



UbuCon India<sup>25</sup>


# Optimizing LLM Inference on Resource-Constrained Hardware

Techniques and Tools for Faster, Cheaper, and Scalable LLM Inference



Abdul Hakkeem P A  
AI Researcher

Qualcomm

 Canonical

IT'S FOSS



# About

- Exploring Inference Optimisation and MLOps.
- R&D Engineer Intern at FinTech Startup.
- Final Year MS Student at CUSAT.
- Previously published two research papers in MLOps.
- Open Source Contributor. (Hugging Face, Yolo, Infisical)
- Community Organiser. (AWS UG Kochi)

# The Rise of LLM-Powered Everything

- 2022: ChatGPT → Everyone saw LLMs as magic.
- 2023: Companies race to integrate “LLM-powered” features.
- 2024–25: Every app wants AI copilots, RAG, chatbots, summarizers.



# The Closed-Source Phase

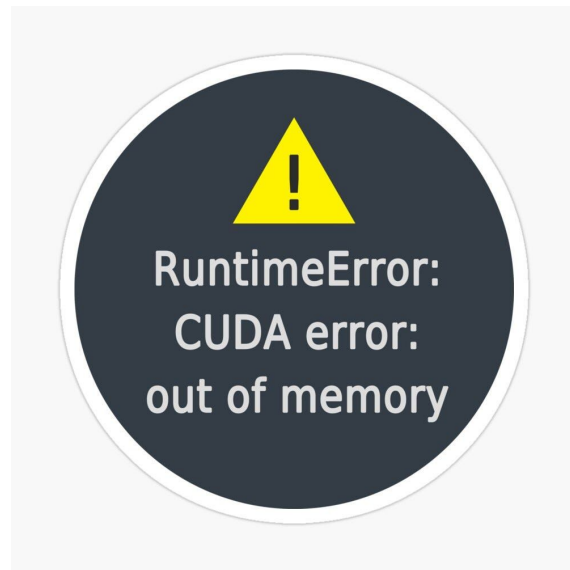
- Easy API access → rapid prototyping.
- Reliable models, but limited control & high cost.
- Privacy & compliance issues → data can't always leave org boundaries.
- Developers hit rate limits & cost ceilings.

# The Open-Source Shift

- Community progress (Llama, Mistral, Phi-3, Gemma, Deepseek, Granite)
- Fine-tuning freedom - tailor for domain-specific use cases
- Cost advantage - one-time setup, infinite queries
- Deployment flexibility - on-prem, edge, or private cloud



# The New Challenge: Running It Yourself



# The New Challenge: Running It Yourself

- Models too large for your GPU
- Slow response times
- Memory bottlenecks
- Expensive infra scaling

