Transparent Hot-Swapping of Containerized AI/ML Workloads

Radostin Stoyanov - PhD Student, Scientific Computing Group

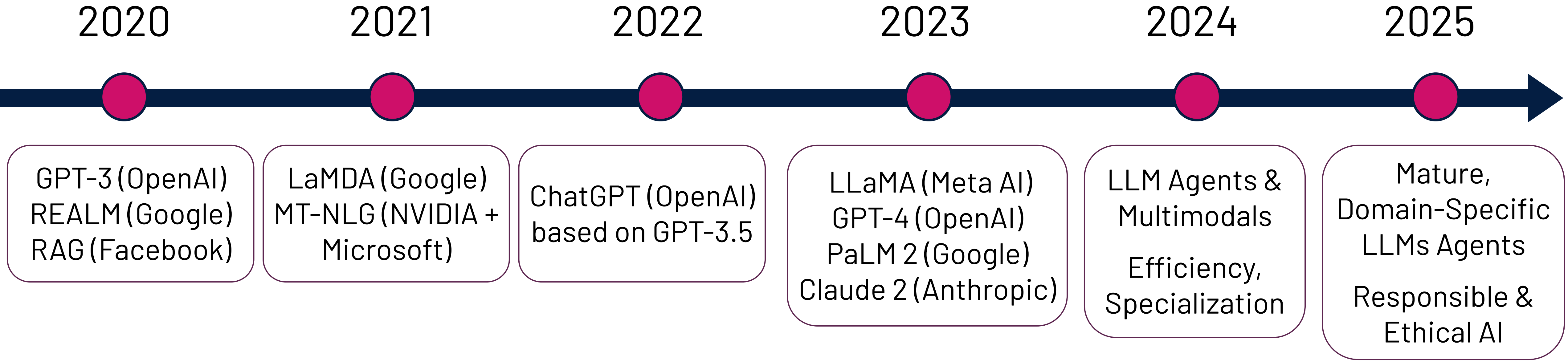
Collaboration with Viktória Spišaková, Marcin Copik, and Adrian Reber

Supervisors: Prof. Rodrigo Bruno, Prof. Wes Armour

AI/ML Workloads in Production

A vast number of applications are leveraging LLMs

- Demand for **inference** far exceeds that of training
- Large number of inferences necessary to **amortize high training costs**



2020

GPT-3 (OpenAI)
REALM (Google)
RAG (Facebook)

2021

LaMDA (Google)
MT-NLG (NVIDIA + Microsoft)

2022

ChatGPT (OpenAI) based on GPT-3.5

2023

LLaMA (Meta AI)
GPT-4 (OpenAI)
PaLM 2 (Google)
Claude 2 (Anthropic)

2024

LLM Agents & Multimodals

Efficiency, Specialization

2025

Mature, Domain-Specific LLMs Agents

Responsible & Ethical AI

LLM Inference Serving

Workloads Characteristics

- Receive large number of queries with strict SLOs
- Unpredictable output lengths
- Run on expensive GPUs with high energy consumption[1]
- GPU resources in multi-tenant clusters (MLaaS) are often underutilized

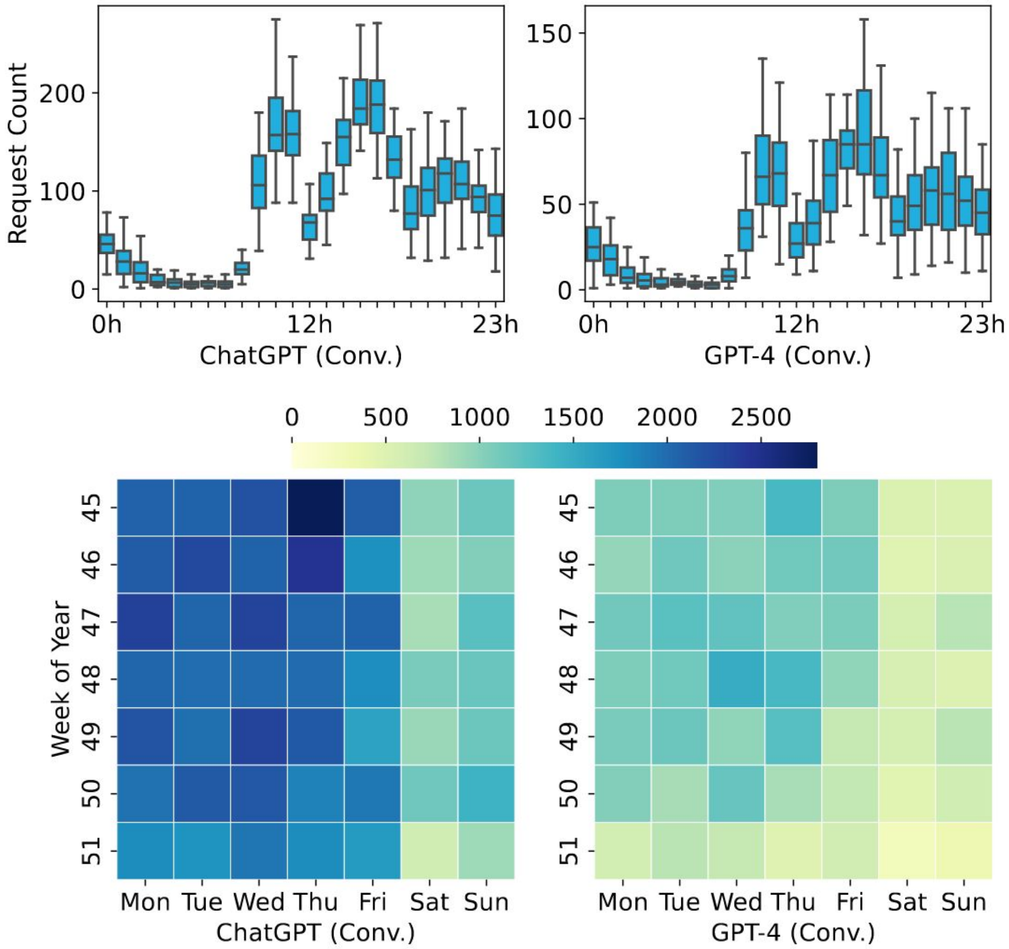
Key Performance Metrics: Latency & Throughput

