Efficient Multi-task LLM Quantization and Serving for Multiple LoRA Adapters

Yifei Xia[1], Fangcheng Fu[1], Wentao Zhang[1], Jiawei Jiang[2],
Bin Cui[1]

[1] Peking University, [2] Wuhan University

NEURAL INFORMATION PROCESSING SYSTEMS

Single-task Serving [vLLM, TensorRT-LLM]

- Current Paradigm: Parameter-Efficient Fine-Tuning (PEFT) + Model Quantization
- PEFT: Only fine-tune small part of parameter
  - LoRA: Using low rank adapters instead of whole matrix.
- Quantization: Low-bit Representation
  - Float point quantization: Direct quantization using only weight information.
  - GPTQ/AWQ: Quantization with calibration set, which is mainstream methods.
- What's the problem?
  - Only support one LoRA adapters concurrently.
  - Directly tweaked to N tasks serving, must N replica of base model.

Current Multi-task Serving [S-LoRA, Punica]

- Key Feature:
  - Shared one base model with LoRA adapter switching.
  - Kernel support to serving multi LoRAs in one batch.

- What's the problem?
  - Can not quantization:
    - Mainstream quantization methods (e.g., GPTQ) cannot support multi-task scenarios due to task-specific calibration needs.
  - No Dynamic Task Support:
    - Existing systems only handle static task sets; adding new tasks requires restarting services.
  - Low Scheduling Inefficiencies:
    - Do not consider the workload difference of different tasks

Our Solution：LoRA_Inlaid, a quantization-based Multi-Task serving system in resource constrained Scene

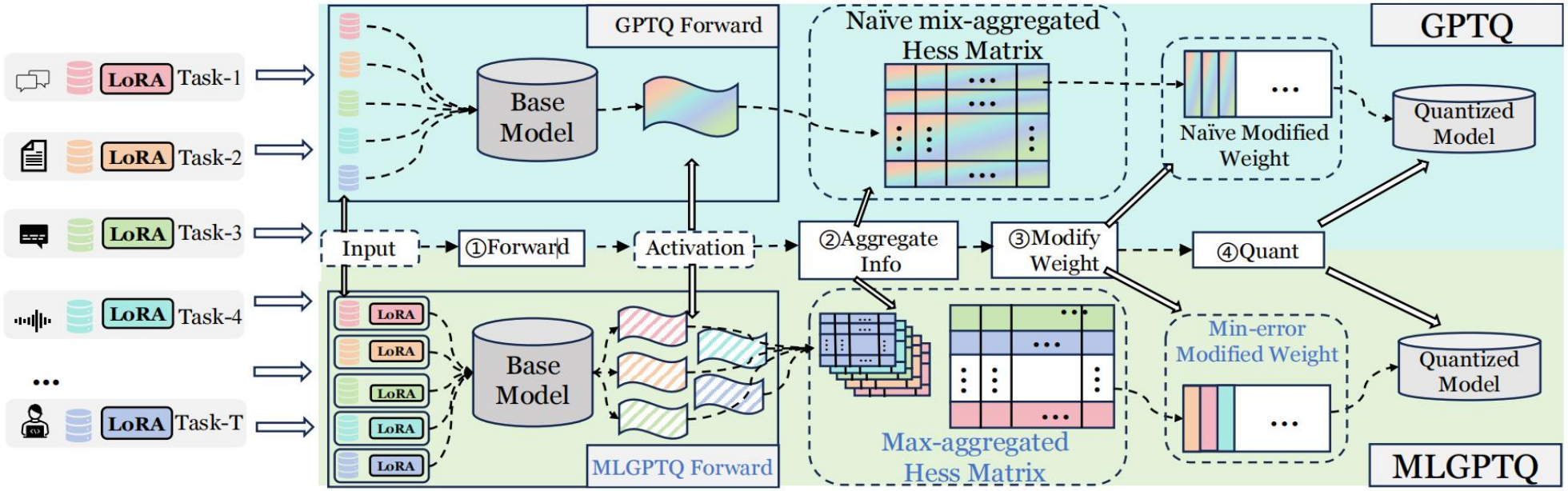
- Key Innovations:
  - Multi-task Quantization Algorithm (MLGPTQ):
    - Jointly quantizes the base model to support multiple LoRA adapters.
    - Enables dynamic task addition without service interruption.
  - Multi-task Scheduling Strategy:
    - Scheduling method designed for multitasking scenarios, Incorporates output length prediction and grouping optimization.

