LLM-OS

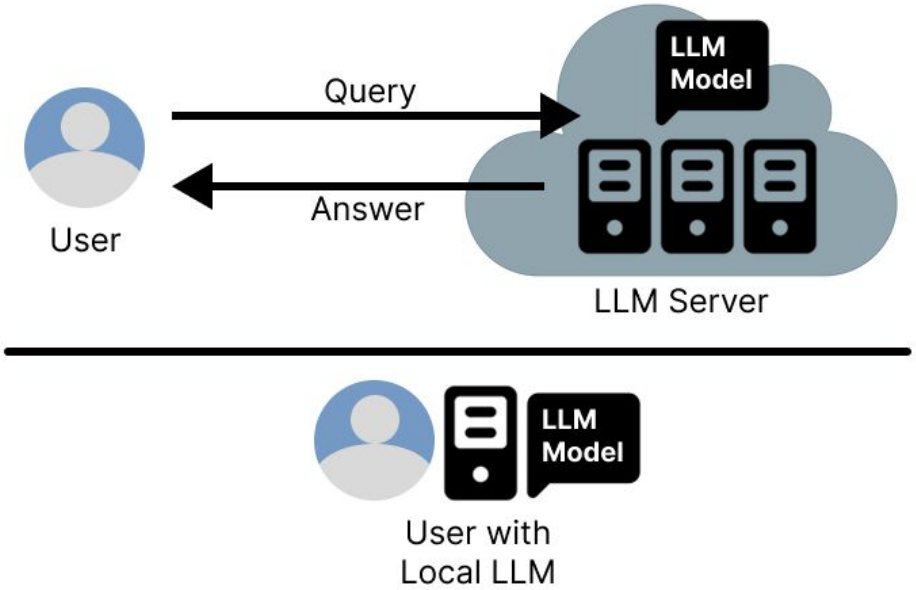
Orchestrating Edge Inference with Priority Scheduling and Adaptive KV Cache Management

Berkay Eren Ürün
Advisor: Teofil Bodea
Chair of Computer Systems
https://dse.in.tum.de/

15.11.2024 – 15.05.2025

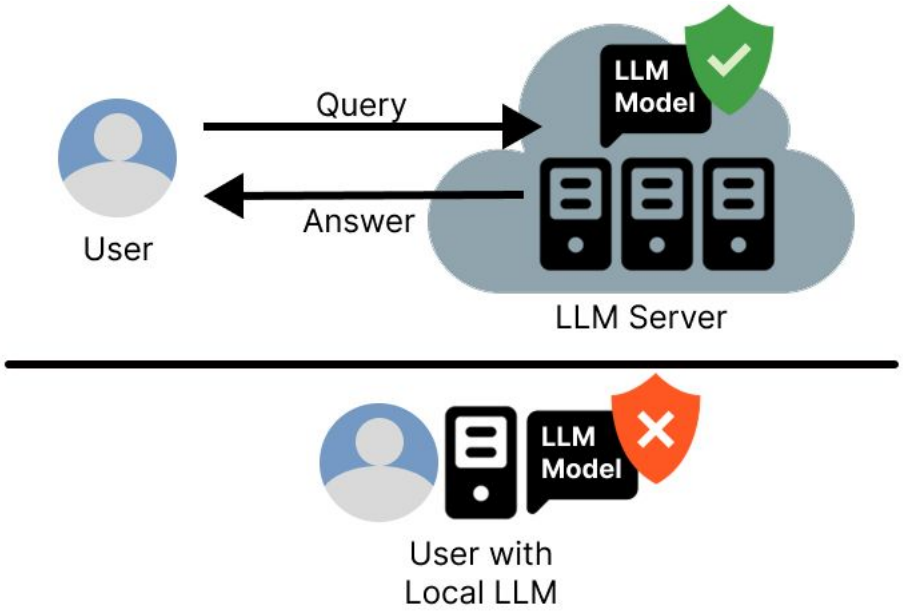
Inference on Edge Devices

- Inference on cloud servers is not always the best option
  - Latency
  - Data privacy
  - Availability
- Local inference can overcome these issues
  - All processing done locally



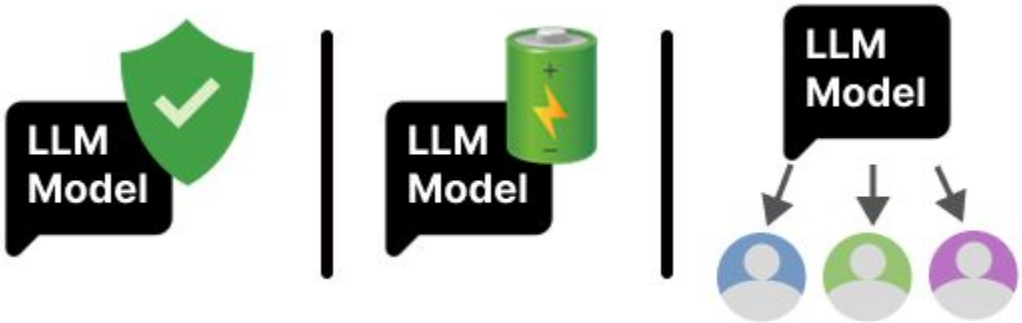
Edge Inference Limitations

- Edge inference introduces different problems
  - Model is stored on untrusted user device
  - Edge devices often have limited resources
  - User application might require specialized models



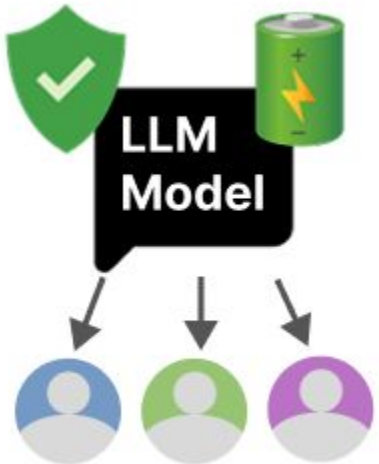
State-of-the-art

- Most work focus on a single aspect
  - Model Security: ASGARD
  - Resource Efficiency/Low Power Consumption: EdgeMoE
  - Adapting and Serving Models for Different Use Cases: S-LoRA



Problem statement

- Designing an LLM inference framework that:
  - Protects the LLM model
  - Focuses on edge devices with limited resources
  - Adapts the model to server multiple clients with different needs



