Nottingham Trent University

Efficient Request Scheduler for LLM Inference

Chen Chen (Nottingham Trent University),
Lei jiao (University of Oregon),
Richard Mortier (University of Cambridge)
22nd April, 2026

Background

AI Stack

LLM-Based applications

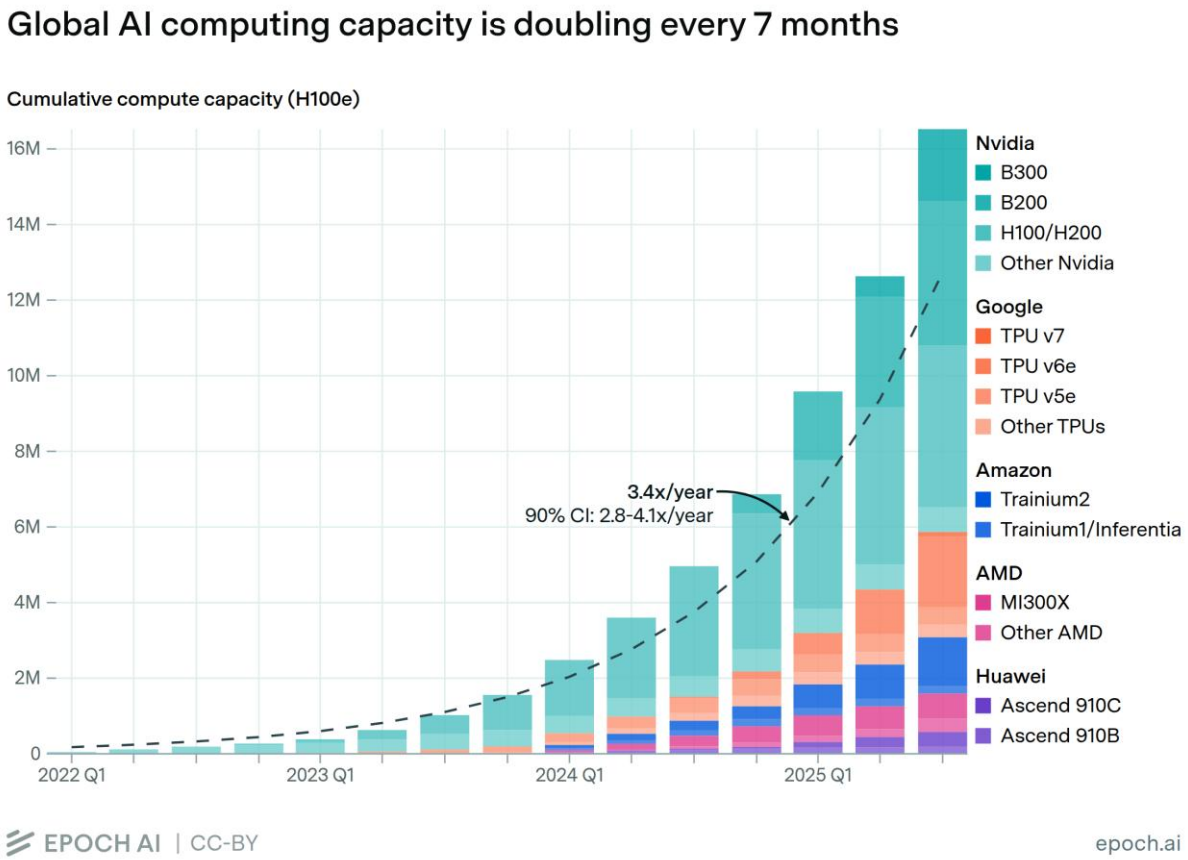
LLM Engines
(LLM inference)

Distributed Frameworks
(run distributed apps)

Infra Layer
(allocate resources)

