

Universal LLM deployment and low-latency serving in MLC LLM

Ruihang Lai, CMU

Oct 16, 2024



Outline

- Part I. Universal LLM deployment in MLCEngine
- Part II. Low-latency serving study in MLCEngine

Outline

- Part I. Universal LLM deployment in MLCEngine
- Part II. Low-latency serving study in MLCEngine

What is MLC LLM?

A *high-performance universal* LLM engine that allows native deployment any large language models with native APIs with *compiler acceleration*.

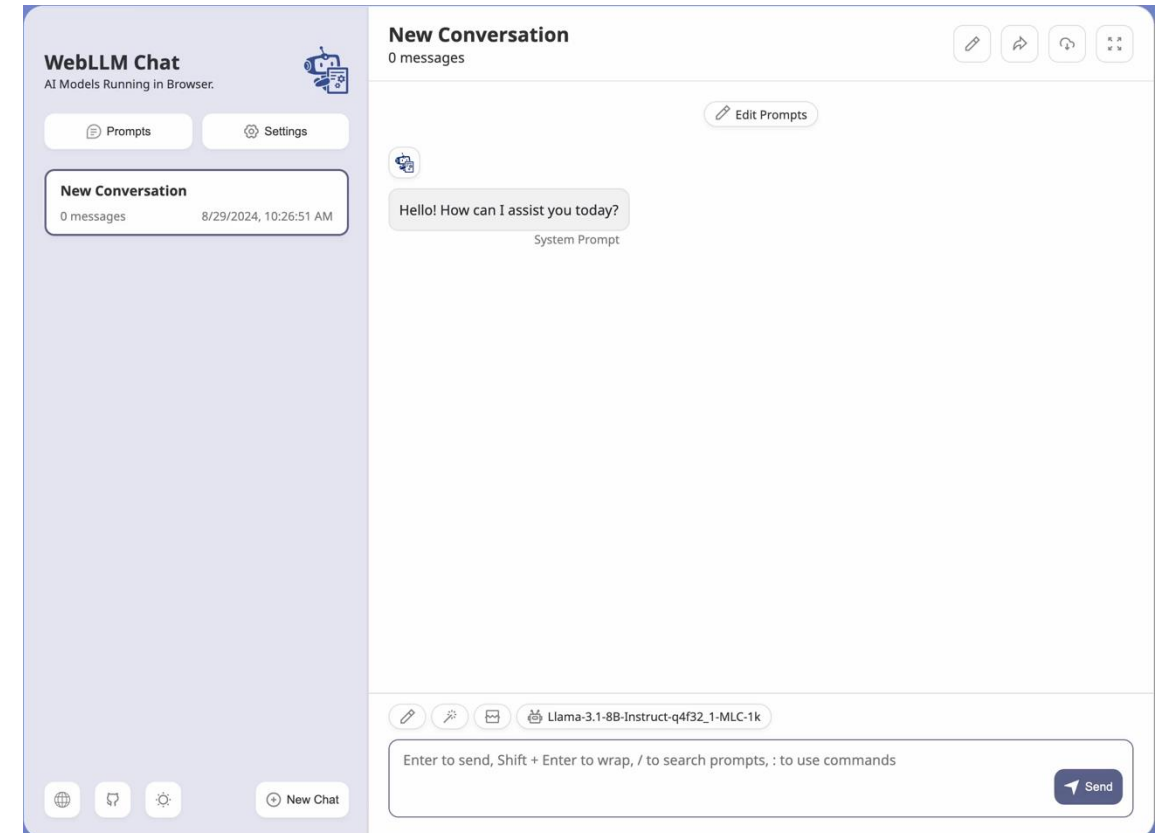
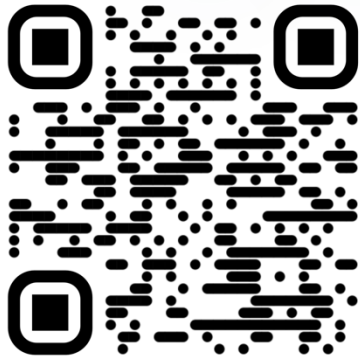


Demo

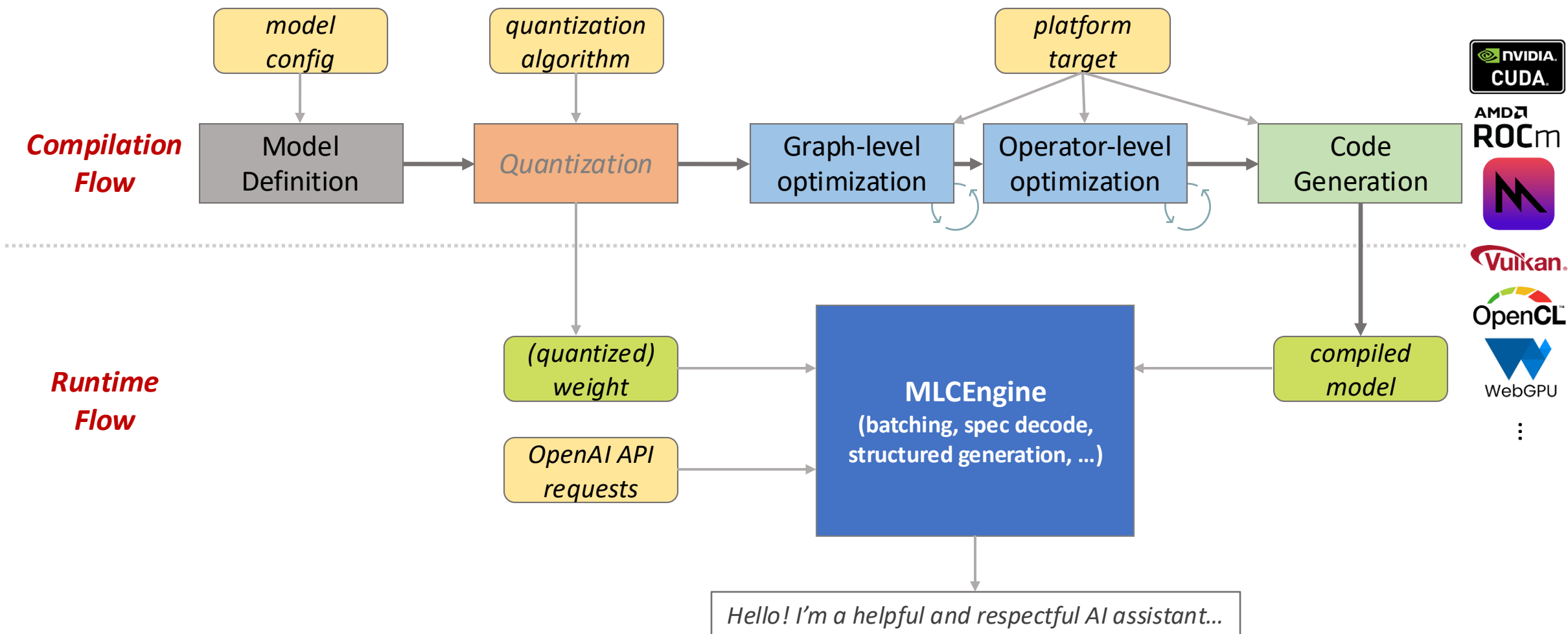
- iOS & Android App <https://llm.mlc.ai>



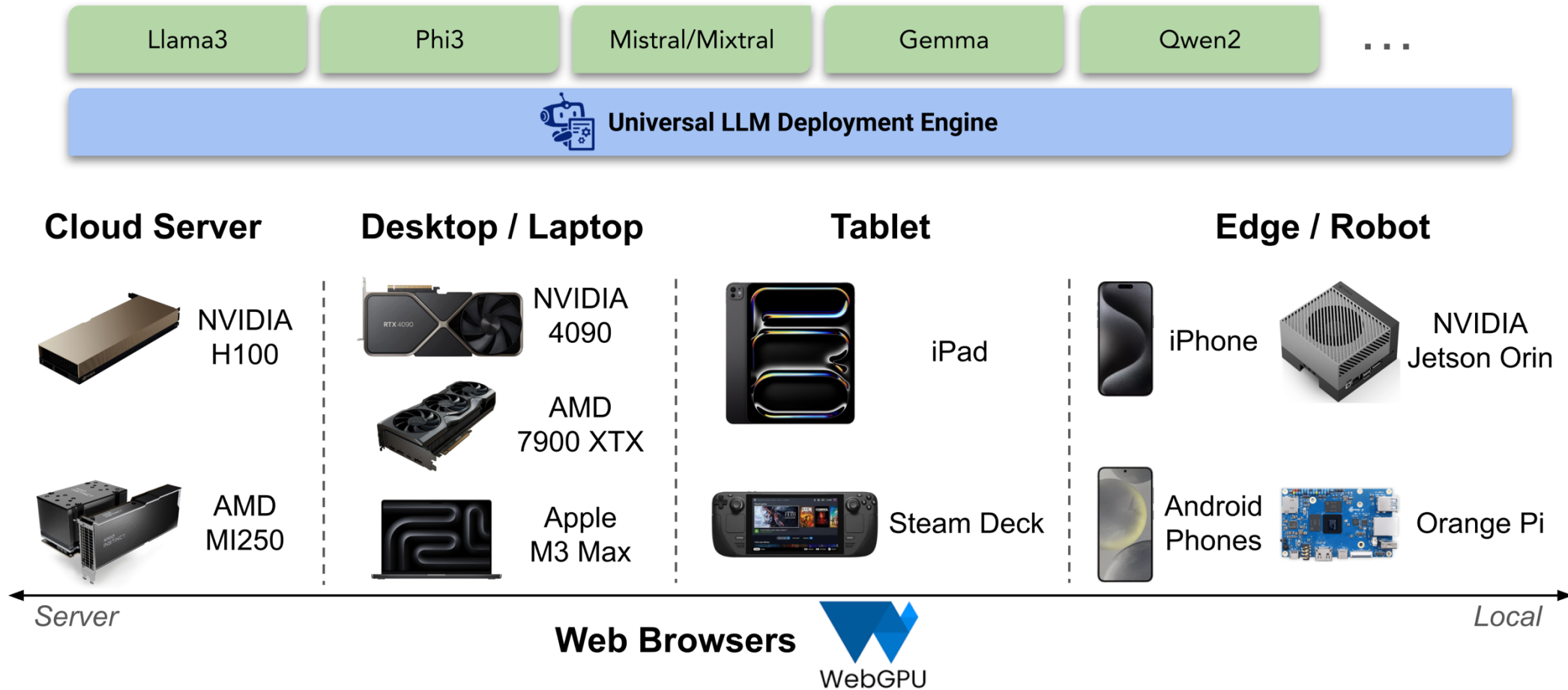
- Web LLM <https://webllm.mlc.ai>



Architecture of MLC LLM



Why Universal Engine?