Table 1: Comparison of supported features of different LLM serving systems

System	Multi-task Serving	Multi-task Quantization	Dynamic Task Addition	Multi-task Scheduling
vLLM [19] & TensorRT-LLM [29]	✗	✗	✗	✗
S-LoRA [35] & Punica [5]	✓	✗	✗	✗
LoRA-Inlaid (this work)	✓	✓	✓	✓

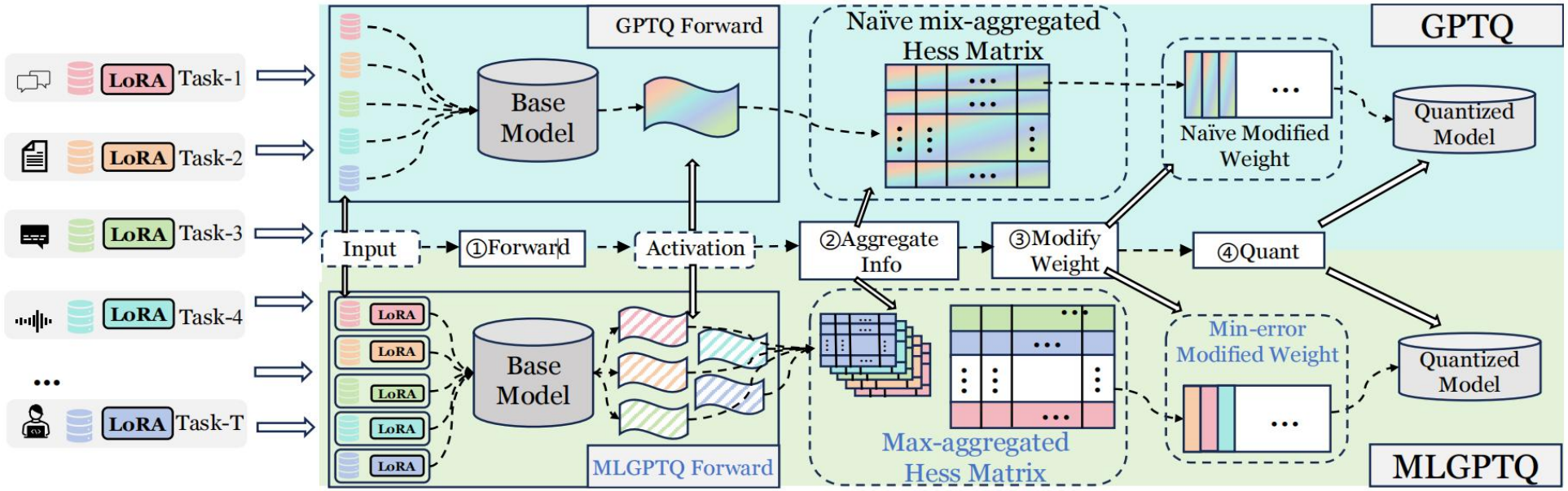
Joint Quantization: Multi-LoRA GPTQ (MLGPTQ)

- Challenge: Existing quantization methods are one task-specific.
  - Quantized models cannot be shared across tasks due to differing calibration requirements.
  - Directly tweaked single-task quantization methods suffers from accuracy loss due to Data distribution disruption.



Joint Quantization: Multi-LoRA GPTQ (MLGPTQ)

- Key idea: Capture and highlight data distribution characteristics of each task.
  - Switch corresponding LoRA adapters when forward(Capture correct distribution).
  - Use a max-aggregated Hessian matrix to retain critical task-specific information(Highlight the most important task).

