LLMShare: Optimizing LLM Inference Serving with Hardware Architecture Exploration

Hongduo Liu[1], Chen Bai[2], Peng Xu[1], Lihao Yin[3], Xianzhi Yu[3], Hui-Ling Zhen[3], Mingxuan Yuan[3], Tsung-Yi Ho[1], Bei Yu[1]

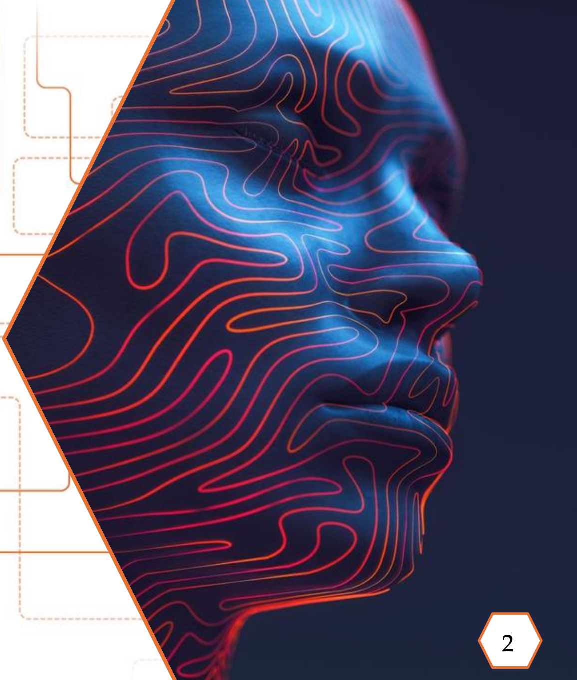
[1]CUHK [2]HKUST [3]Huawei Technologies

62 THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA SIGDA

Outline

1. Introduction
2. Algorithms
3. Experimental Results



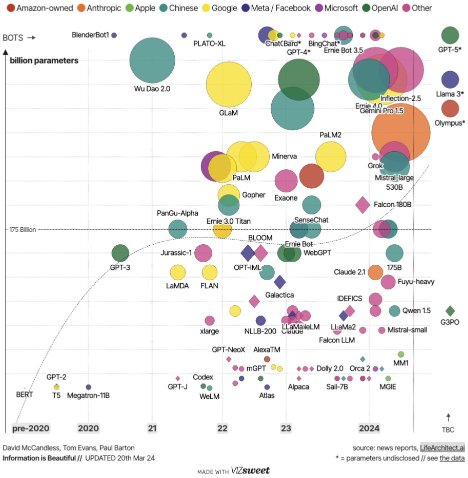
Introduction

THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY

The Rise of LLMs

LLMs are getting smarter, but also larger



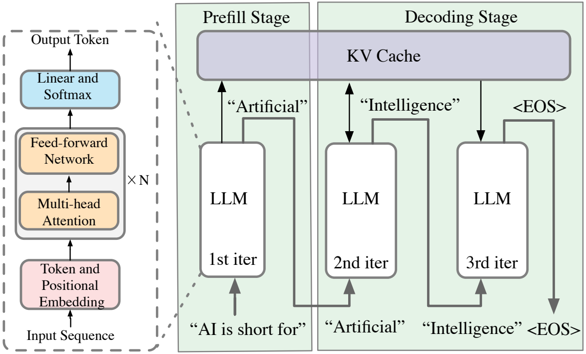
Challenges of Serving LLMs

For commercial applications, serving LLMs can be challenging:

- Requires substantial hardware resources
- Strict service-level objectives
  - End-to-end latency
  - Time to first token
  - Time between tokens
  - Throughput

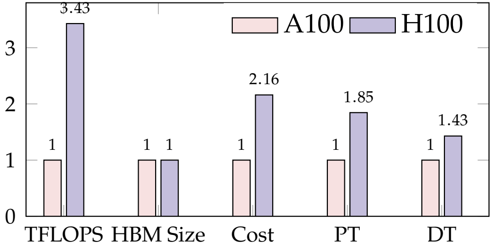
LLM Inference Process Overview

- **Prefill Phase:** Process the entire input prompt to set up context.
- **Decoding Phase:** Generate output tokens autoregressively using KV cache.
- Prefill and Decoding pose different computation and memory requirements