Proposal: LLM-OS

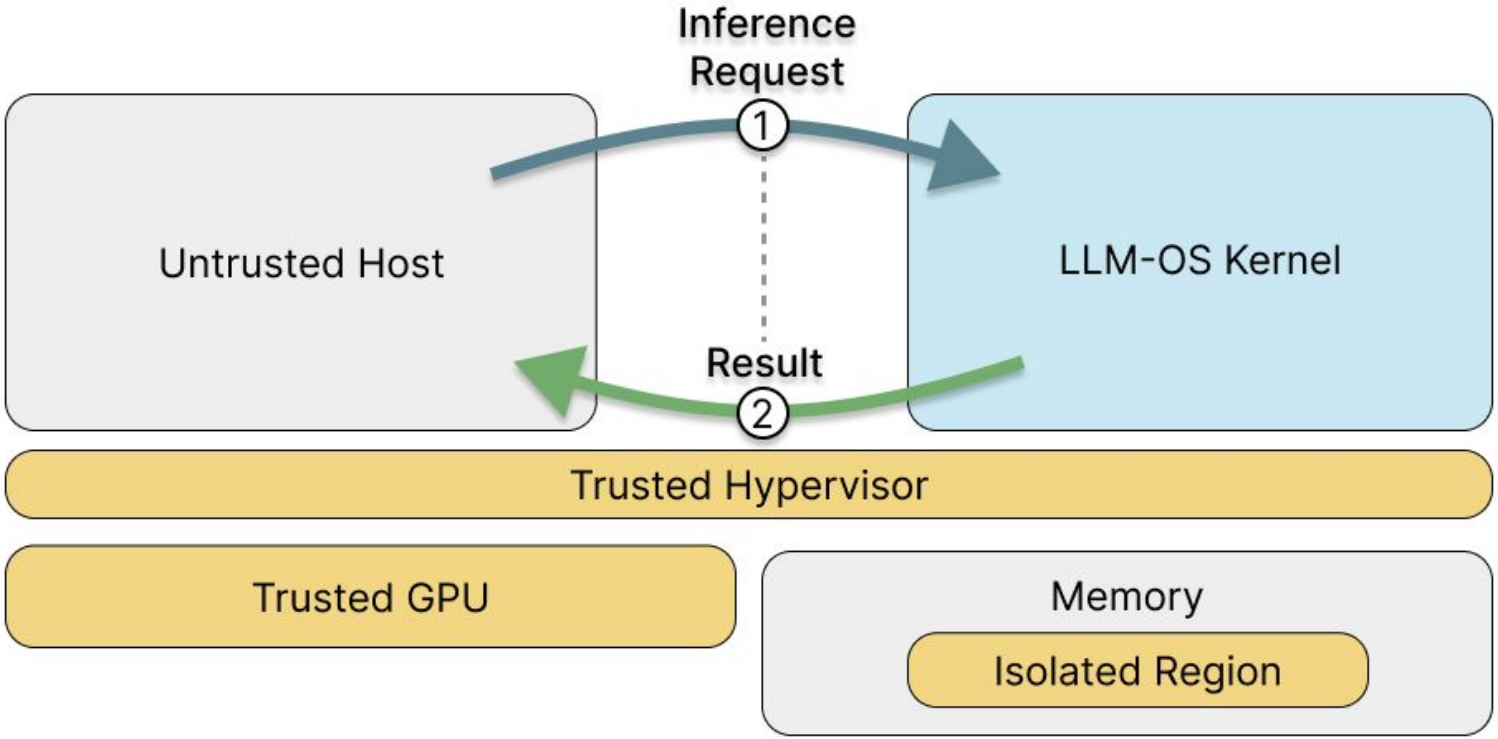
A trusted OS for LLM inference on limited-resource devices

- Three design goals:
  - ***Trustworthiness***: Model must be protected
  - ***Resource Efficiency***: Resource and energy constraints must be respected
  - ***Adaptability***: Model should be able to be specialized by the users