# VRAM vs Model Size

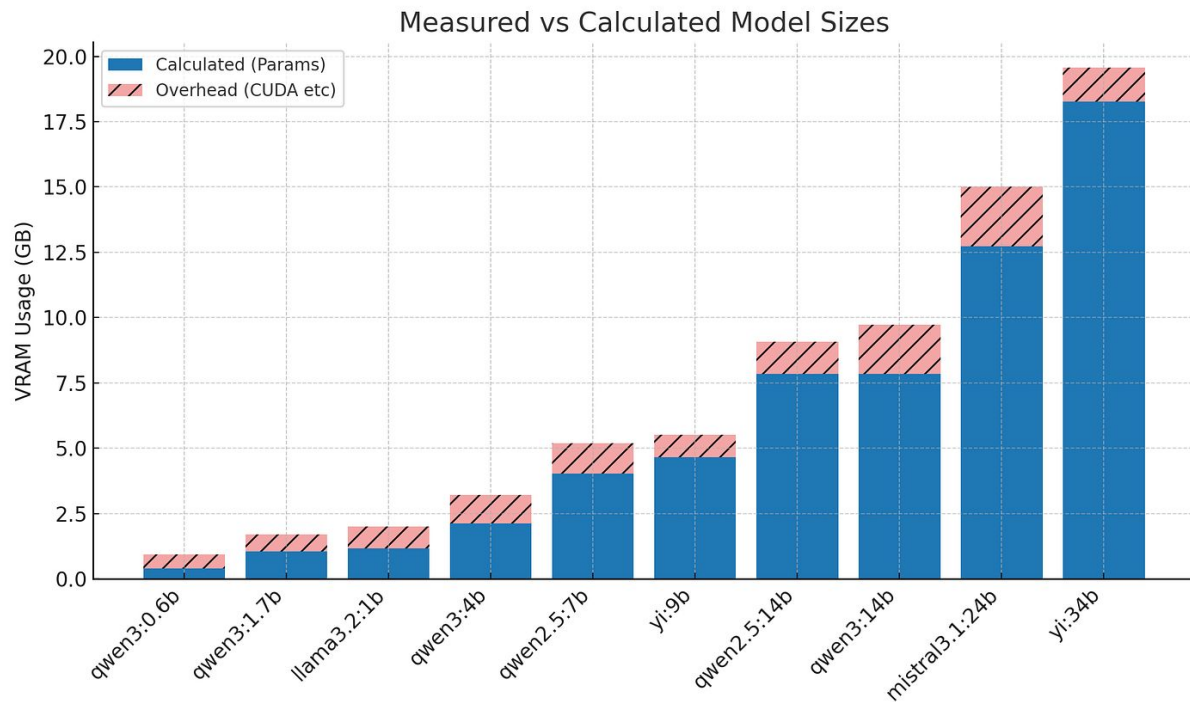


Fig 1. Representing the rise of VRAM requirement according to the model size.



# How much VRAM does your model need?

$$M = \left( \frac{(P \times 4B)}{(32/Q)} \times 1.2 \right)$$

M - Total Memory Required (Bytes)

P - Number of model parameters

B - Bytes per parameter

Q - Bit precision used for quantization

# How to evaluate your LLM System?

- Time To First Token (TTFT) : The time it takes to generate the first token after sending a request.
- Total Latency (E2EL): The time from sending the request to receiving the final token on the user end.
- Inter-Token Latency (ITL): The exact pause between two consecutive tokens.
- Throughput

# What Are We Really Optimizing For?

- Faster Responses (↓ Latency)
- More Parallel Work (↑ Throughput)

# Stages of LLM Inference

- Prefill - model processes the entire input prompt and builds the initial KV cache.
- Decode - model generates tokens one by one, using the context from prefill.

# What is KV Cache in Attention Layer?

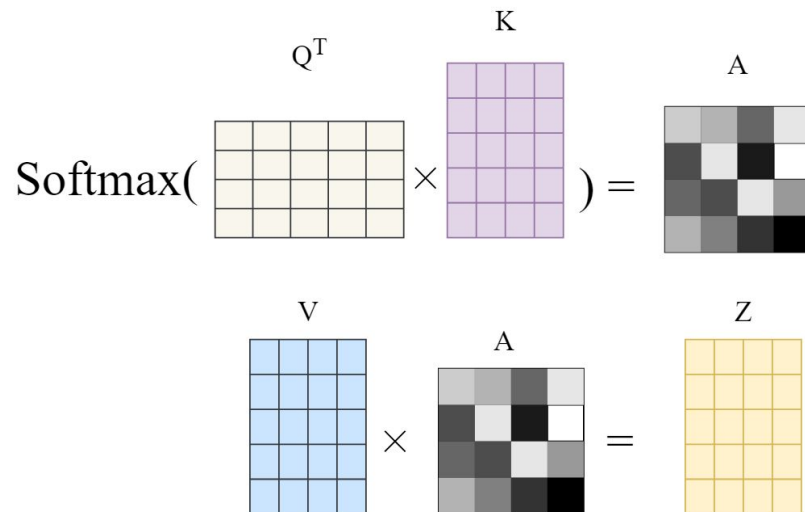


Fig 2. Illustration of Attention Mechanism

# The Real Bottleneck Inside LLM Inference

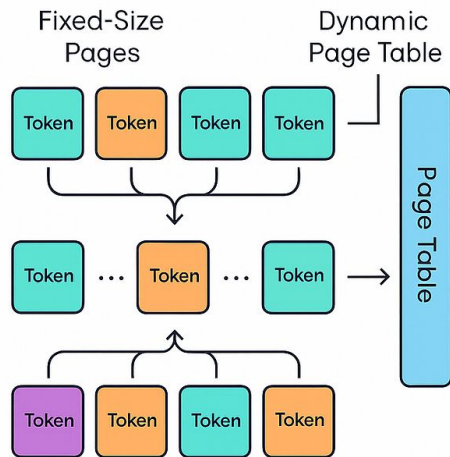
- Vanilla Transformer is not a production ready architecture.
- LLM inference is memory-bound, not compute-bound.
- Inefficient GPU memory management - especially KV Cache.
- Low GPU utilization in real workloads.
- Token generation is inherently sequential.