Optimization Strategies

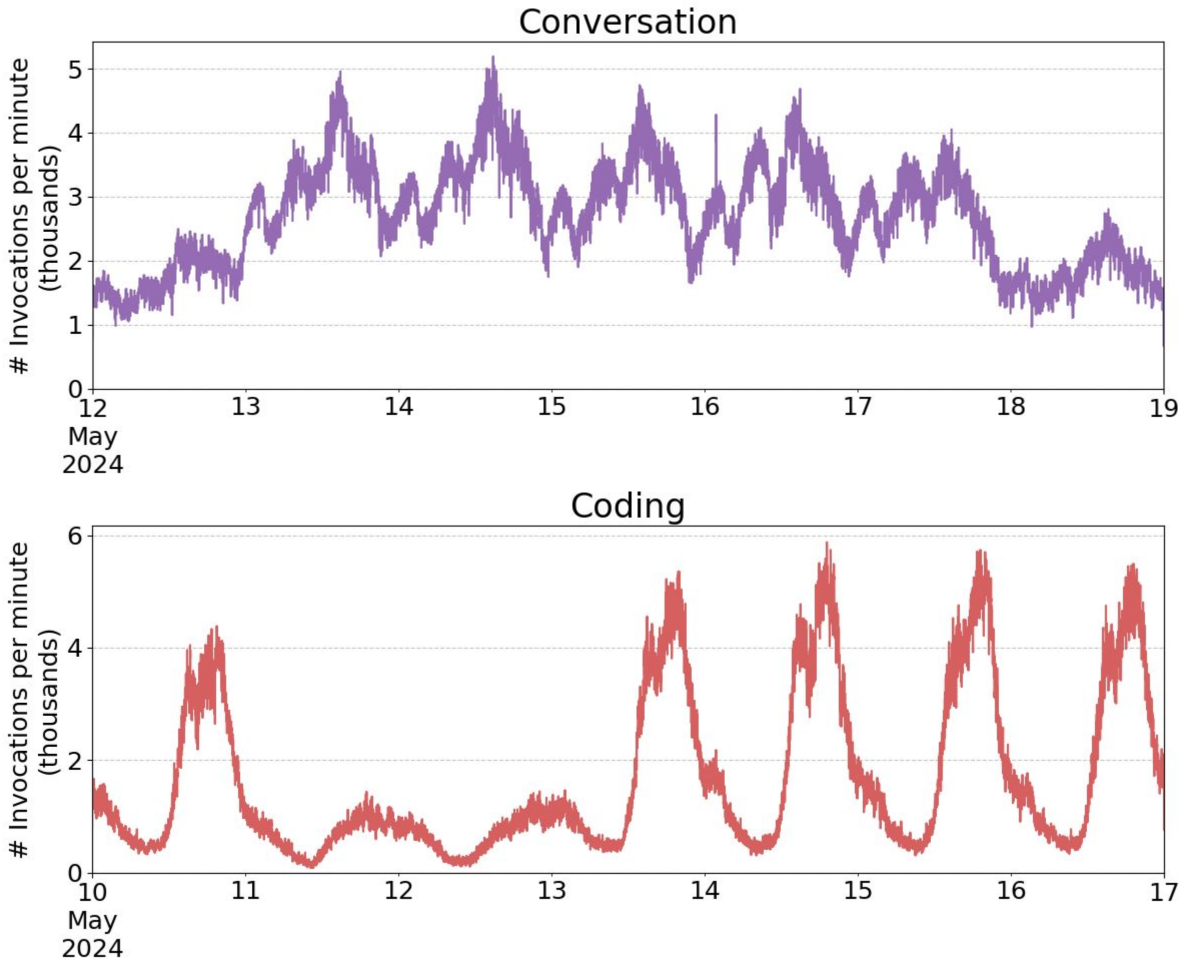
- Efficient KV cache management with distributed inference
- Iteration-level scheduling with continuous batching
- Smaller (specialized) models are faster & less expensive to run

[1] *Double-Exponential Increases in Inference Energy: The Cost of the Race for Accuracy*. Zeyu Yang, Karel Adamek, Wesley Armour. https://arxiv.org/abs/2412.09731

Real-World LLM Deployments



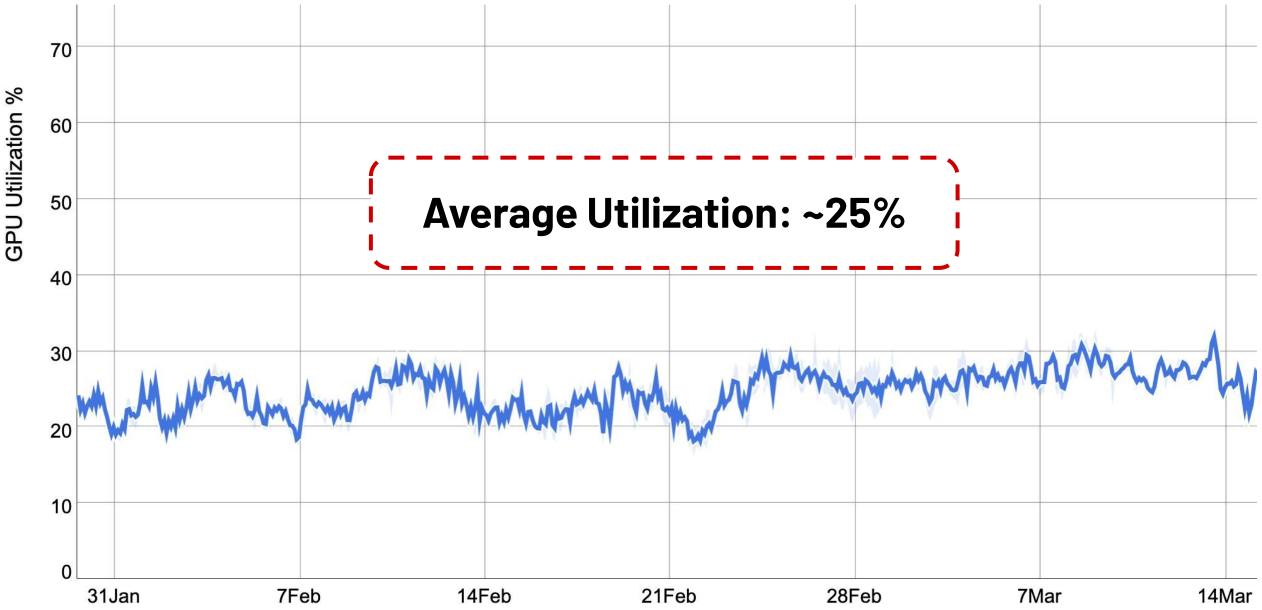
BurstGPT: A Real-world Workload Dataset to Optimize LLM Serving Systems
https://github.com/HPMLL/BurstGPT



Azure LLM Inference Dataset trace
https://github.com/Azure/AzurePublicDataset

Problem: Low GPU Utilization

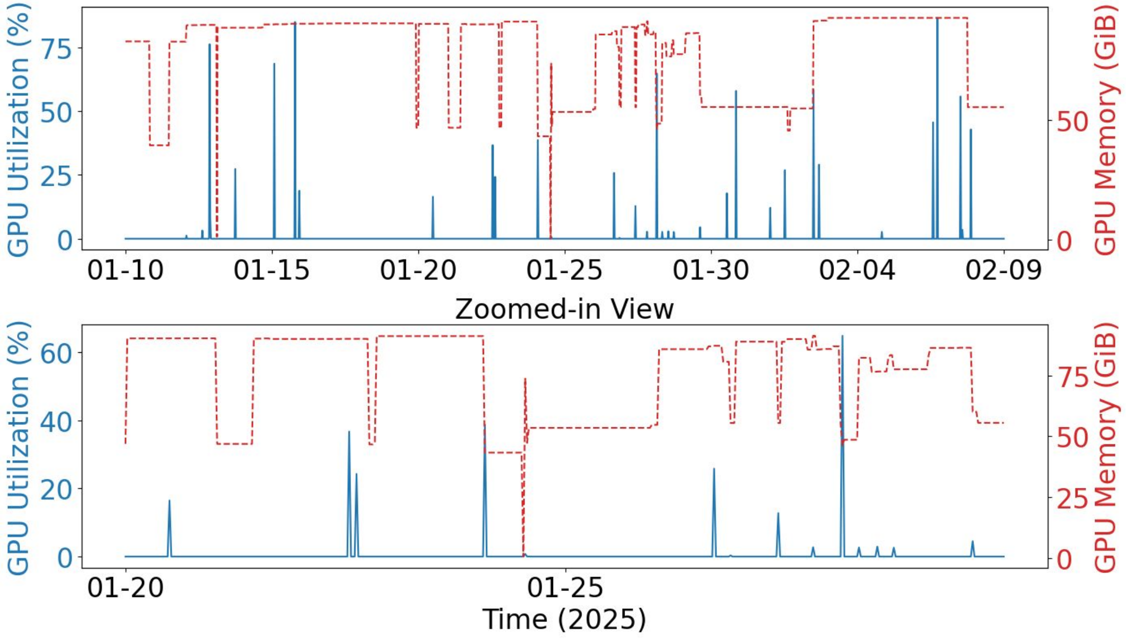
Inference Serving / Notebooks / Visualization Workloads