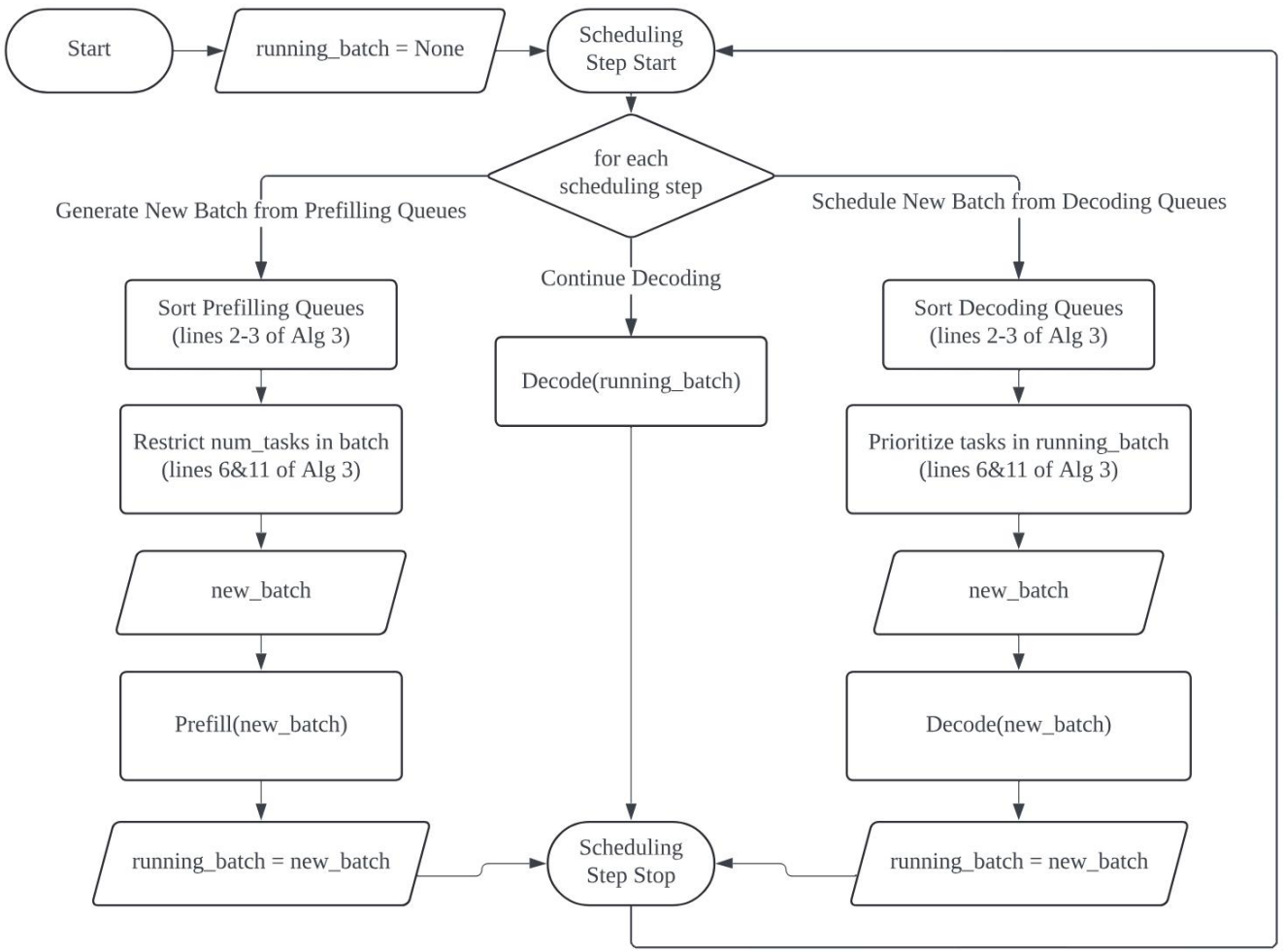


Dynamic Task addition

- Challenge:
  - Unseen Distributions of new task
  - Serving Interruption
  - Memory and computation resources snatching
- Our Solution:
  - Incremental Quantization
  - Layer-by-Layer Quantization
  - Asynchronous Re-Quantization

Multi-task Scheduling

- Challenge:
  - Divergent Workloads
  - Excessive Memory Overhead
- Our Solution:
  - Scheduling Guided by Output Length Prediction
  - Reducing Tasks Involved via Grouping



Experiments: Setup

Platform. RTX 4090 & RTX 3090

Models. Using Llama-2-7b and Llama-2-13b models, with LoRA adapters of ranks {8,16,32,64}.

Dataset and WorkLoad. Accuracy --> 6 translation tasks, 1 text summarization task, 1 table summarization task, 1 code generation task, 2 QA task and 1 classification task.

Performance --> Utilizing Gamma process to generate large-scale virtual load.