# Techniques to Optimise Inference

- Paged Attention
- Continuous Batching
- Speculative Decoding
- Prefix Chunking
- Quantisation

# Paged Attention

- KV Cache stored in non-contiguous memory
- Inspired by OS Virtual Memory
- Reduces GPU memory fragmentation
- Enables swapping and reuse of memory pages
- Improves throughput and parallelism





# Iteration Level Scheduling

- Fine-grained scheduling at each decoding iteration
- Maximizes GPU utilization
- Up to 23x throughput improvement
- Reduces TTFT (Time to First Token)

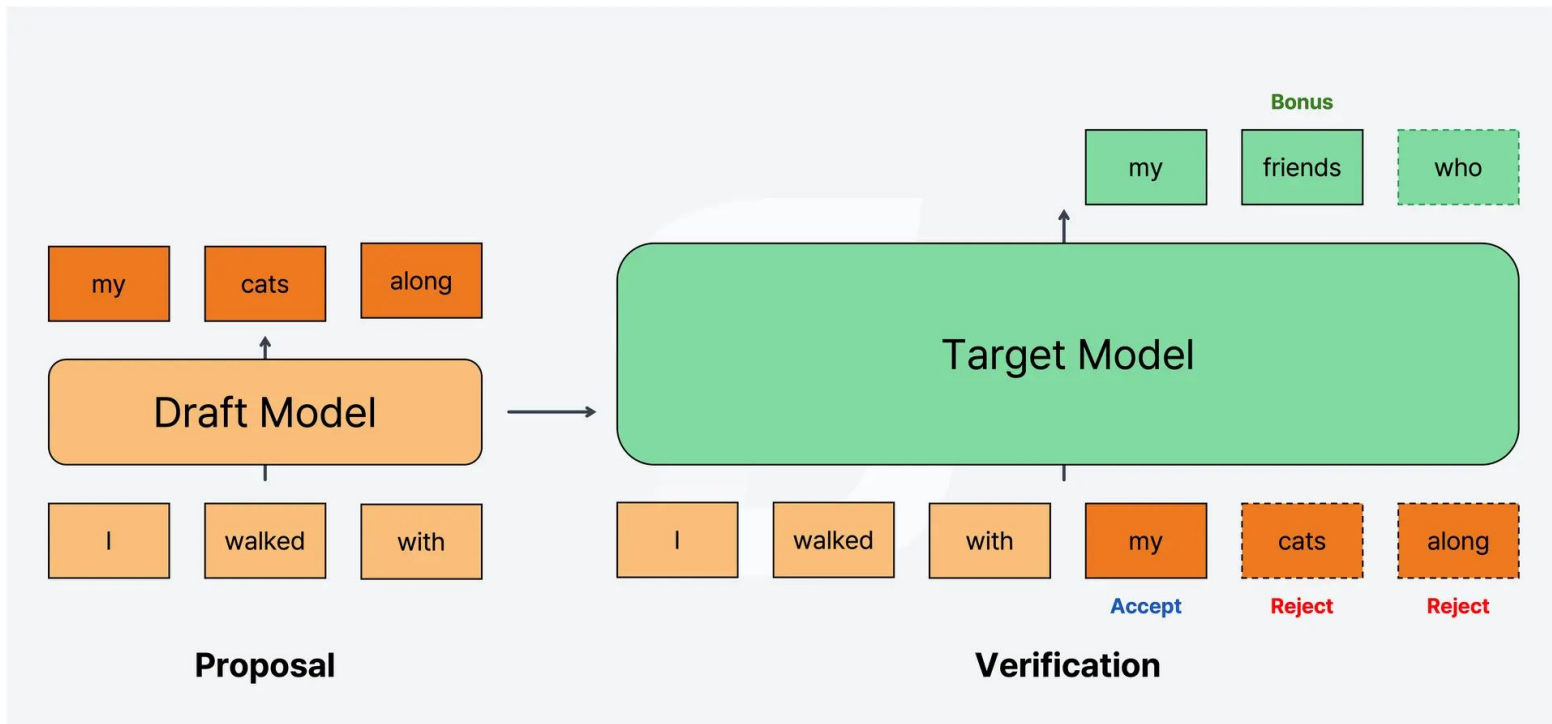
$T_1$	$T_2$	$T_3$	$T_4$	$T_5$	$T_6$	$T_7$	$T_8$
$S_1$	$S_1$	$S_1$	$S_1$				
$S_2$	$S_2$	$S_2$					
$S_3$	$S_3$	$S_3$					
$S_4$	$S_4$	$S_4$					