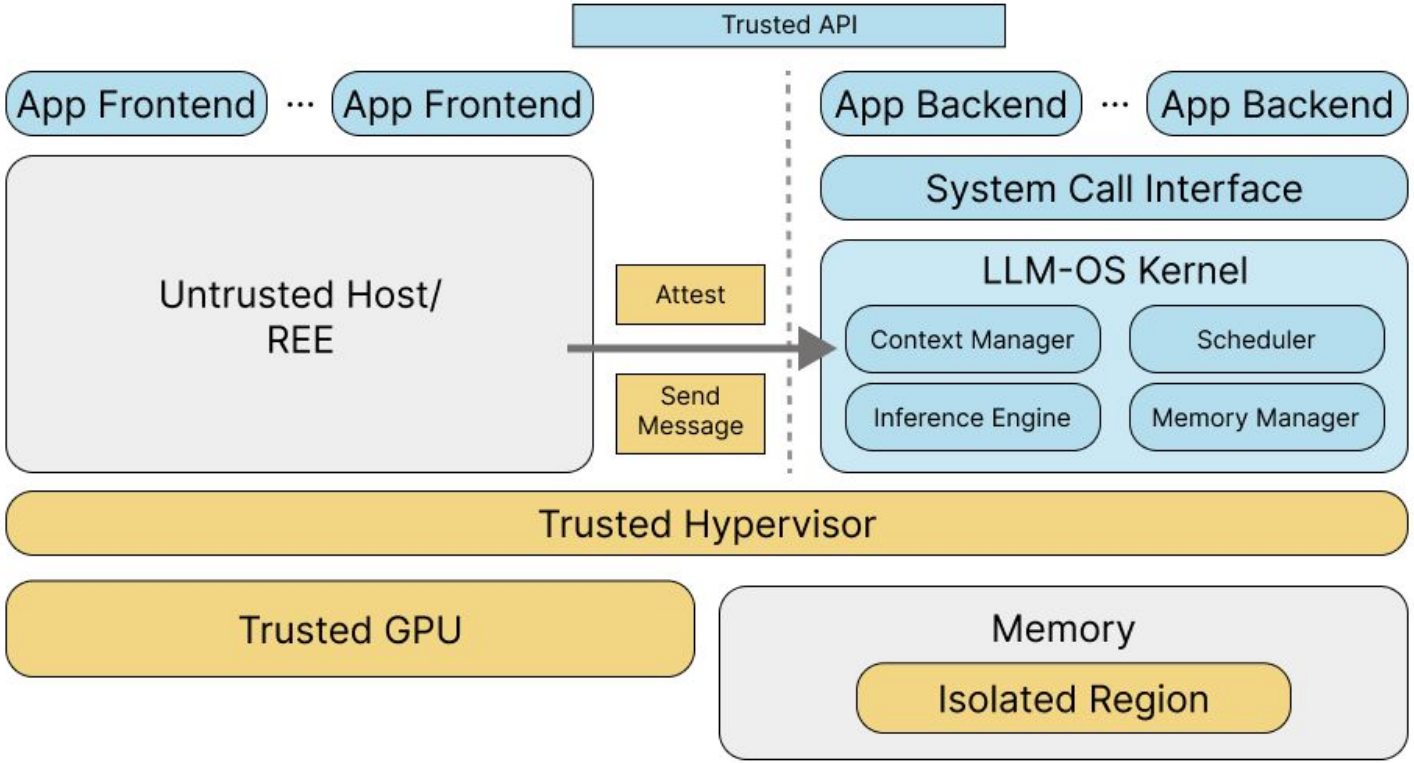
Outline

- ~~Motivation~~
- Overview
- Design
- Implementation
- Evaluation

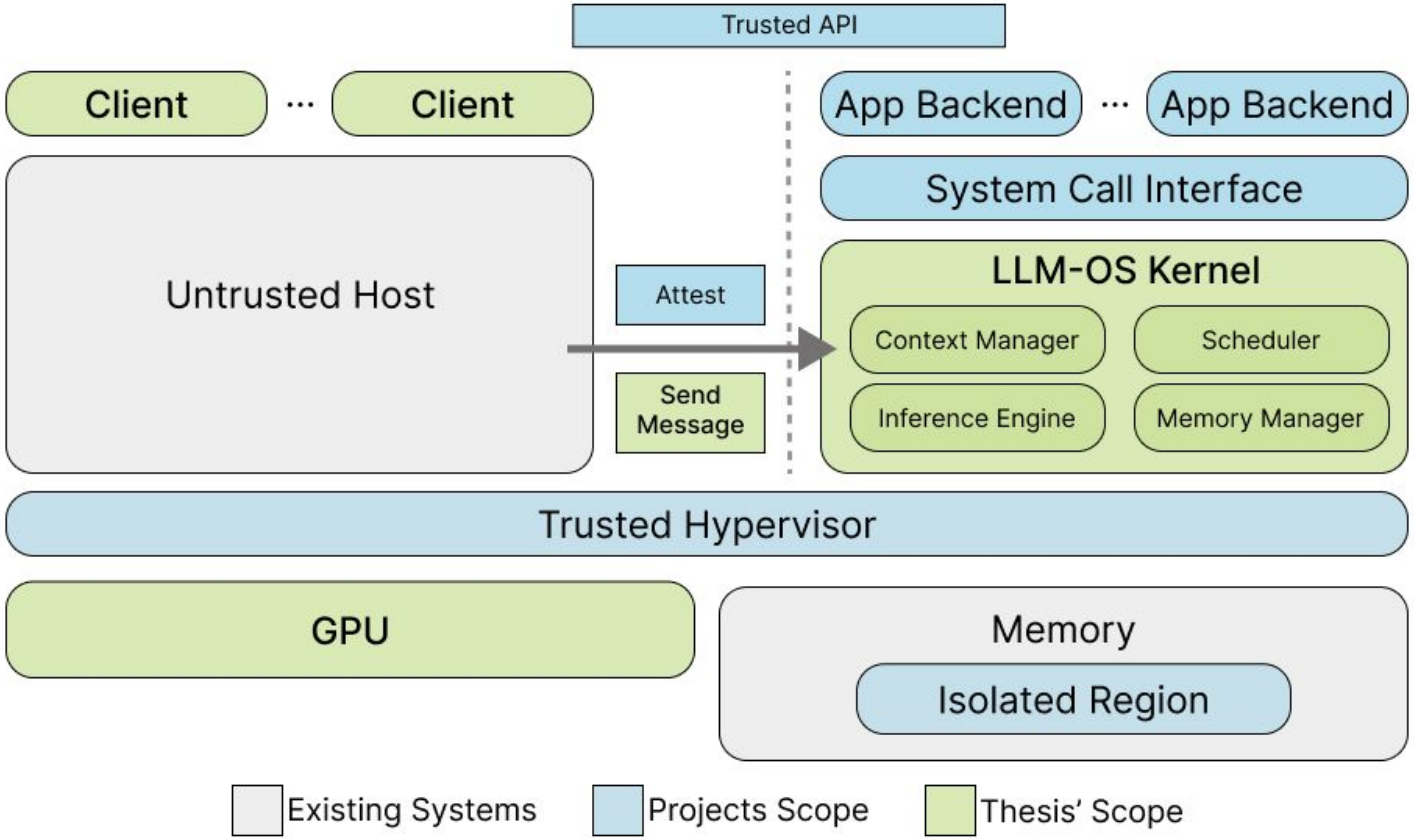
LLM-OS Overview



LLM-OS Overview



Thesis' Scope

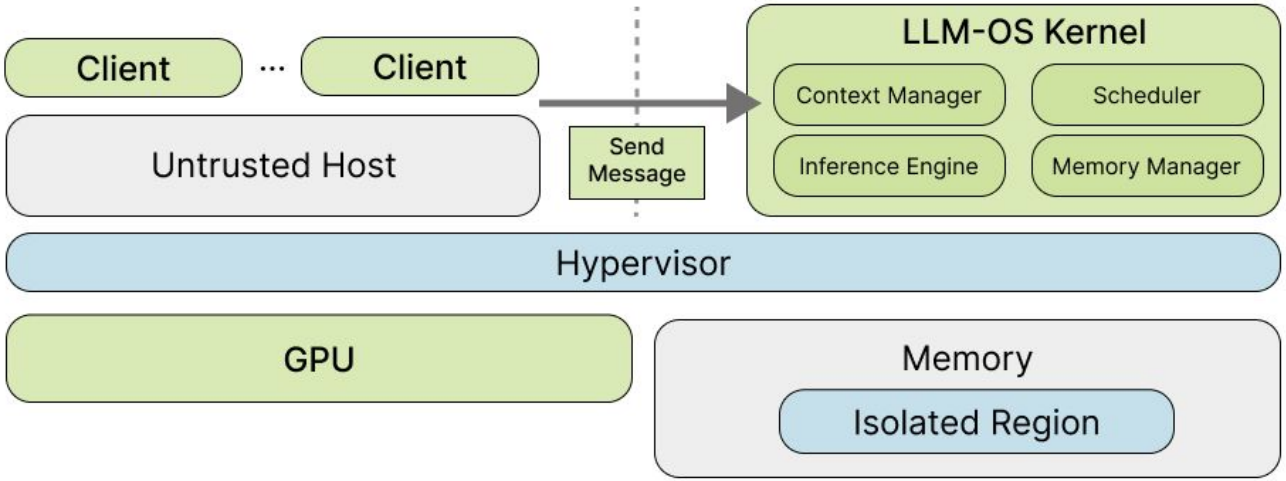


Outline

- ~~Motivation~~
- ~~Overview~~
- Design
- Implementation
- Evaluation

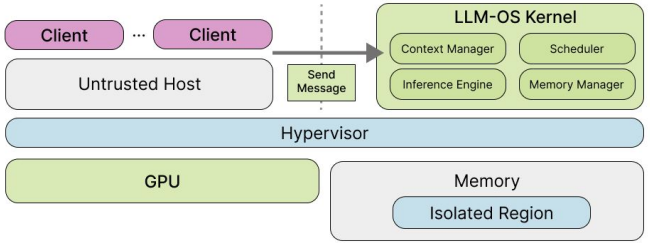
Design

- Untrusted host runs multiple clients requiring LLM service
- LLM-OS Kernel provides the LLM service
- Hypervisor runs both systems, handling the GPU and *Send Message* interface between two sides



