vLLM의 OpenAI-Compatible Server 톺아보기

Hyogeun Oh

Speaker Introduction

Hygeun Oh
(오효근, Zerohertz)

- 기계공학 학/석사 후 Machine Learning Engineer (전문연구요원)로 재직 중입니다.
- Python과 Kubernetes를 주로 다루며, MLOps에 깊은 관심이 있습니다.
- Neovim을 애용하고 생산적인 개발 환경을 추구합니다.
- 더 나은 ML 파이프라인과 자동화를 고민합니다.

GitHub [1]

[1] https://github.com/Zerohertz

Sold Out, Thank You!

vLLM의 OpenAI-Compatible Server 톺아보기 [KR Only]

진행자: Hyogeun Oh
난이도: 중급(Intermediate)
진행 시간: 10:00 AM - 12:00 PM (2시간)

[설명]

최근 대규모 언어 모델(LLM)을 활용한 서비스가 급증함에 따라, 고성능·저지연·효율적인 LLM 서빙 기술에 대한 수요도 함께 증가하고 있습니다.
본 튜토리얼에서는 Meta의 LLaMA, Mistral, OpenChat 등 다양한 오픈소스 LLM을 OpenAI API 호환 형식으로 빠르게 서빙할 수 있는 vLLM 프레임워크를 중심으로 실습과 개념을 다룹니다.

상품이 매진되었어요!

Tutorial Code

- 노트북 가져오신 분들은 아래 QR의 README.md를 참고하여 환경 설정해주세요!
- 실행 환경이 없다면, GitHub에서 코드를 볼 수 있습니다.

Tutorial Code [2]

[2] https://github.com/Zerohertz/PyCon_KR_2025_Tutorial_vLLM

macOS MPS (Metal Performance Shaders)[3]