APIs of MLCEngine

Core: OpenAI API

```
class ChatCompletionRequest(BaseModel):
    """OpenAI chat completion request protocol.
    API reference: https://platform.openai.com/docs/api-reference/chat/create
    """

    messages: List[ChatCompletionMessage]
    model: Optional[str] = None
    frequency_penalty: Optional[float] = None
    presence_penalty: Optional[float] = None
    logprobs: bool = False
    top_logprobs: int = 0
    logit_bias: Optional[Dict[int, float]] = None
    max_tokens: Optional[int] = None
    n: int = 1
    seed: Optional[int] = None
    stop: Optional[Union[str, List[str]]] = None
    stream: bool = False
    stream_options: Optional[StreamOptions] = None
    temperature: Optional[float] = None
    top_p: Optional[float] = None
    tools: Optional[List[ChatTool]] = None
    tool_choice: Optional[Union[Literal["none", "auto"], Dict]] = None
    user: Optional[str] = None
    response_format: Optional[RequestResponseFormat] = None
```

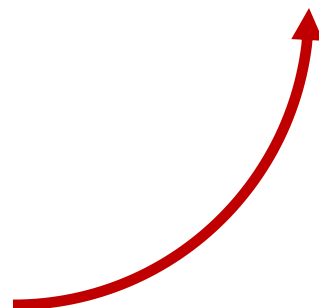
OpenAI API ChatCompletion

```
> mlc_llm serve HF://mlc-ai/Llama-3-8B-Instruct-q0f16-MLC
```

Launch Server

```
> curl -X POST \
  -H "Content-Type: application/json" \
  -d '{
    "messages": [
      {"role": "user", "content": "When is the recent solar eclipse in the U.S.?"}
    ]
  }' \
  http://127.0.0.1:8000/v1/chat/completions
```

Sending HTTP requests



APIs of MLCEngine

Core: OpenAI API

```
from openai import OpenAI

# Initialize client
client = OpenAI(base_url=OPENAI_BASE_URL)

# Run chat completion.
prompt = "What is the meaning of life?"
for response in client.chat.completions.create(
    model="gpt-4",
    messages=[{"role": "user", "content": prompt}],
    stream=True,
):
    ...
```

OpenAI Python package



```
from mlc_llm import MLCEngine

# Create engine
model = "HF://mlc-ai/Llama-3-8B-Instruct-q4f16_1-MLC"
engine = MLCEngine(model)

# Run chat completion in OpenAI API.
prompt = "What is the meaning of life?"
for response in engine.chat.completions.create(
    messages=[{"role": "user", "content": prompt}],
    stream=True,
):
    for choice in response.choices:
        print(choice.delta.content, end="", flush=True)
```

MLCEngine Python API

APIs of MLCEngine

Core: OpenAI API

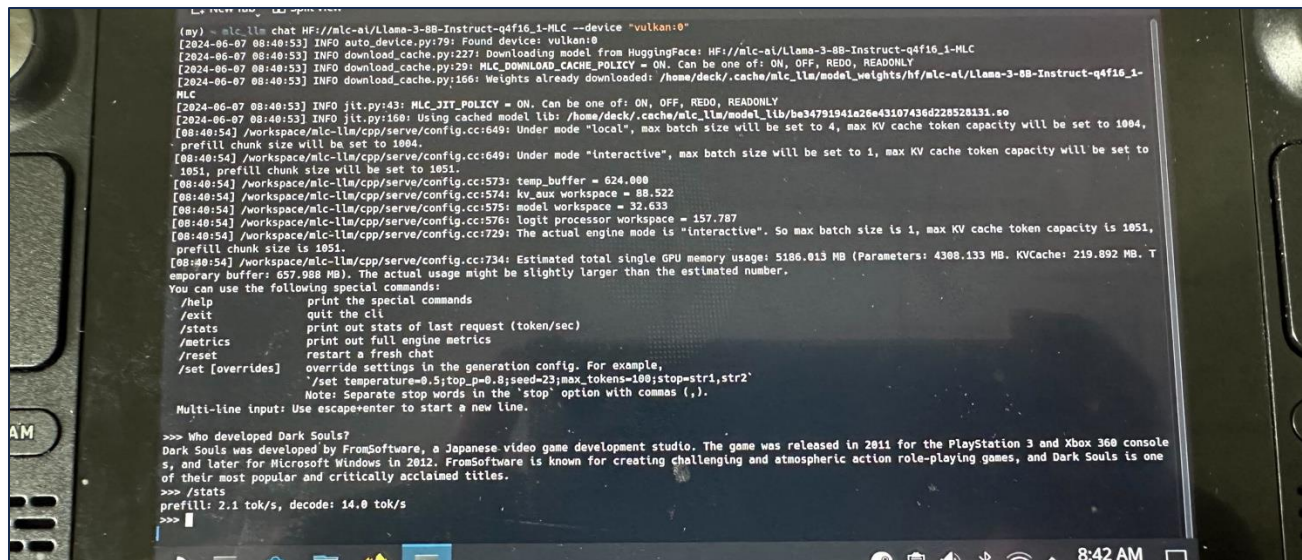
```
from mlc_llm import MLCEngine

# Create engine
model = "HF://mlc-ai/Llama-3-8B-Instruct-q4f16_1-MLC"
engine = MLCEngine(model)

# Run chat completion in OpenAI API.
prompt = "What is the meaning of life?"
for response in engine.chat.completions.create(
    messages=[{"role": "user", "content": prompt}],
    stream=True,
):
    for choice in response.choices:
        print(choice.delta.content, end="", flush=True)
```

MLCEngine Python API

```
> mlc_llm chat HF://mlc-ai/Llama-3-8B-Instruct-q4f16_1-MLC
```



```
(my) ~ mlc_llm chat HF://mlc-ai/Llama-3-8B-Instruct-q4f16_1-MLC --device "vulkan:0"
[2024-06-07 08:40:53] INFO auto_device.py:79: Found device: vulkan:0
[2024-06-07 08:40:53] INFO download_cache.py:227: Downloading model from HuggingFace: HF://mlc-ai/Llama-3-8B-Instruct-q4f16_1-MLC
[2024-06-07 08:40:53] INFO download_cache.py:29: MLC_DOWNLOAD_CACHE_POLICY = ON. Can be one of: ON, OFF, REDO, READONLY
[2024-06-07 08:40:53] INFO download_cache.py:166: Weights already downloaded: /home/deck/.cache/mlc_llm/model_weights/hf/mlc-ai/Llama-3-8B-Instruct-q4f16_1-MLC
[2024-06-07 08:40:53] INFO jit.py:43: MLC_JIT_POLICY = ON. Can be one of: ON, OFF, REDO, READONLY
[2024-06-07 08:40:53] INFO jit.py:160: Using cached model lib: /home/deck/.cache/mlc_llm/model_lib/be34791941e26e43107436d228528131.so
[08:40:54] /workspace/mlc-llm/cpp/serve/config.cc:649: Under mode "local", max batch size will be set to 4, max KV cache token capacity will be set to 1004,
prefill chunk size will be set to 1004.
[08:40:54] /workspace/mlc-llm/cpp/serve/config.cc:649: Under mode "interactive", max batch size will be set to 1, max KV cache token capacity will be set to
1051, prefill chunk size will be set to 1051.
[08:40:54] /workspace/mlc-llm/cpp/serve/config.cc:573: temp_buffer = 624.000
[08:40:54] /workspace/mlc-llm/cpp/serve/config.cc:574: kv_aux workspace = 88.522
[08:40:54] /workspace/mlc-llm/cpp/serve/config.cc:575: model workspace = 32.633
[08:40:54] /workspace/mlc-llm/cpp/serve/config.cc:576: logit processor workspace = 157.787
[08:40:54] /workspace/mlc-llm/cpp/serve/config.cc:729: The actual engine mode is "interactive". So max batch size is 1, max KV cache token capacity is 1051,
prefill chunk size is 1051.
[08:40:54] /workspace/mlc-llm/cpp/serve/config.cc:734: Estimated total single GPU memory usage: 5186.013 MB (Parameters: 4300.133 MB. KVCache: 219.892 MB. T
emporary buffer: 657.988 MB). The actual usage might be slightly larger than the estimated number.
You can use the following special commands:
/help          print the special commands
/exit         quit the cli
/stats        print out stats of last request (token/sec)
/metrics      print out full engine metrics
/reset        restart a fresh chat
/set [overrides] override settings in the generation config. For example,
               /set temperature=0.5;stop_p=0.8;seed=23;max_tokens=100;stop=str1,str2'
               Note: Separate stop words in the 'stop' option with commas (,).
Multi-line input: Use escape+enter to start a new line.

>>> Who developed Dark Souls?
Dark Souls was developed by FromSoftware, a Japanese video game development studio. The game was released in 2011 for the PlayStation 3 and Xbox 360 console
s, and later for Microsoft Windows in 2012. FromSoftware is known for creating challenging and atmospheric action role-playing games, and Dark Souls is one
of their most popular and critically acclaimed titles.
>>> /stats
prefill: 2.1 tok/s, decode: 14.0 tok/s
>>>
```

Interactive chat session

APIs of MLCEngine

Core: OpenAI API

```
from mlc_llm import MLCEngine

# Create engine
model = "HF://mlc-ai/Llama-3-8B-Instruct-q4f16_1-MLC"
engine = MLCEngine(model)

# Run chat completion in OpenAI API.
prompt = "What is the meaning of life?"
for response in engine.chat.completions.create(
    messages=[{"role": "user", "content": prompt}],
    stream=True,
):
    for choice in response.choices:
        print(choice.delta.content, end="", flush=True)
```