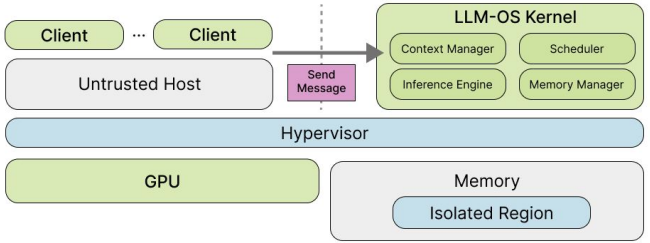
Client

- Simulates a real world client behaviour
- Runs in untrusted host
- Sends periodic LLM inference requests and expects answer
- Customizable
  - Number of tokens to generate
  - Dynamic Priority
  - Specialization
  - Dynamic Throughput Limit



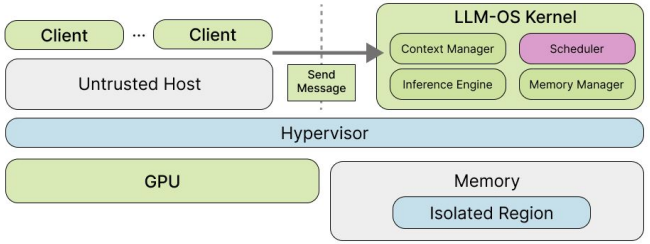
Send Message

- Connects the clients with the LLM-OS Kernel
- Supports multiple concurrent clients
- Clients write their request
- LLM-OS Kernel writes the answer back

