LLM-Inference-Bench: Inference Benchmarking of Large Language Models on AI Accelerators

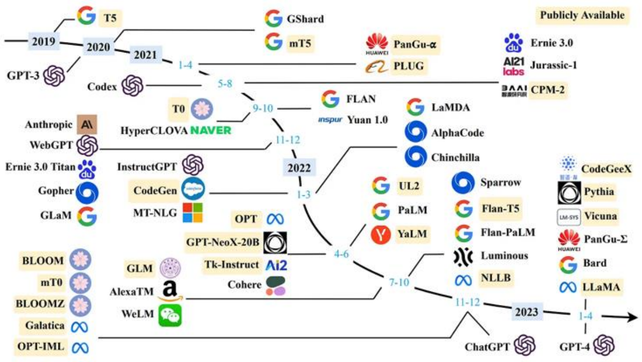
Krishna Teja Chitty-Venkata*, Siddhisanket Raskar*,
Bharat Kale, Farah Ferdaus, Aditya Tanikanti, Ken Raffenetti,
Valerie Taylor, Murali Emani, Venkatram Vishwanath
*Equal Contribution

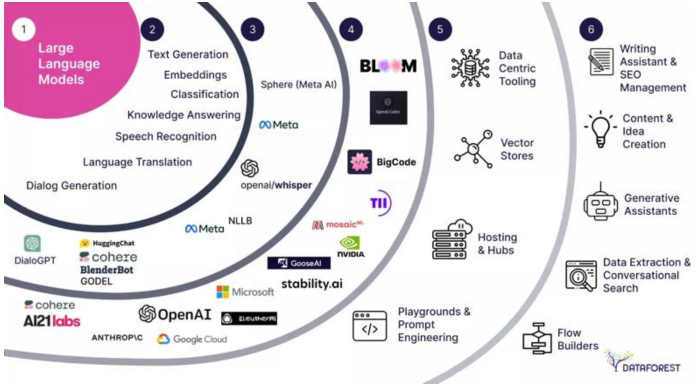
Argonne National Laboratory

15th IEEE International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems held in conjunction with SC24

Large Language Models (LLMs)

- LLM is a deep learning algorithm that's equipped to summarize, translate, predict, and generate human-sounding text to convey ideas and concepts.
- They leverage vast amounts of data and sophisticated algorithms to perform a wide range of tasks
- They rely on a massive number of parameters, which allows them to capture intricate language patterns and context.
- Examples of popular LLMs include OpenAI’s GPT, Google’s BERT, and Meta’s LLaMA.

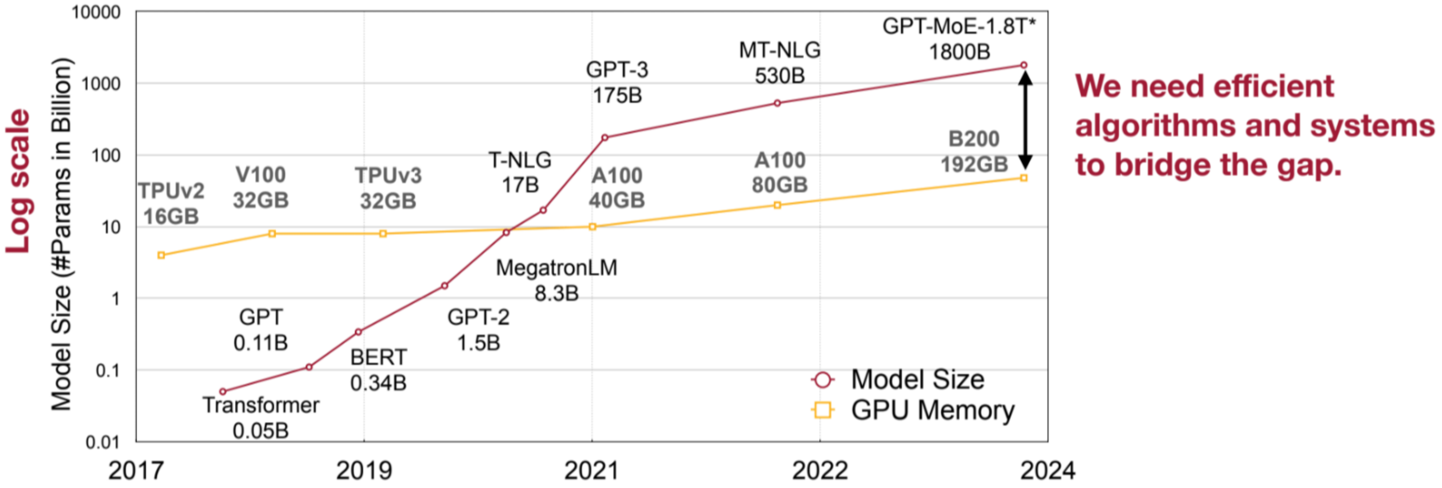




https://www.nextbigfuture.com/2023/04/timeline-of-open-and-proprietary-large-language-models.html
https://dataforest.ai/blog/large-language-models-advanced-communication

Challenge for LLM Inference and deployment: Colossal Sizes

- Despite being powerful, LLMs are hard to serve
- LLM sizes and computation are increasing exponentially
- We need model compression techniques and system support to bridge the gap

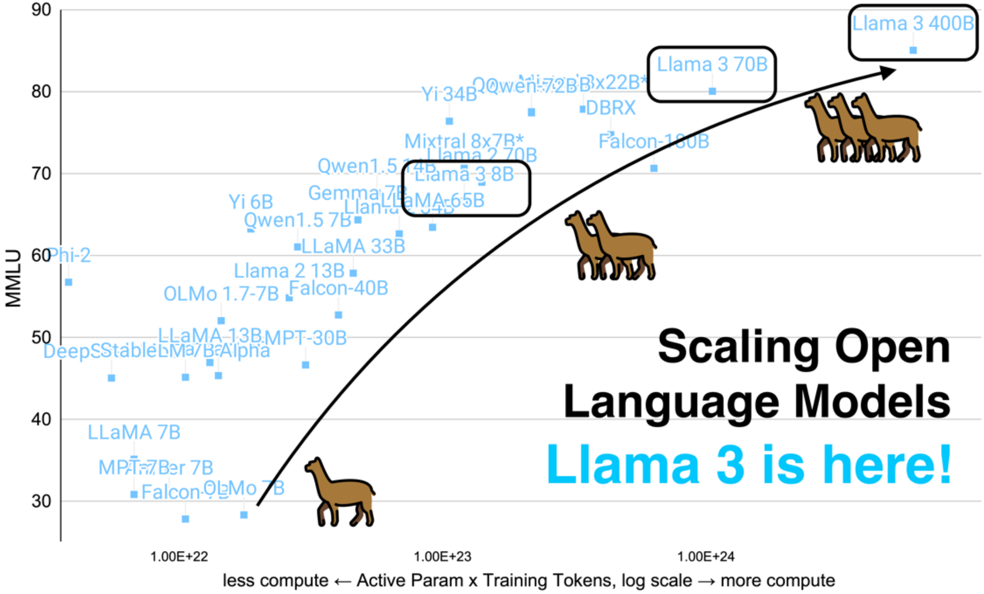


*: Jensen Huang, NVIDIA GTC 2024

Lin et al. AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration

Need for LLM Inference Optimizations

- LLMs, with billions of parameters, can be slow during inference due to the computational load required to process large amounts of data.
- Many applications require real time responses from LLMs, which can be challenging.
- The high computational demands of LLMs translate into significant operational costs.
- LLM Inference optimization methods reduces energy consumption and hardware requirements, making deployments more cost-effective



https://www.interconnects.ai/p/llama-3-and-scaling-open-llms

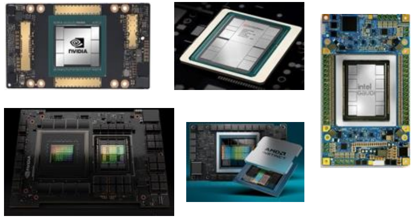
Argonne NATIONAL LABORATORY

LLM-Inference-Bench: Bridging LLMs, Accelerators and Frameworks

Open source LLMs
LLaMA, Mistral, Qwen

LLM-Inference-Bench

AI Accelerators
Nvidia, AMD GPUs,
SambaNova SN40L,
Habana Gaudi



vLLM

TensorRT-LLM

DeepSpeed MII

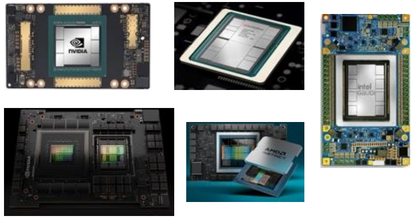
LLaMA C++

LLM-Inference-Bench: Bridging LLMs, Accelerators and Frameworks

Open source LLMs
LLaMA, Mistral, Qwen

LLM-Inference-Bench

AI Accelerators
Nvidia, AMD GPUs,
SambaNova SN40L,
Habana Gaudi



vLLM

TensorRT-LLM

DeepSpeed MII

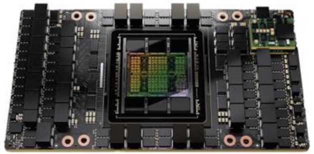
LLaMA C++

AI Accelerators for LLMs

- AI Accelerators for LLMs are key to handle billions and trillions of LLM parameters
- We consider the following Accelerators in our benchmarking study:
  - **Nvidia GPUs**: A100, H100 and GH200
  - **AMD GPUs**: MI300X and MI250
  - **AI accelerators**: SambaNova SN40L and Habana Gaudi2

Nvidia GPUs

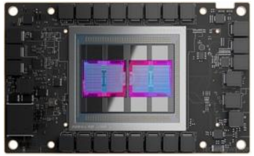
A100



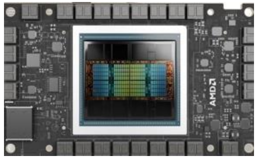
H100

GH200

AMD GPUs



MI250



MI300X

AI Accelerators

Habana Gaudi2

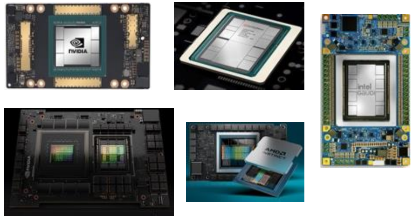
SambaNova SN40L

LLM-Inference-Bench: Bridging LLMs, Accelerators and Frameworks

Open source LLMs
LLaMA, Mistral, Qwen

LLM-Inference-Bench

AI Accelerators
Nvidia, AMD GPUs,
SambaNova SN40L,
Habana Gaudi



vLLM

TensorRT-LLM

DeepSpeed MII

Inference Frameworks

There has been a rise in Inference frameworks for LLM over the last few years

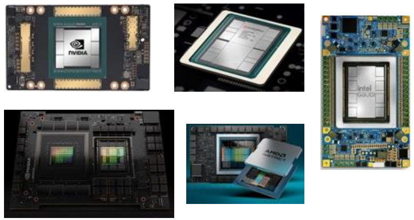
vLLM	TensorRT-LLM	Deepspeed-MII	LLaMA.cpp
Can run on diverse hardware platforms including Intel, Nvidia, AMD GPUs and AI accelerators such as Graphcore and Habana	Limited to Nvidia GPUs, such as A100, H100, GH200 series	Limited to Nvidia GPUs (such as A100, H100, GH200)	Can run on diverse hardware platforms including Intel, Nvidia, AMD GPUs
Supports wide range of Inference Optimizations	Supports wide range of Inference Optimizations	Lacks key LLM optimizations and instead relies on GPU kernel optimizations	Lacks many optimizations and does not scale with increase in number of GPUs
Has wide Community Support	Developed within Nvidia	Developed within Microsoft	Has wide Community support