Motivation

- PD Disaggregation: prefill and decoding are handled by sperate machines
- Mismatches between hardware capabilities and P/D requirements still exist



Comparison of NVIDIA A100 and H100 cluster with 8 GPUs on Llama-70b without batching. ‘PT’ denotes prefill phase throughput, and ‘DT’ denotes decoding phase throughput.

Research Objectives

`Can we find a hardware configuration to achieve a better performance-cost tradeoff?`

1. Model the performance and cost of different hardware configurations
   - A simulator
2. Find a systematic way to explore optimal hardware configurations
   - A design space exploration algorithm

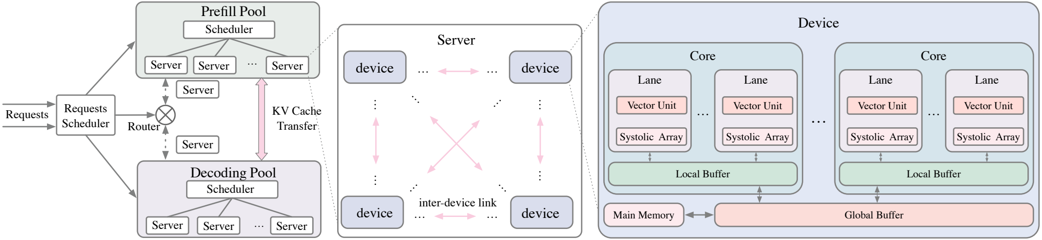
Algorithms

THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY

LLM Serving System Modeling

- The system is divided into two distinct pools
- A dynamic scheduler reallocates servers between pools based on request load
- The architecture of the device models mainstream accelerators like GPUs and TPUs[1]



[1] **Hengrui Zhang et al. (2024). "LLMCompass: Enabling Efficient Hardware Design for Large Language Model Inference". In: *ISCA*. IEEE, pp. 1080–1096.**

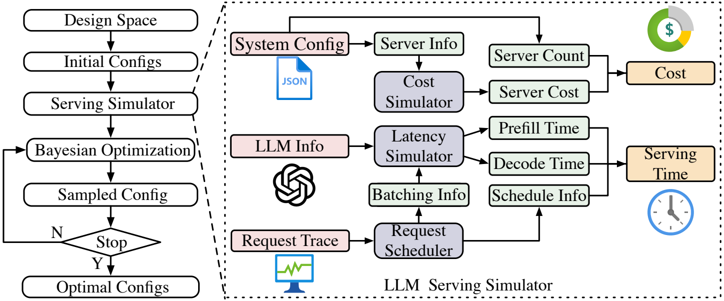
Design Space

Parameter	Notation	Value Range	#
Server Count	sc	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	10
Device Count	dc	4, 8, 12, 16	4
Link Count Per Device	lc	6, 12, 18, 24	4
Main Memory (GB)	mm	40, 64, 80, 96, 112, 128	6
Global Buffer (MB)	gb	20, 30, 40, 50, 60, 70, 80, 90, 100, 110	10
Core Count	cc	72, 96, 108, 132, 156, 180	6
Local Buffer (KB)	lb	64, 128, 192, 256, 320, 384, 448, 512	8
Lane Count	lc	1, 2, 4, 8	4
Array Height	ah	16, 32, 64, 128	4
Vector Width	vw	16, 32, 64, 128	4

Table: Design space of the prefill and decoding pools. The entire design space of an LLM serving system is nearly 9×10^{14}.