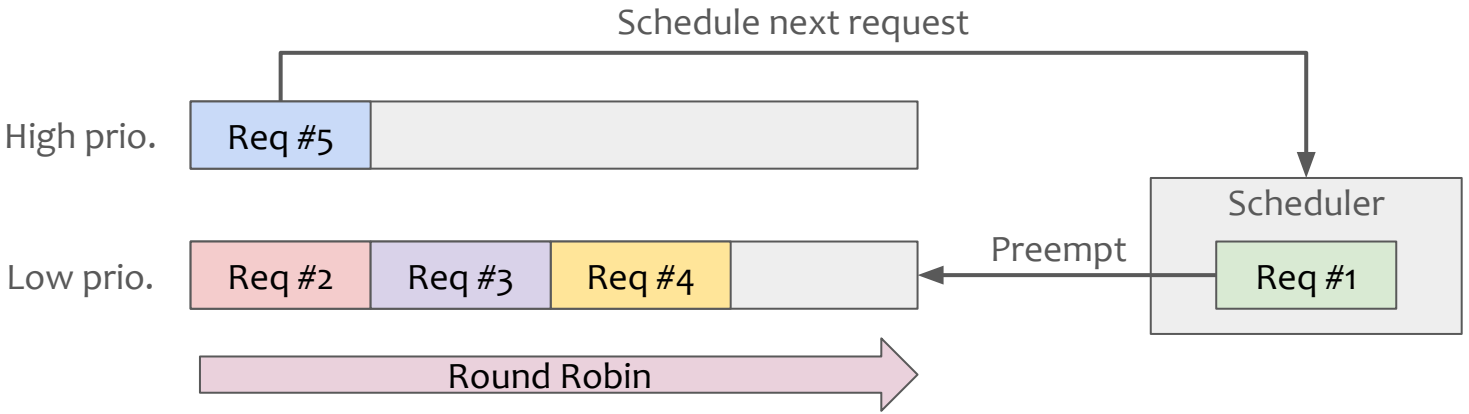


Scheduler

- Schedules user requests based on their priority
- Supports interrupts in case of high priority requests
- Scheduling frequency can be changed

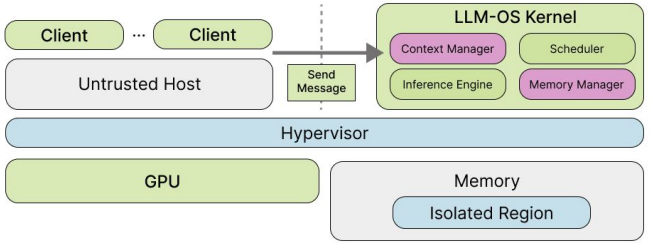


□ frequency == □ waiting time, □ throughput



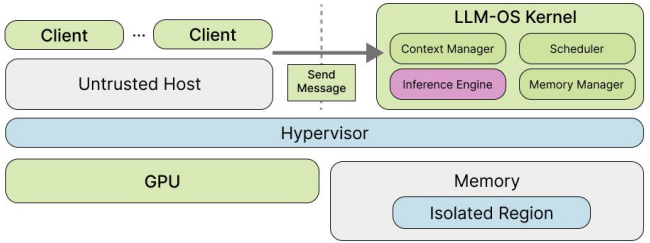
Context Manager & Memory Manager

- Creates context from user request
  - Tokens
  - Information about the connection between the tokens (KV cache)(Optional)
- Handles the context switches
- Can dynamically update its context switching policy
  - Recompute KV cache on short requests
  - Save/Load KV cache on long requests
- Applies adapters to the base model to address different user needs



Inference Engine

- Supports
  - Multiple inference optimizations
  - GPU and CPU inference
  - x86_64 and ARM
  - Hot swapping adapters to specialize the base model

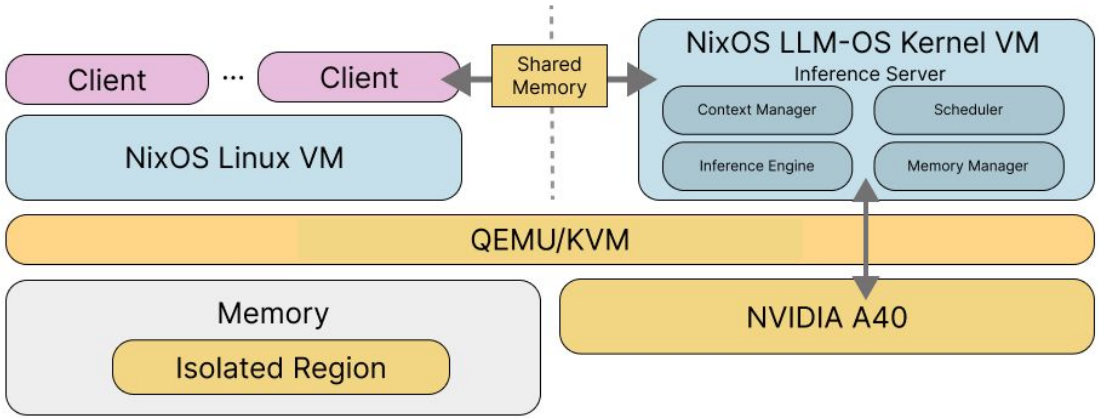


Outline

- ~~Motivation~~
- ~~Overview~~
- ~~Design~~
- Implementation
- Evaluation

Implementation

- QEMU/KVM is used as hypervisor
  - PCI Passthrough used for GPU
- *Send Message* is implemented as a shared memory
  - Memory is divided into separate slots
- llama.cpp is used as the inference engine



Outline

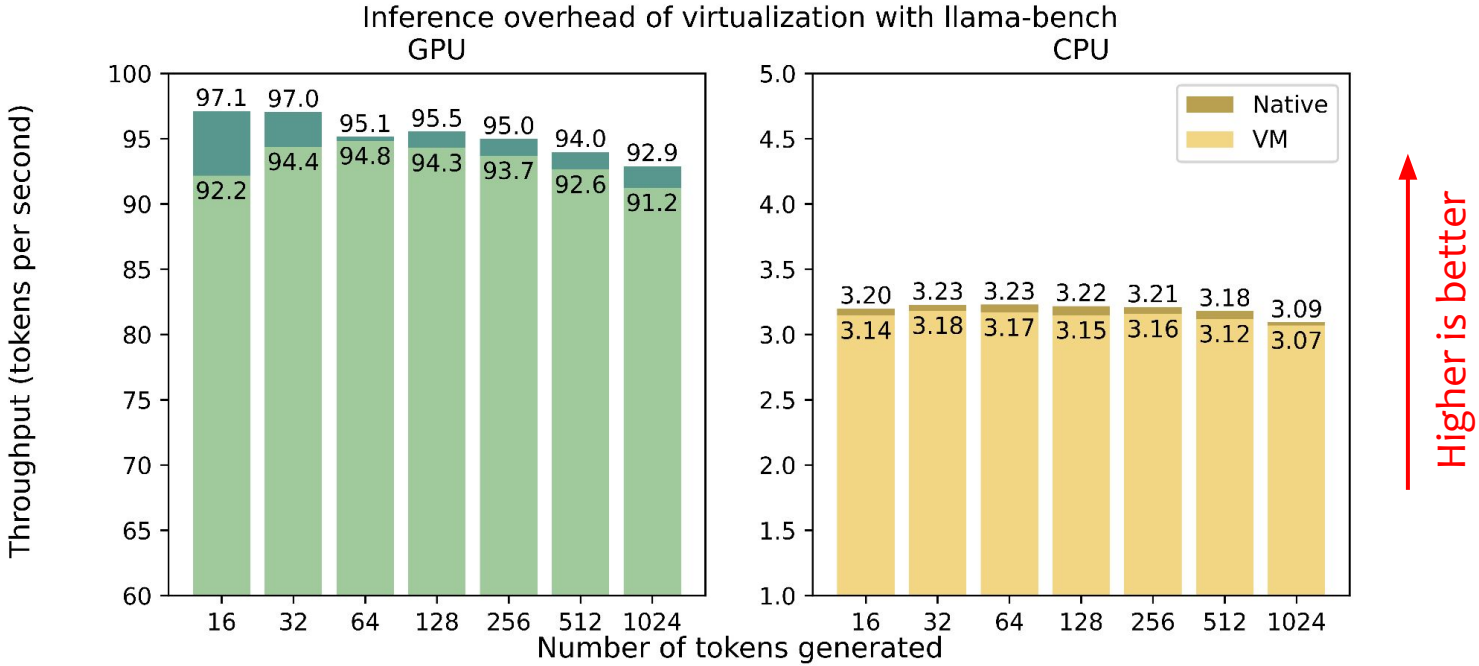
- ~~Motivation~~
- ~~Overview~~
- ~~Design~~
- ~~Implementation~~
- Evaluation

Evaluation

- Hardware:
  - Intel(R) Xeon(R) Gold 6326 CPU @ 2.90GHz
  - 314 GB DRAM
  - NVIDIA A40 GPU with 48 GB VRAM
- Baselines
  - llama-bench with and without virtualization
  - Phoronix OpenSSL SHA256 CPU Benchmark
- Results
  - **Virtualization Overhead**
  - **Implementation Overhead**
  - **Impact on Host OS**
  - Context Switch Latency
  - Scaling with Clients
  - Scheduler Frequency

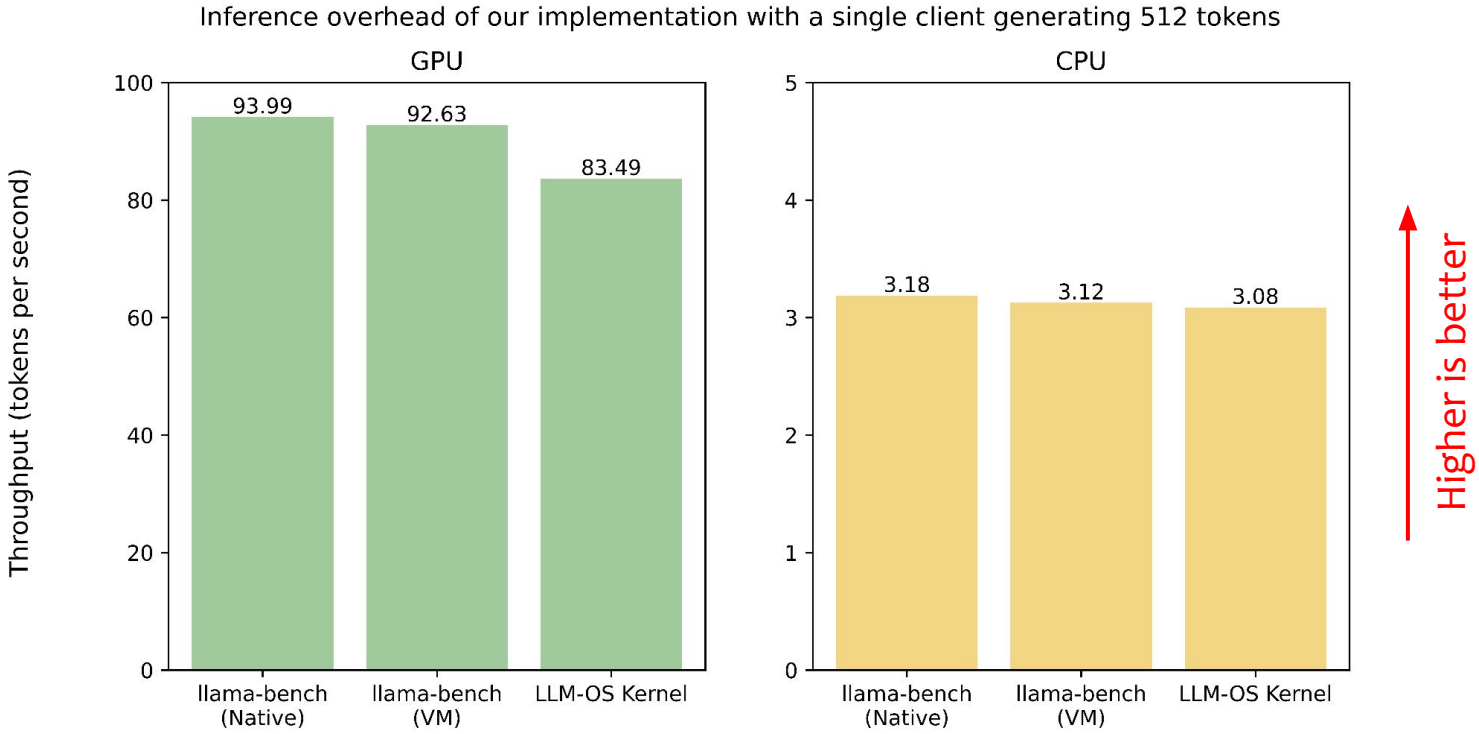
Virtualization overhead

Both on CPU and GPU, the virtualization overhead is less than 2%



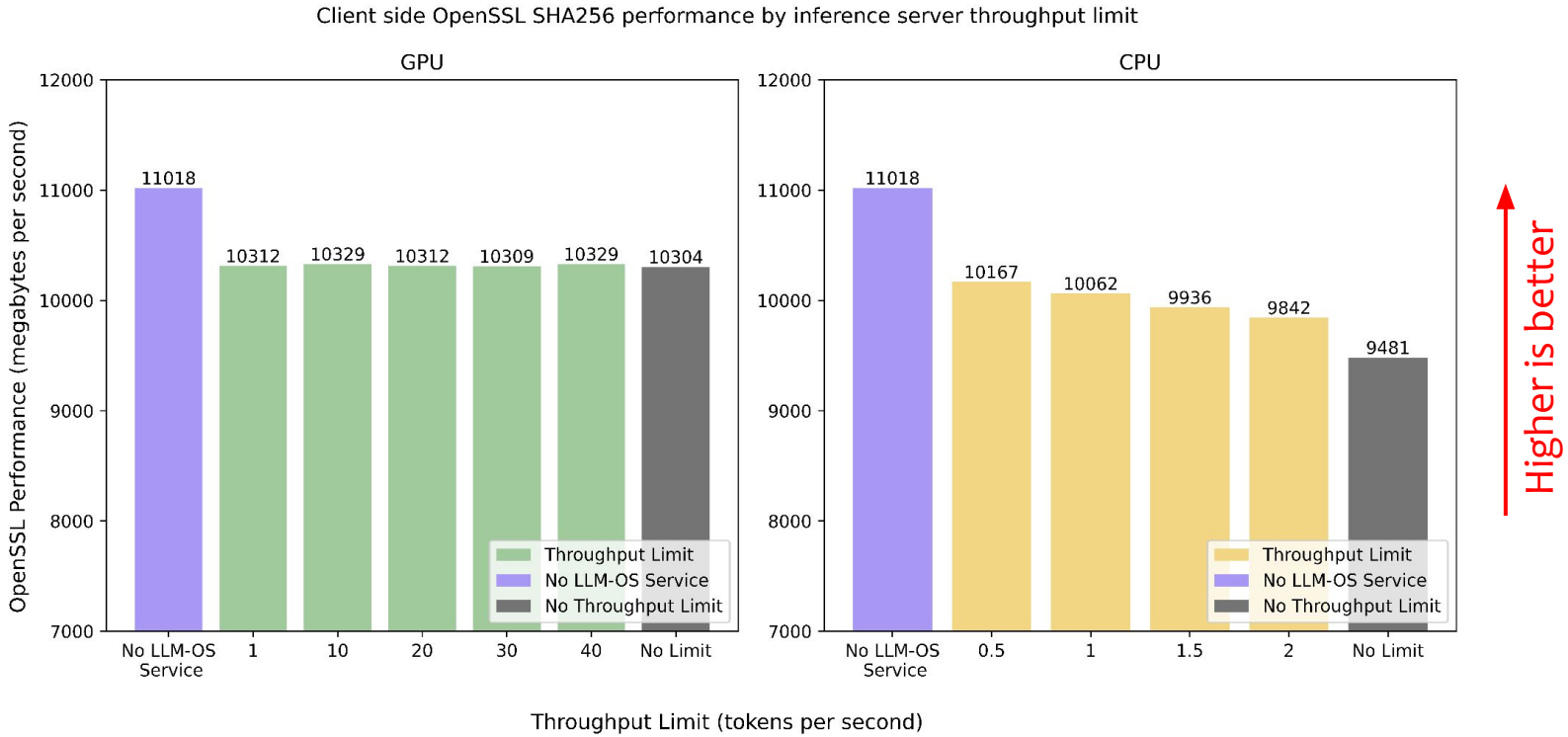
Implementation overhead

When a single client and the LLM-OS service are running in separate VMs, we introduce 2% overhead on CPU inference and 10% on GPU inference



Impact on Host OS

Serving LLM inference impacts the host side applications. On CPU inference, the impact can be reduced by setting a throughput limit.



Summary

LLM-OS project aims to address:

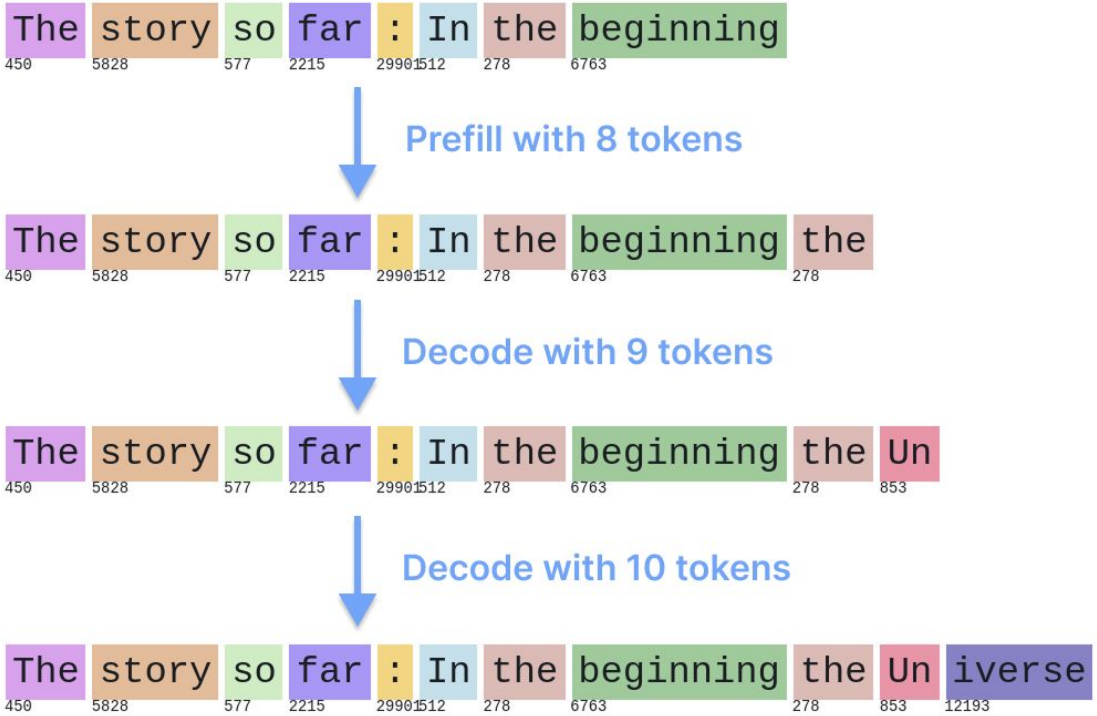
- Model security
- Adaptability of the base model
- Resource efficiency in edge devices

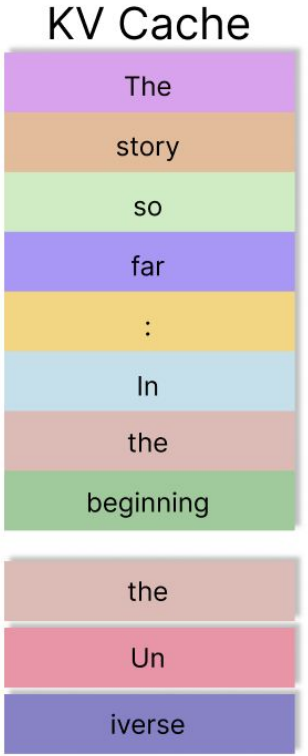
In this thesis we contributed to the larger LLM-OS project by:

- Implementing an example client that can simulate real world environment
- Creating an inter-VM communication mechanism and protocol
- Designing a custom inference server
- Deploying and evaluating the implementation