[3] https://github.com/vllm-project/vllm/issues/1441

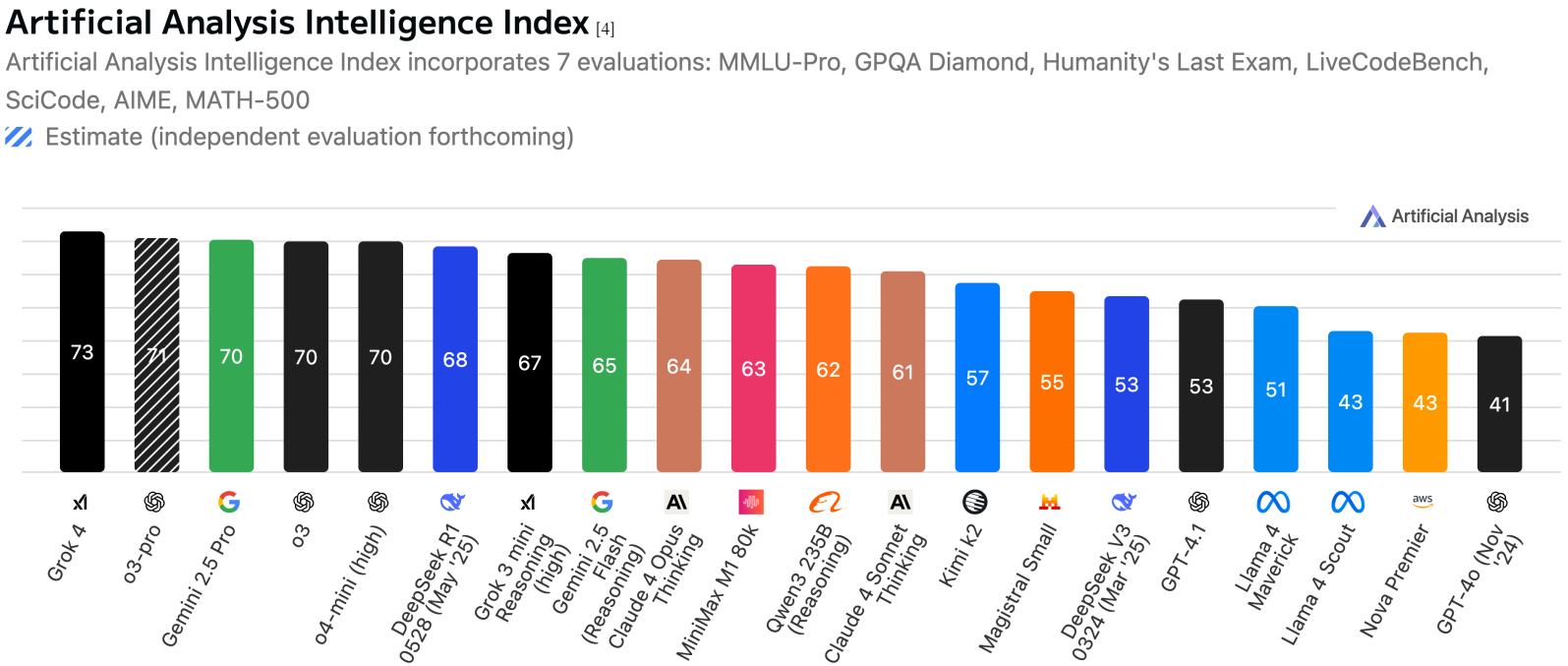
Table of Contents

PART 1

Introduction

Why Self-Host LLMs When Powerful Commercial APIs Exist?

- ❑ 비용 부담: traffic 사용량 증가 시 높은 과금
- ❑ Privacy/보안 이슈: 민감 data 외부 전송 불가
- ❑ Customizing의 어려움: prompt, parameter, 응답 format등 제한적 제어
- ❑ API 가용성: 속도, 안정성, 국가별 접근 제한 가능성



OpenAI
Gemini
Claude
Grok

[4] https://artificialanalysis.ai/

Why Self-Host LLMs When Powerful Commercial APIs Exist?

