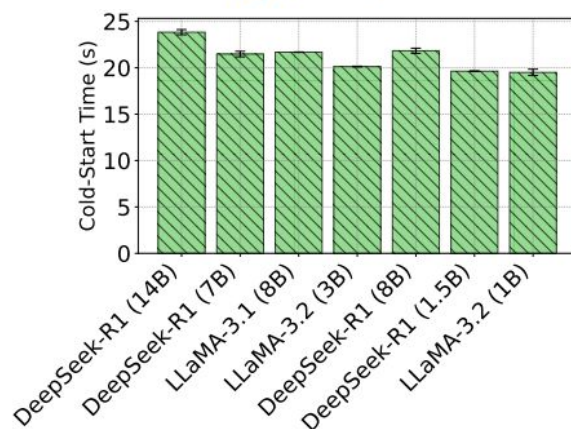
NVIDIA H100 (80 GB HBM3)



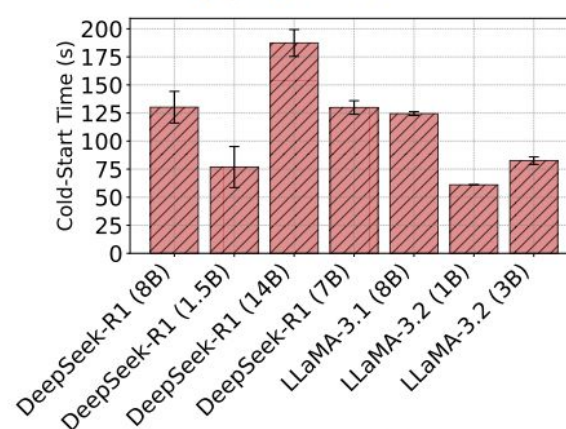
(a) vLLM



(b) Ollama



(c) SGLang



(d) TensorRT-LLM

Container Cold-Start for LLaMA 3.1-8B:

- Ollama: 4 s
- SGLang: 21 s
- vLLM: 87 s
- TensorRT-LLM: 124 s

Start-up overhead is intolerable on the critical path of request-serving.

vLLM Initialization Overheads



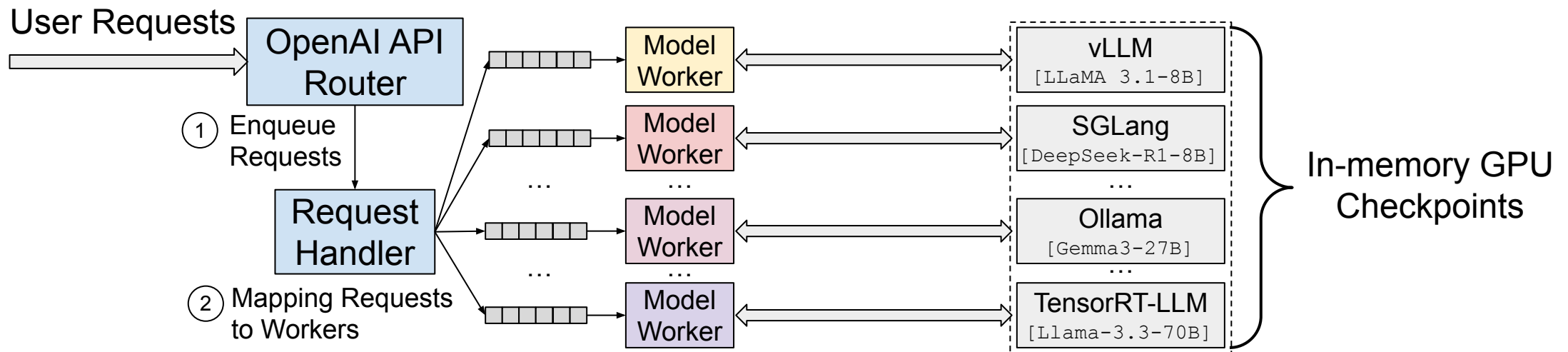
	Gemma 3 (27B)	DeepSeek R1 (14B)	LLaMA 3.1 (8B)
Model Loading	9 sec	5 sec	3 sec
PyTorch Compile	1 min 19 sec	43 sec	29 sec
CUDA Graph Capture	32 sec	21 sec	17 sec
Total Initialization Time	2 mins 40 sec	1 min 22 sec	55 sec