NTU

AI demands growing fast

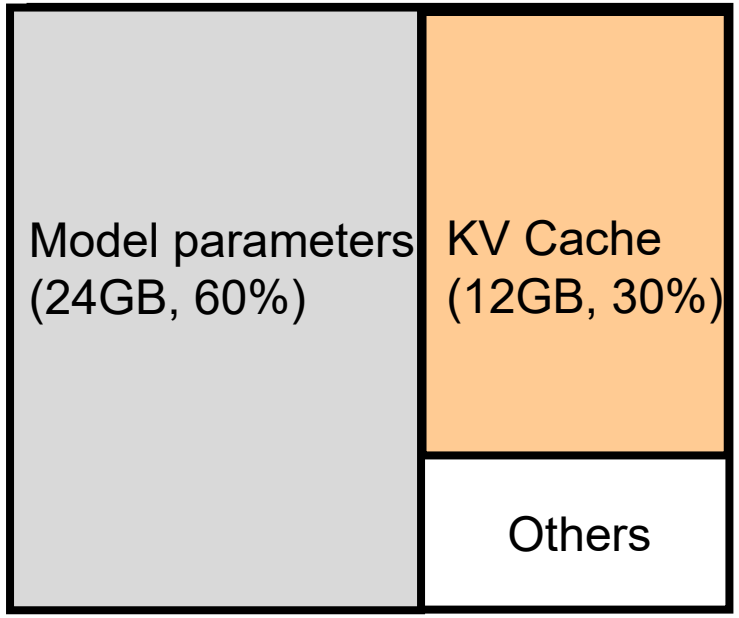


https://epoch.ai/data-insights/ai-chip-production

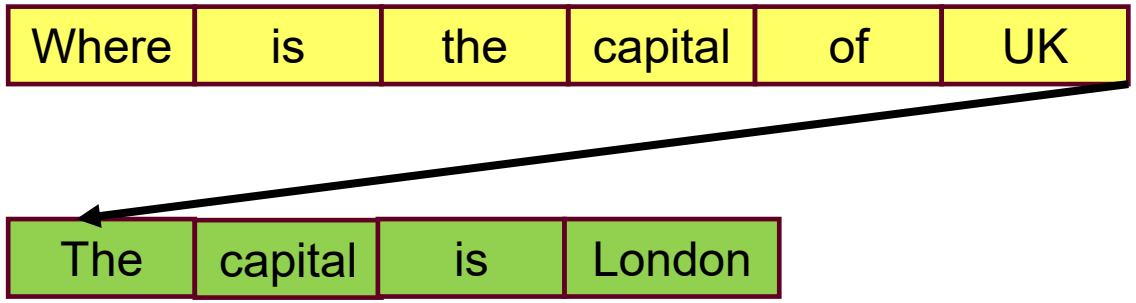
- LLM models are large – 30B-70B parameters for high performance, over 100B for frontier models.
- Inference is also resource-intensive, especially memory.

- LLM applications inject system prompts to define the role and tune.
- Prefill is GPU-heavy while decode is memory-heavy.

Where does memory go?



A 12B LLM on A100-40GB,
presume the KV cache is
4K context x 64 batches



KV cache stores the previous token embeddings:
40~50KB per token

A request can take a few hundreds MB to 1 GB

NTU

Prefix caching

- Extending this caching concept across multiple requests, we call it prefix caching.

- Different requests with identical prefixes can reuse the same cache of the prefix tokens.