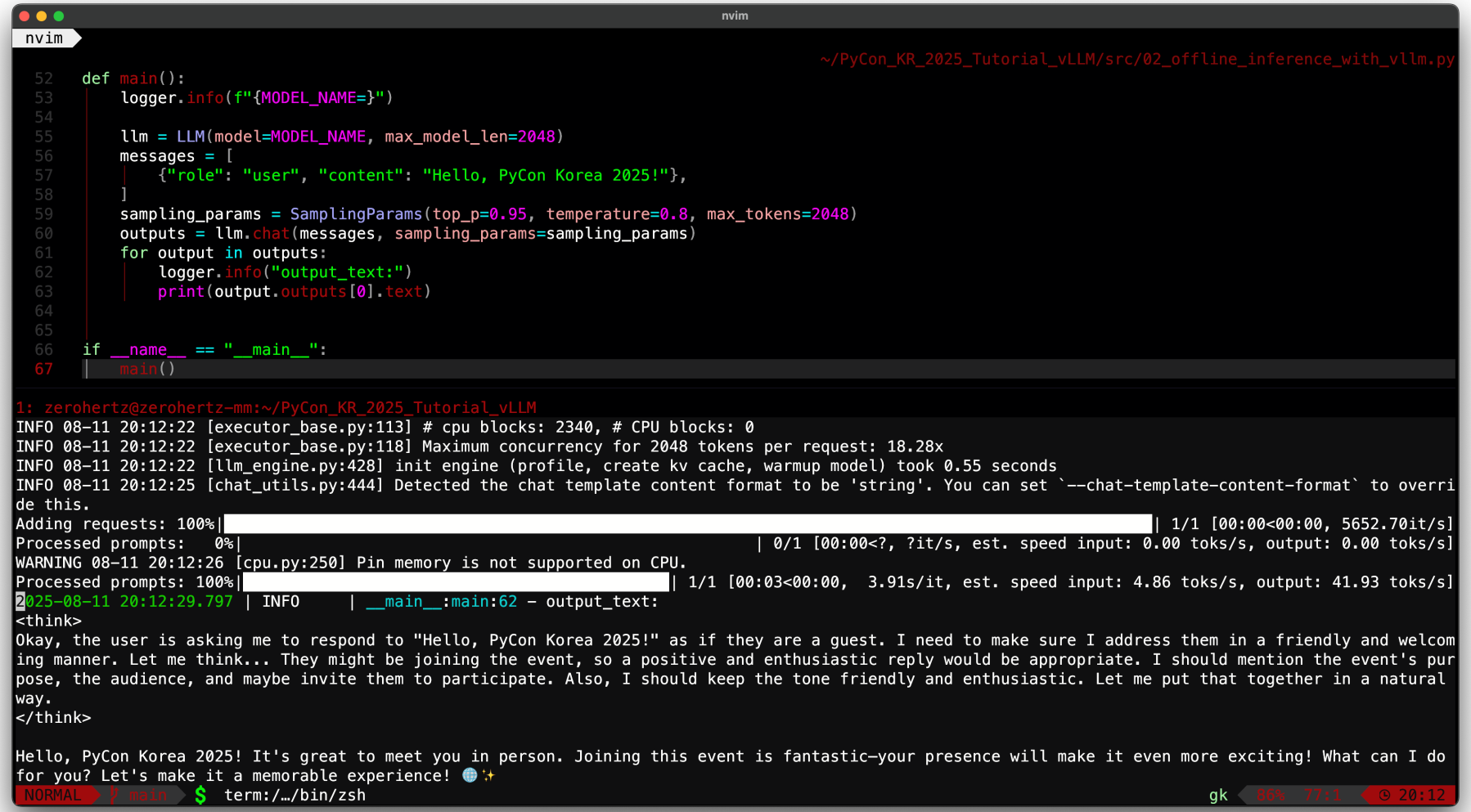
❑ transformers의 AutoModelForCausalLM을 통해 추론하면 되지 않나?

```python
def main():
    logger.info(f"{MODEL_NAME=}")

    processor = AutoProcessor.from_pretrained(MODEL_NAME)
    model = AutoModelForCausalLM.from_pretrained(MODEL_NAME)
    logger.info("Model & processor Loaded!")

    messages = [{"role": "user", "content": "Hello, PyCon Korea 2025!"}]
    prompt = processor.apply_chat_template(
        messages, tokenize=False, add_generation_prompt=True
    )
    logger.info("prompt:")
    print(prompt)
    inputs = processor(prompt, return_tensors="pt")

    with torch.no_grad():
        generated_ids = model.generate(
            **inputs,
            max_new_tokens=1024,
            do_sample=True,
            top_p=0.95,
            temperature=0.8,
            pad_token_id=processor.eos_token_id,
        )

    output_text = processor.batch_decode(generated_ids, skip_special_tokens=True)[0]
    logger.info("output_text:")
    print(output_text)
```

```
2025-07-24 23:51:30.031 | INFO     | __main__:main:31 - MODEL_NAME='Qwen/Qwen3-0.6B'
2025-07-24 23:51:33.554 | INFO     | __main__:main:35 - Model & processor Loaded!
2025-07-24 23:51:33.583 | INFO     | __main__:main:41 - prompt:
<|im_start|>user
Hello, PyCon Korea 2025!<|im_end|>
<|im_start|>assistant

2025-07-24 23:51:44.305 | INFO     | __main__:main:56 - output_text:
user
Hello, PyCon Korea 2025!
assistant
<think>
Okay, the user sent me a message saying "Hello, PyCon Korea 2025!" So I need to respond to that. Let me think. PyCon is a gl
obal event, so I should acknowledge it. Maybe mention that it's happening in Korea. Let me check the date again. Oh, PyCon K
orea 2025 is scheduled for June 10th, 2025. That's important to include.

I should make sure the response is friendly and enthusiastic. Maybe start with a greeting, then mention the event details. A
lso, since they asked for help, I should offer assistance. Let me put that all together in a natural way.
</think>

Hello, PyCon Korea 2025! 🌍🎉
We're excited to have you join us for the event! It's scheduled for June 10th, 2025, and we're looking forward to celebratin
g tech innovation and showcasing the world's best trends! Let me know if you need help or have any questions! 🚀
```

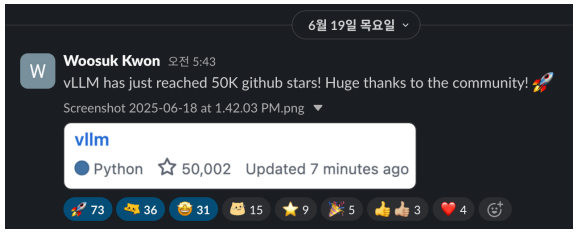
Why Self-Host LLMs When Powerful Commercial APIs Exist?