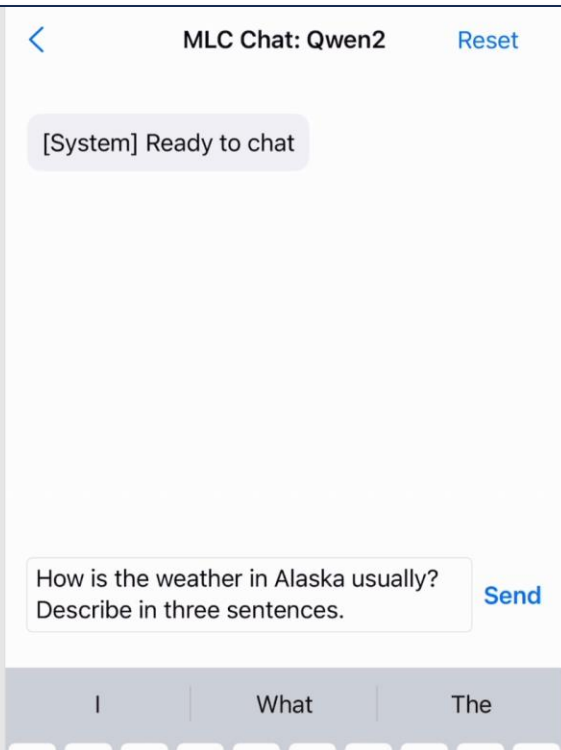
MLCEngine Python API

mimic →

```
func requestGenerate(prompt: String) {
    appendMessage(role: .user, message: prompt)
    appendMessage(role: .assistant, message: "")

    Task {
        self.historyMessages.append(
            ChatCompletionMessage(role: .user, content: prompt)
        )

        var finishReasonLength = false
        for await res in await engine.chat.completions.create(
            messages: self.historyMessages,
            stream_options: StreamOptions(include_usage: true)
        ) {
            for choice in res.choices {
                if let content = choice.delta.content {
                    self.streamingText += content.asText()
                }
                if let finish_reason = choice.finish_reason {
                    if finish_reason == "length" {
                        finishReasonLength = true
                    }
                }
            }
        }
    }
}
```



iOS SDK in Swift

APIs of MLCEngine

Core: OpenAI API

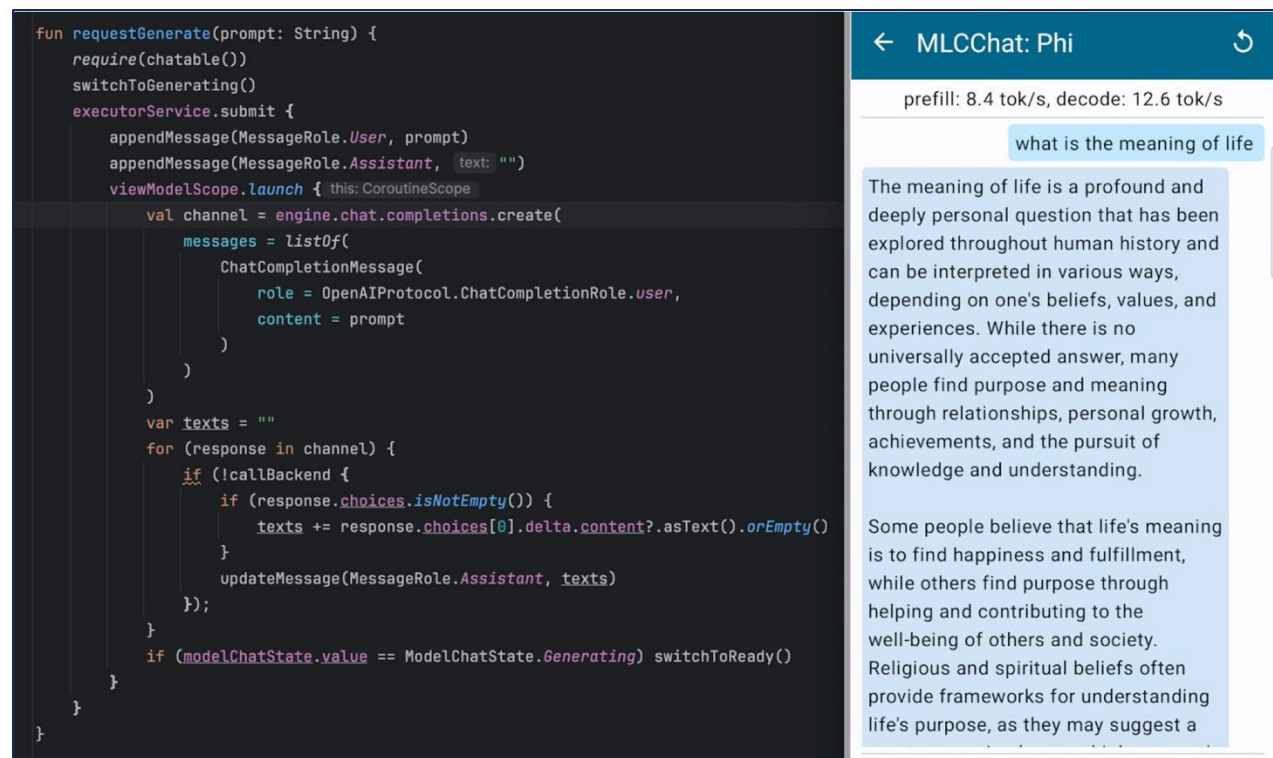
```
from mlc_llm import MLCEngine

# Create engine
model = "HF://mlc-ai/Llama-3-8B-Instruct-q4f16_1-MLC"
engine = MLCEngine(model)

# Run chat completion in OpenAI API.
prompt = "What is the meaning of life?"
for response in engine.chat.completions.create(
    messages=[{"role": "user", "content": prompt}],
    stream=True,
):
    for choice in response.choices:
        print(choice.delta.content, end="", flush=True)
```

MLCEngine Python API

mimic



The screenshot displays an Android application window titled "MLCChat: Phi". The interface includes a status bar at the top showing "prefill: 8.4 tok/s, decode: 12.6 tok/s". Below this, a text input field contains the prompt "what is the meaning of life". The main content area shows a chat response: "The meaning of life is a profound and deeply personal question that has been explored throughout human history and can be interpreted in various ways, depending on one's beliefs, values, and experiences. While there is no universally accepted answer, many people find purpose and meaning through relationships, personal growth, achievements, and the pursuit of knowledge and understanding. Some people believe that life's meaning is to find happiness and fulfillment, while others find purpose through helping and contributing to the well-being of others and society. Religious and spiritual beliefs often provide frameworks for understanding life's purpose, as they may suggest a...". The bottom of the screen shows a portion of the Kotlin code that implements the chat functionality, including a `requestGenerate` function that interacts with the MLCEngine.

```
fun requestGenerate(prompt: String) {
    require(chatable())
    switchToGenerating()
    executorService.submit {
        appendMessage(MessageRole.User, prompt)
        appendMessage(MessageRole.Assistant, text: "")
        viewModelScope.launch { this: CoroutineScope
            val channel = engine.chat.completions.create(
                messages = listOf(
                    ChatCompletionMessage(
                        role = OpenAIProtocol.ChatCompletionRole.user,
                        content = prompt
                    )
                )
            )
            var texts = ""
            for (response in channel) {
                if (!callBackend {
                    if (response.choices.isNotEmpty()) {
                        texts += response.choices[0].delta.content?.asText().orEmpty()
                    }
                }) {
                    updateMessage(MessageRole.Assistant, texts)
                }
            }
            if (modelChatState.value == ModelChatState.Generating) switchToReady()
        }
    }
}
```

Android SDK in Kotlin

APIs of MLCEngine

Core: OpenAI API

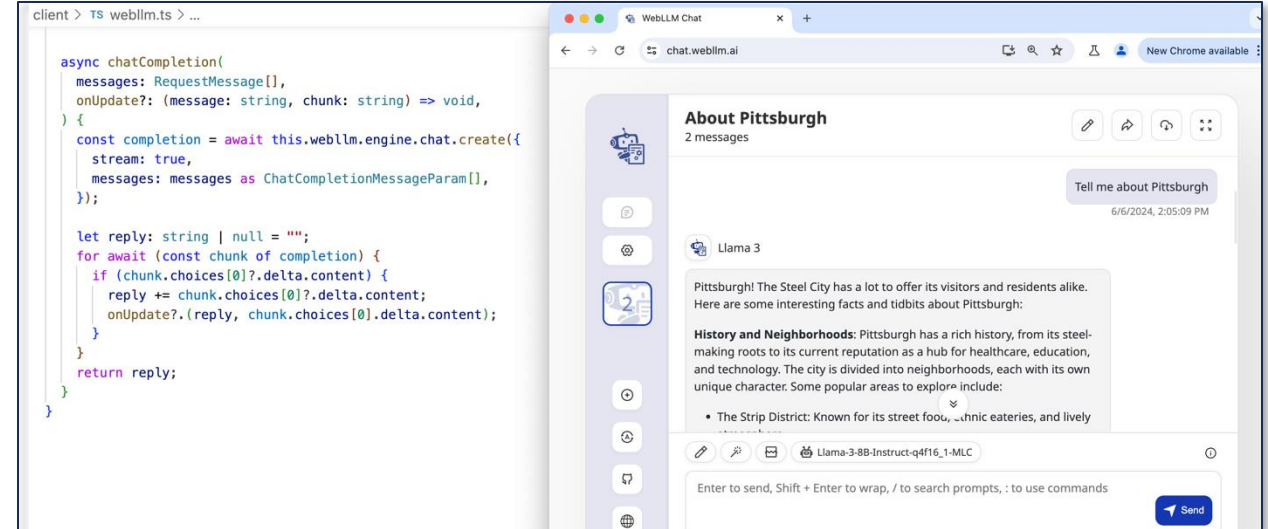
```
from mlc_llm import MLCEngine

# Create engine
model = "HF://mlc-ai/Llama-3-8B-Instruct-q4f16_1-MLC"
engine = MLCEngine(model)

# Run chat completion in OpenAI API.
prompt = "What is the meaning of life?"
for response in engine.chat.completions.create(
    messages=[{"role": "user", "content": prompt}],
    stream=True,
):
    for choice in response.choices:
        print(choice.delta.content, end="", flush=True)
```