LLMShare Overview

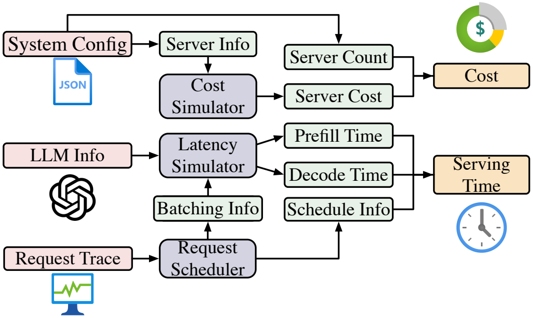
- A LLM serving simulator to get serving time and total cost.
- A Bayesian optimization framework to find optimal serving system configurations



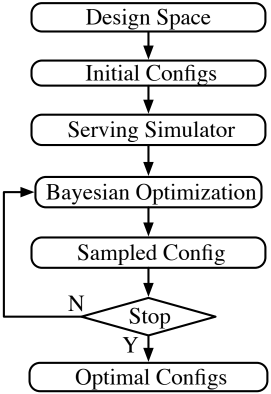
The overview of LLMShare.

Simulator Framework Overview

- **Inputs:**
  - Serving system configuration (number of servers, device specs, etc.).
  - LLM information (e.g., number of layers, attention heads).
  - Request trace (arrival timings, input/output token sizes).
- **Outputs:** Serving time and total cost.



Design Space Exploration Framework



1. Initialize with a set of sample designs.
   - **Memory-Centric Initialization**
2. Get simulated cost and serving time by the simulator
3. Fit a surrogate model
   - **Deep Tree Kernel**
4. Select the most promising design by optimizing the acquisition function
5. Update the surrogate model using the selected design

Memory-Centric Initialization (MCI)

Algorithm 1 Memory-Centric Initialization

Input:
- $\mathcal{U}$: unsampled design space with n configurations;
- t: total number of initial configurations to select;
- u: number of groups used during sampling.

Output: $\mathcal{D}_x$ with $|\mathcal{D}_x| = t$ ▷ Selected initial designs

Compute the total main memory size for each design:
for $i = 1$ to n **do**

$$c_i = \boldsymbol{x}^p_{i,\text{sc}} \cdot \boldsymbol{x}^p_{i,\text{dc}} \cdot \boldsymbol{x}^p_{i,\text{mm}} + \boldsymbol{x}^d_{i,\text{sc}} \cdot \boldsymbol{x}^d_{i,\text{dc}} \cdot \boldsymbol{x}^d_{i,\text{mm}};$$

Determine percentiles: $P = \left\{ \frac{100 \times j}{u} \,\middle|\, j = 0, 1, \ldots, u \right\}$;
Compute bin edges for the percentiles of $\{c_i\}_{i=1}^{n}$:

$$B = \left\{ b_j = \text{Percentile}(\{c_i\}_{i=1}^{n}, p_j) \,\middle|\, j = 0, 1, \ldots, u \right\};$$

Compute base sample count per group: $q \leftarrow \left\lfloor \frac{t}{u} \right\rfloor$;
Compute remainder: $r \leftarrow t \bmod u$;
Initialize $\mathcal{D}_x \leftarrow \emptyset$;
for $j = 1$ to u **do**
    $\mathcal{G}_j = \left\{ i \,\middle|\, b_{j-1} \leq c_i < b_j \right\}$;
    if $j \leq r$ **then**
        $s_j \leftarrow q + 1$;
    else
        $s_j \leftarrow q$;
    Select s_j samples from $\mathcal{G}_j$: $\mathcal{S}_j = \textbf{TED}(\mathcal{G}_j, s_j)$;
    $\mathcal{D}_x \leftarrow \mathcal{D}_x \cup \mathcal{S}_j$;
return $\mathcal{D}_x$

- Memory capacity can largely affect throughput and cost
- Divide the design space into groups based on memory capacity
- Sample in each group using traditional sampling method like transductive experimental design (TED)

Surrogate Model

- The Gaussian Process is used as the surrogate model
- A GP is specified by its mean function $m(\boldsymbol{x})$ and kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$:

$$f(\boldsymbol{x}) \sim \mathcal{GP}\left(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')\right). \tag{1}$$