❑ vLLM의 LLM을 통해 추론하면 되지 않나?

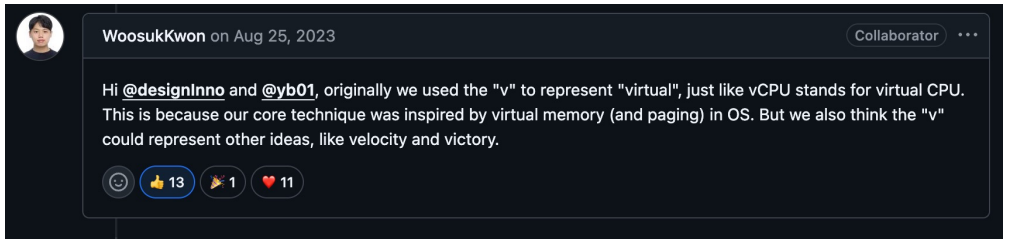


Why vLLM?

- vLLM history
  - 2023년 2월 9일, GitHub에서 CacheFlow [5]라는 이름으로 시작
  - 2023년 6월 17일, vLLM으로 이름 변경 [6]
  - 2023 9월 12일, UC Berkeley Sky Computing Lab에서 “Efficient Memory Management for Large Language Model Serving with PagedAttention” 논문 발표 [7]
  - 2025년 6월 19일, GitHub star 50k 달성



  - 2025년 7월 24일, v0.10.0 release
- License: Apache-2.0 [8]
- v of vLLM [9]



Efficient Memory Management for Large Language Model Serving with *PagedAttention*

Woosuk Kwon[1,*] Zhuohan Li[1,*] Siyuan Zhuang[1] Ying Sheng[1,2] Lianmin Zheng[1] Cody Hao Yu[3]
Joseph E. Gonzalez[1] Hao Zhang[4] Ion Stoica[1]
[1]UC Berkeley [2]Stanford University [3]Independent Researcher [4]UC San Diego

Abstract

High throughput serving of large language models (LLMs) requires batching sufficiently many requests at a time. However, existing systems struggle because the key-value cache (KV cache) memory for each request is huge and grows and shrinks dynamically. When managed inefficiently, this memory can be significantly wasted by fragmentation and redundant duplication, limiting the batch size. To address this problem, we propose PagedAttention, an attention algorithm inspired by the classical virtual memory and paging techniques in operating systems. On top of it, we build vLLM, an LLM serving system that achieves (1) near-zero waste in KV cache memory and (2) flexible sharing of KV cache within and across requests to further reduce memory usage. Our evaluations show that vLLM improves the throughput of popular LLMs by 2-4× with the same level of latency compared to the state-of-the-art systems, such as FasterTransformer and Orca. The improvement is more pronounced with longer sequences, larger models, and more complex decoding algorithms. vLLM's source code is publicly available at https://github.com/vllm-project/vllm.