$T_1$	$T_2$	$T_3$	$T_4$	$T_5$	$T_6$	$T_7$	$T_8$
$S_1$	$S_1$	$S_1$	$S_1$	$S_1$	END	$S_6$	$S_6$
$S_2$	$S_2$	$S_2$	$S_2$	$S_2$	$S_2$	$S_2$	END
$S_3$	$S_3$	$S_3$	$S_3$	END	$S_5$	$S_5$	$S_5$
$S_4$	$S_4$	$S_4$	$S_4$	$S_4$	$S_4$	END	$S_7$

# Speculative Decoding

- A two-model decoding strategy.
- A small draft model predicts several future tokens.
- A larger target model verifies and accepts the correct tokens.
- Reduces generation latency.
- Maintains output quality.
- Improves throughput on constrained hardware.

# Speculative Decoding



# Prefix Caching

- Useful in production workloads with repeated prompt structures
- Applications: Chat systems, AI agents, and RAG pipelines.
- Helps to reduce latency and cost in LLM inference

For example, consider a chatbot with this system prompt:

***System Instruction: You are a helpful AI writer. Please write in a professional manner.***

This prompt doesn't change from one conversation to the next. Instead of recalculating it every time, you store its KV cache once.

# Quantisation

- A model compression technique
- Reduces the precision of model weights and activations
  - e.g. from FP16 / FP32  $\rightarrow$  INT8 / INT4 / NF4.
- PTQ (Post Training Quantization): Apply quantization after training.
- QAT (Quantization-aware Training): Train with quantization effects in mind for better accuracy.

# Introducing vLLM

- Introduced Paged Attention
- State-of-the-art serving throughput
- Streaming outputs
- OpenAI-compatible API server
- Hugging Face integration
- Supports decoding algorithms, including parallel sampling, beam search etc.

# How to use vLLM? - Online Serving

Single Command to Launch the API Server

```
vllm serve NousResearch/Meta-Llama-3-8B-Instruct \  
  --dtype auto \  
  --api-key token-abc123
```

Accessing the model using OpenAI Framework

```
from openai import OpenAI  
client = OpenAI(  
    base_url="http://localhost:8000/v1",  
    api_key="token-abc123",  
)  
  
completion = client.chat.completions.create(  
    model="NousResearch/Meta-Llama-3-8B-Instruct",  
    messages=[  
        {"role": "user", "content": "Hello!"},  
    ],  
)  
  
print(completion.choices[0].message)
```

# How to use vLLM? - Offline Serving

Single Command to Launch the API Server

```
from vllm import LLM

# Initialize the vLLM engine.
llm = LLM(model="facebook/opt-125m")
```



# Speculative Decoding in vLLM

Single Command to Launch the API Server

```
python -m vllm.entrypoints.openai.api_server \
    --host 0.0.0.0 \
    --port 8000 \
    --model facebook/opt-6.7b \
    --seed 42 \
    -tp 1 \
    --gpu_memory_utilization 0.8 \
    --speculative_config '{"model": "facebook/opt-125m",
"num_speculative_tokens": 5}'
```

# Let's connect



[www.abdulahakkeempa.com](http://www.abdulahakkeempa.com)



[abdul-hakkeem-pa](https://www.linkedin.com/in/abdul-hakkeem-pa)



[x.com/abdulahakkeempa](https://x.com/abdulahakkeempa)






UbuCon India<sup>25</sup>

# Thank You!

Qualcomm

 Canonical

IT'S FOSS