- $k(\boldsymbol{x}, \boldsymbol{x}')$ determines how function values vary when inputs changes
- Kernel function is important for the expressiveness of the surrogate model

Deep Tree Kernel (DTK)

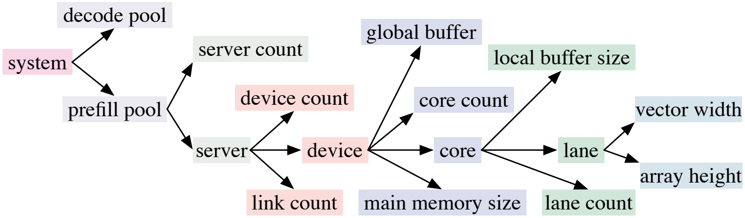
- Each configuration is represented as a tree
- The embedding of a node v is computed based on the embeddings of its child nodes $\{u_1, u_2, \ldots, u_{k_v}\}$

$$\boldsymbol{h}_v = \phi_v \left(\text{concat} \left(\boldsymbol{h}_{u_1}, \boldsymbol{h}_{u_2}, \ldots, \boldsymbol{h}_{u_{k_v}} \right); \theta_v \right), \tag{2}$$

- The deep tree kernel is defined as

$$k_t(\boldsymbol{x}_i, \boldsymbol{x}_j) = k\left(\boldsymbol{h}^{(i)}_{\text{system}}, \boldsymbol{h}^{(j)}_{\text{system}} \right), \tag{3}$$

where $\boldsymbol{x}_i$ is the feature vector of configuration i and k is a traditional kernel function



Multi-Objective Bayesian Optimization

- We want to optimize two conflicting objectives
  - Multi-Objective Bayesian Optimization is used
- Expected Hypervolume Improvement (EHVI)[2] is adopted as our acquisition function

$$\mathrm{EHVI}(\mathbf{x}_*) = \int_{\mathbf{y}} \max\left(\mathrm{HV}(\mathcal{P} \cup \{\mathbf{y}\}) - \mathrm{HV}(\mathcal{P}),\, 0\right) p(\mathbf{y} \mid \mathbf{x}_*, \mathcal{D})\, d\mathbf{y}, \tag{4}$$

- Search the candidate design $\mathbf{x}_*$ that maximizes the EHVI:

$$\boldsymbol{x}_* = \arg\max_{\boldsymbol{x}\in\mathcal{X}} \mathrm{EHVI}(\boldsymbol{x}). \tag{5}$$

[2] **Samuel Daulton, Maximilian Balandat, and Eytan Bakshy (2020). "Differentiable expected hypervolume improvement for parallel multi-objective Bayesian optimization".** In: *NIPS* 33, pp. 9851–9864.