LLM-Inference-Bench: Bridging LLMs, Accelerators and Frameworks

Open source LLMs
LLaMA, Mistral, Qwen

LLM-Inference-Bench

AI Accelerators
Nvidia, AMD GPUs,
SambaNova SN40L,
Habana Gaudi



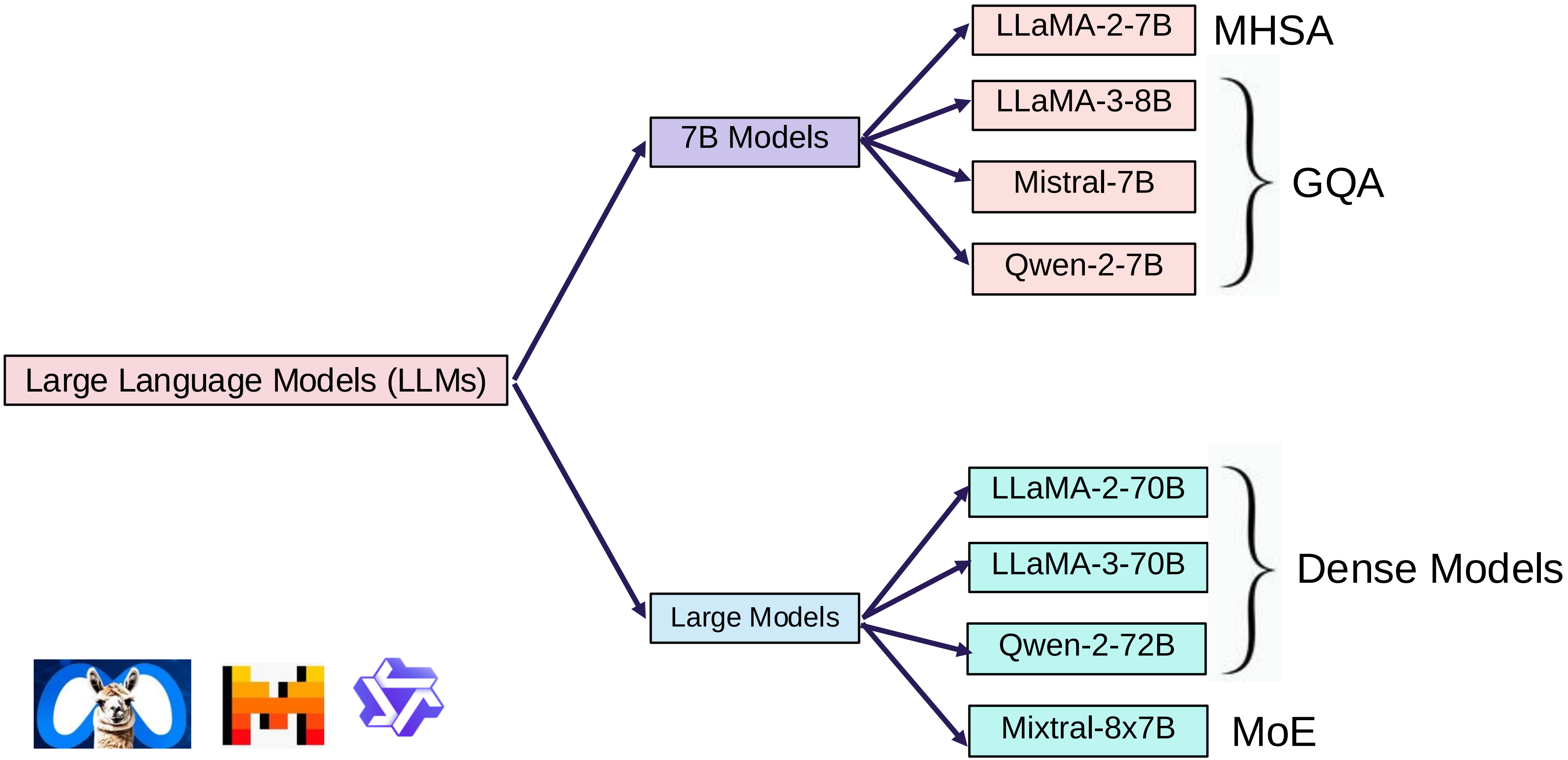
vLLM

TensorRT-LLM

DeepSpeed MII

LLaMA C++

Open Source LLMs



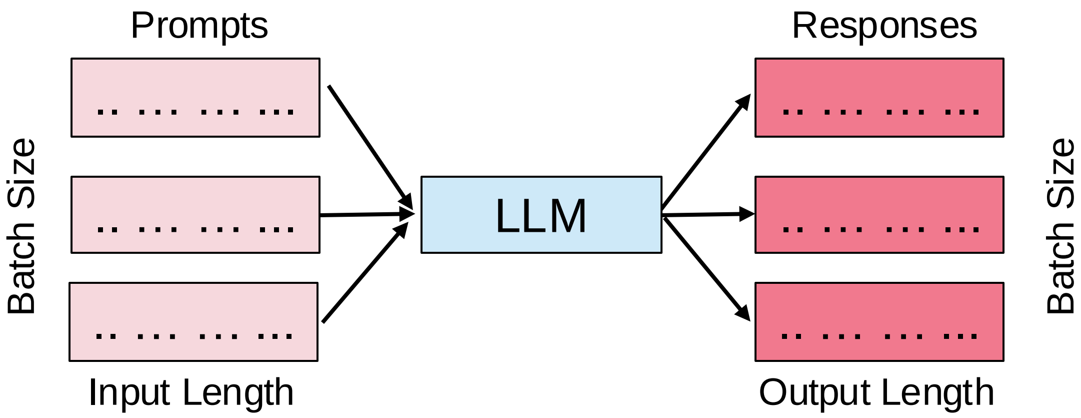
Argonne NATIONAL LABORATORY

LLM Inference – Basic Terms

Input Length: Input Length refers to the total number of tokens given to an LLM as input prompt for a single query.

Output Length: Output Size, also referred to as maximum new tokens is the number of tokens produced by the model as a response to a single input prompt.

Batch Size: Batch Size refers to the number of input sequences processed and new output sequences produced simultaneously.



Performance Metrics

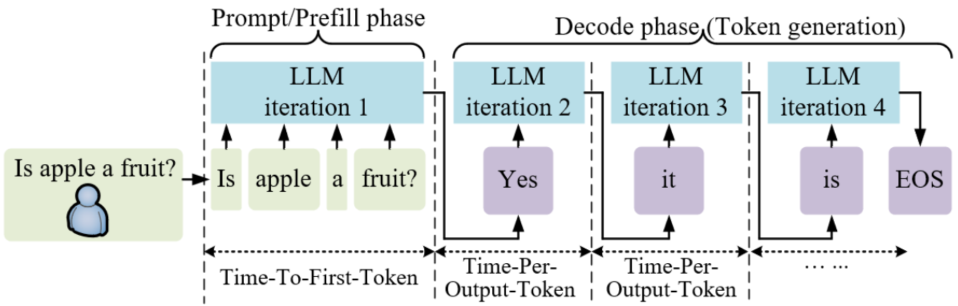
Perplexity quantifies the model's level of surprise when encountering new data to generate a new token. A lower perplexity indicates better performance

Throughput as the total number of tokens (both input and output) processed by the hardware per second.

$$\text{throughput} = \frac{\text{Batch Size} \times (\text{Input} + \text{Output Tokens})}{\text{End-to-End Latency}}$$

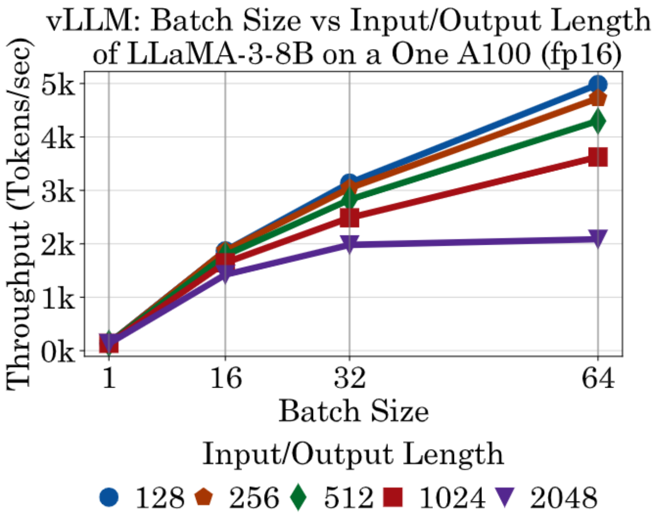
Time to First Token (TTFT) is the amount of time required to produce the first output token after receiving an input prompt.

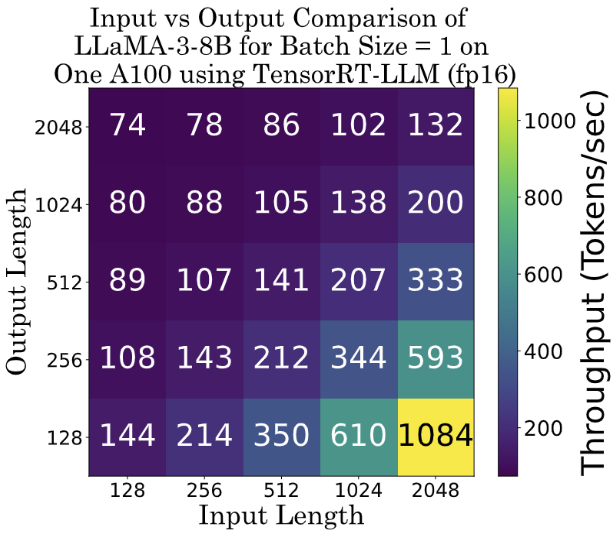
Time Per Output Token refers to the average time interval between generating consecutive tokens.