NVIDIA H100 80GB (HBM3)

Engine-Agnostic Model Hot-Swapping

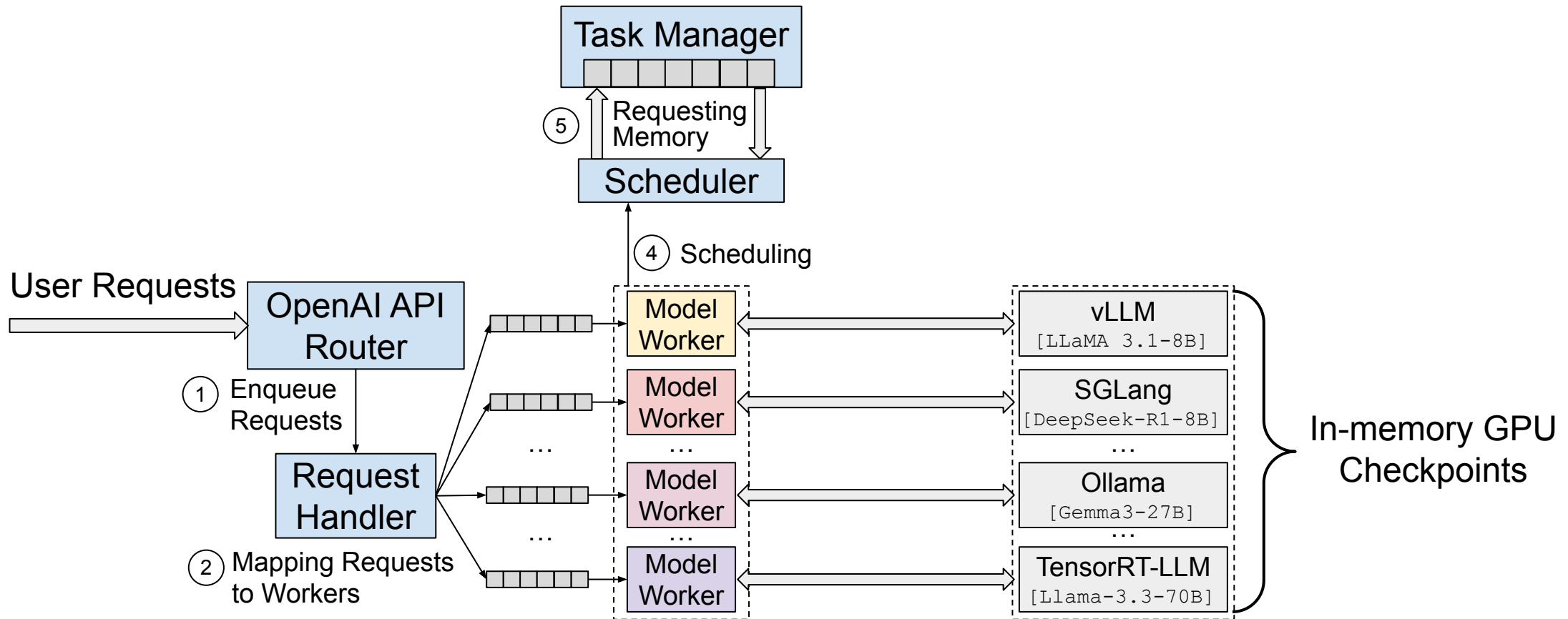
On-demand GPU hot-swapping of inference engines