MLCEngine Python API

mimic



WebLLM SDK in TypeScript

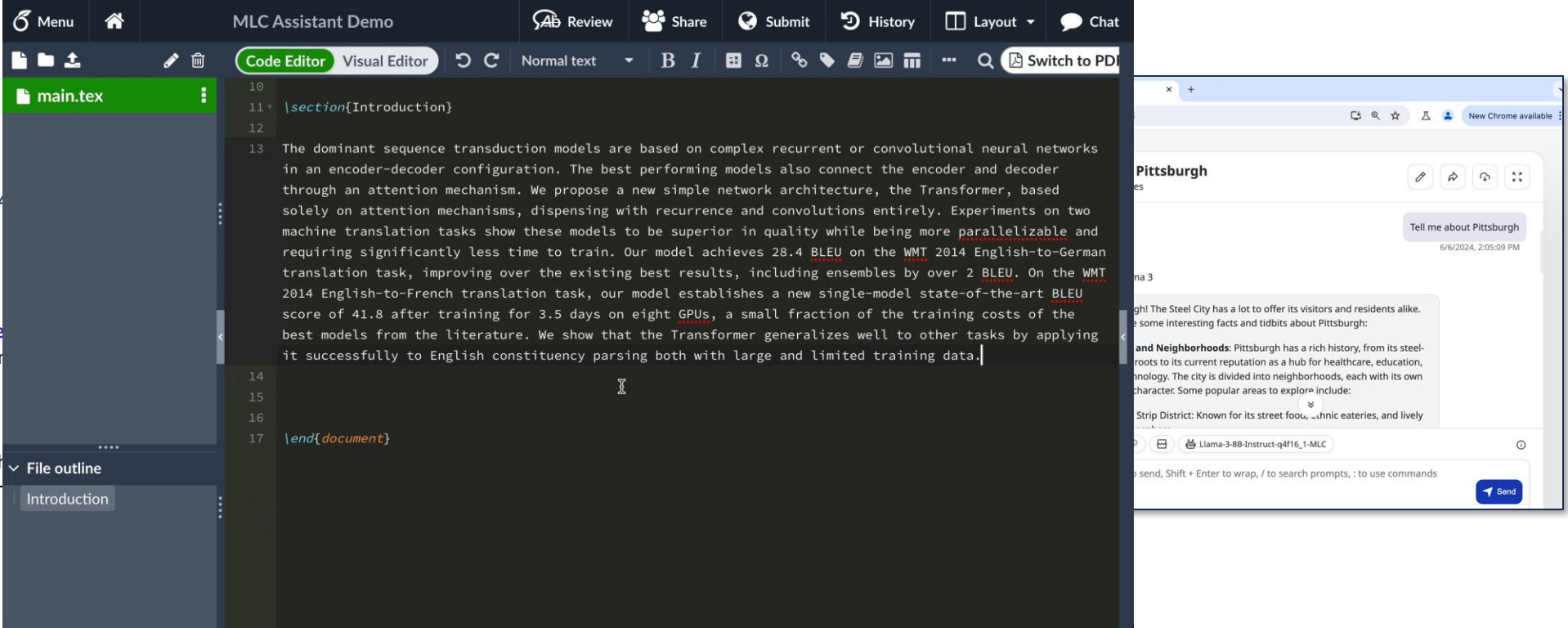
APIs of MLCEngine

Core: OpenAI API

```
from mlc_llm import MLCEngine

# Create engine
model = "HF://mlc-ai/Llama-3-8B-Instruct-q4f16_1-MLC"
engine = MLCEngine(model)

# Run chat completion in OpenAI API.
prompt = "What is the meaning of life?"
for response in engine.chat.completions.create(
    messages=[{"role": "user", "content": prompt}],
    stream=True,
):
    for choice in response.choices:
        print(choice.delta.content, end="", flush=True)
```



The screenshot displays the MLC Assistant Demo web interface. The top navigation bar includes links for Menu, Home, Review, Share, Submit, History, Layout, and Chat. Below this is a toolbar with icons for file operations, editing, and switching between Code Editor and Visual Editor. The main area is divided into two panes. The left pane shows a file explorer with 'main.tex' selected. The right pane is a code editor displaying LaTeX code for a document titled 'Introduction'. The code includes a section header and a paragraph of text about sequence transduction models. A file outline on the bottom left shows the 'Introduction' section. To the right of the code editor is a chat window titled 'Pittsburgh' with a text input field and a 'Send' button. The chat window shows a previous message and a response from the model.

<https://github.com/mlc-ai/Web-LLM-Assistant>

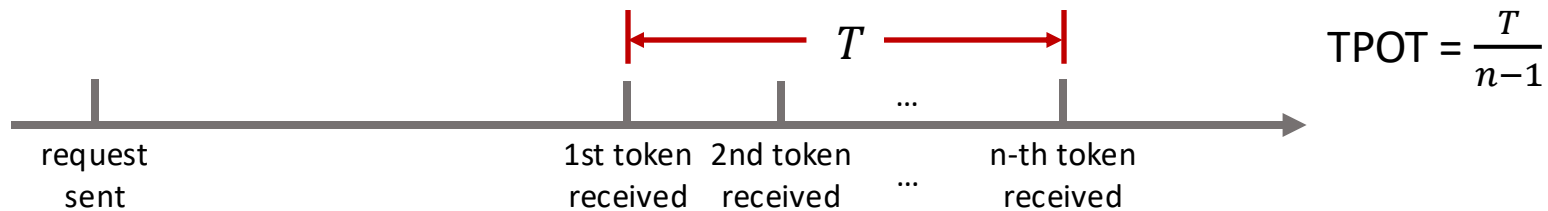
Outline

- Part I. Universal LLM deployment in MLCEngine
- Part II. Low-latency serving study in MLCEngine

Low-latency Serving

Latency is the Top Priority of MLC.

Reflected by TPOT (time per output token, the average number of tokens received per second after the first token is received)



Complicated tradeoffs between latency and throughput:

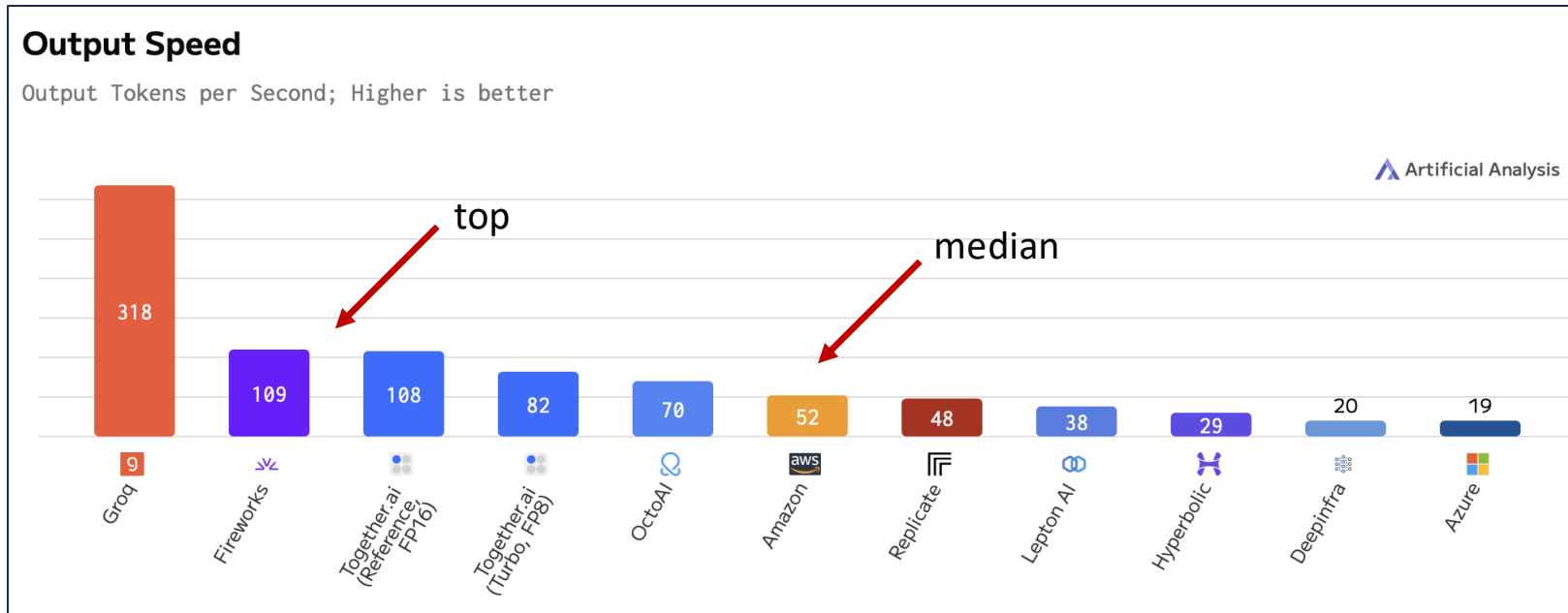
As an endpoint user:

- generation as fast as possible

As an endpoint provider:

- low latency for competitiveness
- reasonable throughput, or waste resource

Low-latency Serving



Llama3 70B provider latency leaderboard

Low-latency Serving

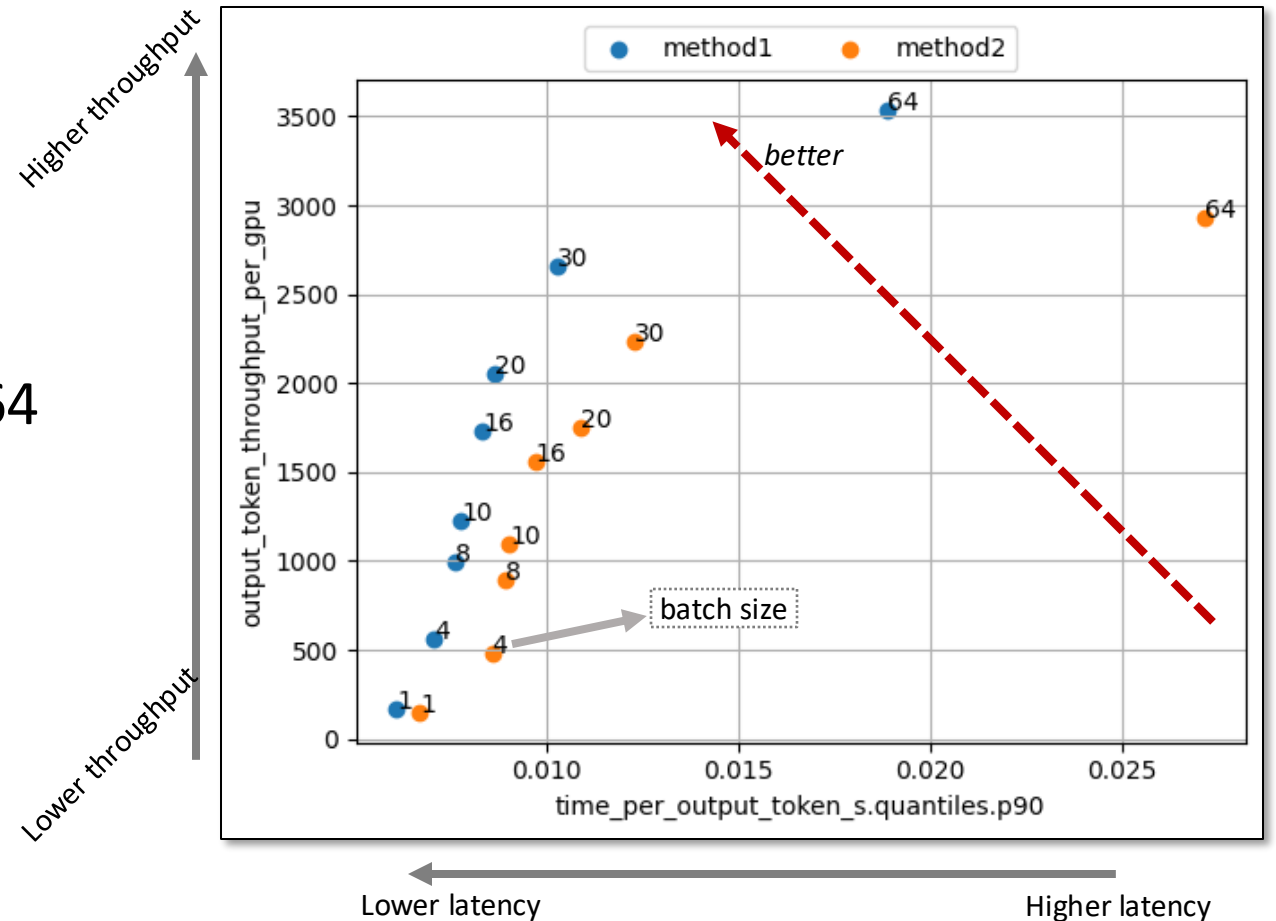
To understand the complication of latency and throughput...

Metrics

- Time-per-output-token (TPOT)
- Output throughput (tok/s)

Fix the request concurrency to be 1 to 64

- Dataset: ShareGPT
- GPU: 4 x H100
- 500 requests

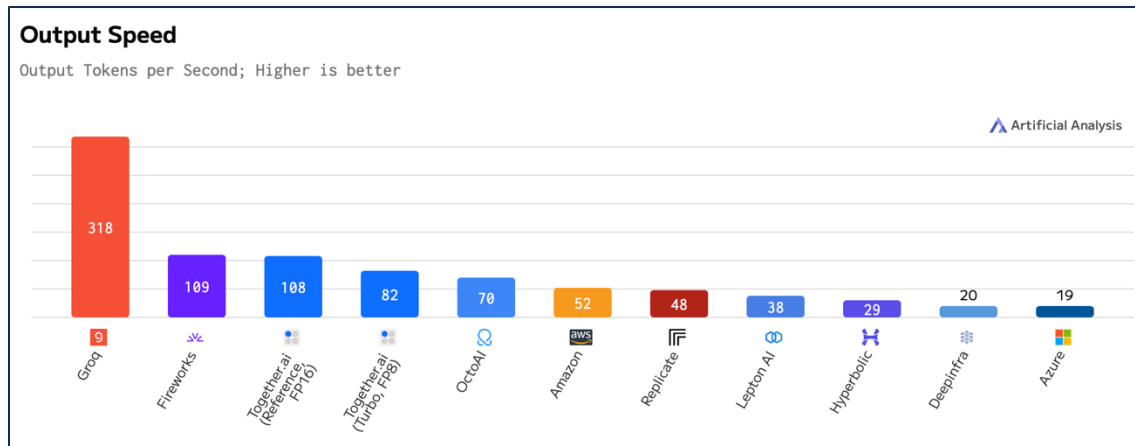


Low-latency Serving

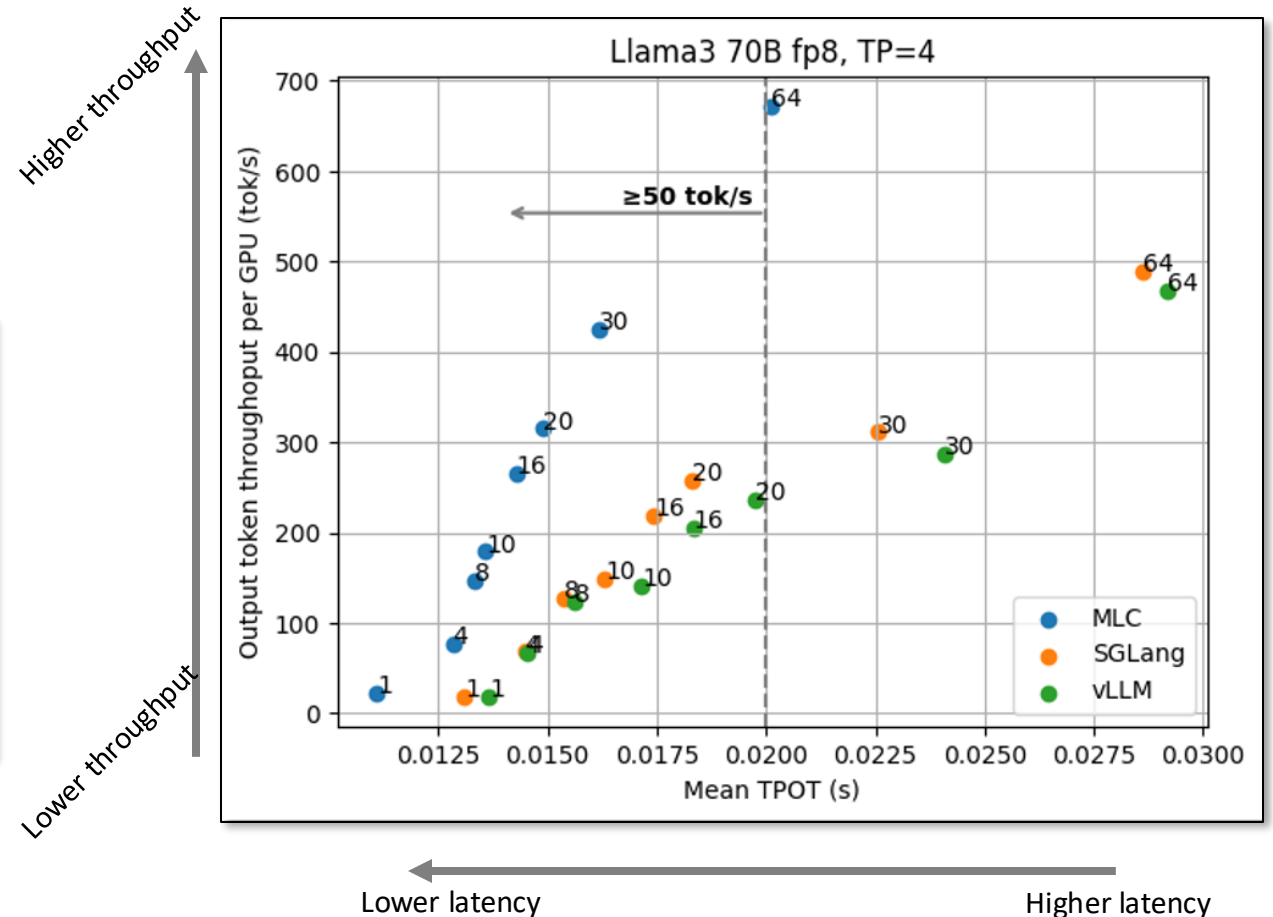
Fix the request concurrency to be 1 to 64

- Dataset: ShareGPT
- GPU: 4 x H100
- 500 requests

Overall 3% of CPU overhead in batch decode



Llama3 70B provider latency leaderboard

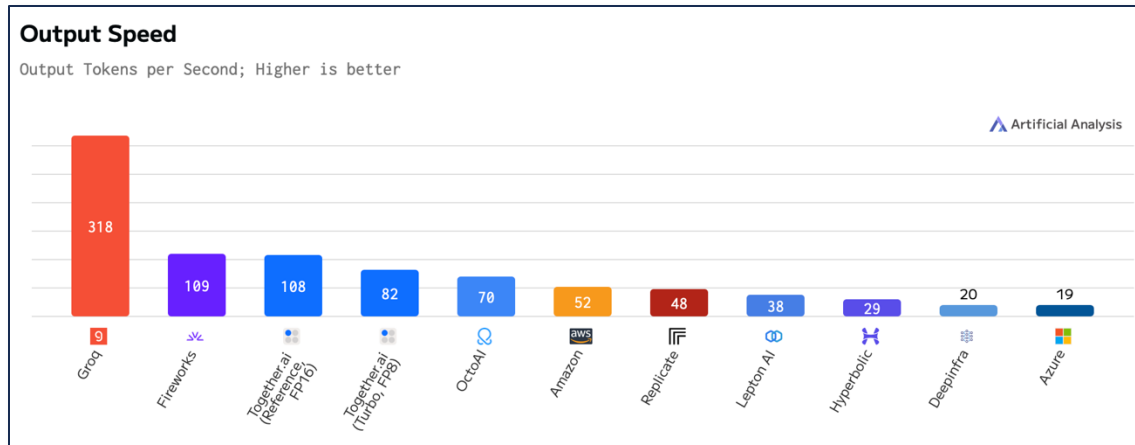


Impact of Tensor Parallelism

Running on 4 GPUs v.s. running on 8 GPUs

1. Reaching 100 tok/s

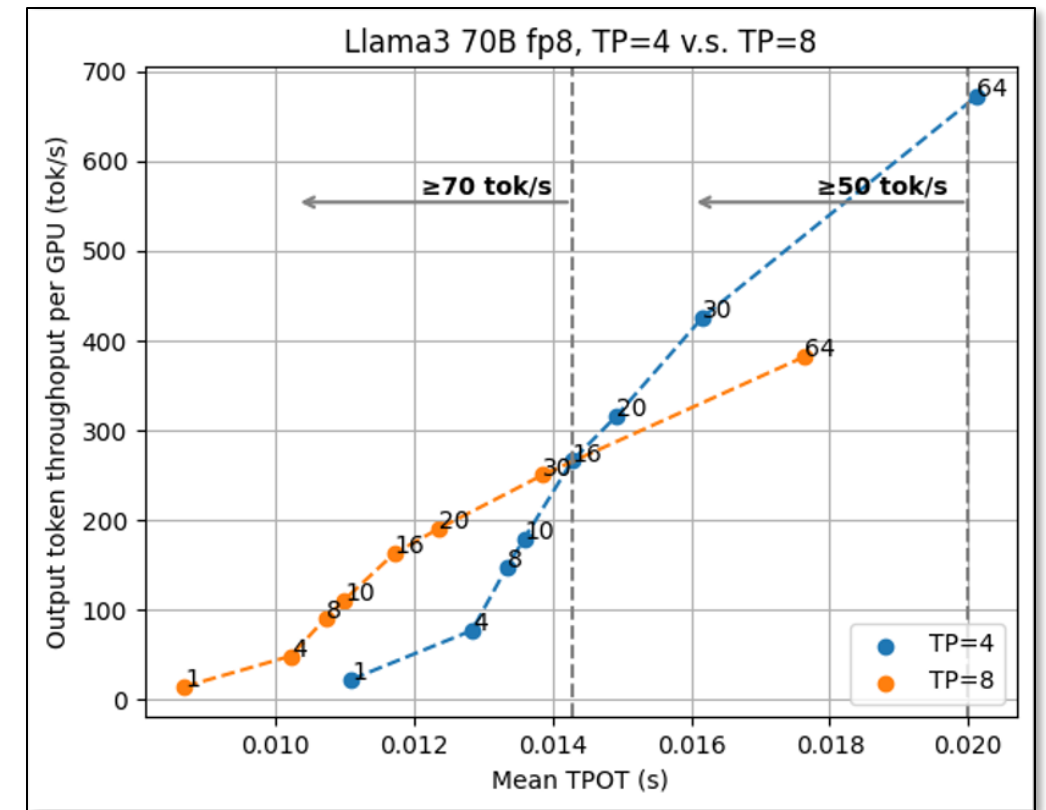
- But with poor throughput



Llama3 70B provider latency leaderboard

Higher throughput

Lower throughput



Lower latency

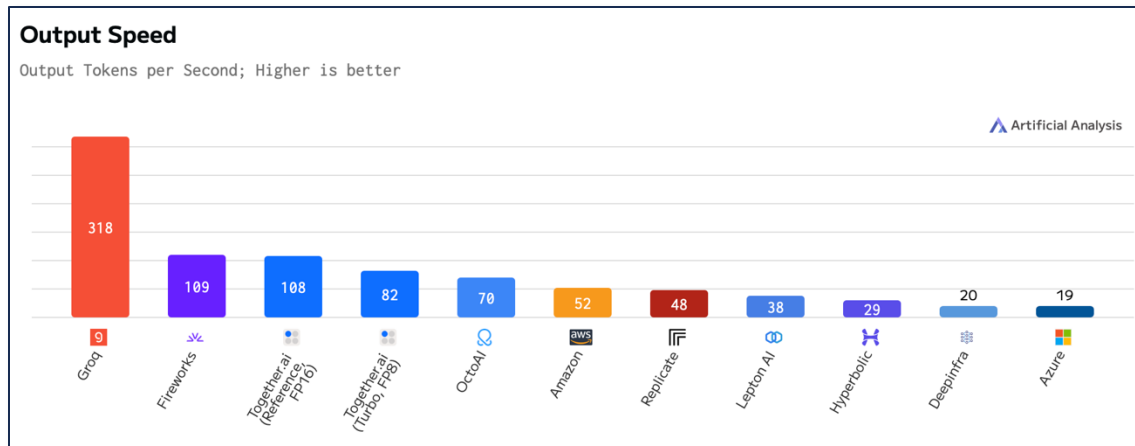
Higher latency

Impact of Tensor Parallelism

Running on 4 GPUs v.s. running on 8 GPUs

2. Choice is driven by the goal.

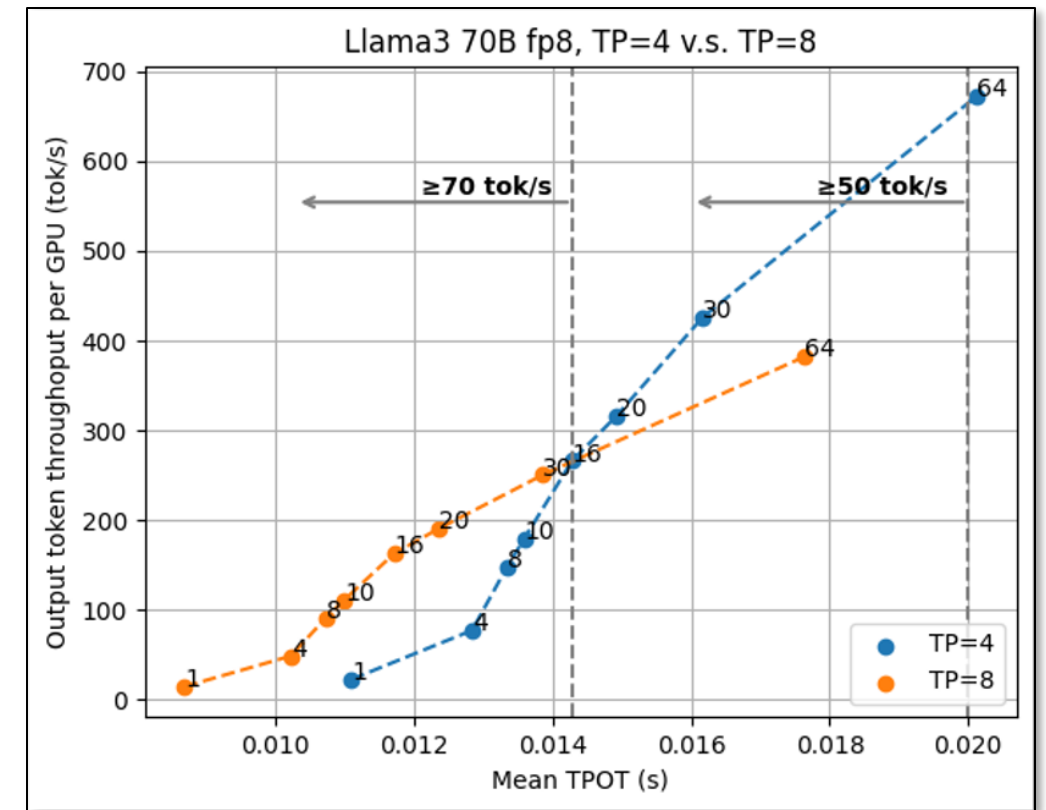
- ≥ 70 tok/s \rightarrow choose TP=8
- < 70 tok/s \rightarrow choose TP=4



Llama3 70B provider latency leaderboard

Higher throughput

Lower throughput



Lower latency

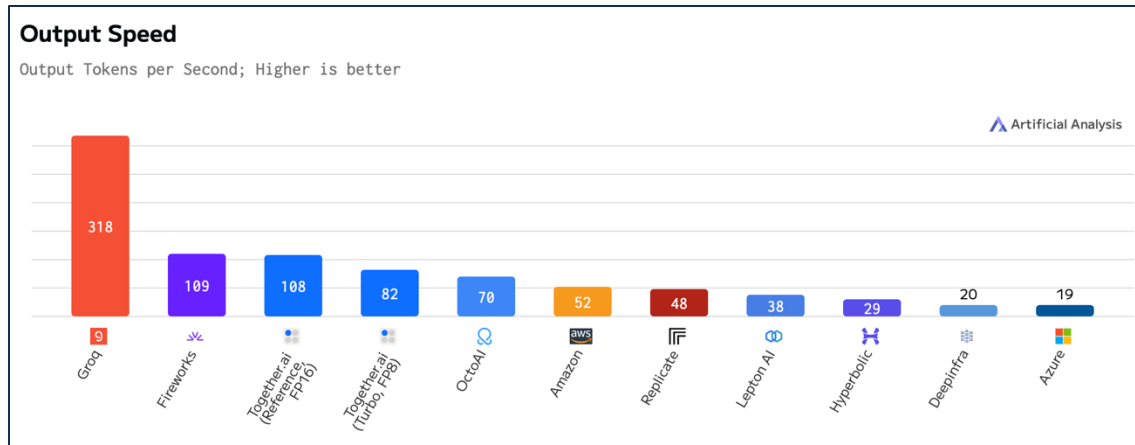
Higher latency

Impact of Tensor Parallelism

Running on 4 GPUs v.s. running on 8 GPUs

3. TP doesn't scale

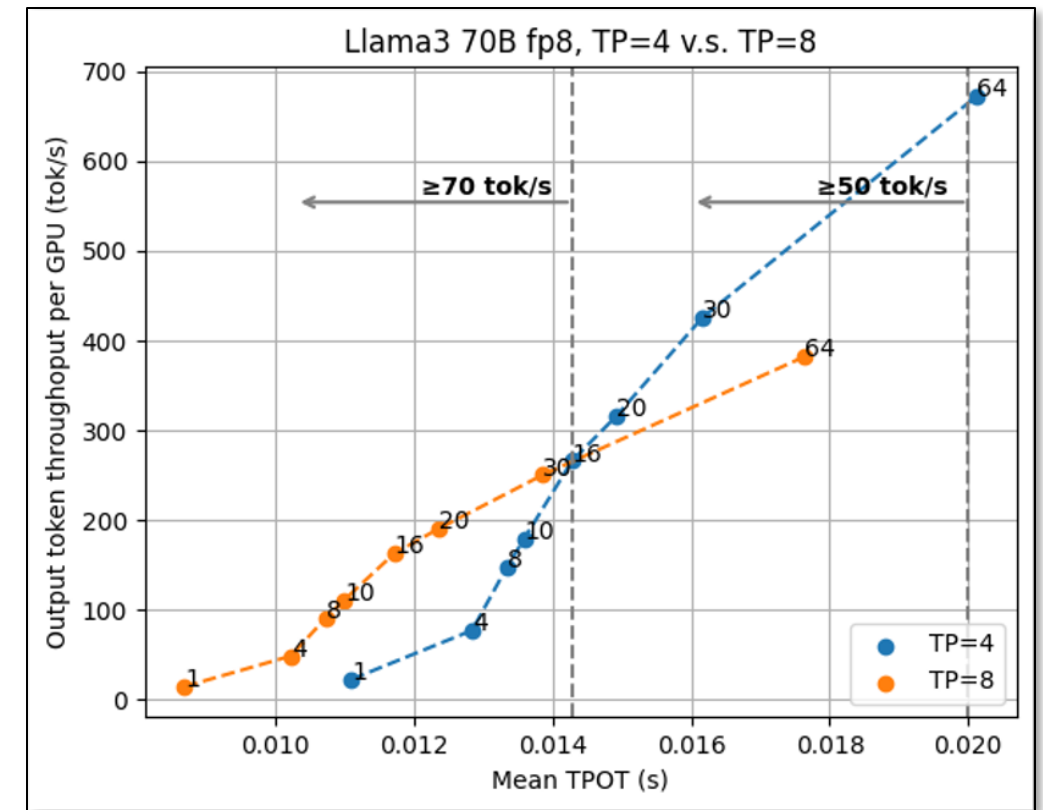
- Some GeMM/attention GPU kernels don't scale well linearly



Llama3 70B provider latency leaderboard

Higher throughput

Lower throughput



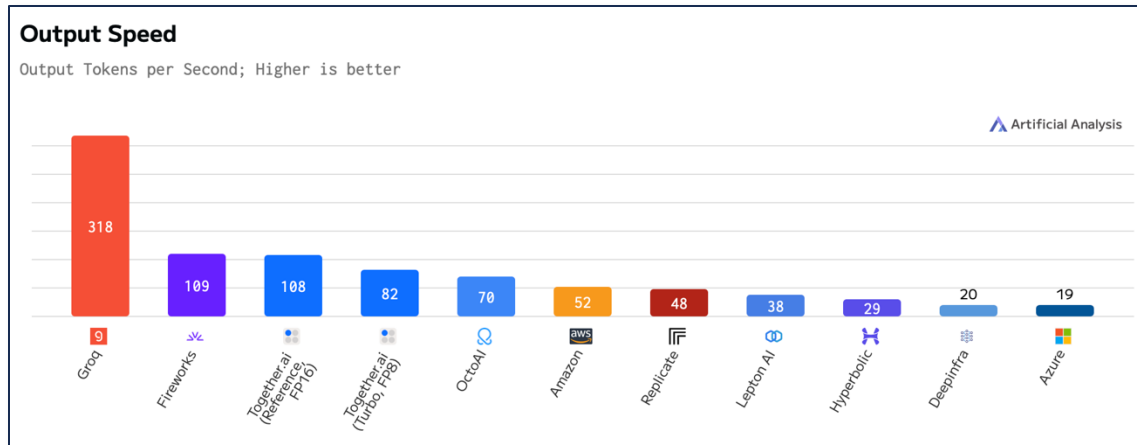
Lower latency

Higher latency

Impact of Speculative Decoding

Use Llama3 8B fp8 to speculate

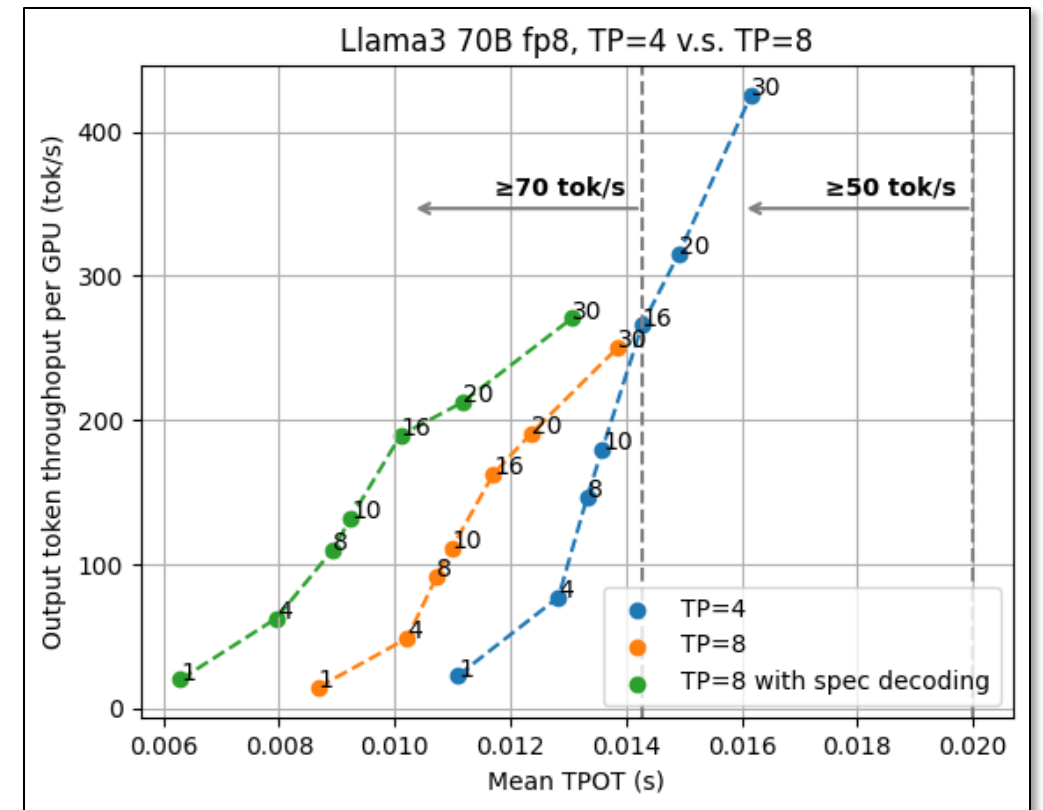
1. Speculative decoding helps further push to the extremity
 - Improvement marginalized by batch size



Llama3 70B provider latency leaderboard

Higher throughput

Lower throughput



Lower latency

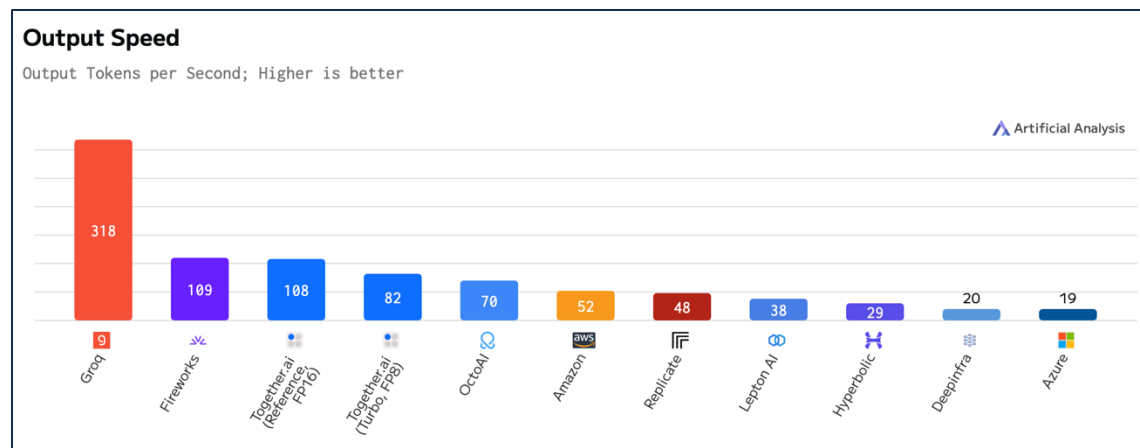
Higher latency

Impact of Speculative Decoding

Use Llama3 8B fp8 to speculate

2. System complexity for spec decoding

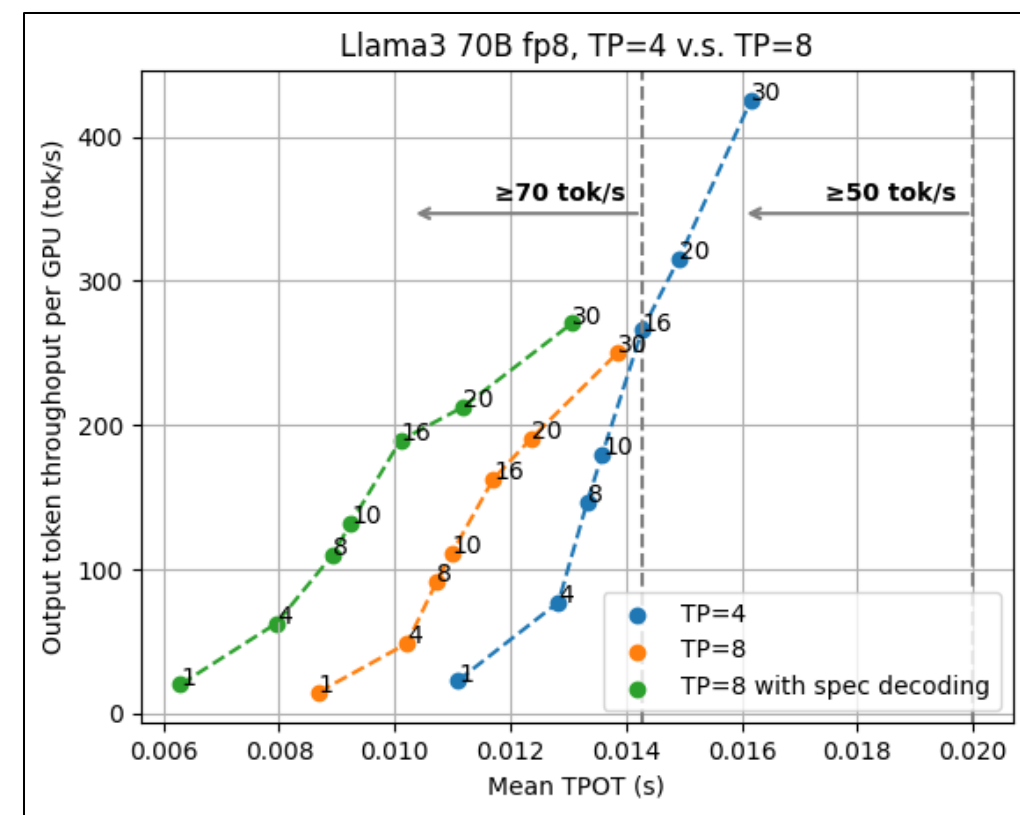
- Draft probability distribution management
- Efficient GPU verification kernel
- ...