Impact of Batch Size, Input & Output Length

- **Batch Size:** Throughput increases with increase in the batch size until memory and compute are saturated
  - This is due to the parallel computing nature of hardware to process batches in parallel
- **Input Length:** Throughput increases with increase in the size of input length
  - This is due to the parallel computing of input sequence
- **Output Length:** Throughput decreases with increase in the size of output length
  - This is due to sequential generation of output tokens based on all the previous tokens





Classification of LLM Inference Methods

Type of Inference Optimizations

Efficient Transformer Neural Network Design

- Group Query Attention (GQA)
- Mixture of Experts (MoE)

LLM Algorithm Optimization Methods

- Quantization
- KV Caching
- Speculative Decoding

Hardware and memory management Techniques

- PaggedAttention
- Parallelism Methods
  - Tensor
  - Pipeline
  - Expert
  - Hybrid

Classification of LLM Inference Methods

Type of Inference Optimizations

Efficient Transformer Neural Network Design

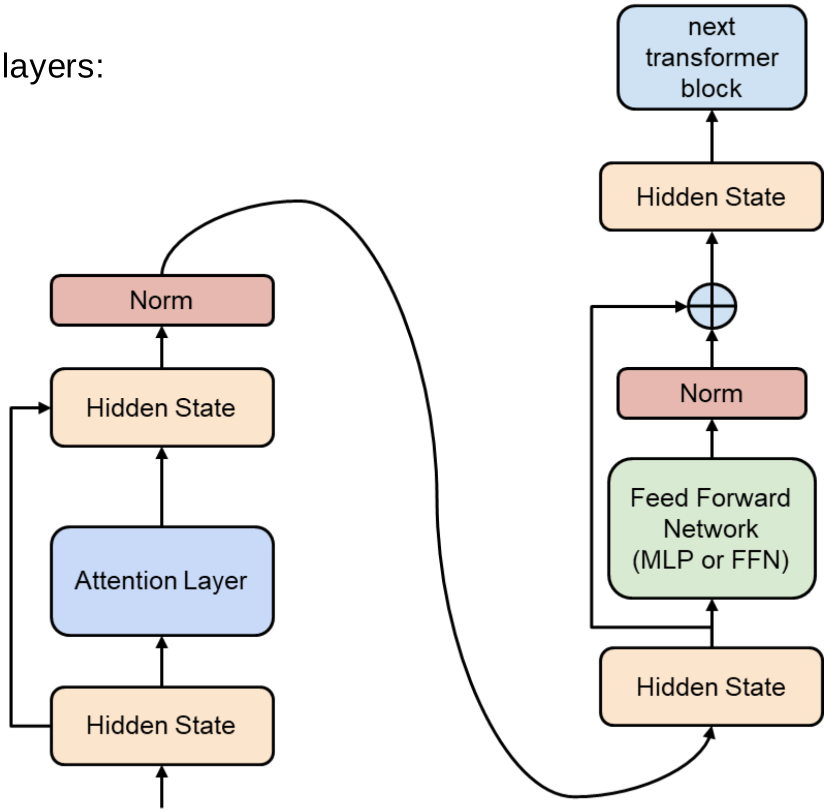
- Group Query Attention (GQA)
- Mixture of Experts (MoE)

Transformer Model

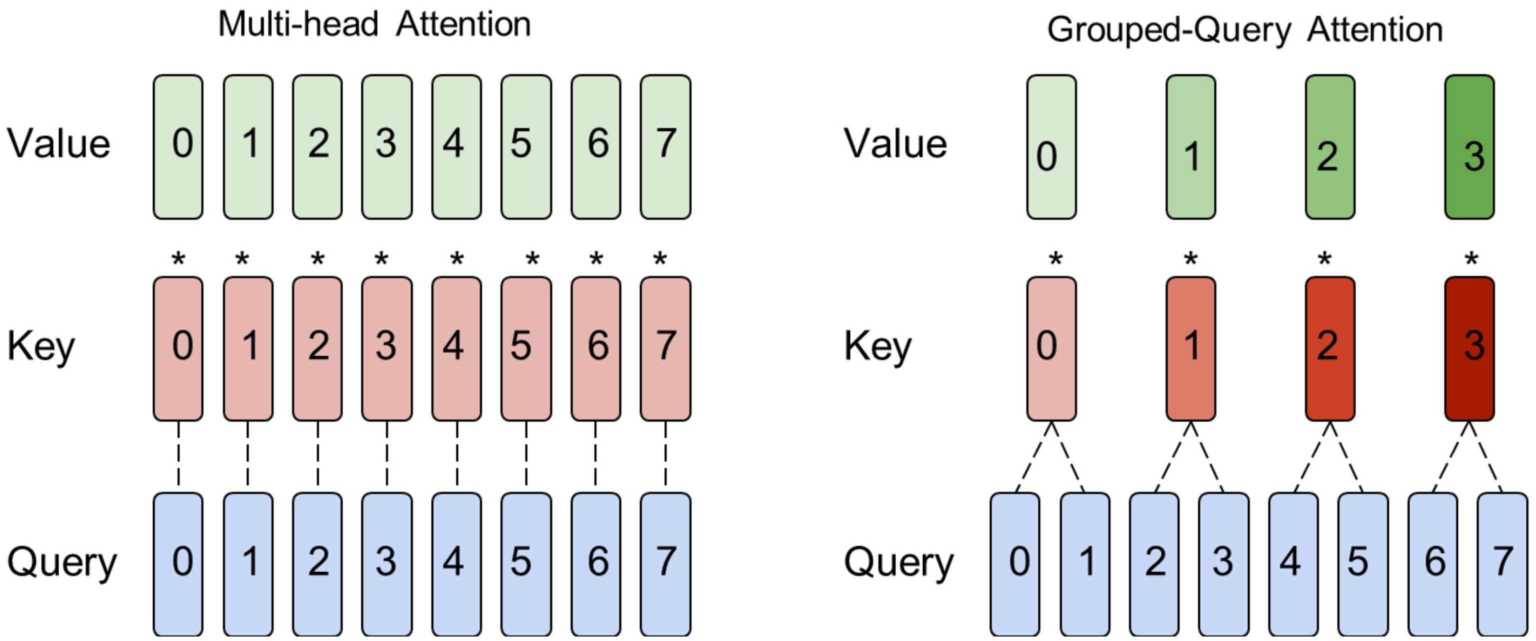
Transformer Neural Network is comprised of two important layers:

1) Attention Layer

2) Feed Forward Network (FFN)



Multi-head Attention (MHA) vs Group Query Attention (GQA)



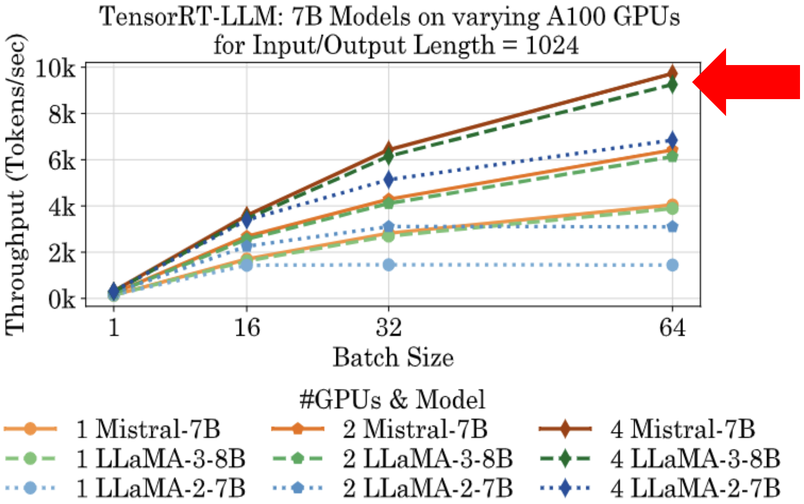
8 Query, Key and Value Heads

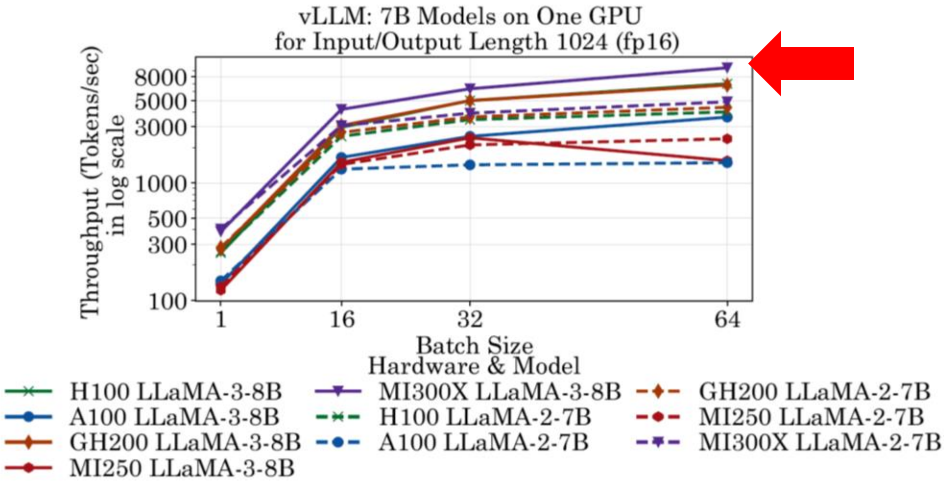
8 Query heads and 4 Key & Value Heads

- GQA reduces memory and compute by a factor of "group size"

MHSA vs GQA Comparison

- MHSA is slower than GQA due to less number of KV heads and KV Cache
- Mistral-7B (**GQA**) > LLaMA-3-8B (**GQA**) > LLaMA-2-7B (**MHSA**)
- vLLM and TensorRT-LLM frameworks demonstrate improved performance using GQA models