SwapServeLLM

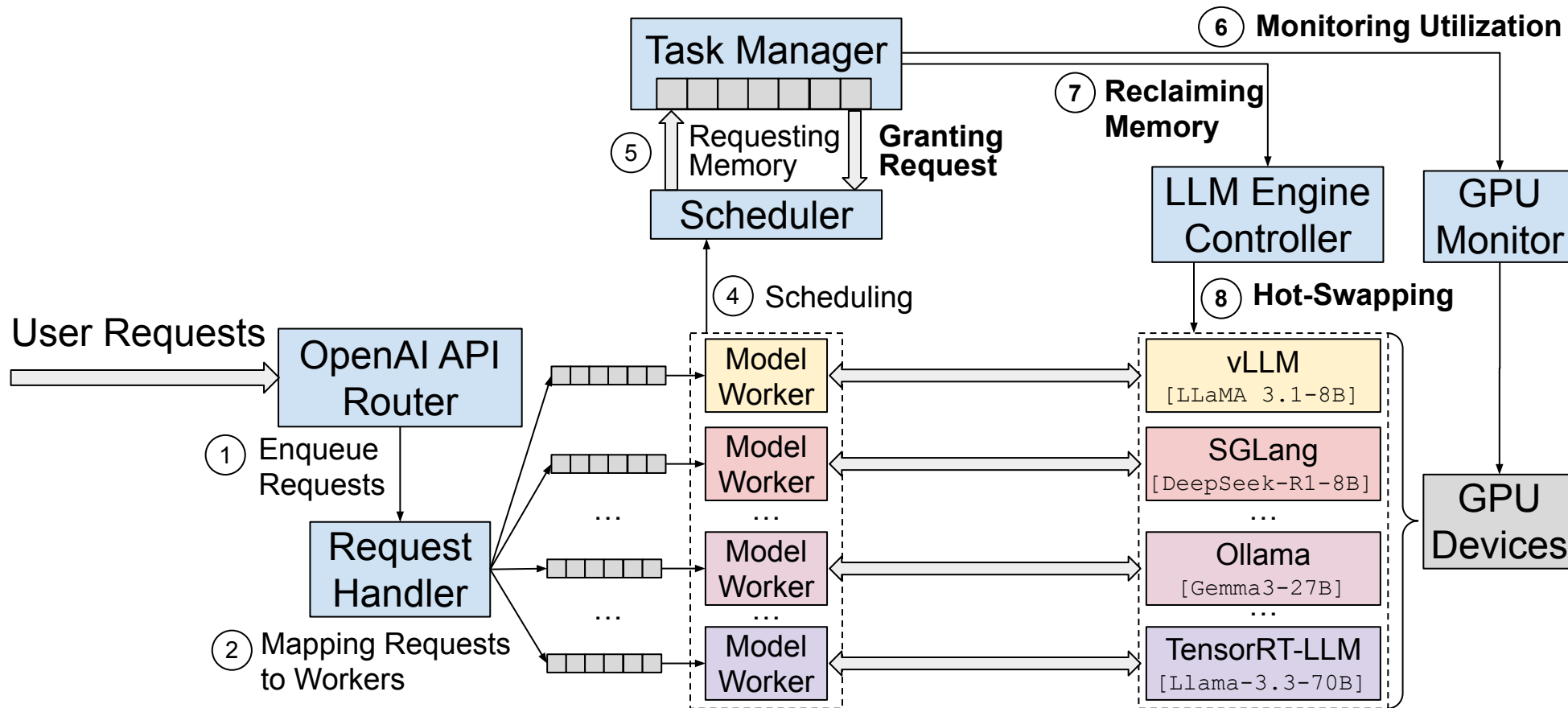


<https://github.com/nvidia/cuda-checkpoint>

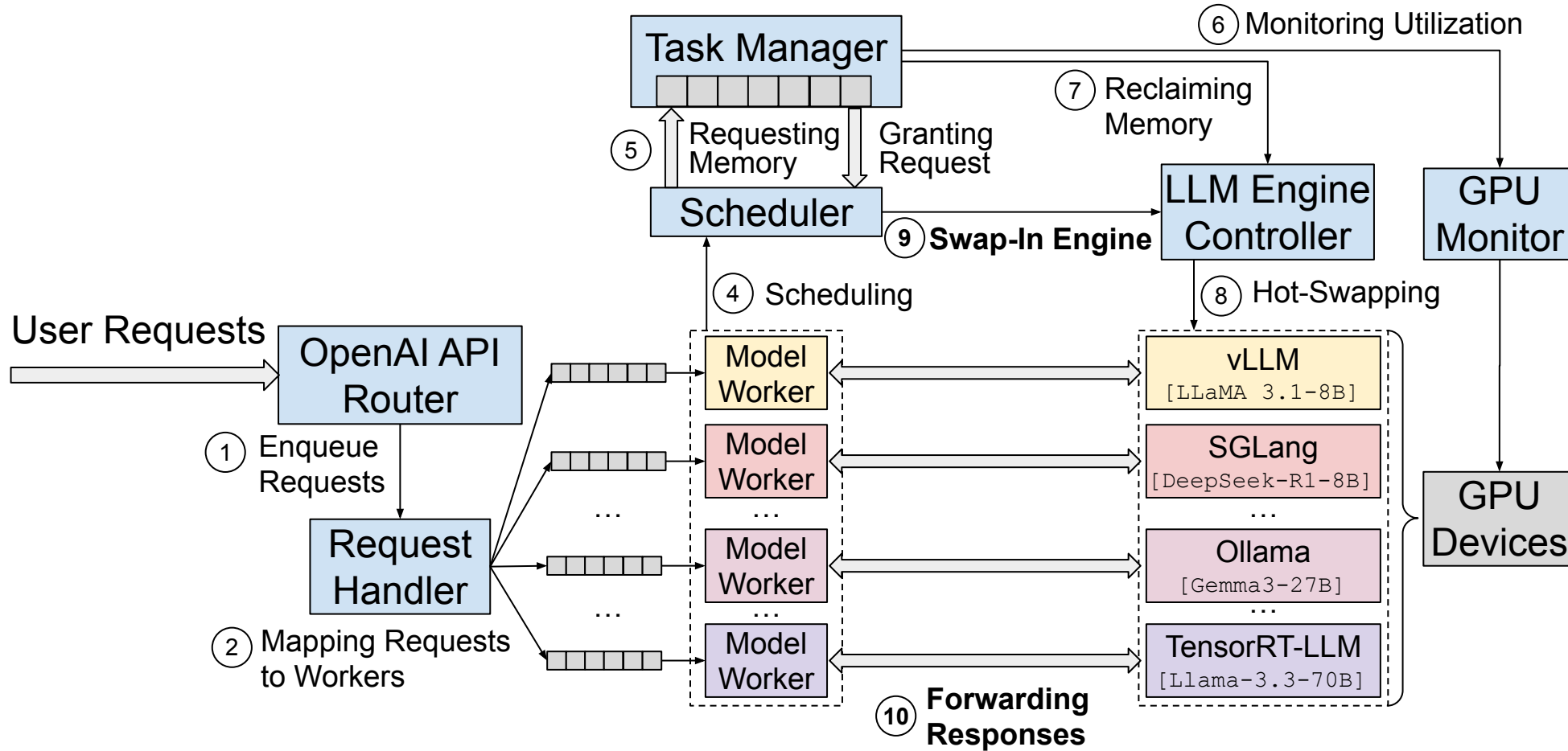
SwapServeLLM



SwapServeLLM



SwapServeLLM



<https://github.com/rst0git/SwapServeLLM>

SwapServeLLM Demo

```
root@h100x1-80gb:~/SwapServeLLM# ./evaluation/openai_client

Available Models:
[0] RedHatAI/Llama-3.1-8B-Instruct