Experimental Results

THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY

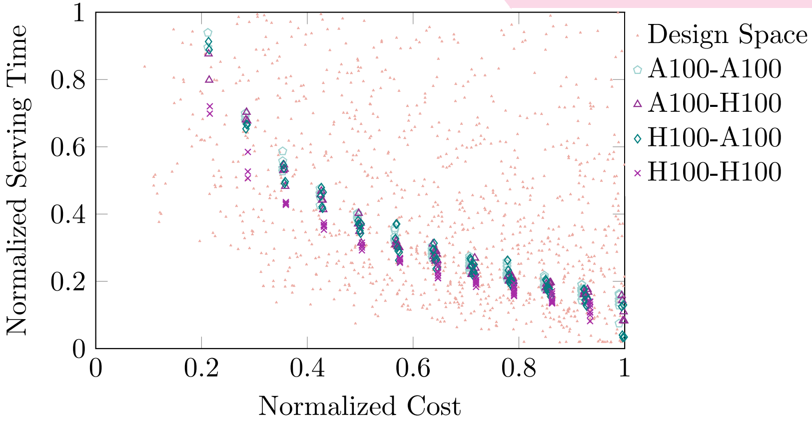
Experimental Setup

- **Underlying LLM:** GPT-3 175B.
- **Serving Trace:** 2454 requests in 2 mins.
  - The distribution of token sizes is derived from a Microsoft Azure production trace
- **Cost and Prefill/Decoding Time Simulation:** LLMCompass[3], which only has 5% simulation error.
- **Request Scheduler:** Splitwise[4]
- **Design Space:** Subset of 1,055 configurations sampled from the whole design space.
- **DSE Process:** Initialization with 10 samples and perform 20 optimization iterations.

[3] Hengrui Zhang et al. (2024). "LLMCompass: Enabling Efficient Hardware Design for Large Language Model Inference". In: *ISCA*. IEEE, pp. 1080–1096.

[4] Pratyush Patel et al. (2024). "Splitwise: Efficient generative llm inference using phase splitting". In: *ISCA*. IEEE, pp. 118–132.

Verification of Motivation



The effectiveness of LLM serving system design space exploration

Results: Comparison of different algorithms

Algorithms	Normalized ADRS	Hypervolume (10^8)
SVR[5]	0.1811	4.9593
DAC'16[6]	0.1718	4.9714
ASPDAC'20[7]	0.1805	4.9513
ICCAD'21[8]	0.2059	4.8134
LLMShare	**0.1589**	**5.0552**

[5] **Mariette Awad et al. (2015). "Support vector regression".** In: *Efficient learning machines: Theories, concepts, and applications for engineers and system designers*, pp. 67–80.

[6] **Dandan Li et al. (2016). "Efficient design space exploration via statistical sampling and AdaBoost learning".** In: *DAC*, pp. 1–6.

[7] **Zhiyao Xie et al. (2020). "FIST: A feature-importance sampling and tree-based method for automatic design flow parameter tuning".** In: *ASP-DAC*. IEEE, pp. 19–25.

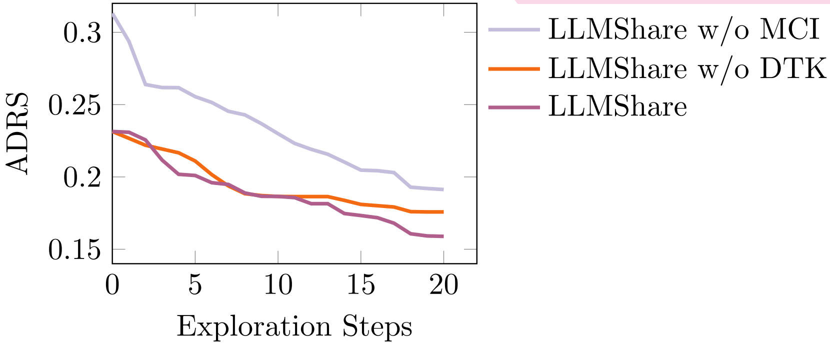
[8] **Chen Bai et al. (2021). "BOOM-Explorer: RISC-V BOOM microarchitecture design space exploration framework".** In: *ICCAD*. IEEE, pp. 1–9.

Results: Pareto Optimal Designs

Table: Normalized cost and request per second (RPS) of a Pareto optimal H100 cluster and a Pareto optimal configuration found by LLMShare. The design parameters are in the same order as Table 1.

Hardware Config			Cost	RPS
H100-cluster	Prefill	7,8,18,80,50,132,256,4,16,32	1.00	1.00
	Decode	6,8,18,80,50,132,256,4,16,32		
LLMShare	Prefill	4,4,24,112,100,156,512,1,128,32	0.87	4.11
	Decode	6,12,18,80,70,72,448,2,32,16		

Ablation Study



Ablation study on the effectiveness of DTK and MCI

Conclusion

- Developed a simulator to model LLM serving system performance and cost.
- Introduced a DSE framework with specialized techniques.
- Significant improvements: 13% cost reduction and $4\times$ throughput gain.

THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY