ExeGPT: Constraint-Aware Resource Scheduling for LLM Inference

Hyungjun Oh Kihong Kim Jaemin Kim Sungkyun Kim Junyeol Lee
Du-seong Chang Jiwon Seo
Hanyang University KT Corporation

ASPLOS 2024

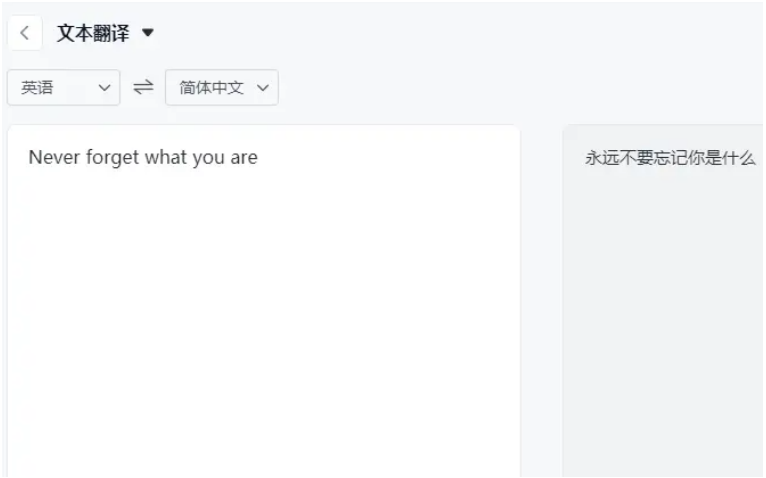
汇报人：陆松涛

2024年10月28日

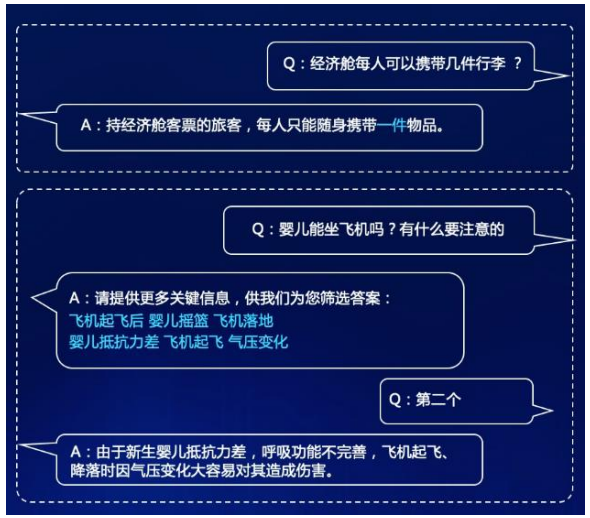
- 研究背景
- 研究问题
- 研究内容
- 实验评估
- 工作总结

- **研究背景**
- 研究问题
- 研究内容
- 实验评估
- 工作总结

研究背景



Text Translation



Intelligent Answer System

Automatic Code Generation

Large language models (LLMs) have significantly advanced the field of natural language processing (NLP) and enabled a wide range of NLP applications.

研究背景

High Computational Costs

- **Hundreds of billions** of parameters
- **Multiple** GPUs for model parallel execution

optimization

Efficiency

Irregular Executions

- **Different** workload of encoding and decoding
- **Variable** length of each input and output

High Throughput

- 研究背景
- **研究问题**
- 研究内容
- 实验评估
- 工作总结

研究问题

LLM Inference Optimization

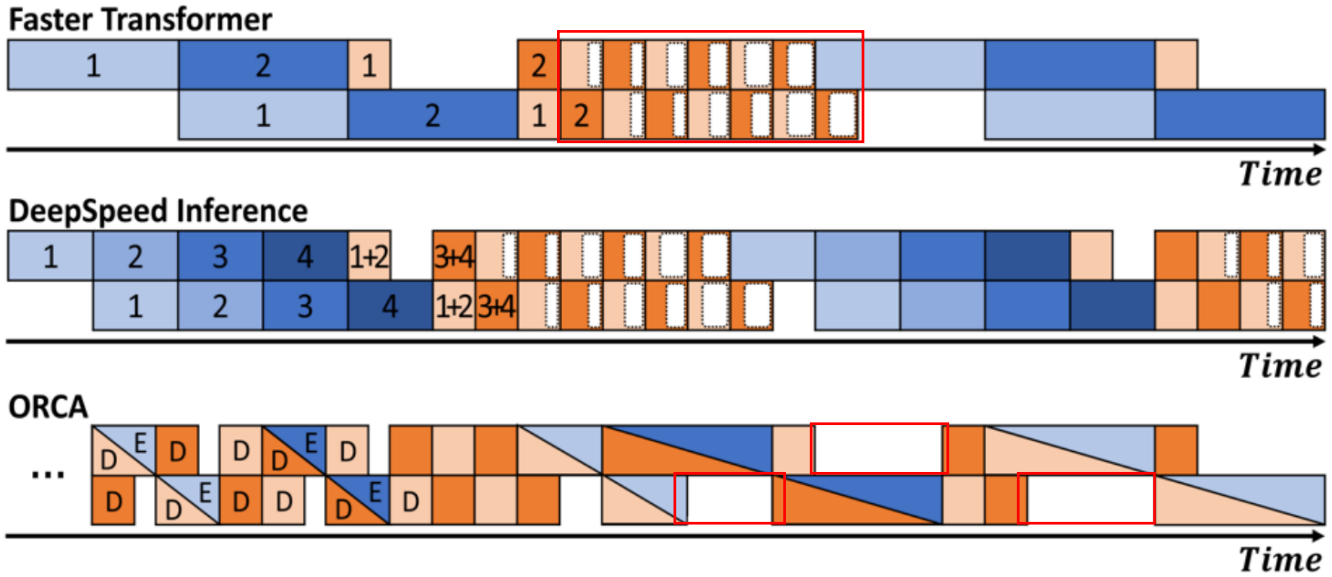
1. Model Compression Techniques

2. Kernel Acceleration Techniques

3. Resource Scheduling Optimizations

研究问题——现有工作局限性

Limitation of Existing work



(b) Execution on two workers. The numbers are micro-batch IDs. E and D denote encoder and decoder.

Batch sizes decrease

↓

A **degradation** of hardware resource utilization and the inference throughput

Differences in encoding and decoding workloads

↓

Pipeline bubbles

Inefficient trade-off mechanism between throughput and latency

研究问题——挑战

Challenge 1　　Balance between encoding and decoding operations

- Small decoding batch sizes
- Pipeline bubbles

Challenge 2　　Flexible trade-off between latency and throughput

NLP applications have diverse latency constraints for generating a sequence within guaranteed timeframes.

- 研究背景
- 研究问题
- **研究内容**
- 实验评估
- 工作总结

研究内容——ExeGPT

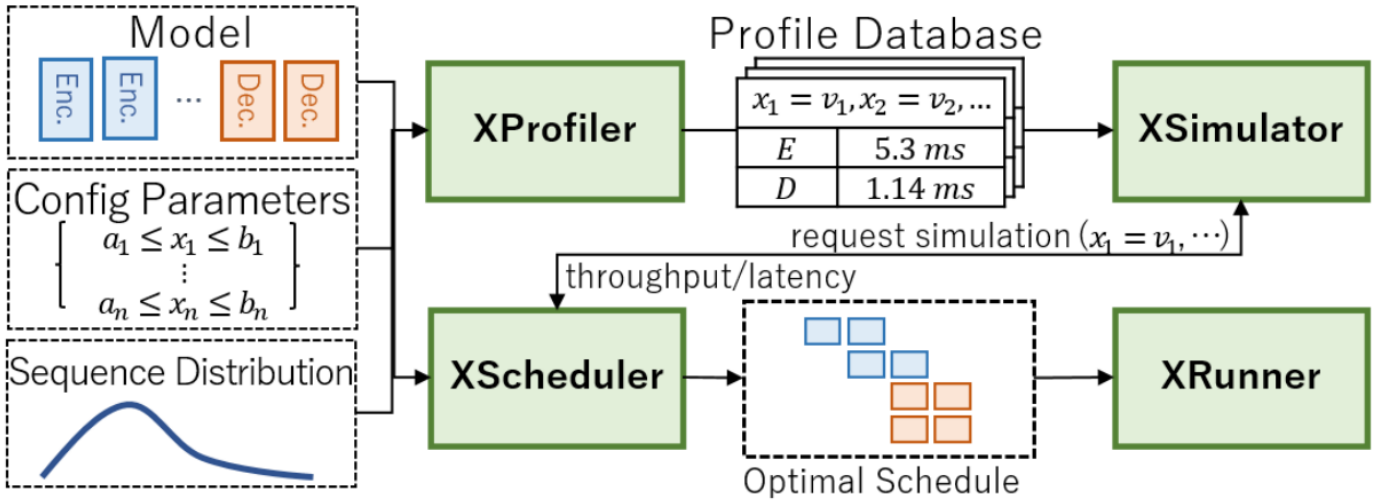


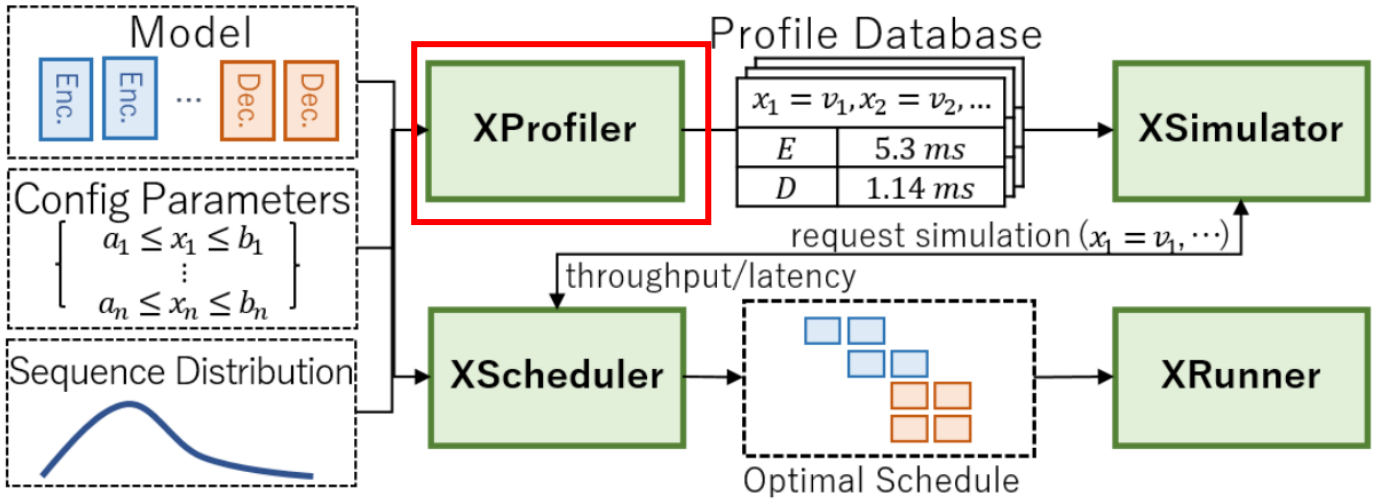
Figure 2. High-level architecture of ExeGPT

ExeGPT, a **distributed** system designed for **constraint-aware** LLM inference.

ExeGPT finds and runs with an optimal execution schedule to **maximize inference throughput while satisfying a given latency constraint.**

By leveraging the distribution of input and output sequences, it effectively **allocates resources** and **determines optimal execution configurations**, including batch sizes and partial tensor parallelism.

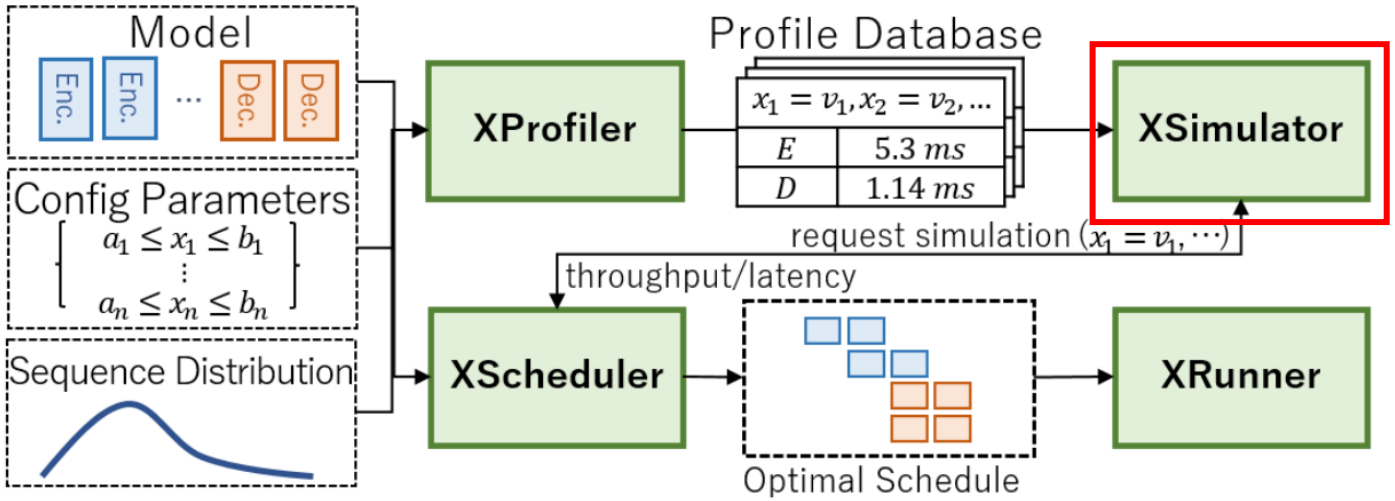
研究内容——ExeGPT



XProfiler

For a single encoding and decoding layer, the profiler separately measures the **execution times** of the attention kernel and the rest of the encoding/decoding layer, considering all feasible tensor-parallel degrees.

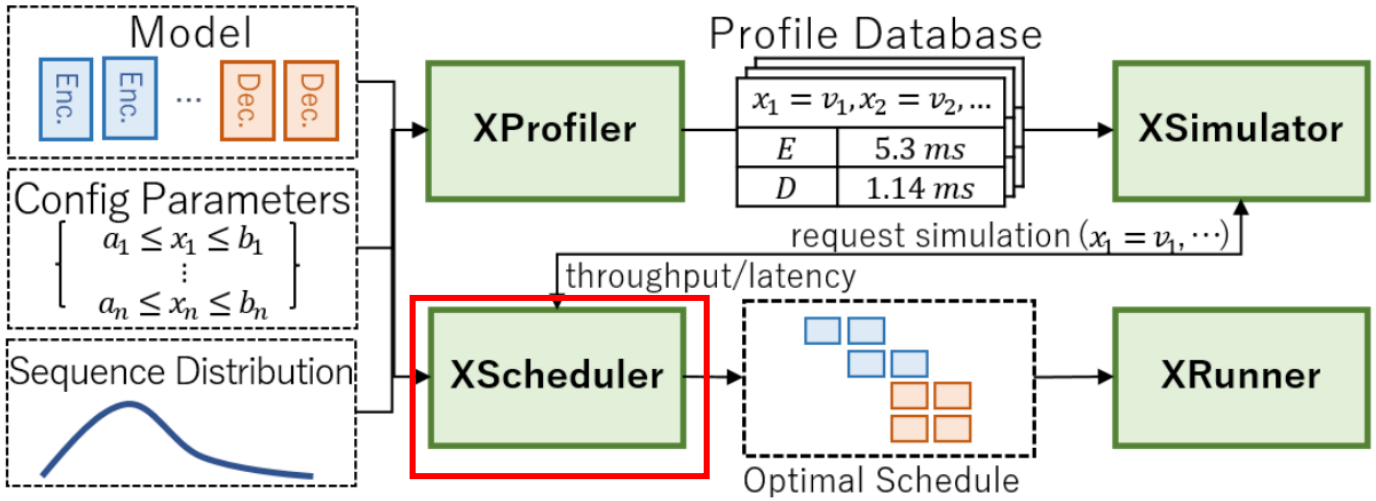
研究内容——ExeGPT



XSimulator

Using the profiling results, the simulator constructs the execution timeline based on the configuration parameters, such as batch size and tensor-parallel degree, given by the scheduler.

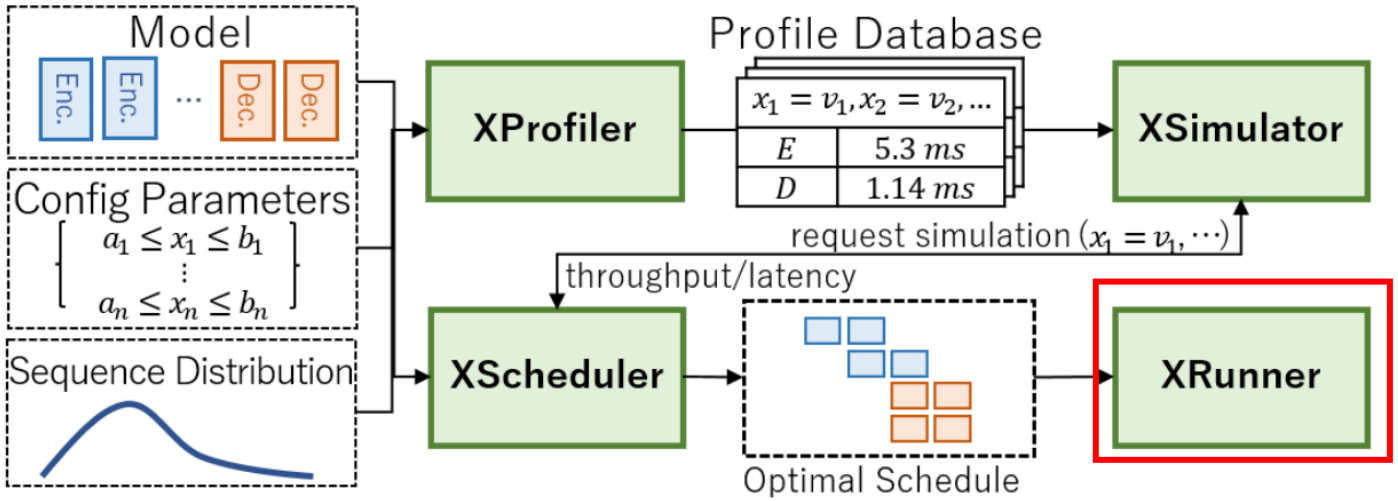
研究内容——ExeGPT



XScheduler

The scheduler searches over the solution space to find the optimal values of the configuration parameters that maximize throughput under a given latency bound.

研究内容——ExeGPT

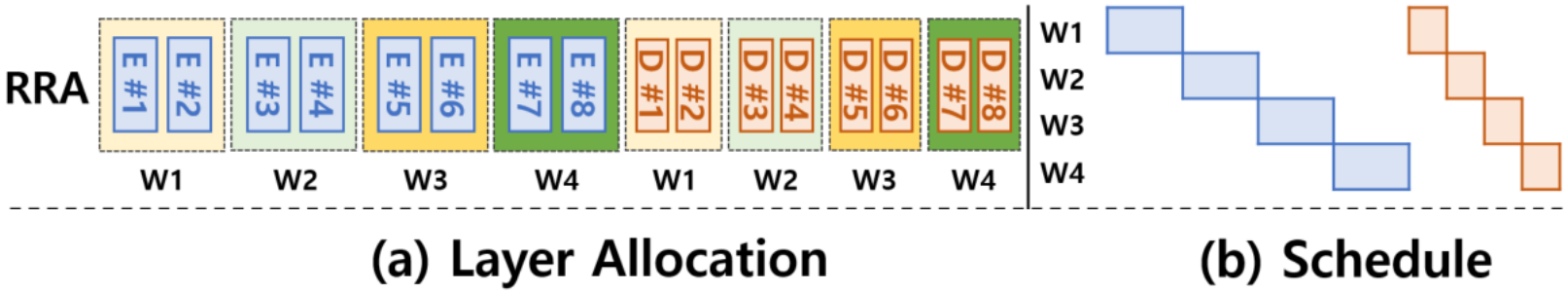


XRunner

This serves as the execution engine for running LLM inference on multiple distributed GPUs. It takes as input the execution schedule provided by XScheduler and ensures that the schedule is enforced during the execution.

研究内容——Decoupled Scheduling of Encoders and Decoders

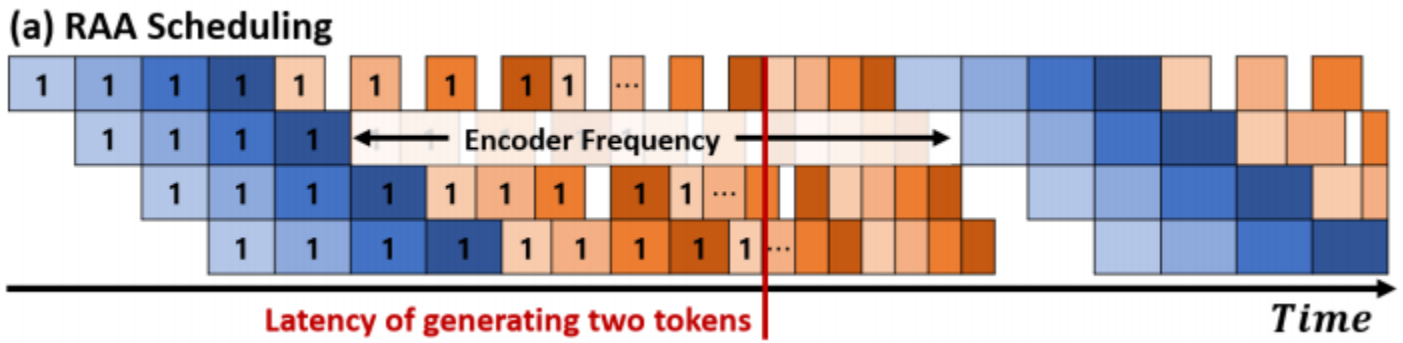
Round-Robin Allocation (RRA) and Workload-Aware Allocation (WAA)



RRA Round-Robin Allocation

Execute input encoding once

Fix the number of decoding iterations



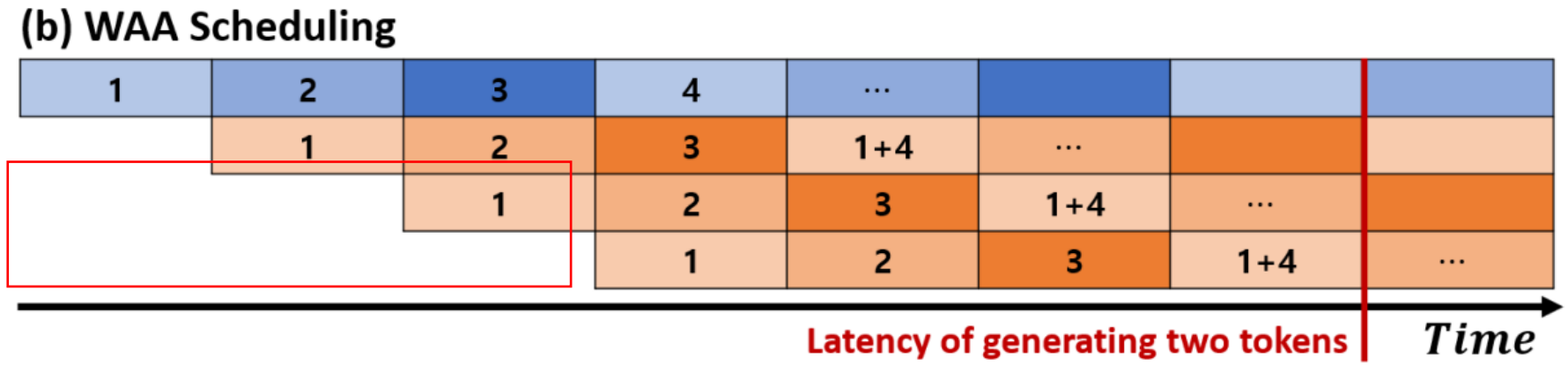
研究内容——Decoupled Scheduling of Encoders and Decoders

Round-Robin Allocation (RRA) and Workload-Aware Allocation (WAA)

WAA Workload-Aware Allocation

WAA-C based on computation time

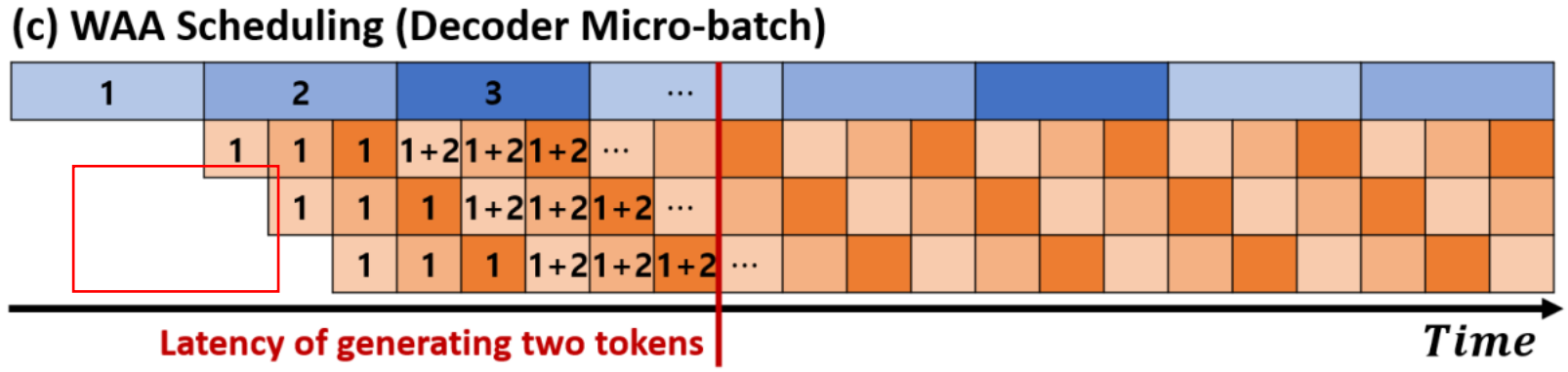
WAA-M based on memory consumption

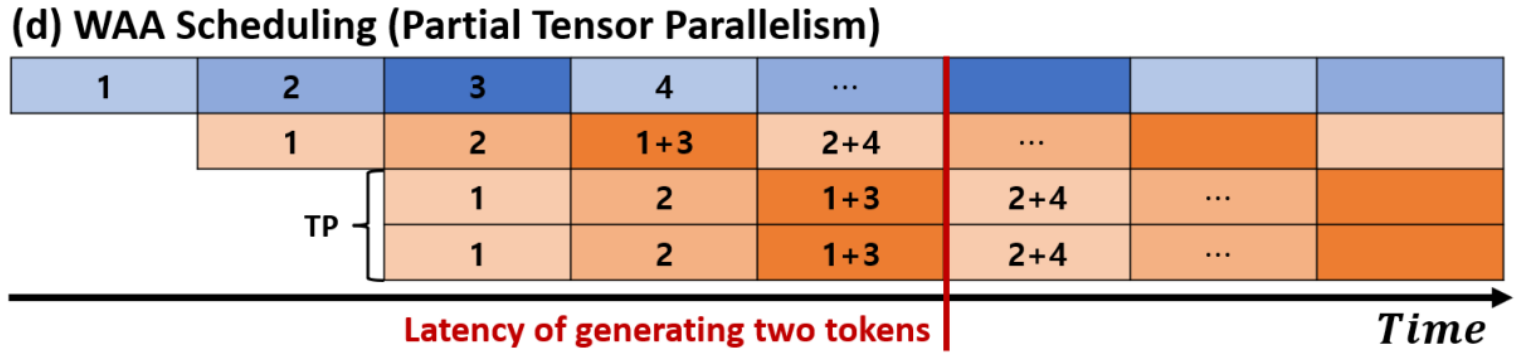


研究内容——Decoupled Scheduling of Encoders and Decoders

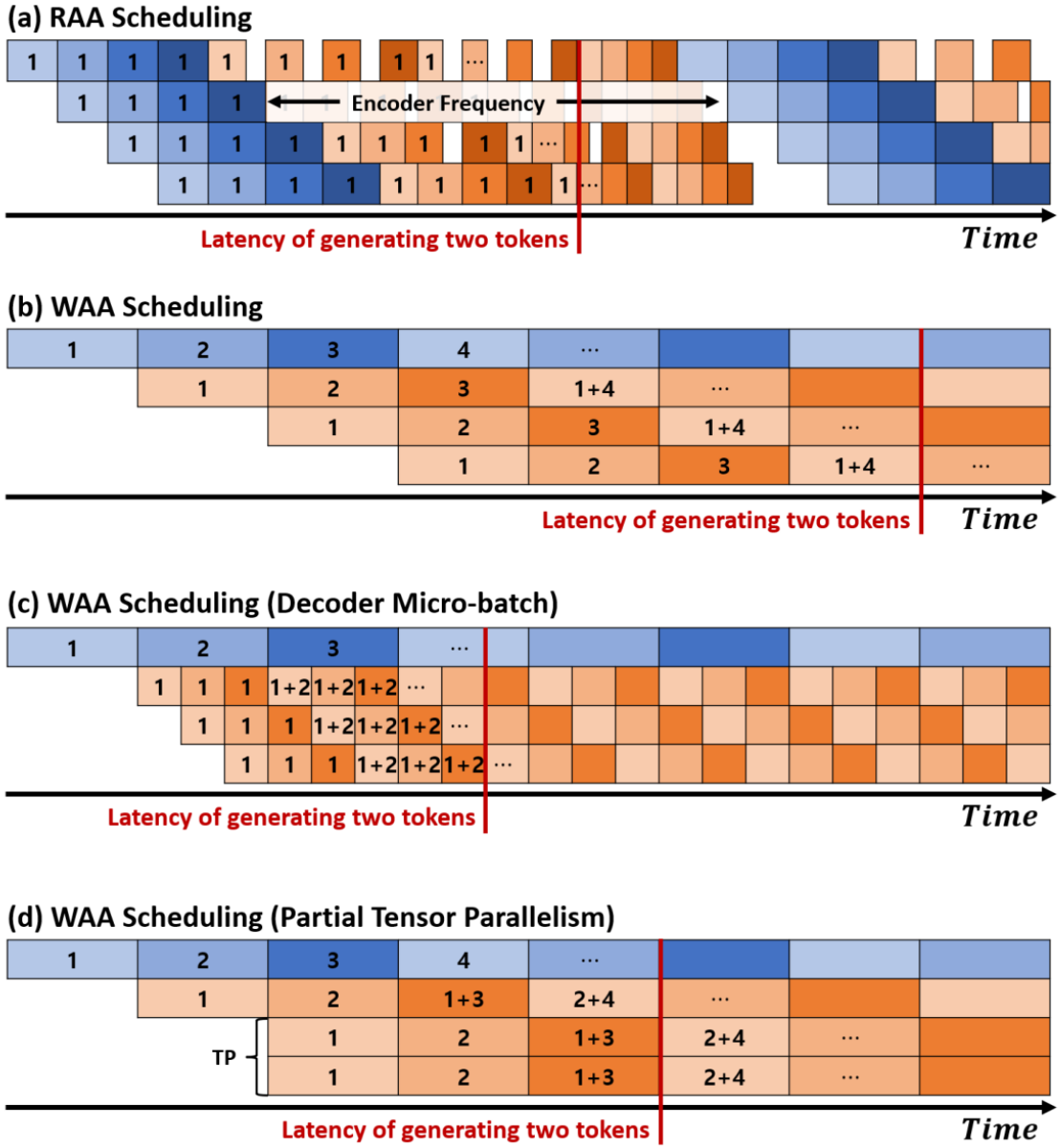
Round-Robin Allocation (RRA) and Workload-Aware Allocation (WAA)

WAA Workload-Aware Allocation





研究内容——Decoupled Scheduling of Encoders and Decoders



→ Different algorithms are applicable to different tasks and application scenarios

研究内容——Controlling Latency and Throughput Trade-off

Batch size

Increasing the batch size improves the parallelism of the inference kernels and thus improves the throughput but it also increases the latency.

Decoder micro-batch

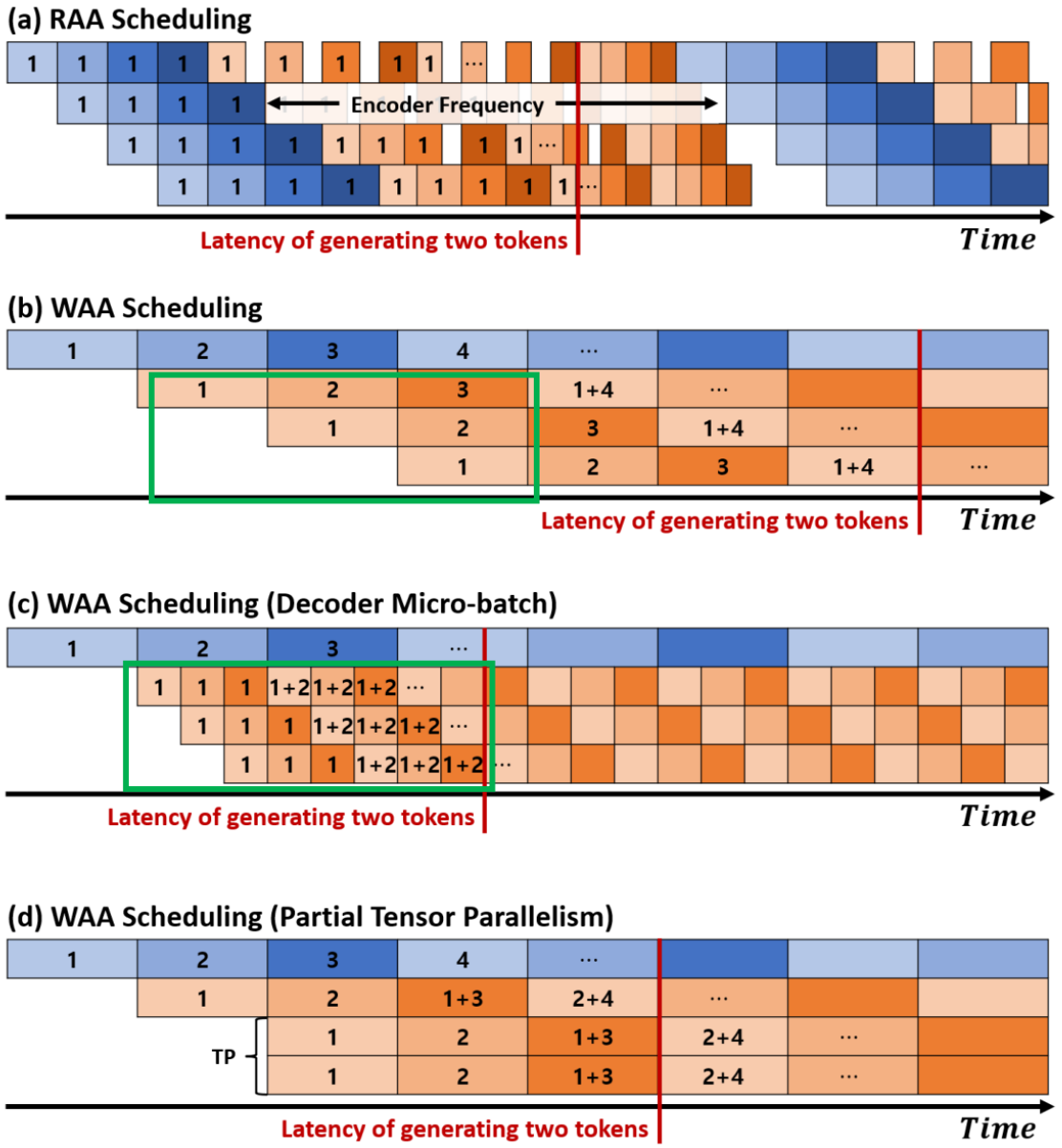
In WAA, we make it possible to split a decoding batch into multiple micro-batches, which can largely reduce the latency.

Partial tensor parallelism

By splitting a layer and executing it on multiple GPUs in a tensor parallel way, we can reduce the depth of the execution pipeline and decrease latency at the cost of lower throughput due to synchronization overhead.

Encoding frequency

In RRA Scheduling, adjusting the encoding frequency can increase throughput or decrease latency.



研究内容——Constraint-Aware Scheduling Algorithm

Scheduling Algorithm for Monotonic Optimization

$$\underset{B_{\{E,D,m\}},T_P,F_E,S}{\arg\max} \quad Throughput(B_E, B_D, B_m, T_P, F_E, S, P_E, P_D)$$

$$\text{s.t. } Latency(B_E, B_D, B_m, T_P, F_E, S, P_E, P_D) < L_{Bound},$$

Monotonic

Branch-and-bound method

Traverse and evaluate

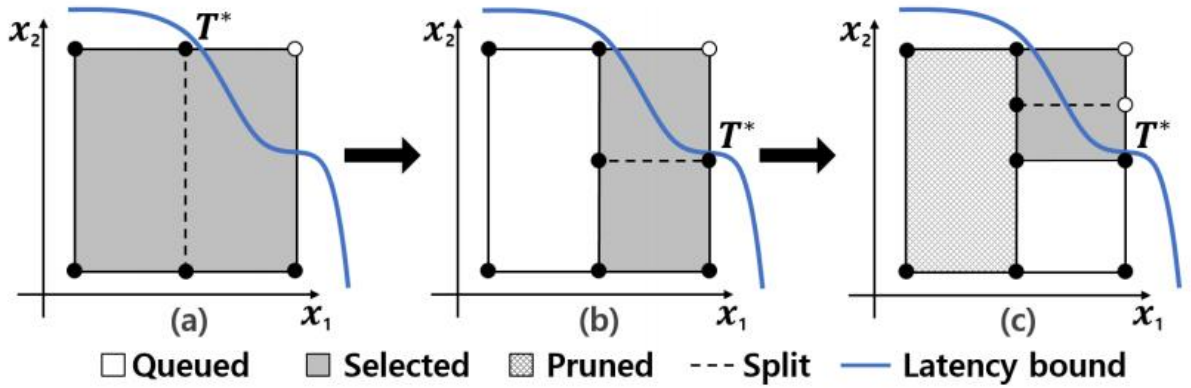


Figure 5. Illustration of the branch-and-bound scheduling. The filled circles are configuration points within the latency bound and empty circles are those that are not. T^* is the current optimal point.

研究内容——Constraint-Aware Scheduling Algorithm

Scheduling Algorithm for Monotonic Optimization

Monotonic

The control variables generally exhibit **monotonic** behavior with throughput and latency.

Table 5. Percentage of non-monotonic points*

Task	Tol.†	RRA		WAA		
		B_E	N_D	B_E	TP	B_m
S	2%	(0.6, 2.3)	(0.0, 0.0)	(0.0, 0.0)	(0.0, 0.0)	(0.8, 0.0)
	5%	(0.6, 0.3)	(0.0, 0.0)	(0.0, 0.0)	(0.0, 0.0)	(0.5, 0.0)
	10%	(0.3, 0.0)	(0.0, 0.0)	(0.0, 0.0)	(0.0, 0.0)	(0.4, 0.0)
T	2%	(0.2, 0.1)	(0.0, 0.0)	(0.1, 0.0)	(1.2, 0.0)	(10.9, 0.0)
	5%	(0.2, 0.0)	(0.0, 0.0)	(0.0, 0.0)	(0.8, 0.0)	(10.6, 0.0)
	10%	(0.2, 0.0)	(0.0, 0.0)	(0.0, 0.0)	(0.4, 0.0)	(10.4, 0.0)

∗ Each cell represents (Latency, Throughput).

† Tolerance values are calculated as % of L_B (70^{th} pctl) and the achieved tput.

研究内容——Constraint-Aware Scheduling Algorithm

Scheduling Algorithm for Monotonic Optimization

$$\underset{B_{\{E,D,m\}},T_P,F_E,S}{\arg\max} \quad Throughput(B_E, B_D, B_m, T_P, F_E, S, P_E, P_D)$$

$$\text{s.t. } Latency(B_E, B_D, B_m, T_P, F_E, S, P_E, P_D) < L_{Bound},$$

Monotonic

Branch-and-bound method

Traverse and evaluate

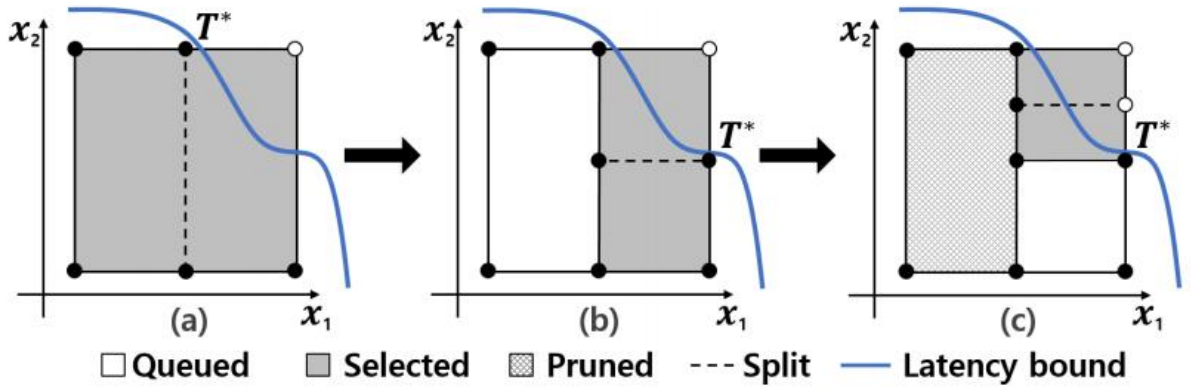


Figure 5. Illustration of the branch-and-bound scheduling. The filled circles are configuration points within the latency bound and empty circles are those that are not. T^* is the current optimal point.

研究内容——Controlling Latency and Throughput Trade-off

Dynamic Scheduling for Consistent Workload

Workload ⟵ Input and Output Sequence Lengths

Dynamic Workload Adjustment at Runtime ⟶ Consistent and Reliable Inference Execution

Dynamically adjust the encoder batch size to ensure that the encoder workload stays within a predetermined threshold of the average workload.

研究内容——Simulation with Sequence Distribution

Maintain consistent batch size

Estimate each decoding batch size

研究内容——Simulation with Sequence Distribution

probability distributions of input and output sequence lengths

$P_E(S)$ $P_D(S)$

a sequence of length S

the number of decoding iterations N_D

the encoder batch size B_E

the probability of completing the decoding of a query at U' th iteration after the previous encoding phase

$$P_D(U|S)=\begin{cases}\left\{\begin{array}{ll}1, & \text{if } U=S\\ 0, & \text{otherwise}\end{array}\right\}, & \text{if } S\le N_D\\ \left\{\begin{array}{ll}\frac{1}{\lceil \frac{S}{N_D}\rceil}, & \text{if } U=1+(S-1 \bmod N_D)\\ 0, & \text{otherwise}\end{array}\right\}, & \text{if } S>N_D\end{cases}$$

$$P_D(U)=\sum_S P_D(U|S)P_D(S)$$

the decoding batch size $B_D = \frac{B_E}{\sum_U P_D(U)}$

Use B_E and $P_E(S)$ to estimate the expected encoding workload and execution time.

Use B_D and $P_D(U)$ to compute the expected decoding workload and execution time.

实验评估

- 研究背景
- 研究问题
- 方法设计
- **实验评估**
- 工作总结

实验评估——Settings

Evaluated Models:

FT, OPT, GPT-3

Encoderdecoder and decoder-only models

System Settings:

GPU (Mem)	Cluster Size (per node×# node)	Interconn. (Intra/Inter)	Model: # GPUs
A100 (80GB)	16 (8×2)	NVLink/Infini.	GPT-3 (101B): 16 GPT-3 (175B): 16
A40 (48GB)	48 (8×6)	PCIe 4.0/Infini.	T5 (11B): 8 OPT (13B): 4 GPT-3 (39B): 16 GPT-3 (175B): 32 GPT-3 (341B): 48

Baseline:

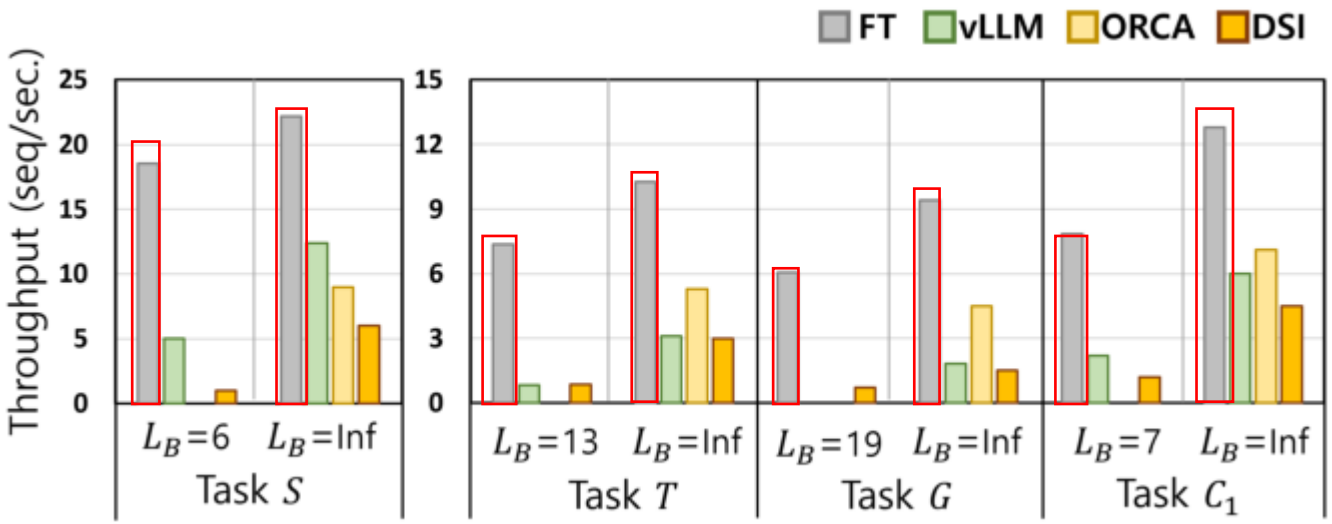
FasterTransformer (FT), DSI, ORCA, vLLM

Evaluation Scenarios:

Four representative NLP tasks:

Task	Task ID	Input Length (Avg., Std., Max)	Output Length (Avg., Std., 99^{th}, Max)
Summarization	S	(256, 252, 512)	(32, 13, 63, 80)
Translation	T	(128, 81, 256)	(128, 68, 292, 320)
Code Generation	G	(64, 23, 128)	(192, 93, 417, 480)
Conversational	C_1	(256, 115, 512)	(64, 30, 137, 160)
Q & A	C_2	(512, 252, 1024)	(256, 134, 579, 640)

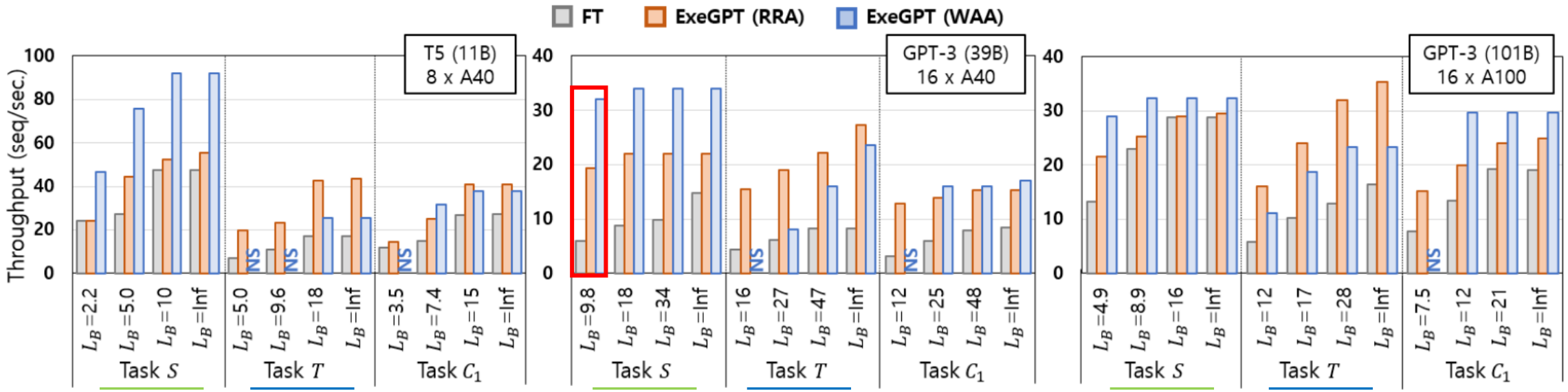
实验评估——the Performance of Existing Systems



Throughput comparison of LLM inference systems

Task	Task ID
Summarization	S
Translation	T
Code Generation	G
Conversational	C_1
Q & A	C_2

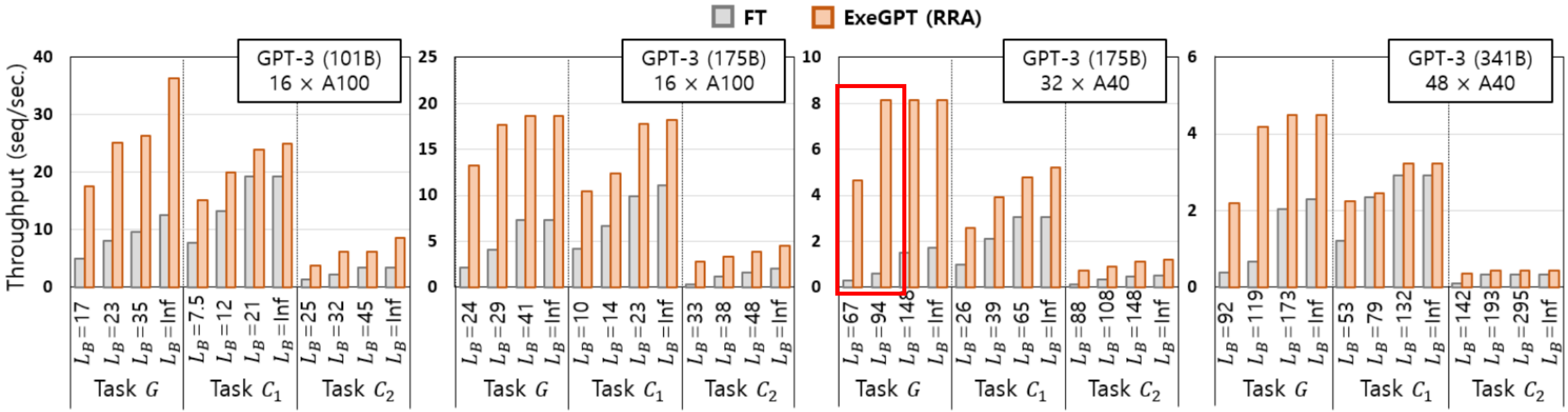
实验评估——Performance Evaluation of Small to Sid-sized LLMs



Task	Task ID
Summarization	S
Translation	T
Code Generation	G
Conversational	C_1
Q & A	C_2

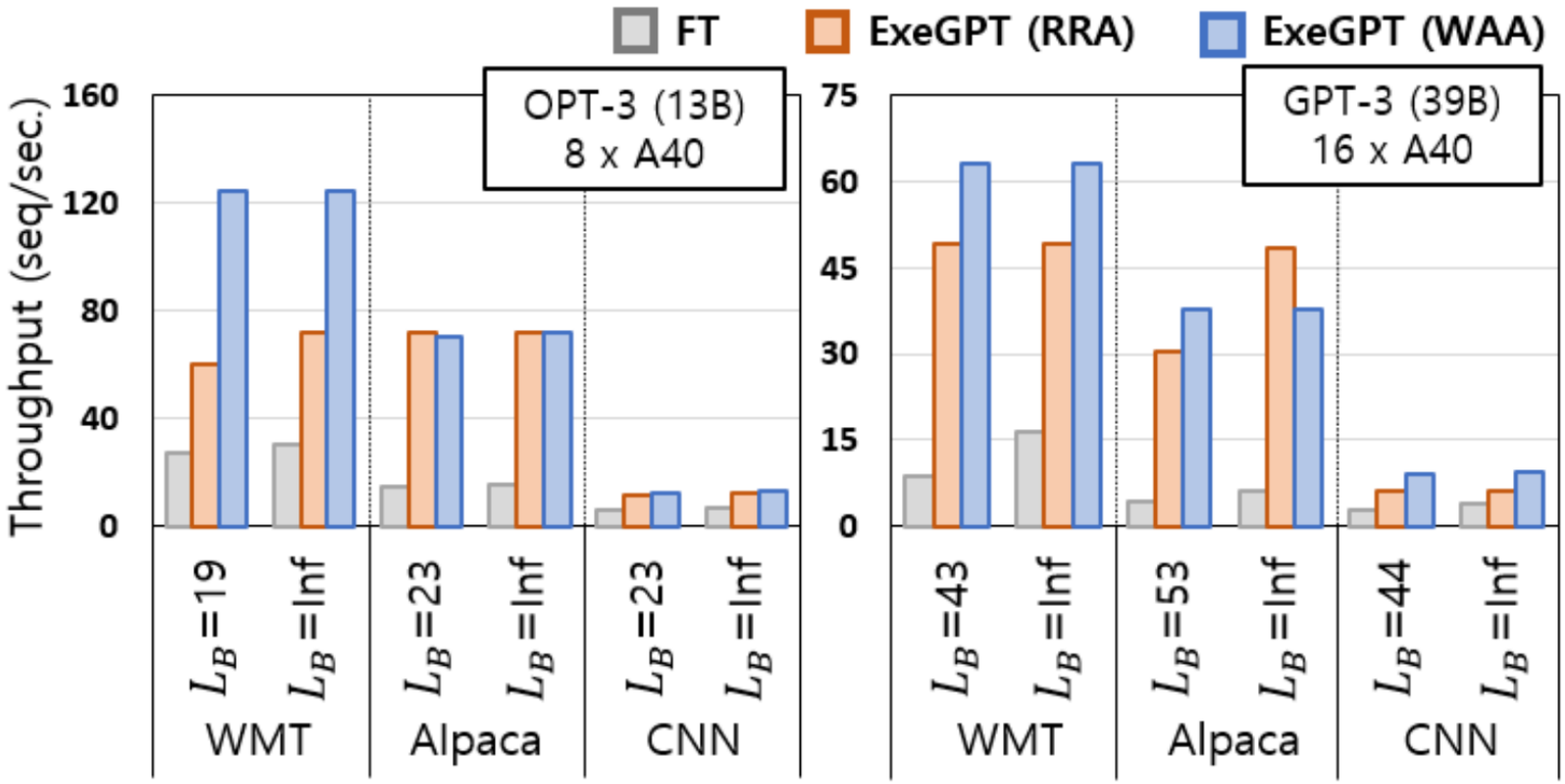
Overall, ExeGPT is on average 2× faster than FT; maximum 5.4× faster.

实验评估——Performance Evaluation of Large LLMs



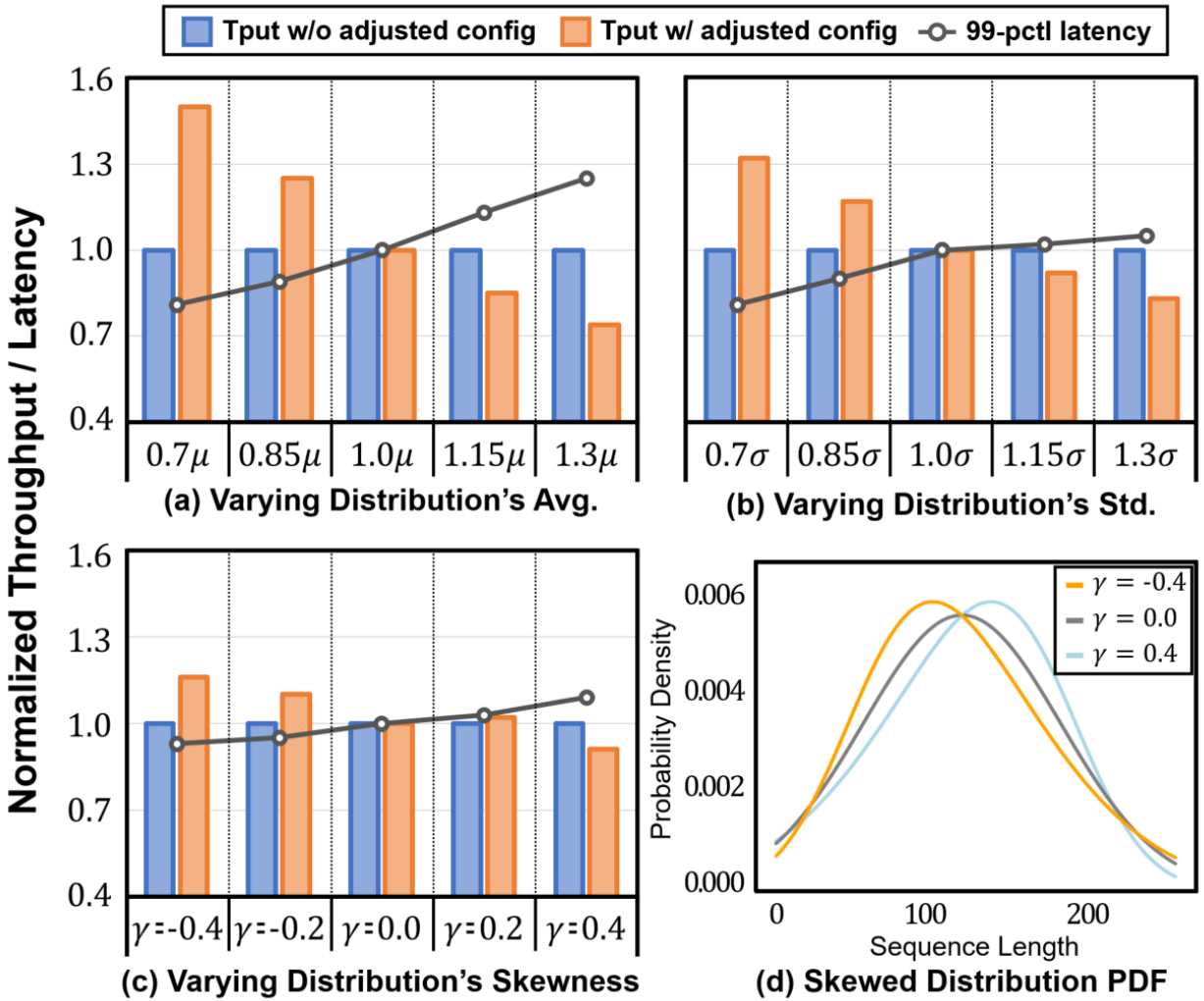
Across all tasks and models, ExeGPT is 3.2× faster than FT on average; maximum 15× faster.

实验评估——Evaluation with Real-World Datasets



ExeGPT (the faster of RRA and WAA) performs on average 4.4× faster than FT with a maximum speedup of 8.7×.

实验评估——Scheduling with Limited Information on Distribution



实验评估——Cost of Profiling, Scheduling, and (Re-)Deploying

Profiling < two hours

Scheduling 3s to 2min for RRA
1min to 5min for WAA

Scheduling with Exhaustive Search

> five hours
EVEN can extend to an entire day

Table 4. The cost of loading LLMs from SSD or CPU's DRAM

Model	#GPUs	Loading from DRAM	Loading from SSD
39B	16	0.9 secs.	2.1 secs.
101B	32	1.3 secs.	7.1 secs.
175B	32	2.1 secs.	11.9 secs.
341B	48	3.5 secs.	15.1 secs.

reasonably modest

实验评估——Other

Table 6. Selected Schedule and the Optimal Configurations

L_B	Selected Schedule	Control Variables	Latency (sec.)	Tput (seq./sec.)
3.1	WAA	B_E=4, T_P=2	3.01	22.41
5.9	WAA	B_E=10, T_P=2	4.11	23.76
11.5	RRA	B_E=52, N_D=13, T_P=4	10.23	25.23
Inf.	RRA	B_E=49, N_D=7, T_P=4	16.87	26.15

L_B ↑ B_E ↑

L_B ∞ N_D halve

While the encoder workload variance is not negligible，its impact on latency remains **minor.**

Table 7. Variance of encoder/decoder exec. times in seconds

Schedule	Encoder (99^{th} pctl Range)	Decoder (99^{th} pctl Range)
RRA	1.68 (±0.12, ±7.1%)	0.038 (±0.001, ±2.4%)
WAA	0.85 (±0.10, ±11.8%)	0.041 (±0.002, ±5.5%)

工作总结

- 研究背景
- 研究问题
- 方法设计
- 实验评估
- **工作总结**

工作总结——Conribution

Conceptually

- It introduces ExeGPT, a system for executing LLM inference that maximizes throughput under a given latency bound.

Technically

- The system utilizes two scheduling strategies, based on RRA and WAA policies.
- It implementes a scheduling algorithm based on the branchand-bound method, to efficiently find the optimal values of the variables.

Experimentally

- They evaluate ExeGPT on six LLM instances of T5, OPT, and GPT-3 and five NLP tasks, each with four distinct latency constraints.
- The evaluation shows that ExeGPT achieves up to 15.2× higher throughput when latency is the same and 6× better latency when throughput is the same than FT.

工作总结——Inspiration

Pipeline

Constraint-Aware Scheduling Algorithm replace into ML

Long output length in large LLM

Q&A

2024年11月10日

陆松涛