Challenges

System prompts are everywhere

Examples of System prompts

Amazon user

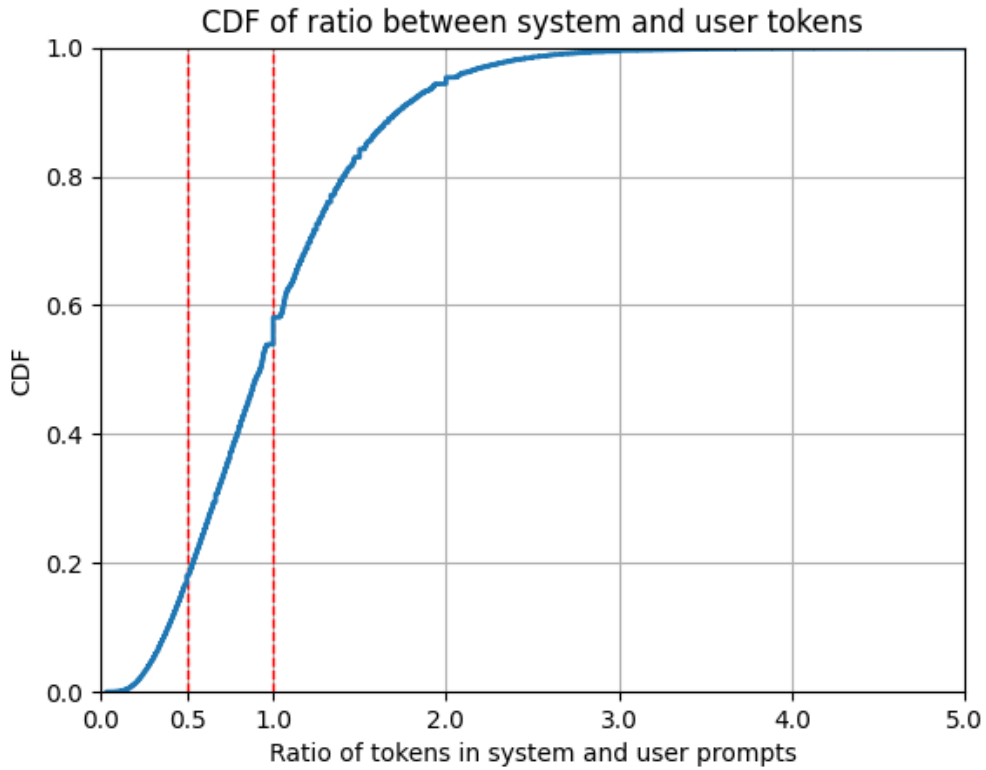
```
"role": "system", "content": "I am Rufus, Amazon's AI
shopping assistant. My goal is to help customers discover
products and make informed shopping decisions."
"role": "user", "content": "{input_shopping_question}"
```

Gemini user

```
"role": "system", "content": "You are Gemini, a helpful AI
assistant built by Google. I am going to ask you some
questions. Your response should be accurate without
hallucination.."
"role": "user", "content": "{input_search_question}"
```

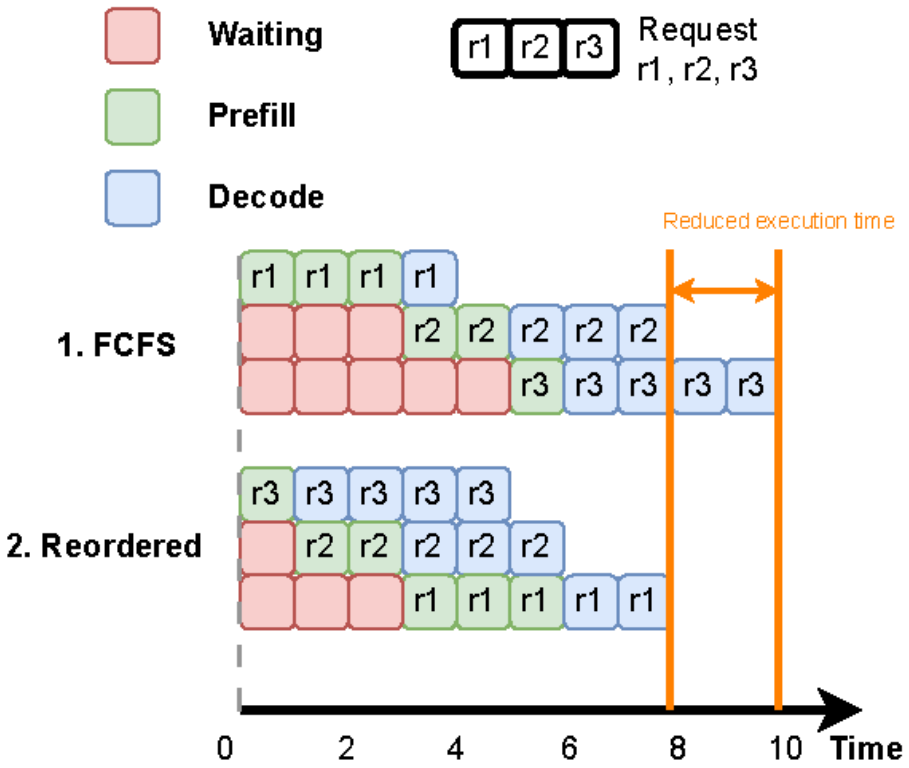
Claude user

```
"role": "system", "content": "You are Claude Code,
Anthropic's official CLI for Claude.
You are an interactive CLI tool that helps users with
software engineering tasks. Use the instructions below
and the tools available to you to assist the user."
"role": "user", "content": "{input_coding_question}"
```



https://github.com/LouisShark/chatgpt_system_prompt/tree/main
https://github.com/FoundationAgents/MetaGPT

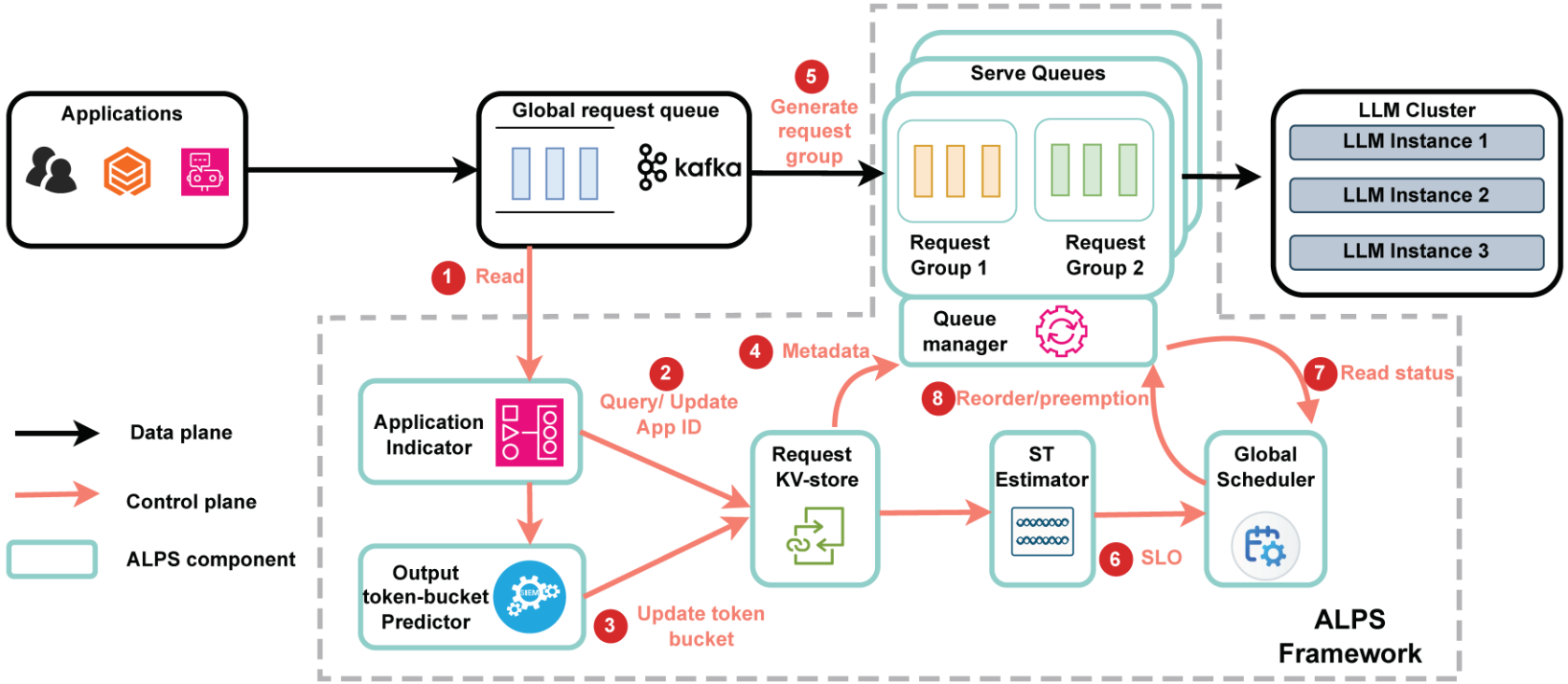
Continuous batching



- New requests get inserted (prefill)
- Ongoing requests (decode) continue
- The scheduler mixes them into the same batch