Llama3 70B provider latency leaderboard

Higher throughput

Lower throughput



Lower latency

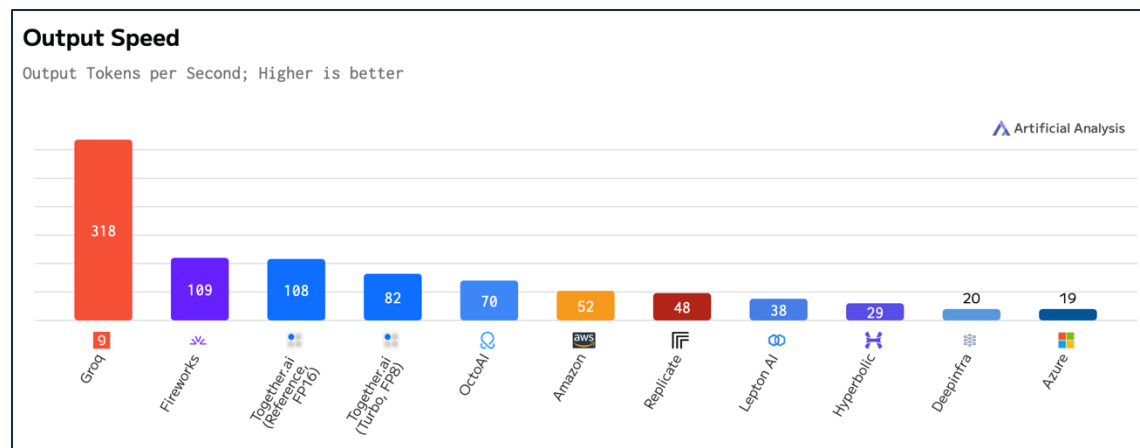
Higher latency

Impact of Speculative Decoding

Use Llama3 8B fp8 to speculate

3. The potential of being applied on mobile/edge platforms

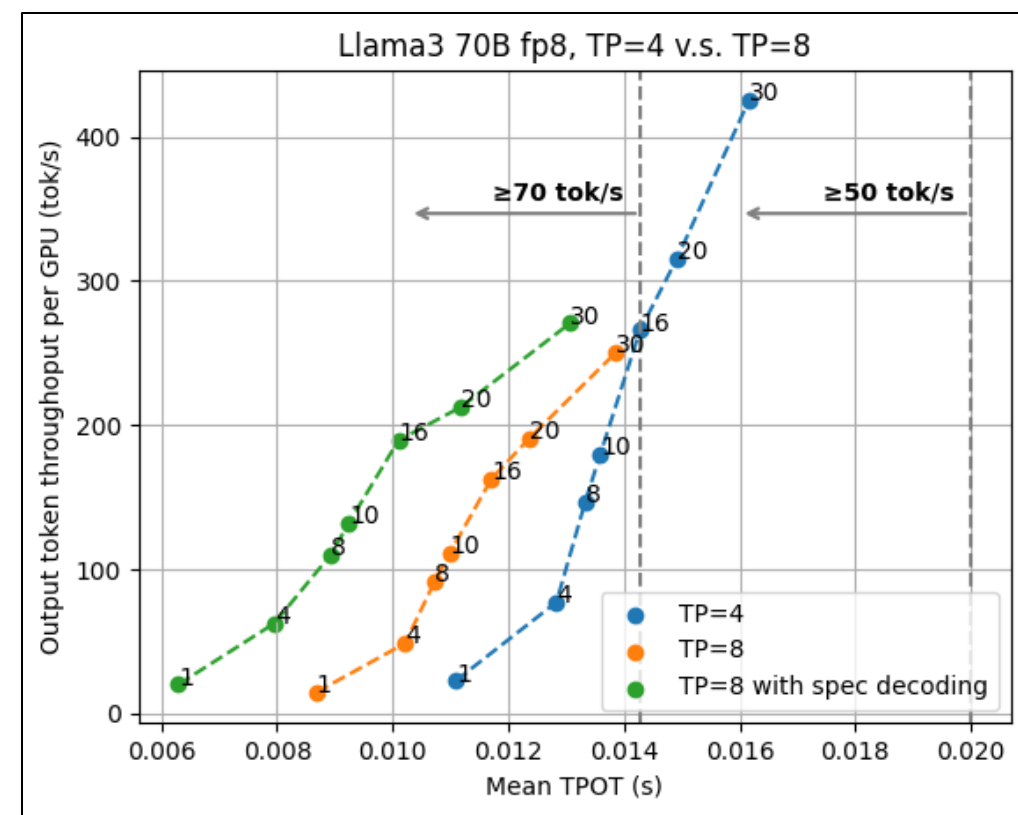
- Llama3 8B speculated with 1B: 80% 1-step acceptance on ShareGPT



Llama3 70B provider latency leaderboard

Higher throughput

Lower throughput

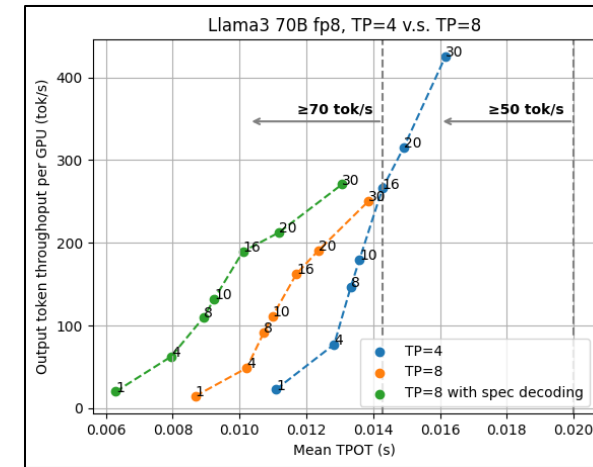


Lower latency

Higher latency

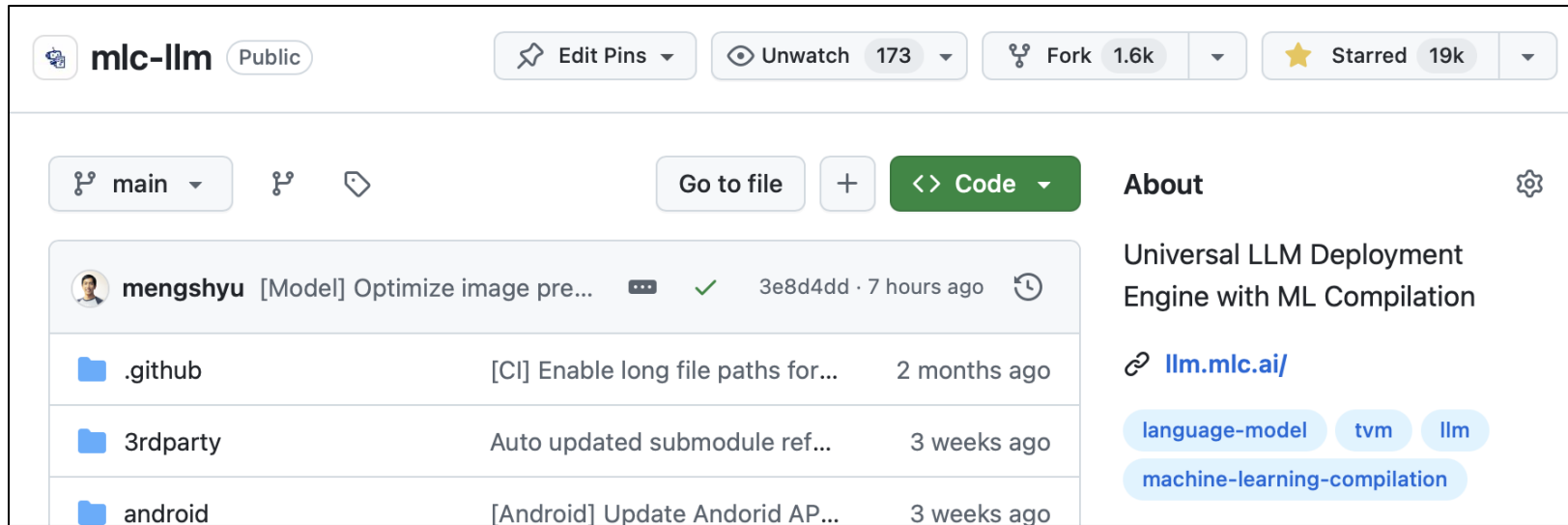
MLC LLM

- Today: Universal deployment and low-latency serving in MLCEngine



- Future roadmap
 - Multi-modality completeness across platforms
 - Other speculative decoding methods
 - And more...

MLC LLM



- GitHub: <https://github.com/mlc-ai/mlc-llm>
- Documentation: <https://llm.mlc.ai/docs/>
- HuggingFace: <https://huggingface.co/mlc-ai>
- Blogs: <https://blog.mlc.ai>
- Discord: <https://discord.com/invite/9Xpy2HGBuD>