Average Utilization: ~25%

"Improving GPU Utilization using Kubernetes Engine"
Maulin Patel, Pradeep Venkatachalam (Google)

e-INFRA CZ Kubernetes Cluster [1,2,3]
GPU Utilization of LLM Inference Workloads



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

[1] *Efficient Transparent Checkpointing of AI/ML Workloads in Kubernetes.* Viktória Spišaková, Radostin Stoyanov, Adrian Reber. KubeCon 2025
[2] *Optimizing Resource Utilization for Interactive GPU Workloads with Container Checkpointing.* Viktória Spišaková, Radostin Stoyanov. FOSDEM 2025
[3] *Kubernetes Scheduling with Checkpoint/Restore: Challenges & Open Problems.* Viktória Spišaková et. al., JSSPP 2025

Transparent Checkpointing

Transparent Checkpointing

Checkpoint/Restore in Userspace

- Transparent checkpointing of CPU-GPU state
- Supports both AMD and NVIDIA GPUs
- Integrated with Docker, Podman, Kubernetes

github.com/checkpoint-restore/criu
github.com/nvidia/cuda-checkpoint

CRIUgpu: Transparent Checkpointing of GPU-Accelerated Workloads. Radostin Stoyanov, Viktória Spišaková, Jesús Ramos, Steven Gurfinkel, Andrei Vagin, Adrian Reber, Wesley Armour, Rodrigo Bruno. (2025). https://arxiv.org/abs/2502.16631

Challenges

cuda-checkpoint

- Only for GPU state; CPU processes continue running
- Checkpoints a single process, not a process tree (container)
- Without CRIU, it does not support migration

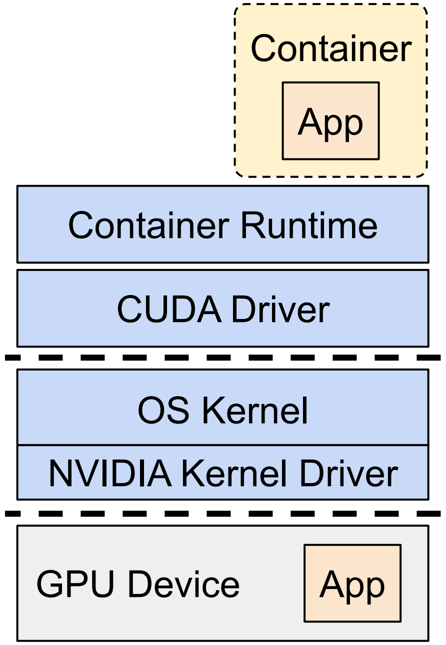
CRIU + GPU Plugins

- Saving checkpoint data to disk can be very slow
- Container engines are not optimized for large container checkpoints[1]

[1] https://github.com/checkpoint-restore/criu/issues/2519

Transparent Hot-Swapping

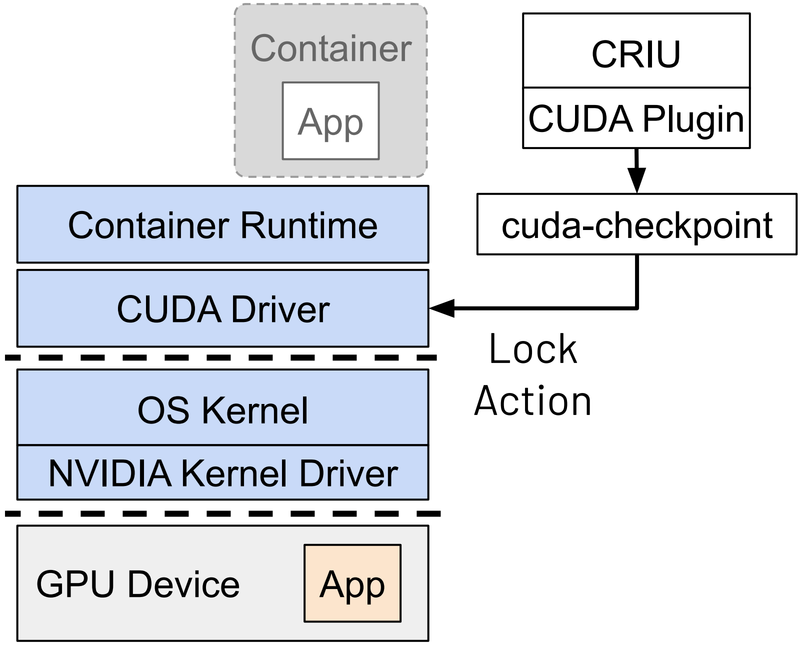
Transparent Hot-Swapping



Starting Container A

- Mounting GPU libraries & device files in container
- Loading model, allocating GPU memory, KV Cache
- Ready to accept user requests

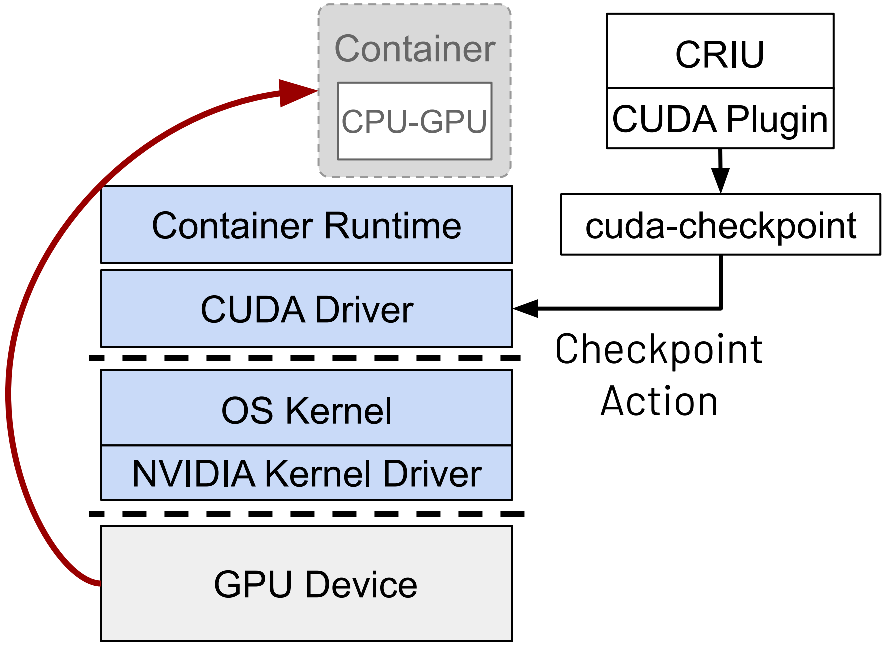
Transparent Hot-Swapping



Transparent Preemption of GPU Workloads

- Stop execution of CPU processes: `ptrace(SEIZE+INTERRUPT)`
- Lock CUDA driver APIs

Transparent Hot-Swapping

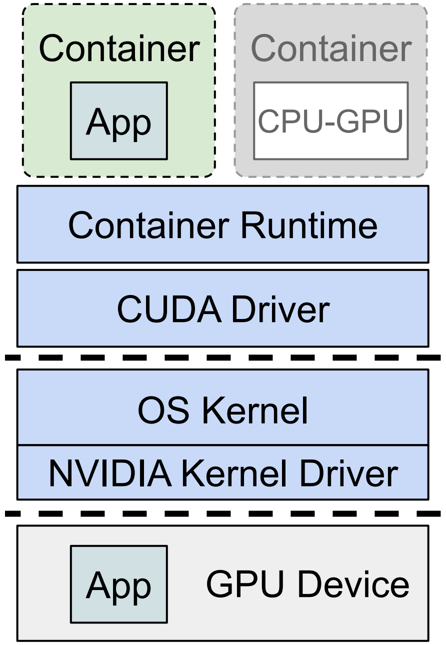


Transparent Preemption of GPU Workloads

- Stop execution of CPU processes: `ptrace(SEIZE+INTERRUPT)`
- Lock CUDA driver APIs
- Checkpoint GPU state to host memory
- Leave container container A in a “stopped” state

⇨ GPU resources have been released

Transparent Hot-Swapping

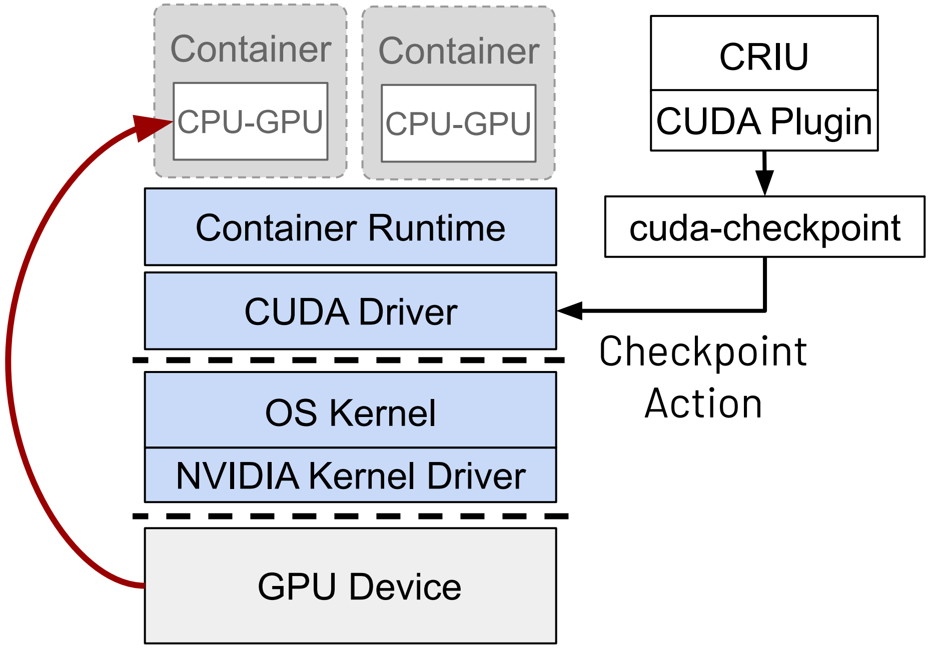


Starting Container B

- Mounting GPU libraries & device files in container
- Loading model, allocating GPU memory, KV Cache

⇨ Container B is ready to accept user requests

Transparent Hot-Swapping

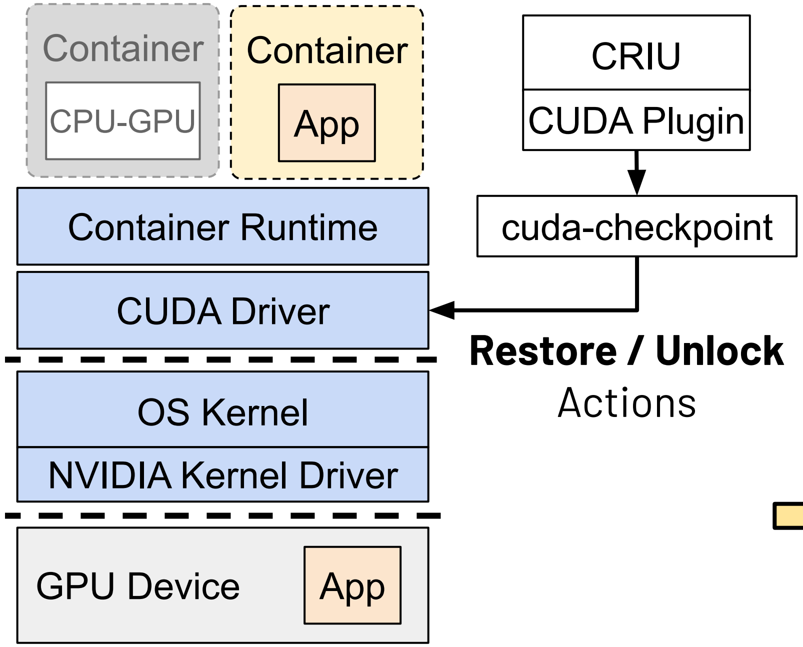


Transparent Preemption of GPU Workloads

- Stop execution of CPU processes: `ptrace(SEIZE+INTERRUPT)`
- Lock CUDA driver APIs
- Checkpoint GPU state to host memory
- Leave container container B in a “stopped” state

⇨ GPU resources have been released

Transparent Hot-Swapping

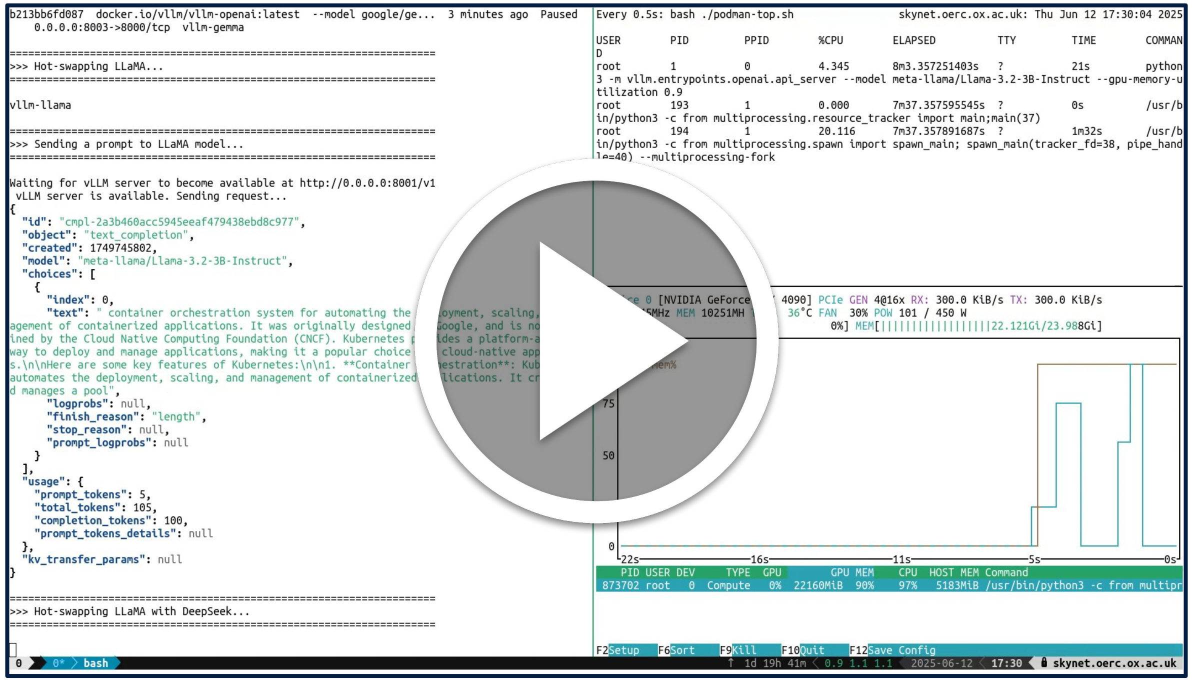


Resuming Preempted GPU Workloads

- Resume execution of CPU processes
- Restore GPU state from host memory
- Unlock CUDA driver APIs

⇨ Container A is ready to accept user requests

Transparent Hot-Swapping Demo



```
b213bb6fd087  docker.io/vllm/vllm-openai:latest  --model google/ge...  3 minutes ago  Paused
    0.0.0.0:8003->8000/tcp  vllm-gemma

======================================================================
>>> Hot-swapping LLaMA...
======================================================================

vllm-llama

======================================================================
>>> Sending a prompt to LLaMA model...
======================================================================

Waiting for vLLM server to become available at http://0.0.0.0:8001/v1
 vLLM server is available. Sending request...
{
  "id": "cmpl-2a3b460acc5945eeaf479438ebd8c977",
  "object": "text_completion",
  "created": 1749745802,
  "model": "meta-llama/Llama-3.2-3B-Instruct",
  "choices": [
    {
      "index": 0,
      "text": " container orchestration system for automating the     oyment, scaling,
agement of containerized applications. It was originally designed    Google, and is no
ined by the Cloud Native Computing Foundation (CNCF). Kubernetes p    ides a platform-a
way to deploy and manage applications, making it a popular choice     cloud-native app
s.\n\nHere are some key features of Kubernetes:\n\n1. **Container     estration**: Kub
automates the deployment, scaling, and management of containerized    lications. It cr
d manages a pool",
      "logprobs": null,
      "finish_reason": "length",
      "stop_reason": null,
      "prompt_logprobs": null
    }
  ],
  "usage": {
    "prompt_tokens": 5,
    "total_tokens": 105,
    "completion_tokens": 100,
    "prompt_tokens_details": null
  },
  "kv_transfer_params": null
}

======================================================================
>>> Hot-swapping LLaMA with DeepSeek...
======================================================================

0  0*  bash
```

```
Every 0.5s: bash ./podman-top.sh                    skynet.oerc.ox.ac.uk: Thu Jun 12 17:30:04 2025

USER        PID         PPID        %CPU        ELAPSED           TTY         TIME        COMMAN
D
root        1           0           4.345       8m3.357251403s    ?           21s         python
3 -m vllm.entrypoints.openai.api_server --model meta-llama/Llama-3.2-3B-Instruct --gpu-memory-u
tilization 0.9
root        193         1           0.000       7m37.357595545s   ?           0s          /usr/b
in/python3 -c from multiprocessing.resource_tracker import main;main(37)
root        194         1           20.116      7m37.357891687s   ?           1m32s       /usr/b
in/python3 -c from multiprocessing.spawn import spawn_main; spawn_main(tracker_fd=38, pipe_hand
le=40) --multiprocessing-fork
```

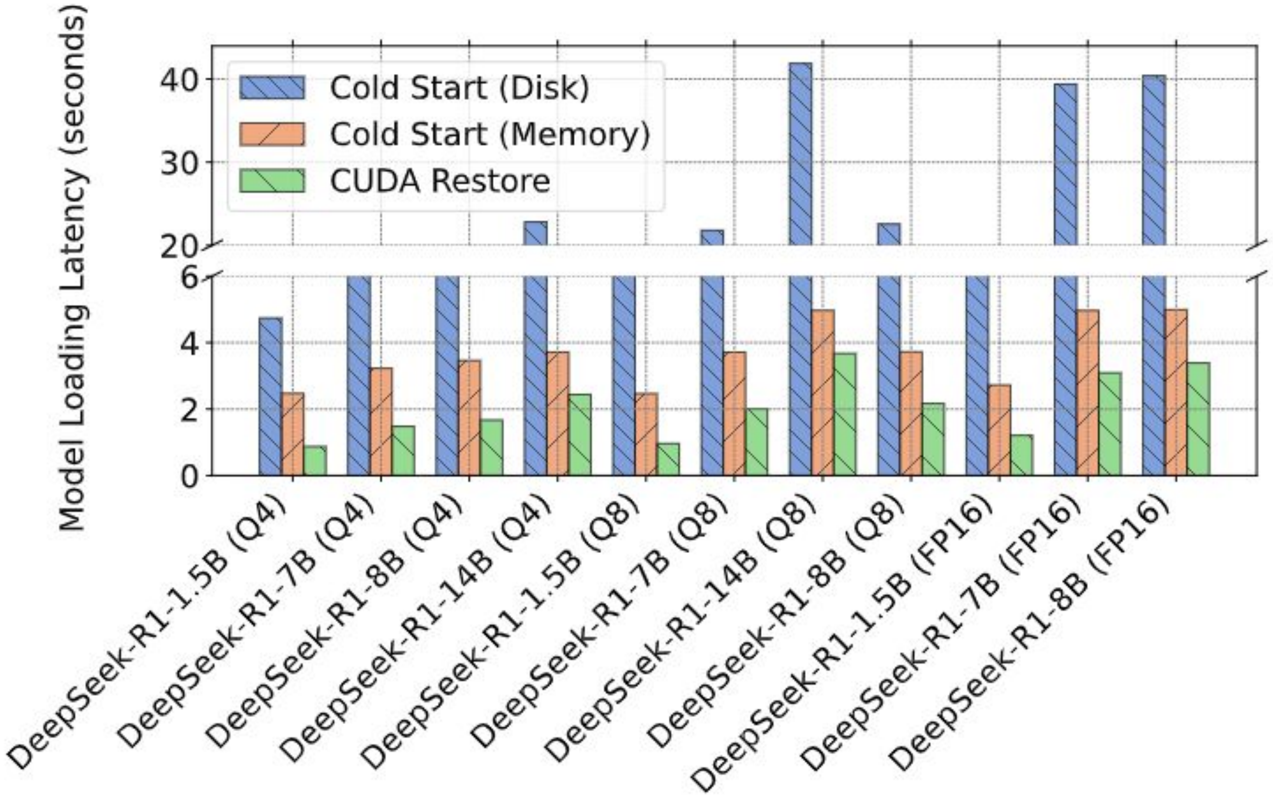
```
  ce 0 [NVIDIA GeForce    4090] PCIe GEN 4@16x RX: 300.0 KiB/s TX: 300.0 KiB/s
   5MHz MEM 10251MH       36°C FAN  30% POW 101 / 450 W
                        0%] MEM[||||||||||||||||||22.121Gi/23.988Gi]
```

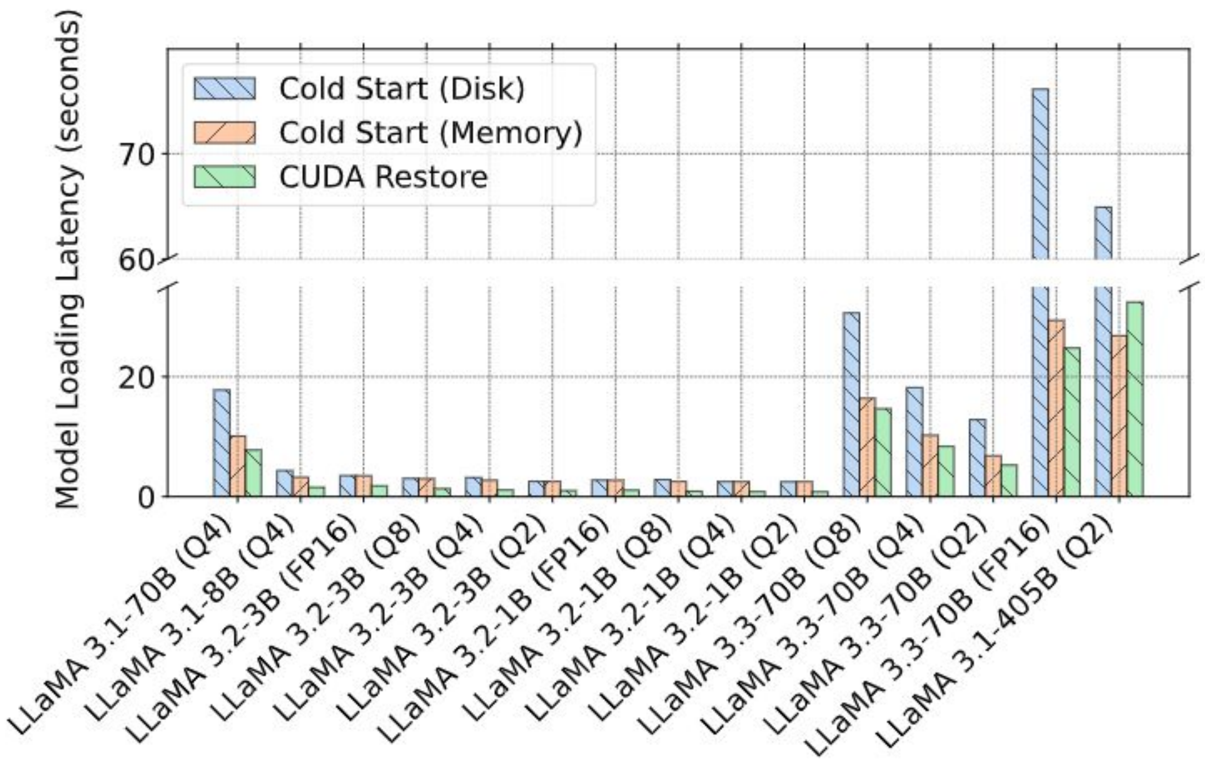
```
   PID USER DEV     TYPE  GPU       GPU MEM    CPU  HOST MEM Command
873702 root   0  Compute   0%  22160MiB  90%    97%   5183MiB /usr/bin/python3 -c from multipr

F2Setup  F6Sort  F9Kill  F10Quit  F12Save Config
↑ 1d 19h 41m  0.9 1.1 1.1  2025-06-12  17:30  skynet.oerc.ox.ac.uk
```

Part 2: Transparent Hot-Swapping of vLLM Containers

Part 1: vLLM Initialization + Checkpoint Creation

Evaluation





Ollama LLM Inference Workloads on NVIDIA A100 (SXM4 80GB)

Summary & Questions

- Transparent Hot-Swapping of GPU Workloads
- Optimized Resource Utilization for LLM Inference
- Out-of-the-box Integration with Container Platforms

github.com/checkpoint-restore/criu

github.com/nvidia/cuda-checkpoint