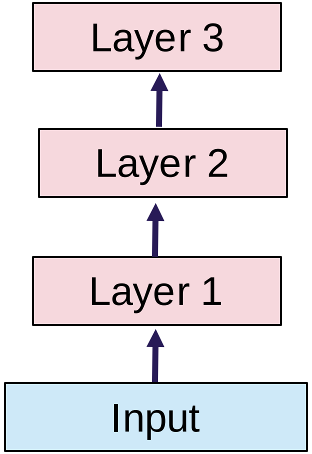




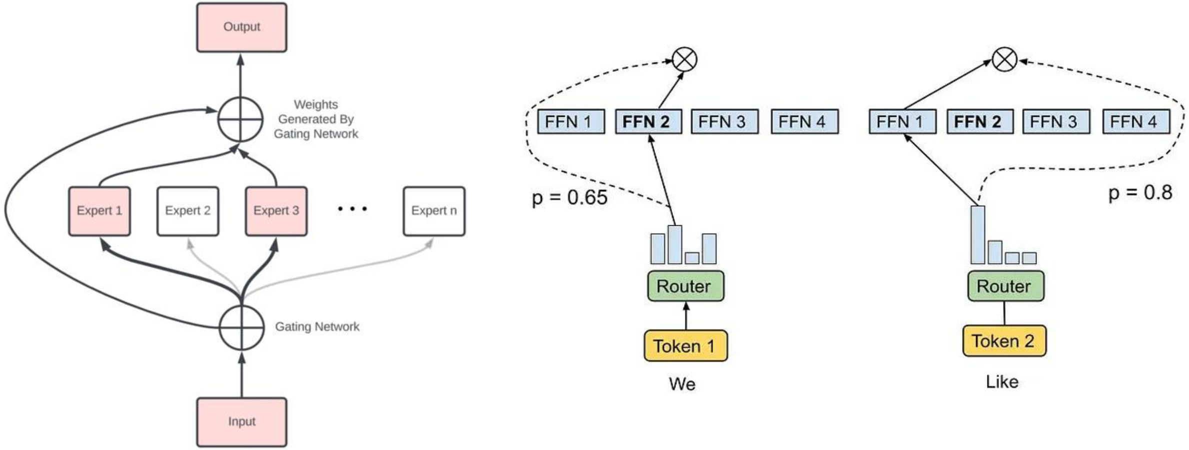
Mixture of Experts (MoE)

- **Dense Models:** Dense LLMs are the traditional layer-by-layer transformer models connected in series
- **Mixture-of-Experts (MoE):** MoE employs a combination of specialized sub-networks called experts and a gating mechanism to selectively activate only a subset of parameters for each input
- In a regular dense LLM, all parameters are active while in MoE only a few model weights are utilized
- Example: Mixtral-8x7B, GPT-MoE 1.8T

Dense Model



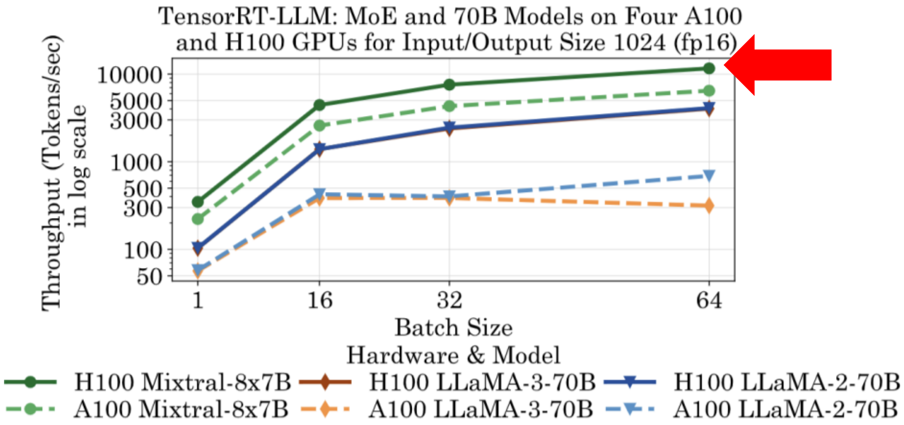
MoE Model

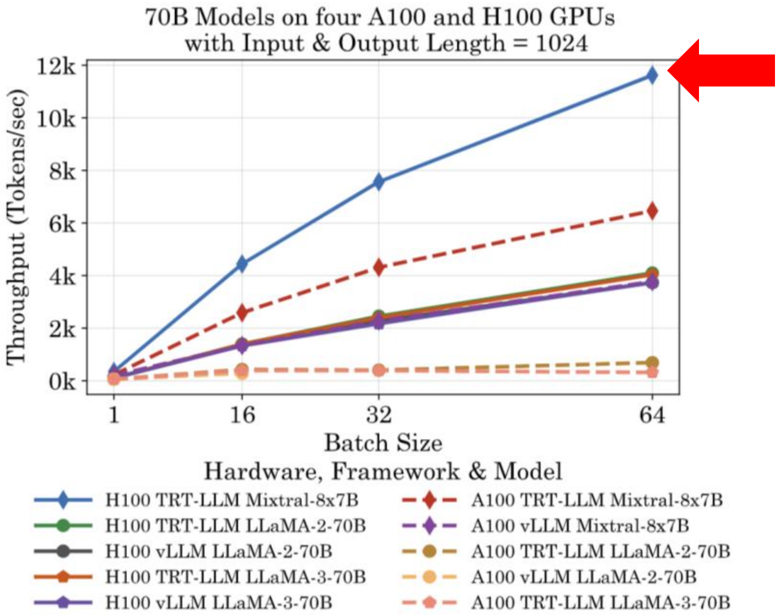


https://deepgram.com/learn/mixture-of-experts-ml-model-guide

MoE vs Dense LLMs

- Mixture of Experts (MoE) models are faster than Dense models for the similar parameter sizes due to less number of active parameters during inference.
- Mixtral-8x7B (**MoE**) > LLaMA-2-70B (**Dense**) > LLaMA-3-70B (**Dense**)
- vLLM and TensorRT-LLM frameworks demonstrate improved performance using MoE models





Classification of LLM Inference Methods

Type of Inference Optimizations

- **Efficient Transformer Neural Network Design**
  - Group Query Attention (GQA)
  - Mixture of Experts (MoE)
- **LLM Algorithm Optimization Methods**
  - Quantization
  - KV Caching
  - Speculative Decoding
- **Hardware and memory management Techniques**
  - PaggedAttention
  - Parallelism Methods
    - Tensor
    - Pipeline
    - Expert
    - Hybrid

Classification of LLM Inference Methods

Type of Inference Optimizations

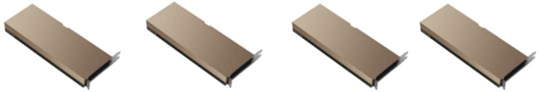
LLM Algorithm Optimization Methods

- Quantization
- KV Caching
- Speculative Decoding

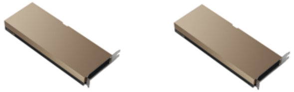
LLM Quantization

LLaMa-3-70B model requires at least :

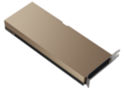
- **FP16:** 140GB memory → 4 x 40GB A100 GPUs



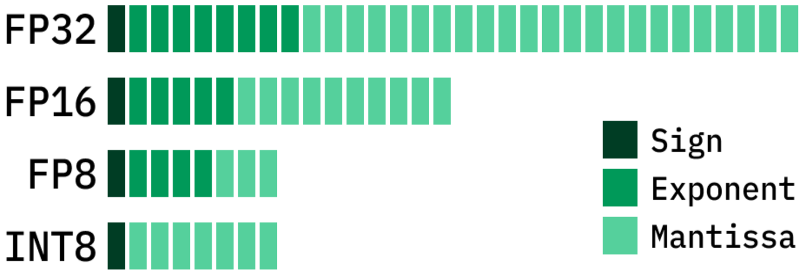
- **INT8:** 70GB memory → 2 x 40GB A100 GPU
- **FP8:** 70GB memory → 2 x 40GB A100 GPU