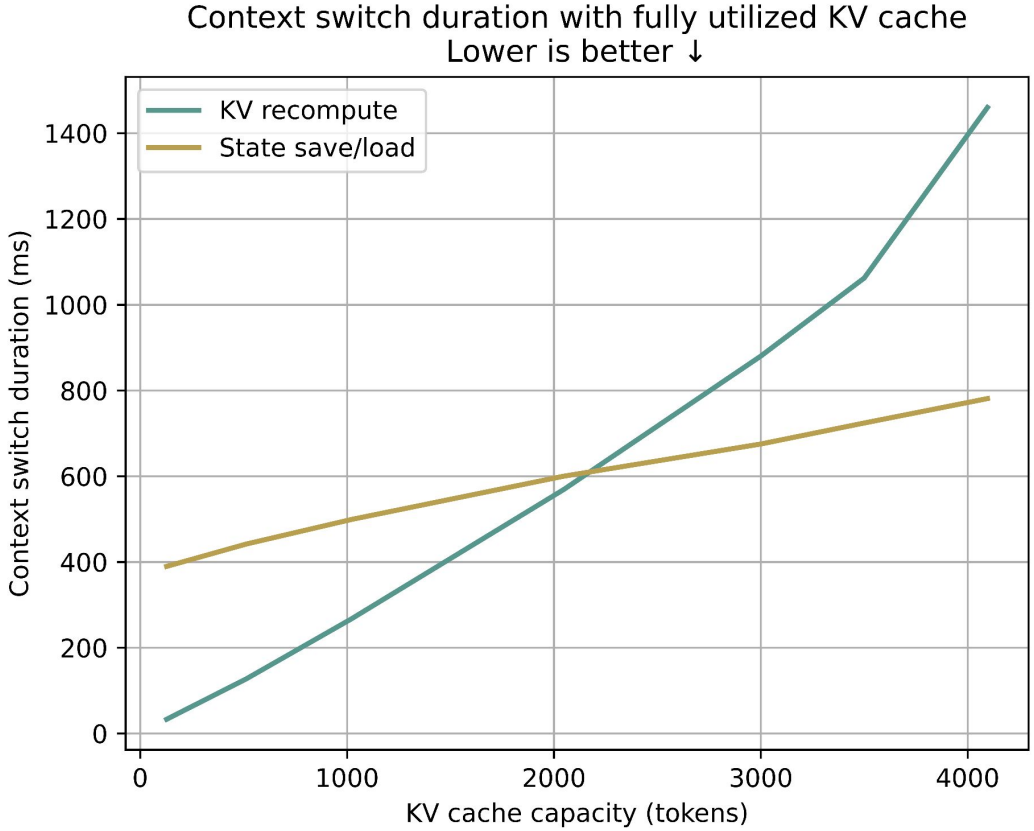
Backup

Autoregressive Generation and KV Cache

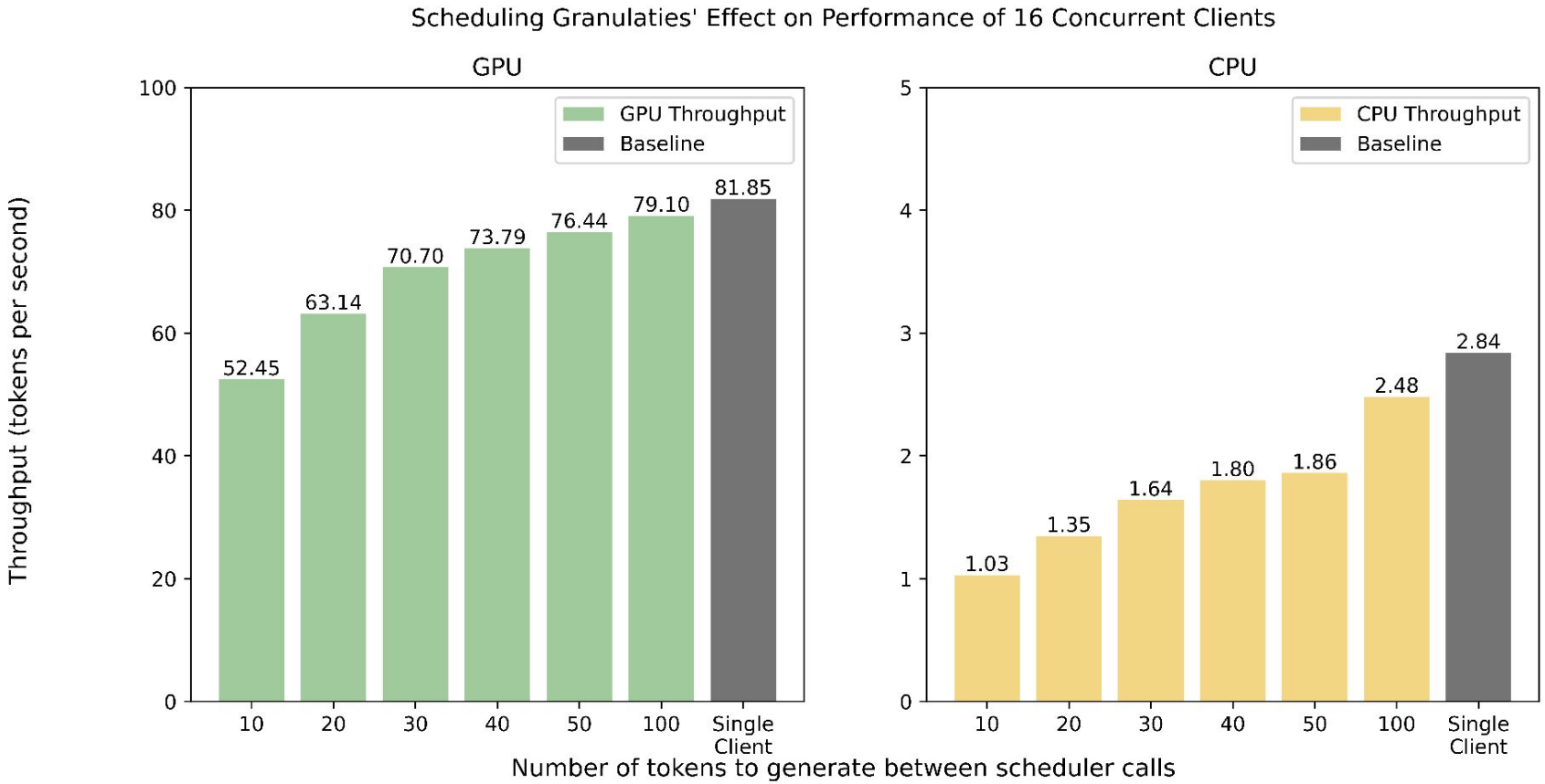




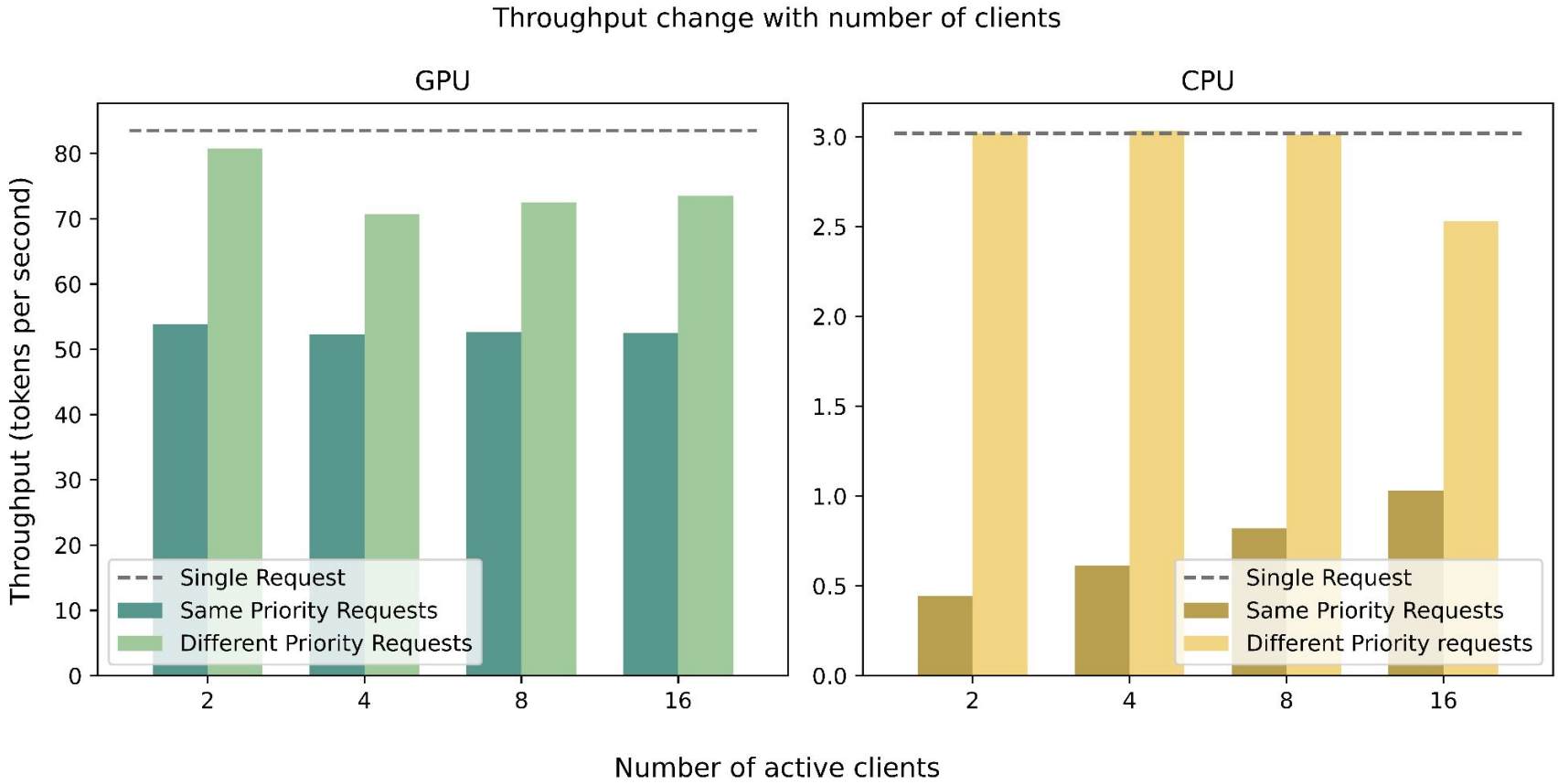
Context Switch Duration by Policy