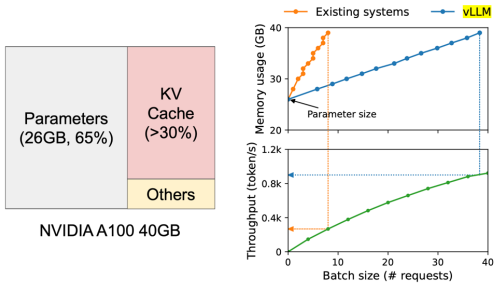
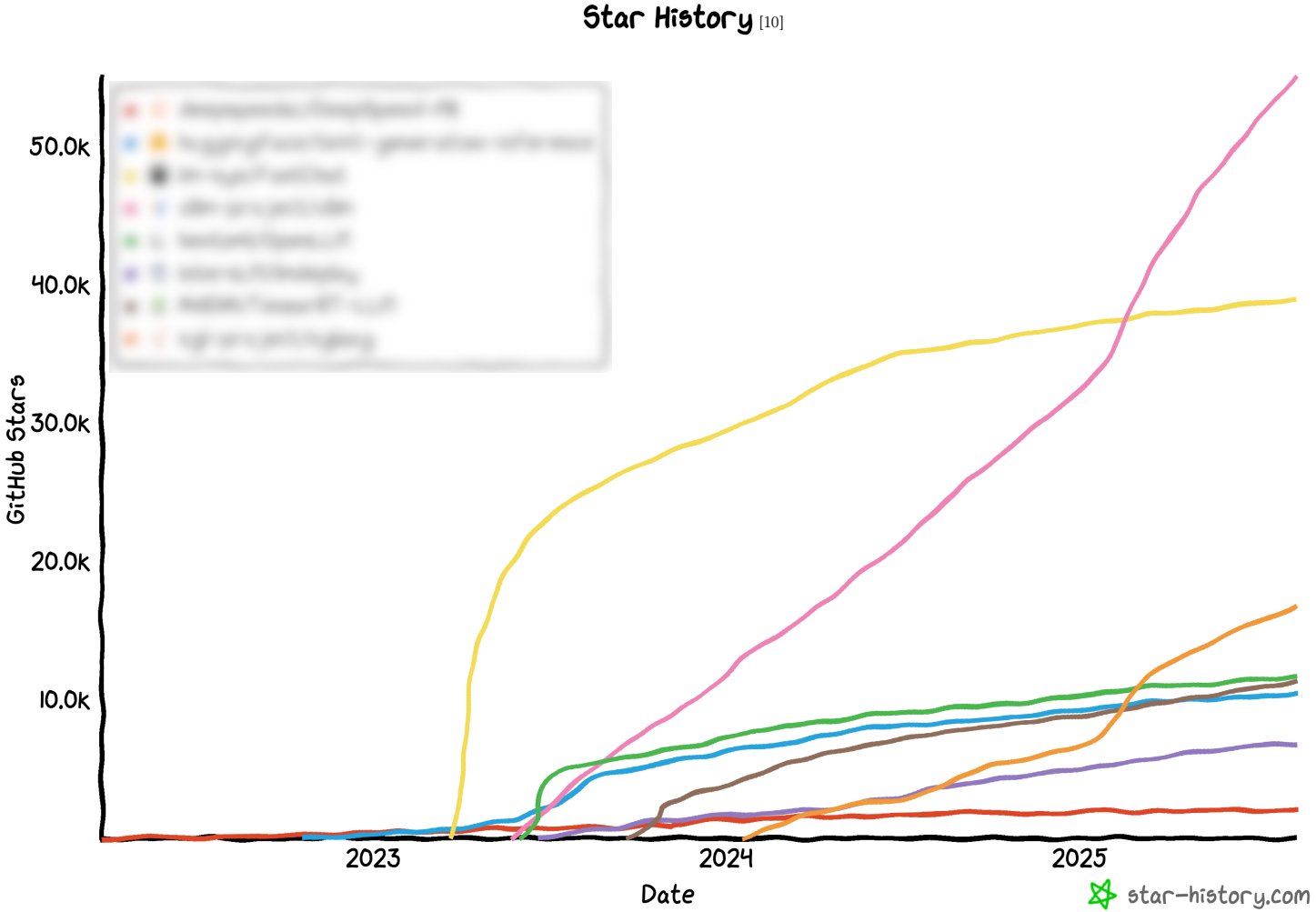


Figure 1. *Left:* Memory layout when serving an LLM with 13B parameters on NVIDIA A100. The parameters (gray) persist in GPU memory throughout serving. The memory for the KV cache (red) is (de)allocated per serving request. A small amount of memory (yellow) is used ephemerally for activation. *Right:* vLLM smooths out the rapid growth curve of KV cache memory seen in existing systems [31, 60], leading to a notable boost in serving throughput.

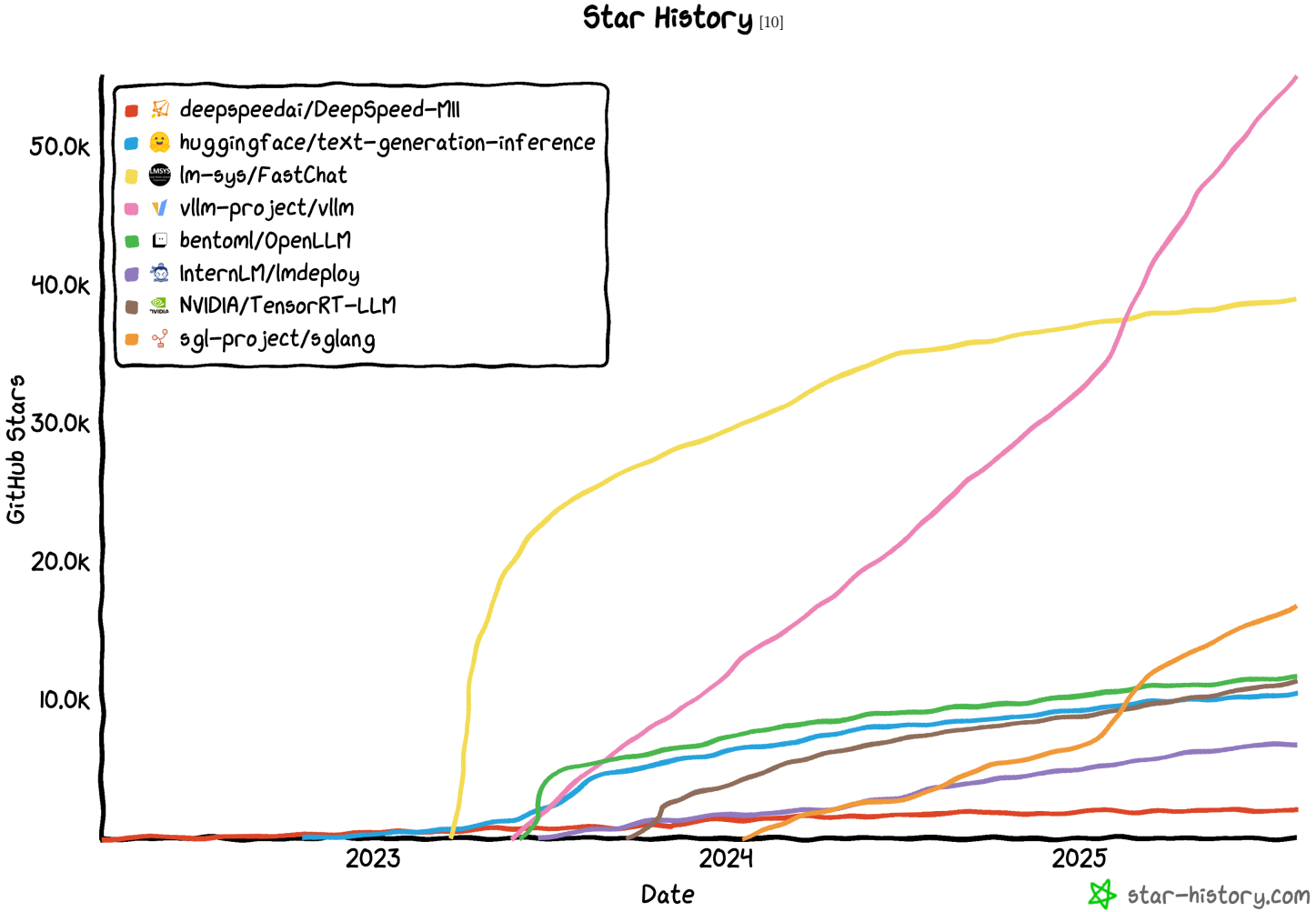
[5] https://github.com/vllm-project/vllm/commit/e7d9d9c08c79b386f6d0477e87b77a5723903l7d
[6] https://github.com/vllm-project/vllm/commit/0b98ba15c744f1dfb0ea4f2135e85ca23d572ae1
[7] https://arxiv.org/abs/2309.06180
[8] https://github.com/vllm-project/vllm/blob/main/LICENSE
[9] https://github.com/vllm-project/vllm/issues/835

Why vLLM?



[10] https://www.star-history.com/

Why vLLM?



[10] https://www.star-history.com/

Why vLLM?

❑ 초고속 LLM 서빙 성능
- **State-of-the-art throughput**: 대규모 요청 처리
- **Continuous batching**: 실시간 요청에 대한 효율적 처리

❑ 메모리 효율 및 최적화된 연산
- **PagedAttention** [7]: Attention key/value 메모리의 효율적 관리
- **Optimized CUDA kernels**: FlashAttention [11], FlashInfer [12] 등 최신 커널 통합
- **Chunked prefill** [13], **Speculative decoding** [14]

❑ 유연성 및 확장성
- **HuggingFace 통합**: 손쉽게 다양한 모델 서빙
- **다양한 디코딩 알고리즘**: 병렬 샘플링, 빔서치 등 고성능 추론 지원
- **분산 추론 지원**: Tensor, pipeline (Ray 기반), data, expert parallelism 지원

❑ 실용적 API 및 운영 편의성
- **OpenAI-Compatible API**: 기존 AI 서비스 (e.g., LangChain, Gemini CLI, ⋯)와 손쉽게 연동
- **Streaming output**: 스트리밍 방식 결과 제공
- **Prefix caching, Multi-LoRA**

[7] https://arxiv.org/abs/2309.06180
[11] https://github.com/vllm-project/flash-attention
[12] https://github.com/flashinfer-ai/flashinfer
[13] https://docs.vllm.ai/en/v0.9.2/configuration/optimization.html#chunked-prefill_1
[14] https://docs.vllm.ai/en/v0.9.2/features/spec_decode.html

How to serving LLM with vLLM?

- Installation
  - Local (CPU) 사용 시

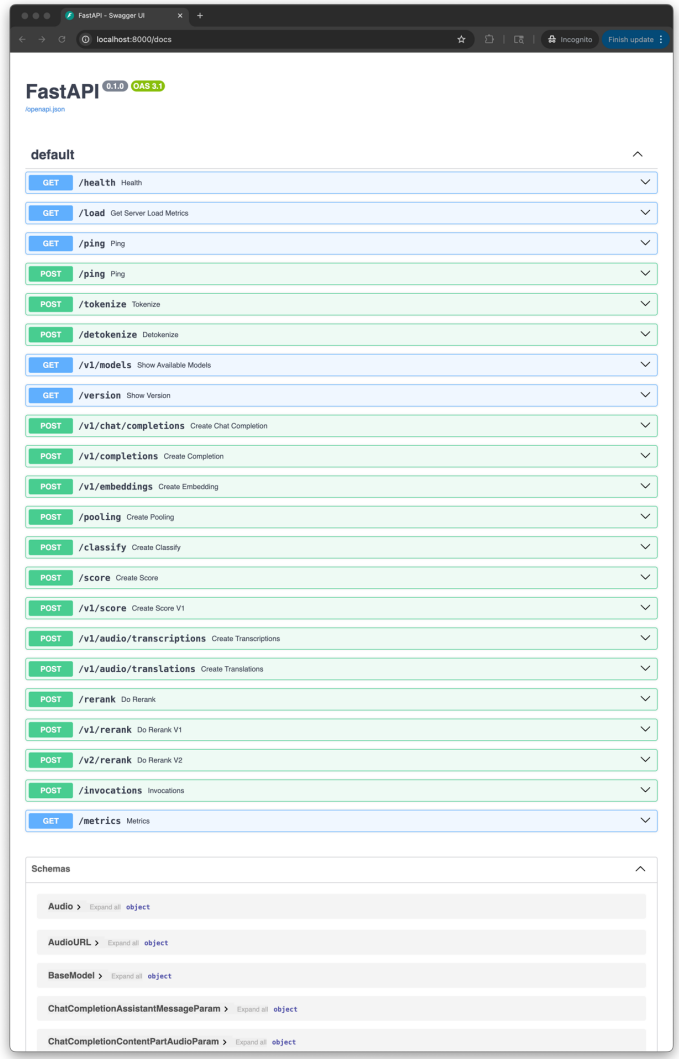
```
$ uv pip install vllm==0.9.2
Using Python 3.12.11 environment at: /opt/venv/main
Resolved 120 packages in 47ms
Installed 1 package in 10ms
 + vllm==0.9.2
```