- **INT4:** 35GB memory → 1 x 40GB A100 GPU



- LLM Quantization Methods: GPTQ, AWQ



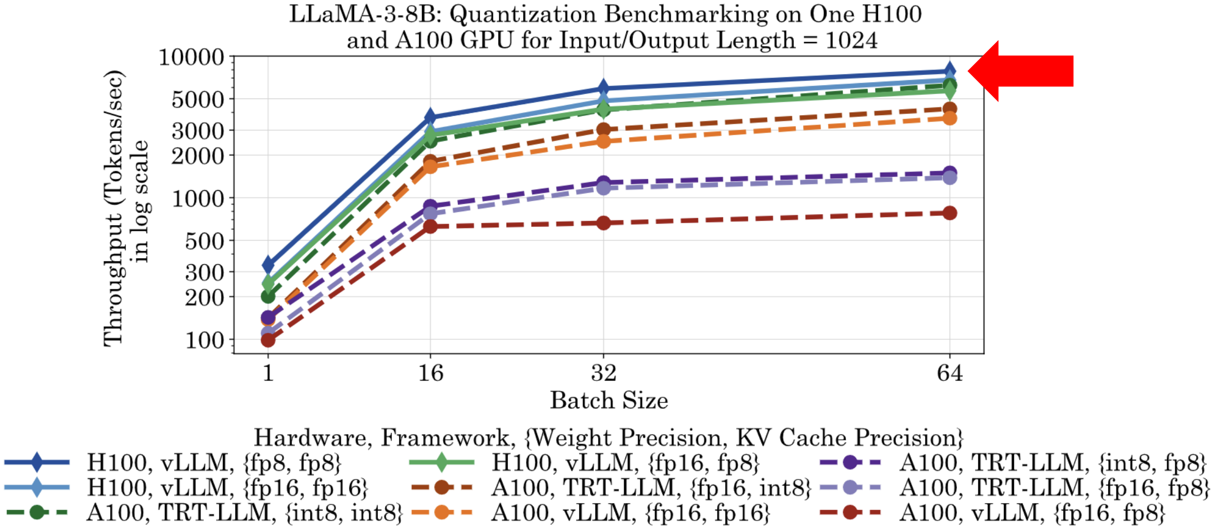
Argonne NATIONAL LABORATORY

Quantization Comparison

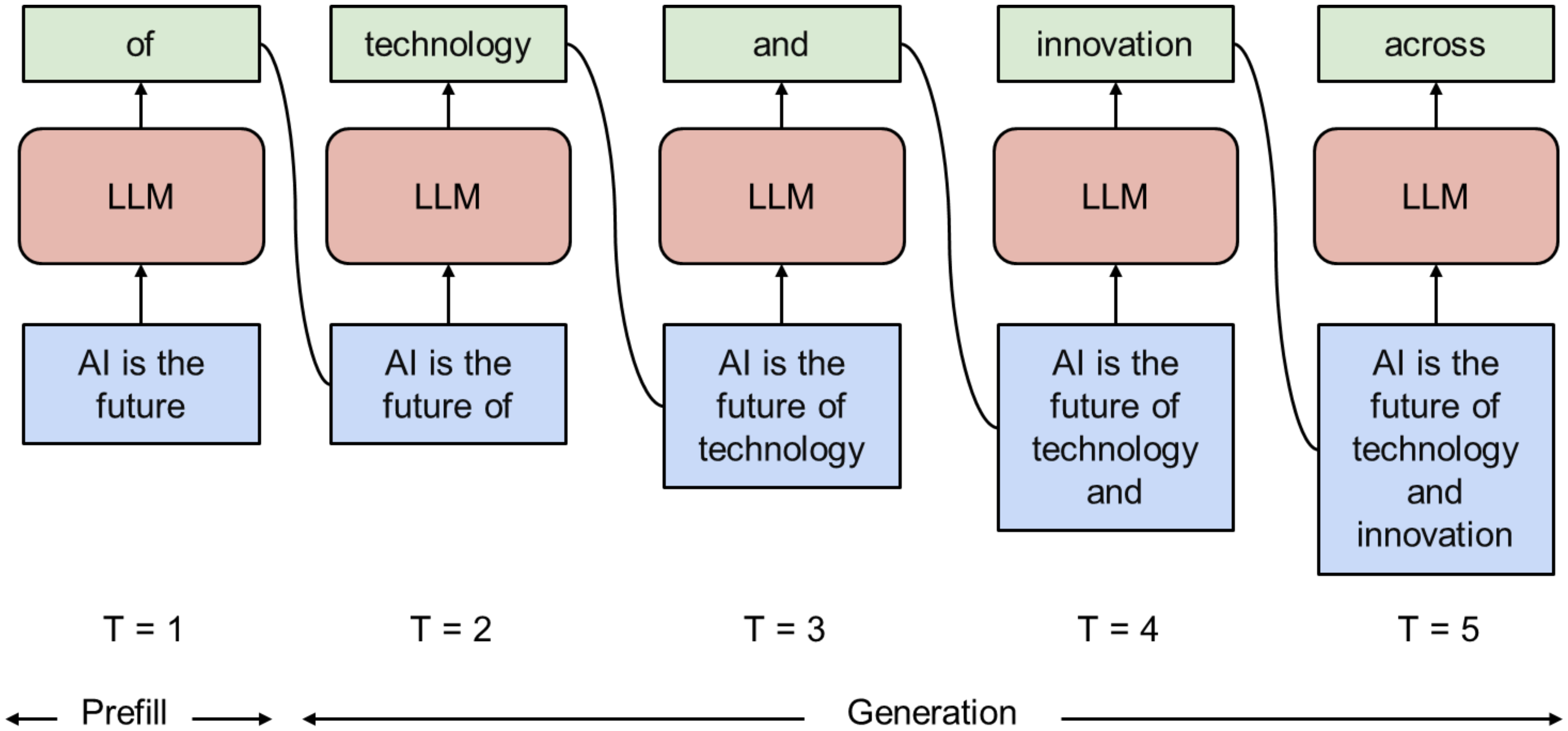
- Nvidia A100 Support:
  - FP32, Fp16, Int8
- Nvidia H100 Support:
  - FP32, FP16, Int8, FP8

- Quantization of the parameters (weights and activations) boosts the throughput of LLMs

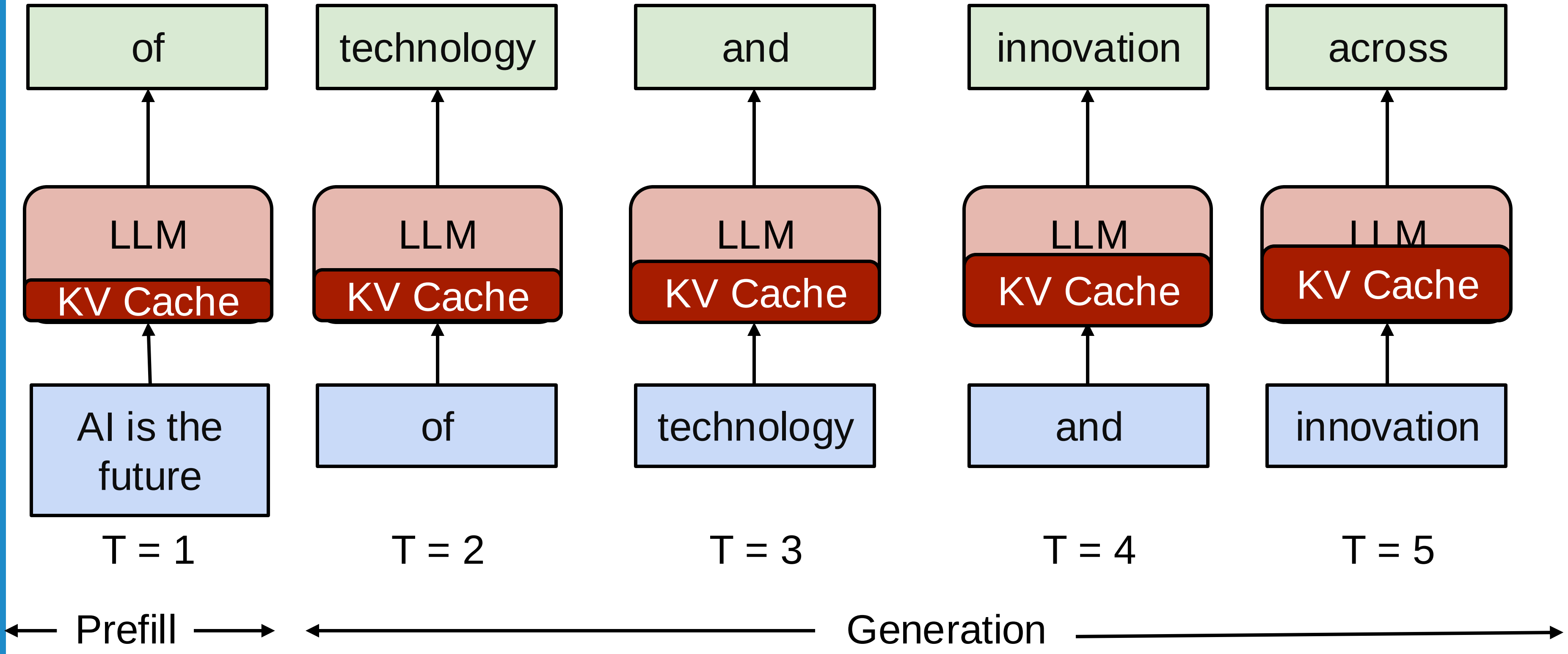
- FP8 on H100 > Int8 on A100 for LLaMA-3-8B



LLM Inference

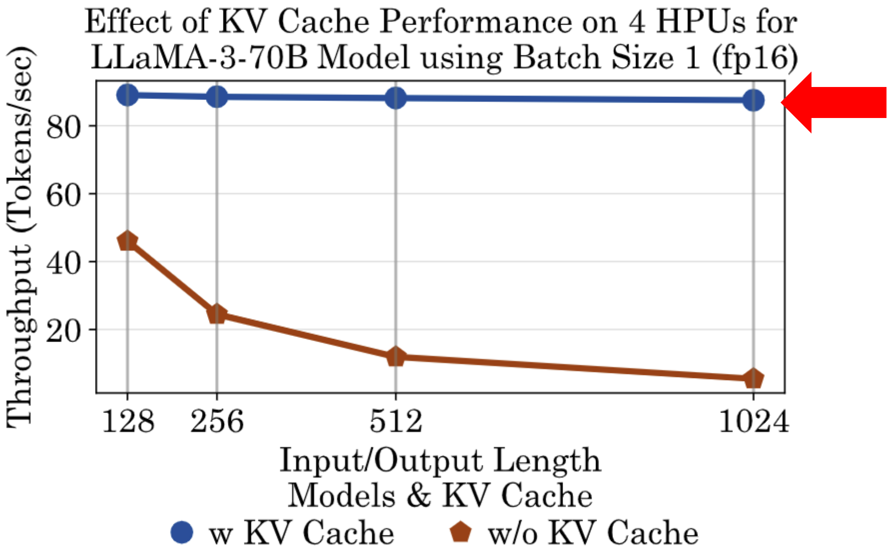


LLM Inference with KV Cache



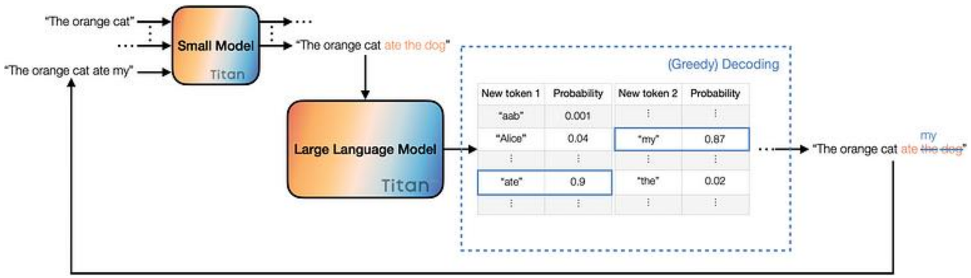
Impact of KV Cache

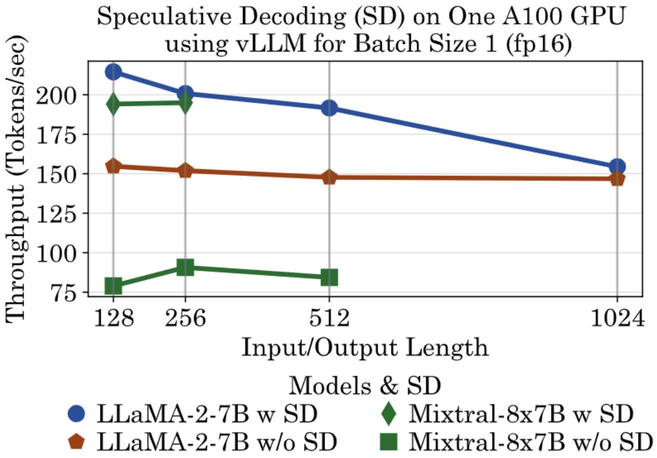
- KV Cache boosts the LLM Inference and becomes significant for longer sequence lengths
- VLLM, TensorRT-LLM, Deepspeed supports KV caching by default
- No option to not use KV Cache in State-of-the-art frameworks
- We can unset KV Cache in Habana Processing Unit (HPU)



Speculative Decoding

- Speculative Decoding is a widely used technique to speed up inference for LLMs without greatly compromising the output quality
- During inference, the speculative decoding method utilizes a smaller draft model (Eg: OPT-125M) to generate speculative tokens and then uses the larger LLM (LLaMA-2-7B) to verify those draft tokens.
- Both draft model and the main model should have the same vocab size





https://www.titanml.co/glossary/speculative-decoding

Argonne NATIONAL LABORATORY

Classification of LLM Inference Methods

Type of Inference Optimizations

- Efficient Transformer Neural Network Design
  - Group Query Attention (GQA)
  - Mixture of Experts (MoE)
- LLM Algorithm Optimization Methods
  - Quantization
  - KV Caching
  - Speculative Decoding
- Hardware and memory management Techniques
  - PaggedAttention
  - Parallelism Methods
    - Tensor
    - Pipeline
    - Expert
    - Hybrid