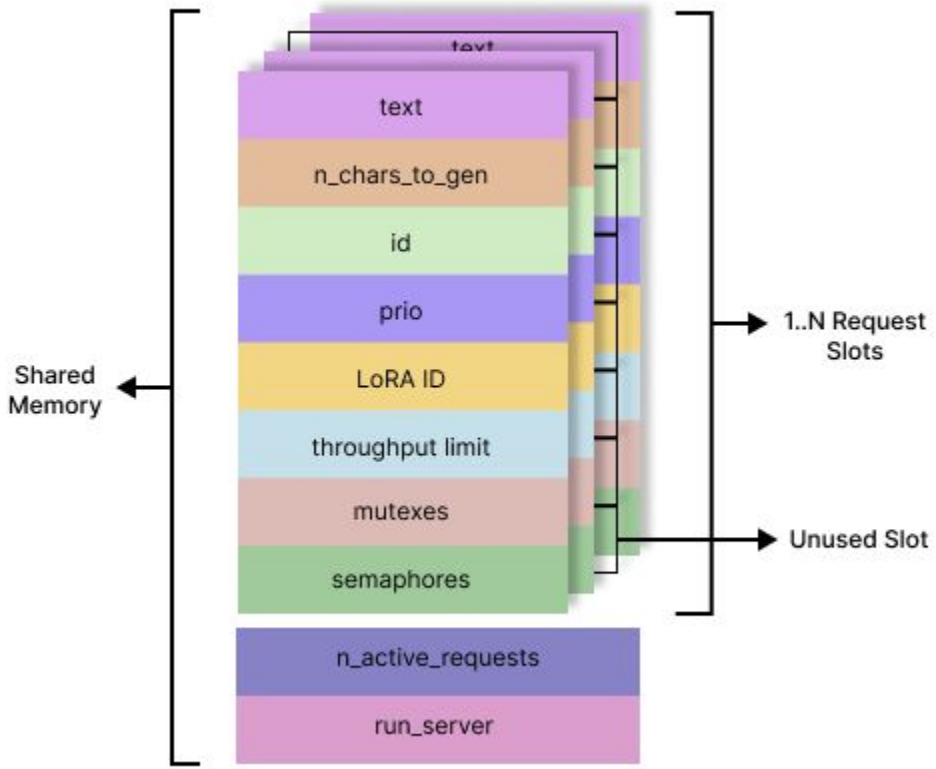
Scheduling Frequency



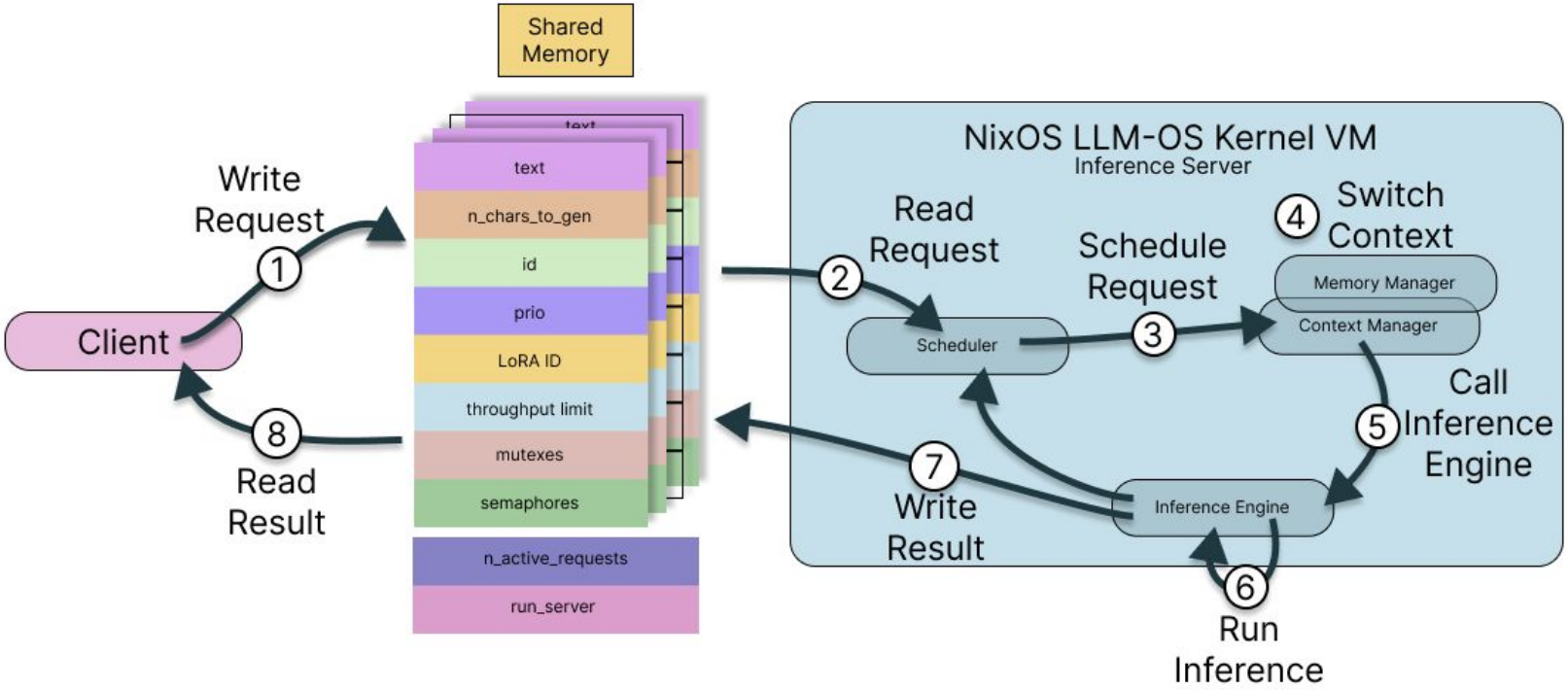
Client Scaling



Shared Memory Slots



General Workflow



Shared Memory Workflow