Evaluation Metrics. Translation task accuracy --> Sacrebleu; Other task accuracy --> ROUGE.

Performance --> Throughput, average request latency, task completion time, average response time, and SLO (service level object) satisfaction rate.

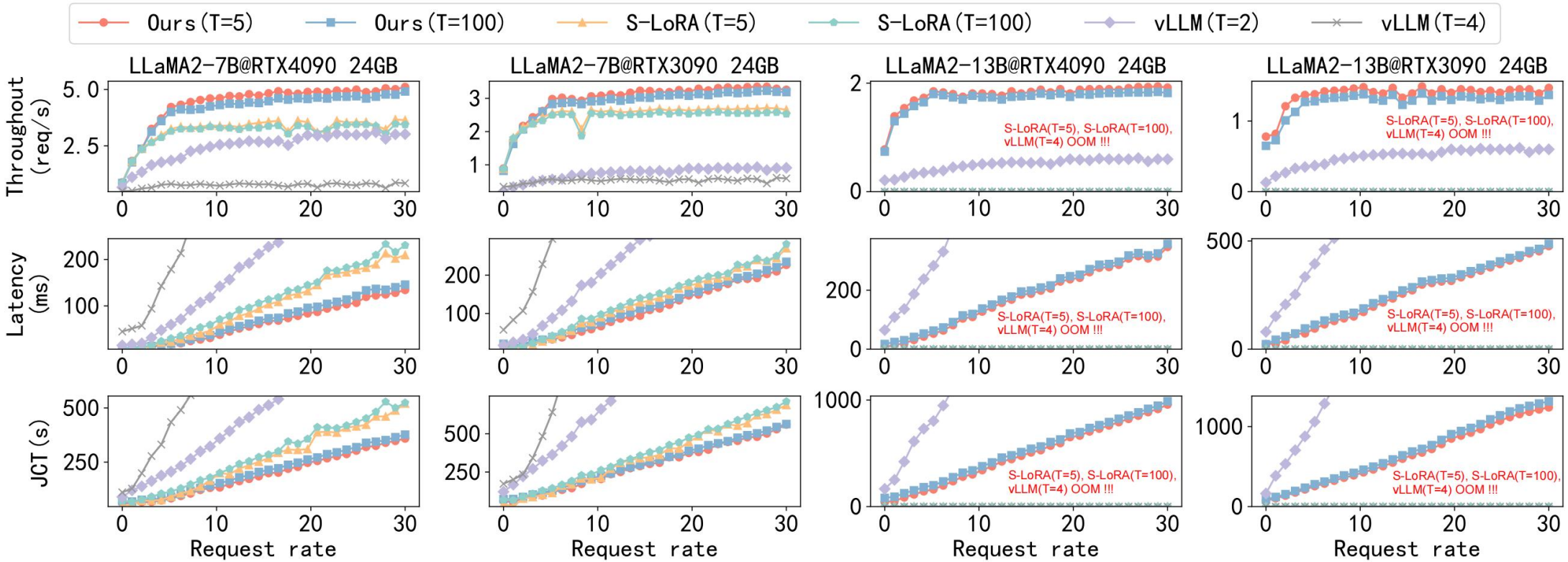
Experiments: Model quality

- In all 12 different tasks, MLGPTQ quantization achieved the best results, comparable to single-task quantization effects.

	Dataset Metric	trans-fr S_BLEU	trans-cs S_BLEU	trans-id S_BLEU	trans-nl S_BLEU	trans-da S_BLEU	trans-sw S_BLEU	QTsum ROUGE-1	xlsum ROUGE-1	tiny-codes ROUGE-1	GSM8k Acc (%)	med-qa Acc (%)	malicious Acc (%)
	Unquantized	34.45 (0.01)	31.89 (0.02)	33.94 (0.01)	30.94 (0.01)	35.04 (0)	31.14 (0.01)	49.38 (0)	41.28 (0.01)	31.72 (0.02)	32.14 (0)	80.9 (0)	37.44 (0)
4-bit	MLGPTQ	**34.05 (0.02)**	**31.16 (0.02)**	**33.63 (0.01)**	**30.73 (0.04)**	**34.39 (0.01)**	**31.20 (0.01)**	**49.02 (0)**	**40.96 (0.02)**	**30.85 (0.05)**	**31.62 (0)**	**76.7 (0.68)**	**36.44 (0.6)**
	$GPTQ_{tweaked}$	33.91 (0.02)	28.95 (0.21)	32.88 (0.08)	30.48 (0.07)	33.47 (0.04)	28.94 (0.06)	48.23 (0.02)	39.77 (0.06)	29.25 (0.10)	31.51 (0)	74.81 (2.53)	35.05 (1.15)
	$AWQ_{tweaked}$	33.88 (0.04)	29.45 (0.11)	33.01 (0.06)	29.99 (0.09)	33.34 (0.11)	30.11 (0.07)	47.96 (0.03)	40.12 (0.08)	30.23 (0.07)	30.51 (0.01)	75.42 (1.13)	35.68 (0.33)
	RTN	33.79 (0)	29.64 (0.01)	32.96 (0.01)	30.33 (0)	33.96 (0)	30.46 (0.02)	47.54 (0.01)	40.27 (0.02)	30.63 (0.02)	31.01 (0)	76.15 (0)	33.78 (0)
	GPTQ	34.07 (0.02)	31.19 (0.03)	33.79 (0.02)	30.86 (0.15)	34.57 (0.02)	31.08 (0.08)	49.26 (0.02)	40.89 (0.06)	30.92 (0.06)	31.35 (0)	76.7 (2.66)	36.25 (0.44)
	AWQ	34.17 (0.03)	31.19 (0.05)	33.72 (0.07)	30.69 (0.08)	34.21 (0.08)	31.07 (0)	49.04 (0.12)	41.10 (0.02)	31.03 (0.04)	31.45 (0)	75.42 (1.26)	36.18 (0.28)
3-bit	MLGPTQ	**31.72 (0.39)**	**26.93 (0.58)**	**30.11 (0.63)**	**27.97 (1.04)**	**30.77 (0.50)**	**28.06 (0.53)**	**47 (0.38)**	**39.07 (0.22)**	**27.62 (0.47)**	**28.74 (0)**	**54.84 (7.94)**	**31.90 (0.47)**
	$GPTQ_{tweaked}$	31.3 (0.62)	25.89 (0.7)	28.18 (0.75)	23.54 (1.01)	24.09 (0.51)	21.12 (0.39)	45.99 (0.26)	38.32 (0.15)	23.80 (0.46)	28.30 (0)	54.02 (7.9)	30.93 (0.24)
	$AWQ_{tweaked}$	31.57 (0.94)	26.45 (0.59)	25.13 (0.62)	24.46 (1.02)	26.77 (0.7)	19.79 (1.02)	45.13 (0.17)	37.62 (0.46)	21.83 (0.46)	28.24 (0)	53.66 (19)	31.03 (0.34)
	RTN	26.02 (0.01)	0.03 (0)	0.03 (0)	0.06 (0)	0.05 (0)	0.05 (0)	0.9 (0)	0.10 (0)	0.34 (0)	26.38 (0)	51.3 (0)	31.4 (0)
	GPTQ	30.83 (0.31)	26.19 (0.58)	31.88 (0.65)	28.21 (0.81)	32.93 (0.26)	29.75 (0.3)	47.22 (0.17)	39.53 (0.17)	26.12 (0.11)	28.16 (0)	56.03 (12.39)	31.26 (0.5)
	AWQ	31.23 (0.63)	25.35 (0.18)	30.35 (0.73)	28.65 (1.15)	31.23 (0.32)	28.77 (0.62)	47.13 (0.17)	39.23 (0.44)	27.16 (0.57)	28.63 (0)	53.66 (4.08)	31.77 (0.29)

Experiments: Performance

- LoRA-Inlaid outperforms existing SOTA multi-task serving systems by up to 1.58× in terms of throughput, 1.76× in terms of average latency, 2× in terms of job completion time.



Conclusion and significance

- **We developed a multi-LoRA task serving system, namely *LoRA-Inlaid*, which:**
  - Equiped with a dynamic multi-task quantization algorithm.
  - Facilitated real-time dynamic task addition
  - Introduced a novel multi-task scheduling strategy
- **Significance:**
  - The first one introduce a multi-task joint quantization method, significantly reducing the memory requirements for multi-task serving deployment.
  - Make multi-task serving systems truly usable in resource-constrained environments.

Thanks for your attention!