[1] RedHatAI/gemma-3-27b-it-FP8-dynamic
[2] RedHatAI/DeepSeek-R1-Distill-Qwen-32B-FP8-dynamic
[3] llama3.2:3b-instruct-fp16
[4] gpt-oss:20b
[5] mistral:7b-text-fp16

💬 Chatting with model: RedHatAI/Llama-3.1-8B-Instruct

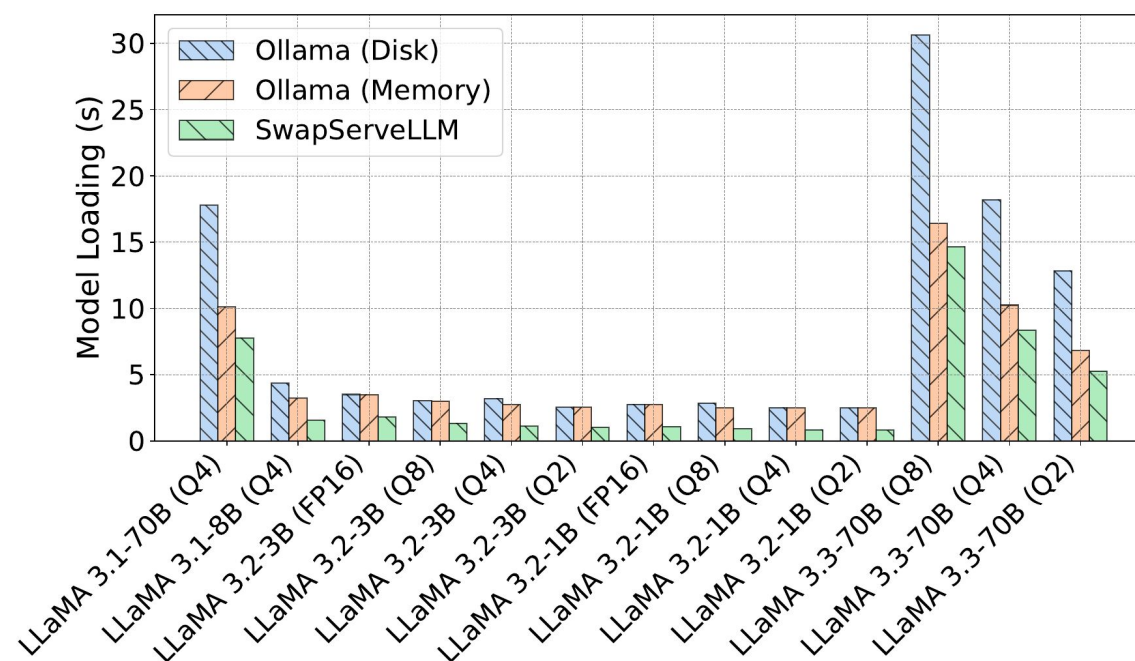
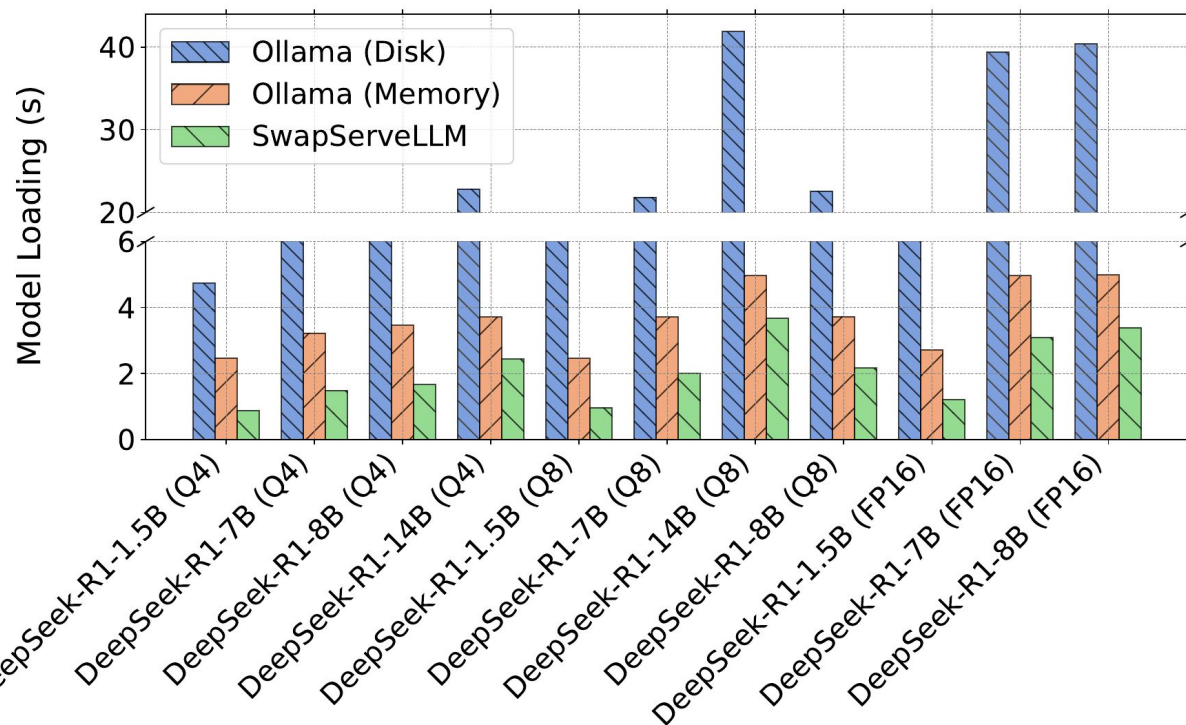
* Supported Commands:
/model or /m <model_id or index> - Switch to a different model
/models - List all available models
/benchmark or /b [prompt] - Run benchmark across all models (sequentially)
/multi-user-benchmark or /mub [prompt] - Run benchmark in parallel across all models
/itl [prompt] - Measure Inter-Token Latency for all models
/itl-model or /itl-m <model> [prompt] - Measure ITL for a specific model
/swapout or /so [model_id or index] - Swap out a specific model (or all if omitted)
/swapin or /si [model_id or index] - Swap in a specific model (or all if omitted)
/tokens or /t <number> - Set or show max tokens per response
/help or /h - Show this help message
/q or /quit or /exit - Exit the chat

You [RedHatAI/Llama-3.1-8B-Instruct]: █
```