Classification of LLM Inference Methods

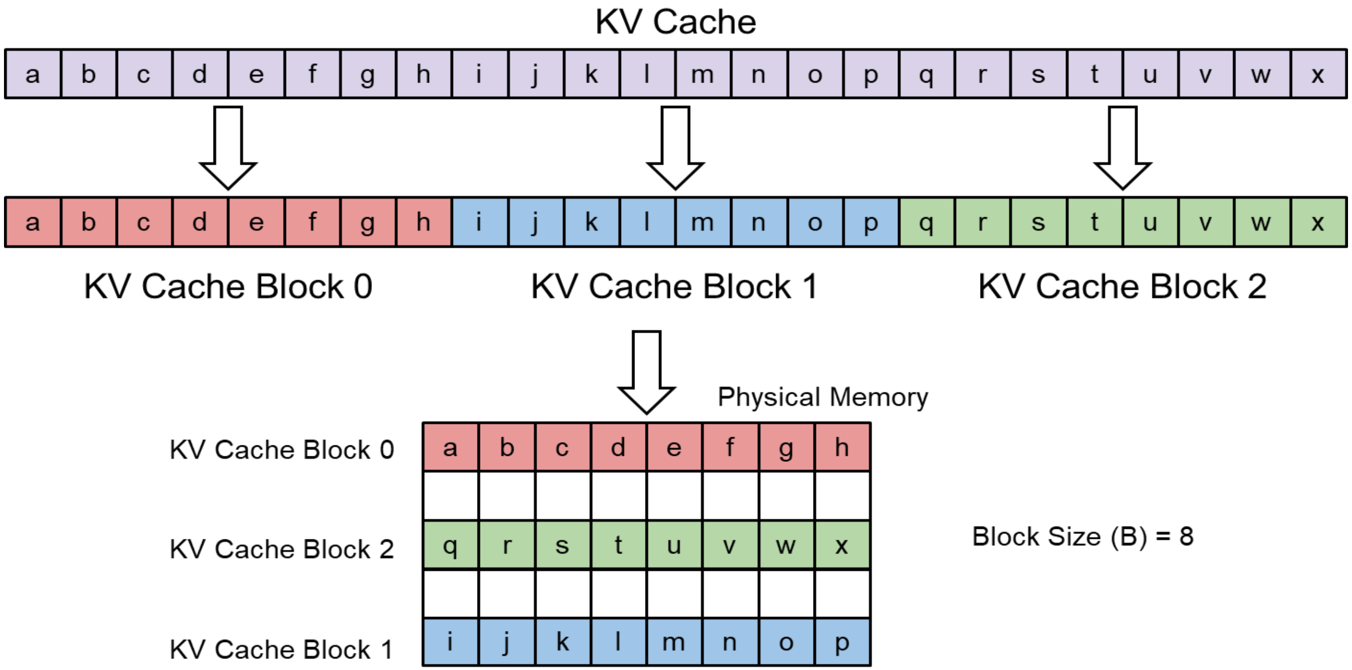
Type of Inference Optimizations

Hardware and memory management Techniques

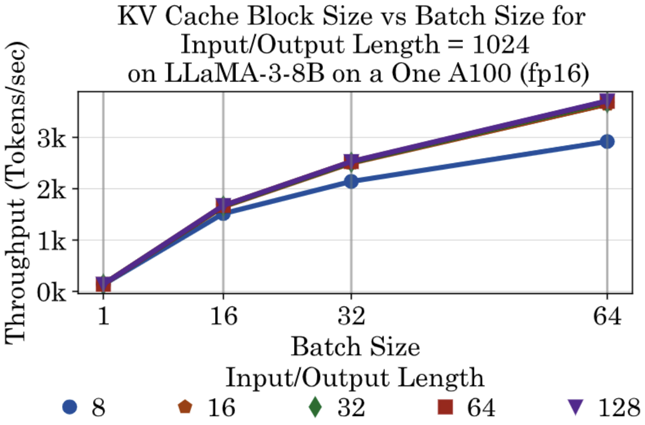
- PaggedAttention
- Parallelism Methods
  - Tensor
  - Pipeline
  - Expert
  - Hybrid

PagedAttention

- Paged Attention partitions the KV cache of each sequence into smaller, more manageable "**pages**" or "**blocks**". Each block contains key-value vectors for a fixed number of tokens.

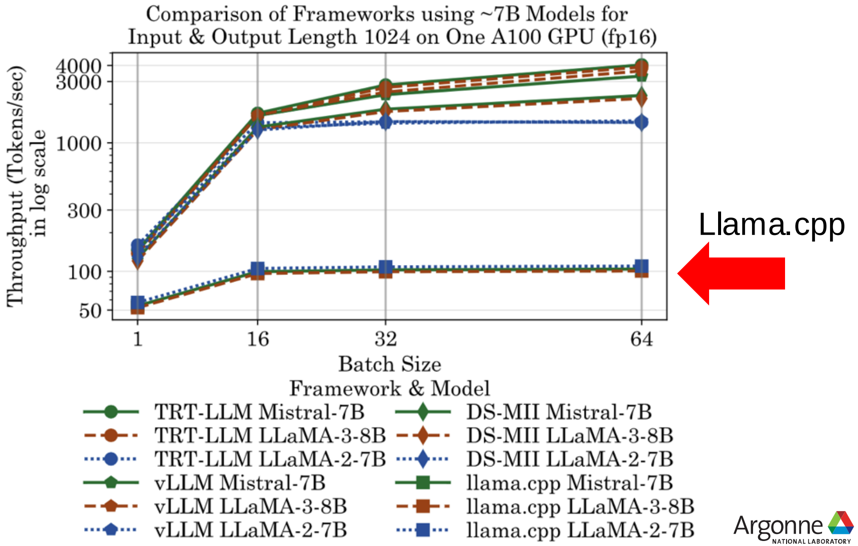


Impact of PagedAttention



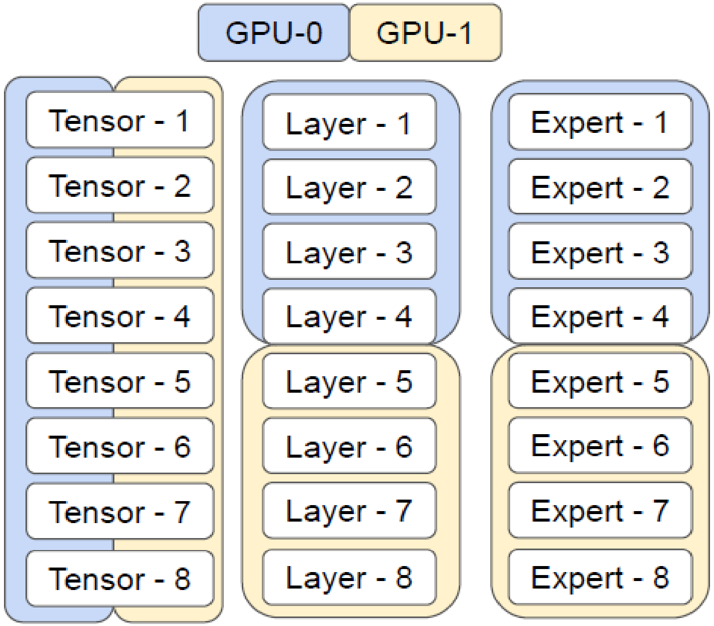
- Low KV Cache block sizes like 2,4 and 8 hurts the inference performance
- Block size greater than or equal to 16 produces optimal throughput

- Frameworks supporting PagedAttention perform much better than the frameworks which do not have efficient implementation of KV Cache Blocking
- For example, TensorRT-LLM, vLLM and Deepspeed have support for PagedAttention and perform better llama.cpp



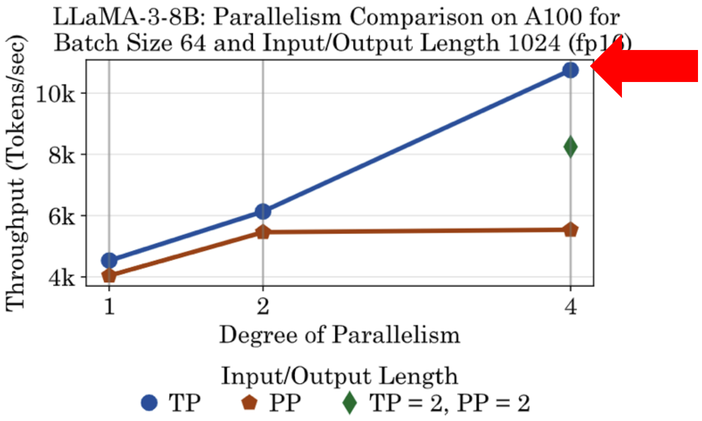
Parallelism Techniques

- **Tensor Parallelism (TP)**
  - Distributes the weight tensor of a layer across multiple devices.
  - The devices communicate with each other to share the input and output activations.
- **Pipeline Parallelism (PP)**
  - Divides the model into different layers, and each device computes its assigned layers and passes the output to the next device in the pipeline.
- **Expert Parallelism (EP)**
  - Distributes the experts of the MoE model across multiple devices
- **Hybrid Parallelism (EP)**
  - Combines one or more parallelism techniques

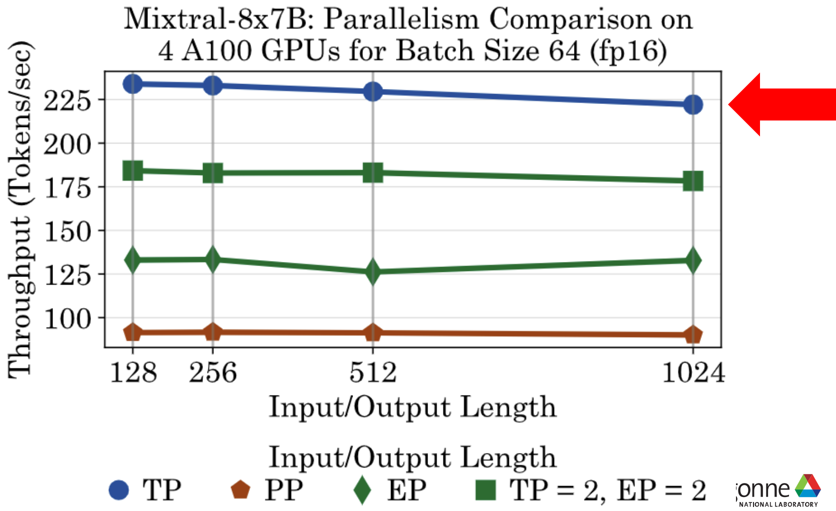


(a) Tensor Parallelism (b) Pipeline Parallelism (c) Expert Parallelism

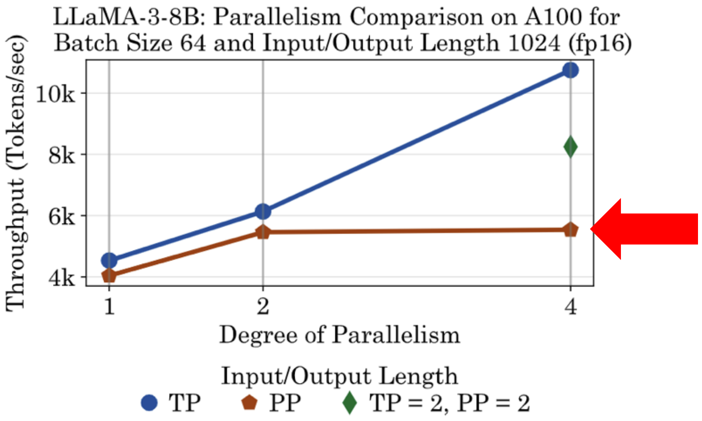
Parallelism Comparison – Tensor Parallelism