The idea

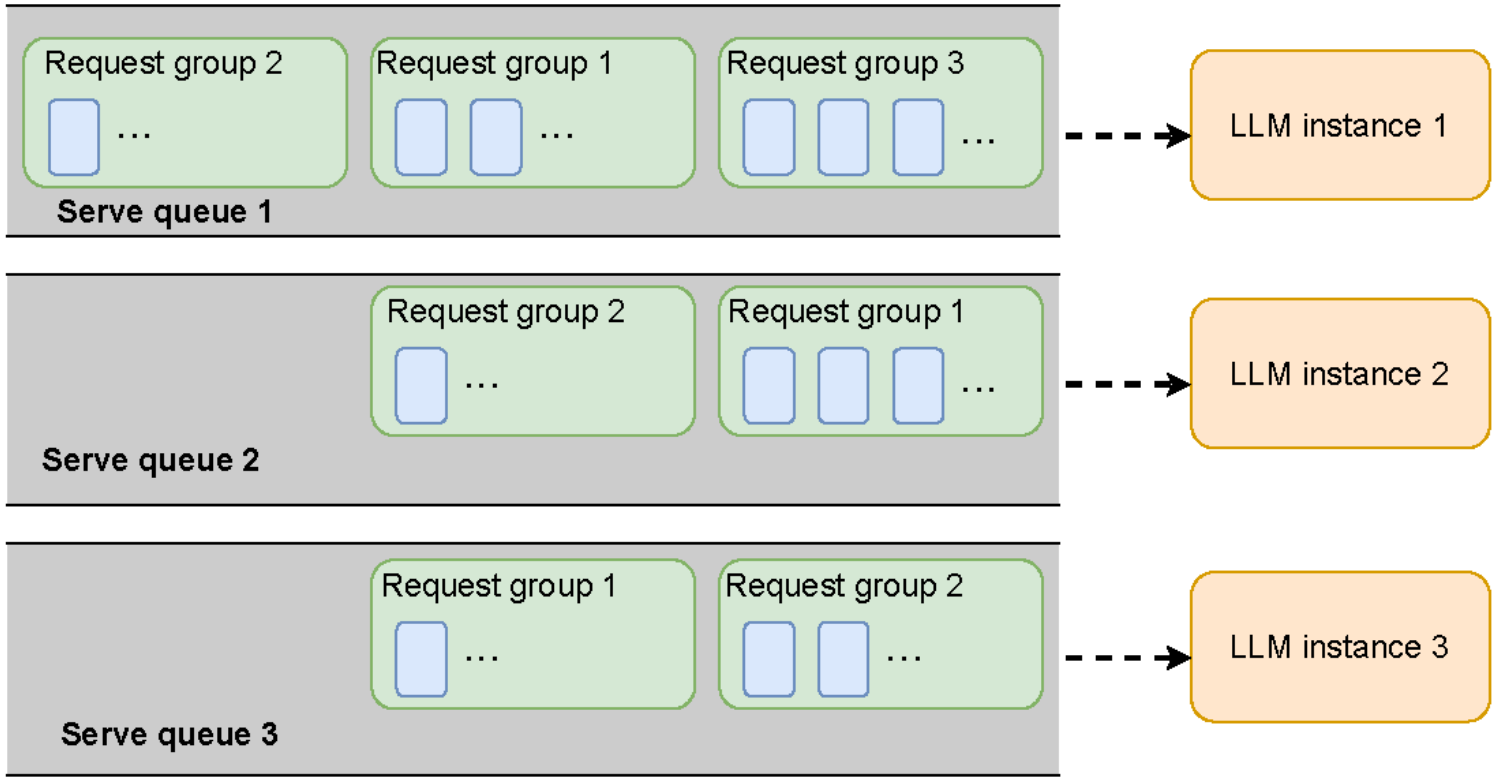
Aggregate requests in request groups



Maximize prefill and decode overlap

- prioritize requests with large generation ratio ϵ.

- $\epsilon = \frac{L_{decode}}{L_{prefill}}$



NTU

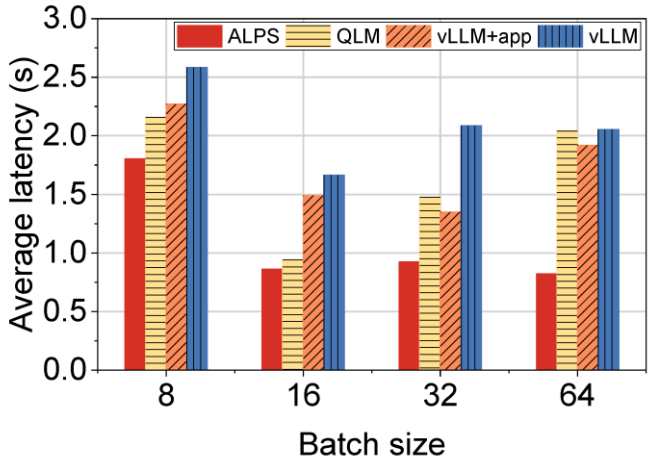
Performance

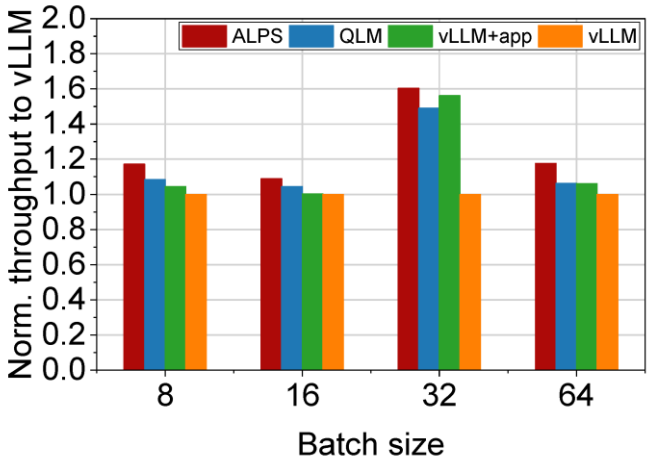
Set up

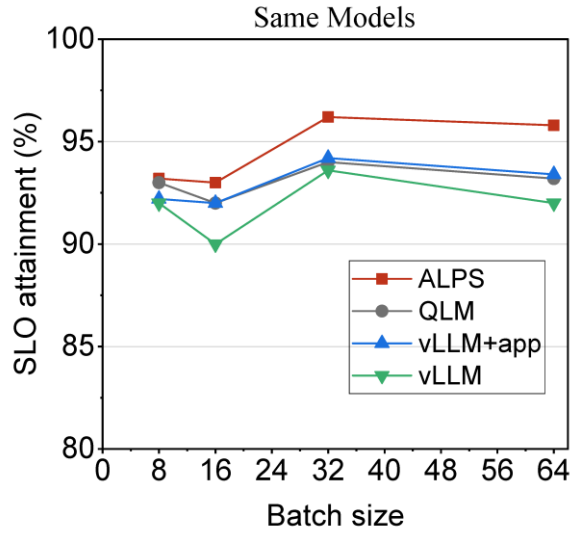
Type	Dataset	Request size
Conversation	AlpacaEval	805
Question Q&A	MS MARCO	102k
Code generation	Nl2bash	609
Math problem	Orca-math	200k
Finance	Finance-alpaca	68.9k