Evaluation Results

Ollama Cold-start Latencies

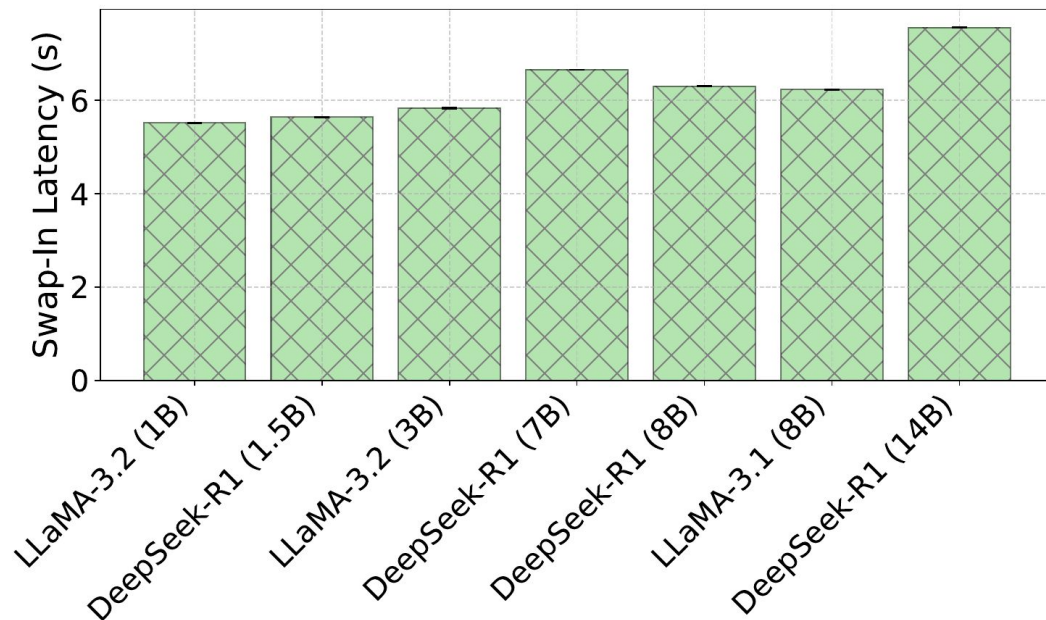


NVIDIA A100 (SXM4 80 GB)

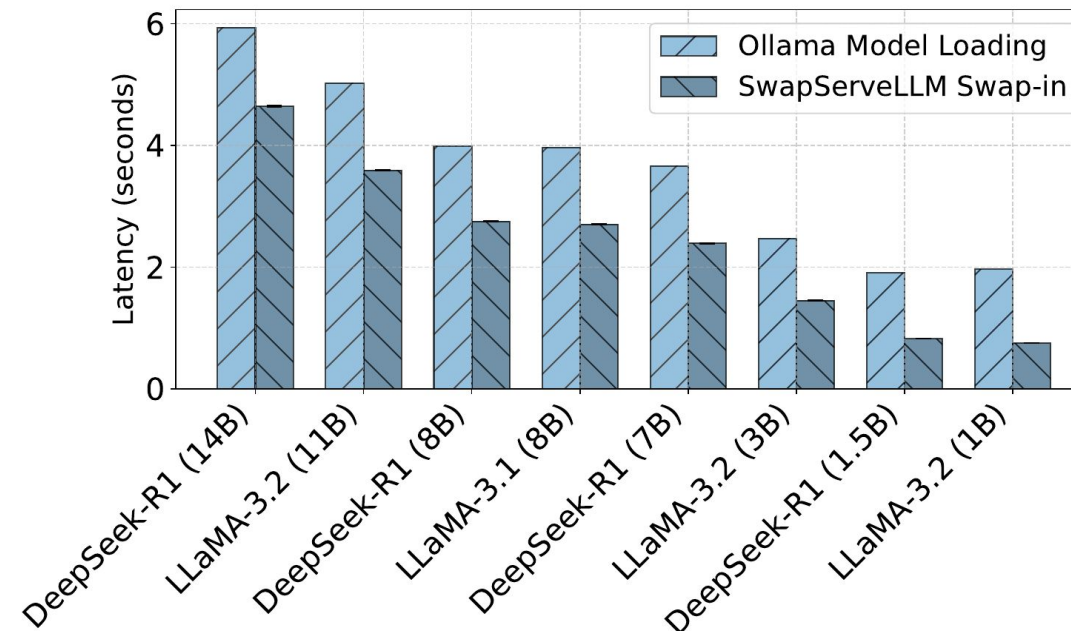
Evaluation Results

On-demand swap-in latency with SwapServeLLM

Swap-in latency with vLLM backend



Ollama (baseline) vs SwapServeLLM



NVIDIA H100 (80 GB HBM3)

Summary & Questions

- **Problem:** Running LLM inference at scale is inefficient due to long cold-start times and bursty request patterns.
- **Proposed Solution:** Engine-Agnostic Model Hot-Swapping
- **Results:** Up to 31× faster model loading with vLLM and 29% with Ollama



<https://github.com/rst0git/SwapServeLLM>

<https://github.com/nvidia/cuda-checkpoint>

<https://doi.org/10.1145/3731599.3767354>