- Tensor Parallelism (within a single node) performs best due to better device utilization

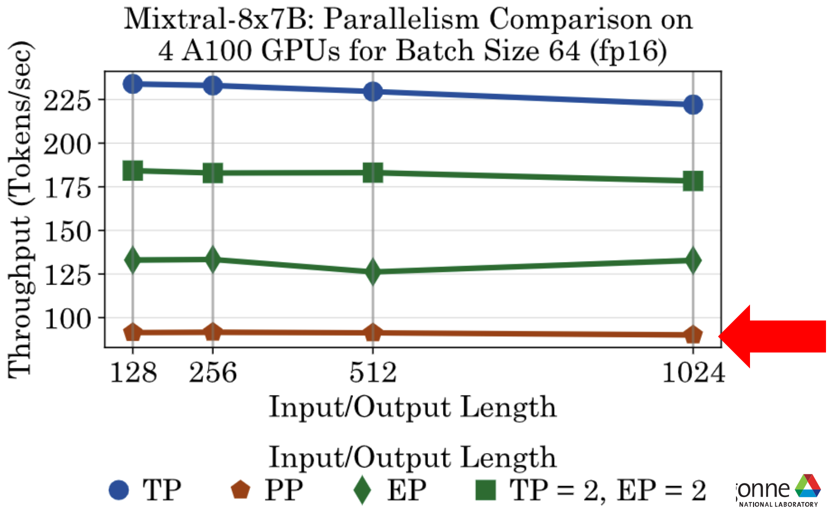


Parallelism Comparison – Pipeline Parallelism

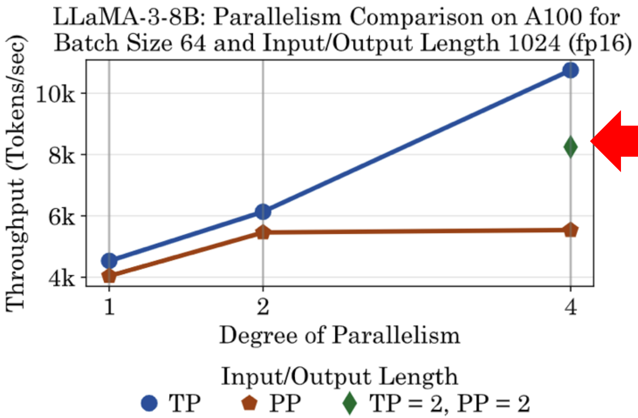


- Tensor Parallelism (within a single node) performs best due to better device utilization

- Pipeline Parallelism (within a single node) has the least performance due to least utilization as only one device is active

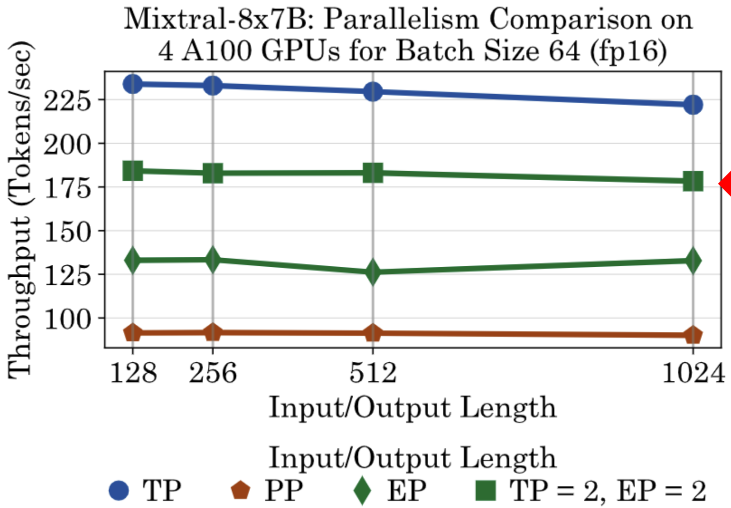


Parallelism Comparison – Hybrid Parallelism



- Tensor Parallelism (within a single node) performs best due to better device utilization

- Hybrid Pipeline Parallelism (a combination of Tensor and Pipeline Parallelism) offers flexibility and the performance is between the two methods

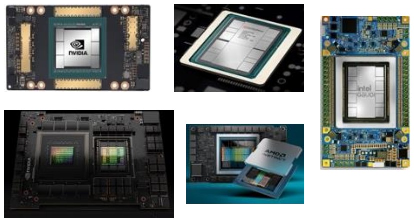


LLM-Inference-Bench: Bridging LLMs, Accelerators and Frameworks

Open source LLMs
LLaMA, Mistral, Qwen

LLM-Inference-Bench

AI Accelerators
Nvidia, AMD GPUs,
SambaNova SN40L,
Habana Gaudi



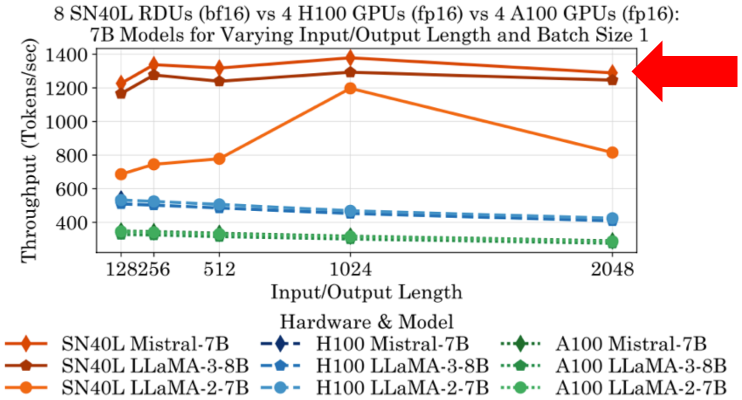
vLLM

TensorRT-LLM

DeepSpeed MII

LLaMA C++

Model Comparison

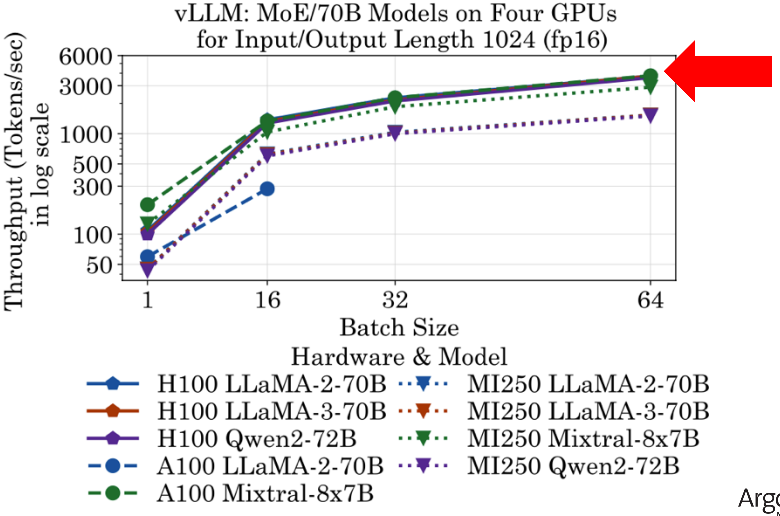


7B Models

- Mistral-7B performs better than LLaMA-3-8B due to one billion less parameters
- LLaMA-3-8B performs better than LLaMA-2-7B due to Group Query Attention (GQA) despite one billion more parameters

Large Models

- Mixtral-8x7B performs better than LLaMA-3-70B and LLaMA-2-70B due to mixture of experts layer utilizing less active parameters
- LLaMA-2-70B performs better than LLaMA-3-70B due to smaller vocabulary size