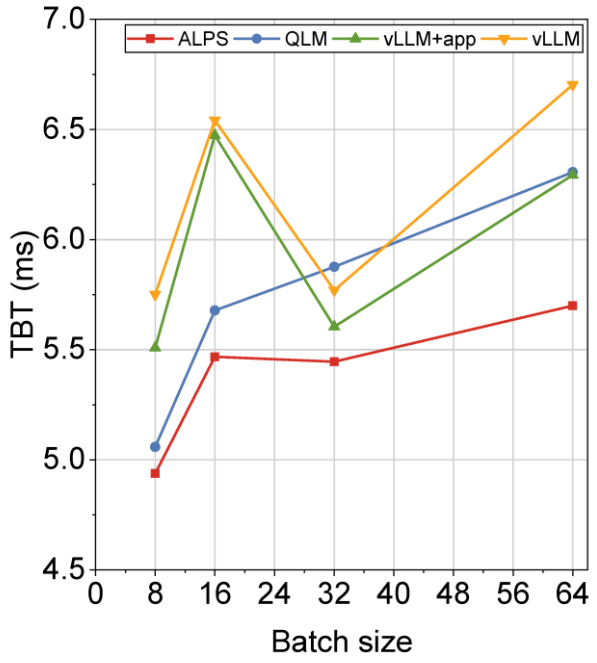
Nvidia Jetson AGX Orin, Gemma-3-1B, Llama-3-1B, SmoiLM2-360M over vLLM.

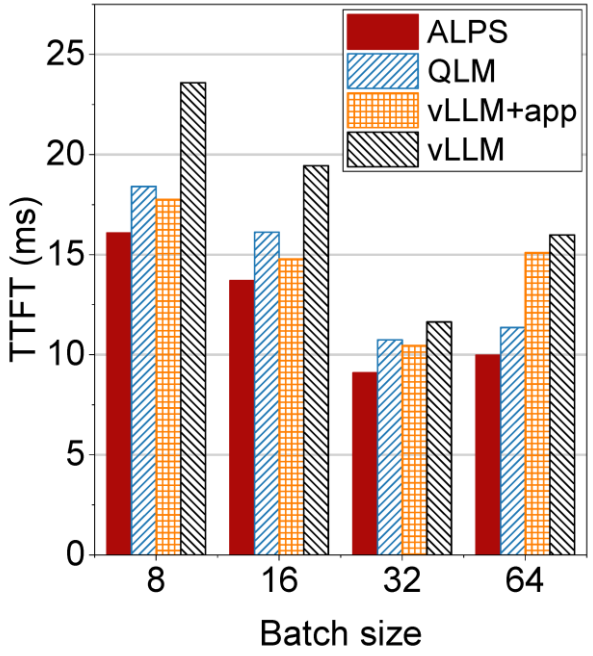
NTU

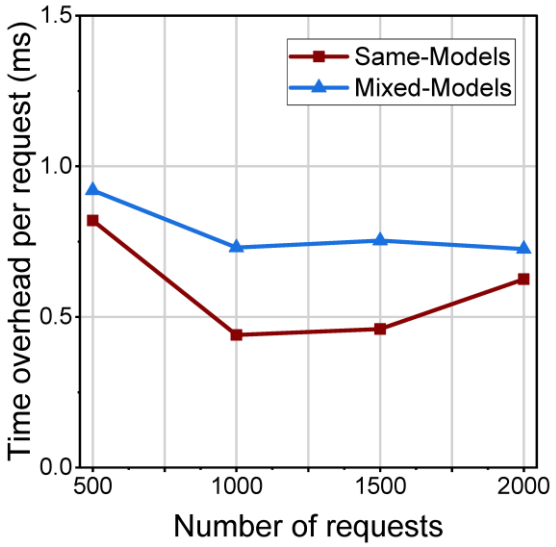

ALPS
QLM
vLLM+app
vLLM
Average latency (s)
Batch size


ALPS
QLM
vLLM+app
vLLM
Norm. throughput to vLLM
Batch size


Same Models
SLO attainment (%)
ALPS
QLM
vLLM+app
vLLM
Batch size


ALPS
QLM
vLLM+app
vLLM
TBT (ms)
Batch size


ALPS
QLM
vLLM+app
vLLM
TTFT (ms)
Batch size


Same-Models
Mixed-Models
Time overhead per request (ms)
Number of requests

Future work

- Can we partition a model into smaller components?
- Can we make those components distributed?
- Can we only scale the components that need to be scaled up/out?

Nottingham Trent University

Thank you

chen.chen04@ntu.ac.uk