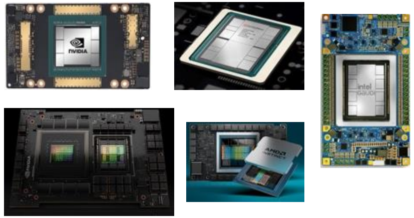


LLM-Inference-Bench: Bridging LLMs, Accelerators and Frameworks

Open source LLMs
LLaMA, Mistral, Qwen

LLM-Inference-Bench

AI Accelerators
Nvidia, AMD GPUs,
SambaNova SN40L,
Habana Gaudi



vLLM

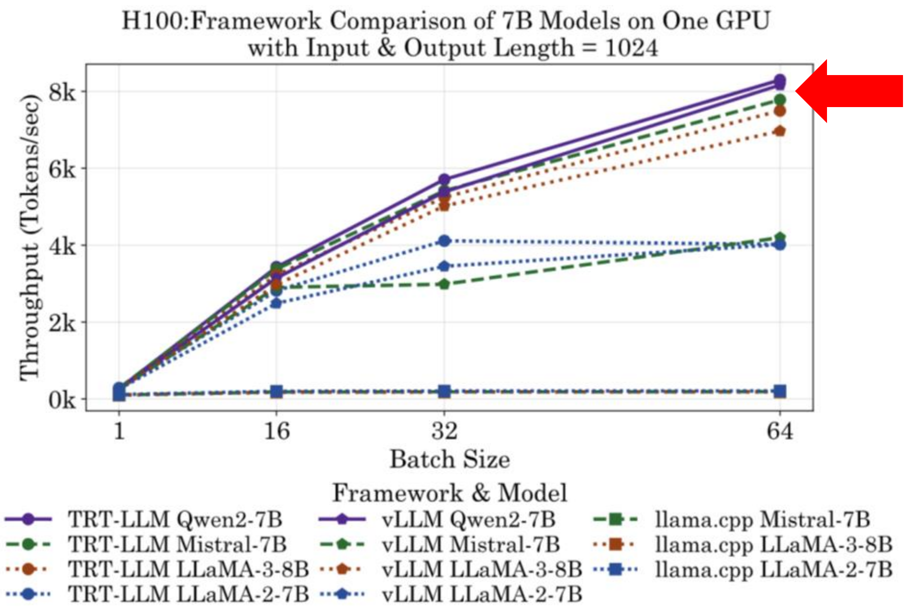
TensorRT-LLM

DeepSpeed MII

LLaMA C++

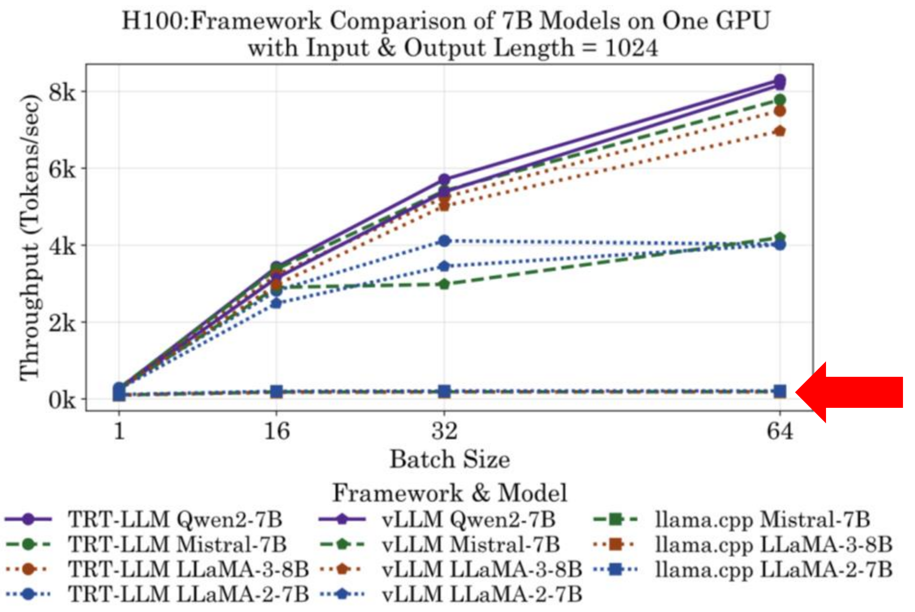
Framework Comparison

- TensorRT-LLM attains the highest throughput on Nvidia GPUs across different Large Language Models
- vLLM is the second best performer
- llama.cpp shows least performance due to lack of efficient transformer algorithm methods such as GQA and PaggedAttention



Framework Comparison

- TensorRT-LLM attains the highest throughput on Nvidia GPUs across different Large Language Models
- vLLM is the second best performer
- llama.cpp shows least performance due to lack of efficient transformer algorithm methods such as GQA and PaggedAttention

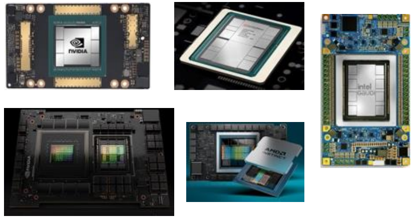


LLM-Inference-Bench: Bridging LLMs, Accelerators and Frameworks

Open source LLMs
LLaMA, Mistral, Qwen

LLM-Inference-Bench

AI Accelerators
Nvidia, AMD GPUs,
SambaNova SN40L,
Habana Gaudi



vLLM

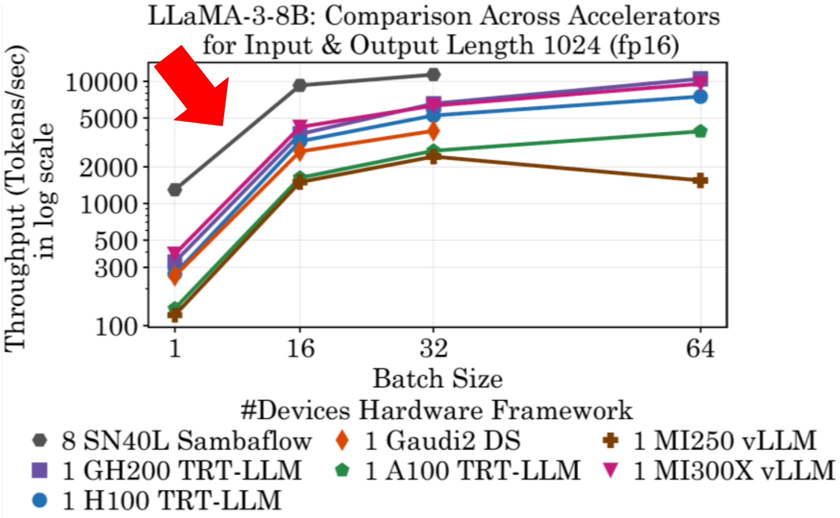
TensorRT-LLM

DeepSpeed MII

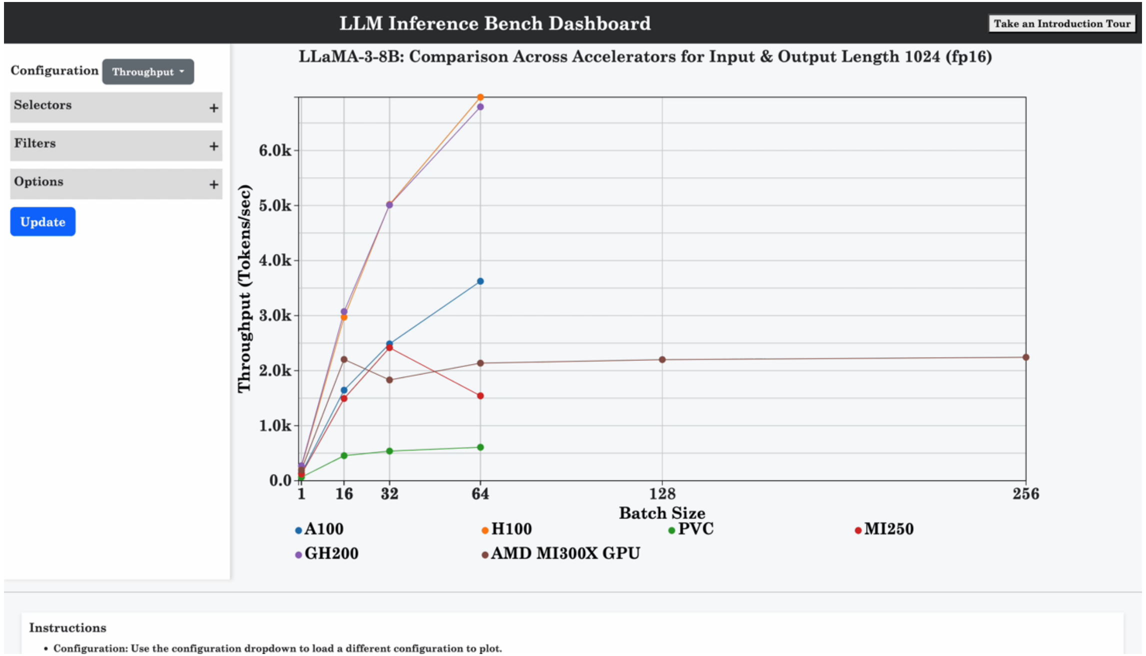
LLaMA C++

Accelerator Comparison

- SambaNova SN40L achieves has the best performance among all the accelerators we benchmarked
  - However, as of July 2024, the maximum batch size SN40L supports is 32
- Nvidia GH200 > H100 > A100 (in terms of throughput)
- MI300X and GH200 are comparable
- Habana Gaudi's performance is between A100 and H100
- The performance of AMD MI250 saturates for large batch sizes



Performance Dashboard and Code



Dashboard

Code

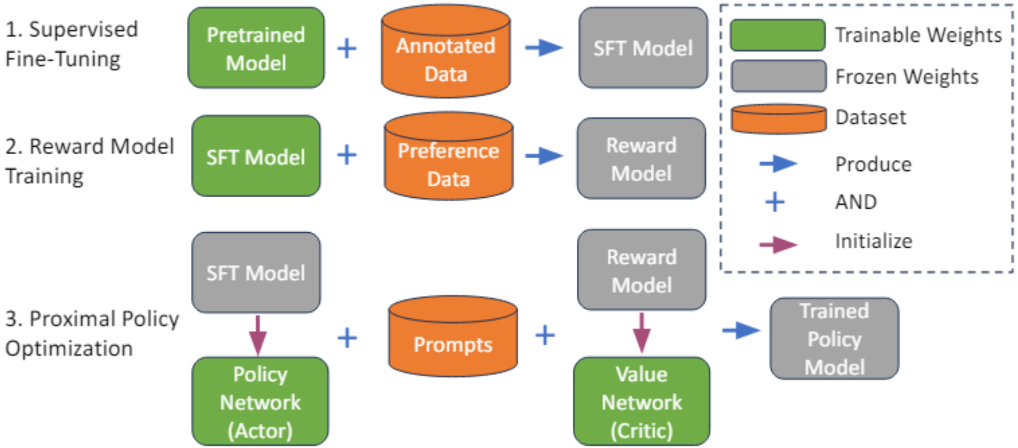
Explore all the experimental results with interactive dashboard

https://github.com/argonne-lcf/LLM-Inference-Bench

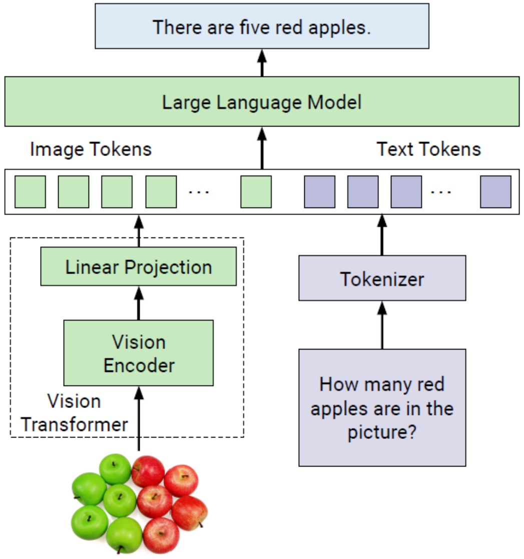
Argonne NATIONAL LABORATORY

Future Works

LLaMA-Finetune-Bench: Benchmarking Finetuning of Large Language and Multimodal Models on AI Accelerators



Multimodal Models



Thank You

Any Questions?

- This research used resources of the Argonne Leadership Computing Facility, a U.S. Department of Energy (DOE) Office of Science user facility at Argonne National Laboratory and is based on research supported by the U.S. DOE Office of Science-Advanced Scientific Computing Research Program, under Contract No. DE-AC02-06CH11357.

- We gratefully acknowledge the computing resources provided and operated by the Joint Laboratory for System Evaluation (JLSE) at Argonne National Laboratory.

- We would like to thank collaborators at NVIDIA, AMD, SambaNova, Intel Habana

Argonne NATIONAL LABORATORY

U.S. DEPARTMENT OF ENERGY