  - GPU 서버 사용 시 [15]

```
docker run --runtime nvidia --gpus all \
    --name vllm \
    -v ~/.cache/huggingface:/root/.cache/huggingface \
    -p 8000:8000 \
    --ipc=host \
    vllm/vllm-openai:v0.9.2 \
    --model Qwen/Qwen3-0.6B \
    --max-model-len 8192
```

- vllm serve

```
$ vllm serve Qwen/Qwen3-0.6B --max-model-len 8192
INFO 07-27 01:38:11 [__init__.py:244] Automatically detected platform cpu.
INFO 07-27 01:38:12 [api_server.py:1395] vLLM API server version 0.9.2
INFO 07-27 01:38:12 [cli_args.py:325] non-default args: {'model': 'Qwen/Qwen3-0.6B', 'max_model_len': 8192}
INFO 07-27 01:38:15 [config.py:841] This model supports multiple tasks: {'reward', 'classify', 'generate', 'embed'}. Defaulting to 'generate'.
WARNING 07-27 01:38:15 [config.py:3320] Your device 'cpu' doesn't support torch.bfloat16. Falling back to torch.float16 for compatibility.
WARNING 07-27 01:38:15 [config.py:3371] Casting torch.bfloat16 to torch.float16.
INFO 07-27 01:38:15 [config.py:1472] Using max model len 8192
INFO 07-27 01:38:15 [arg_utils.py:1746] cpu is experimental on VLLM_USE_V1=1. Falling back to V0 Engine.
WARNING 07-27 01:38:15 [cpu.py:131] Environment variable VLLM_CPU_KVCACHE_SPACE (GiB) for CPU backend is not set, using 4 by default.
INFO 07-27 01:38:16 [api_server.py:268] Started engine process with PID 83075
INFO 07-27 01:38:17 [__init__.py:244] Automatically detected platform cpu.
INFO 07-27 01:38:18 [llm_engine.py:230] Initializing a V0 LLM engine (v0.9.2) with config: model='Qwen/Qwen3-0.6B', speculative_config=None, tokenizer='Qwen/Qwen3-0.6B', skip_tokenizer_init=False, tokenizer_mode=auto, revision=None, override_neuron
_config={}, tokenizer_revision=None, trust_remote_code=False, dtype=torch.float16, max_seq_len=8192, download_dir=None, load_format=LoadFormat.AUTO, tensor_parallel_size=1, pipeline_parallel_size=1, disable_custom_all_reduce=True, quantization=None
, enforce_eager=False, kv_cache_dtype=auto,  device_config=cpu, decoding_config=DecodingConfig(backend='auto', disable_fallback=False, disable_any_whitespace=False, disable_additional_properties=False, reasoning_backend=''), observability_config=Ob
servabilityConfig(show_hidden_metrics_for_version=None, otlp_traces_endpoint=None, collect_detailed_traces=None), seed=0, served_model_name=Qwen/Qwen3-0.6B, num_scheduler_steps=1, multi_step_stream_outputs=True, enable_prefix_caching=None, chunked_
prefill_enabled=False, use_async_output_proc=False, pooler_config=None, compilation_config={"level":0,"debug_dump_path":"","cache_dir":"","backend":"","custom_ops":[],"splitting_ops":[],"use_inductor":true,"compile_sizes":[],"inductor_compile_confi
g":{"enable_auto_functionalized_v2":false},"inductor_passes":{},"use_cudagraph":true,"cudagraph_num_of_warmups":0,"cudagraph_capture_sizes":[],"cudagraph_copy_inputs":false,"full_cuda_graph":false,"max_capture_size":256,"local_cache_dir":null}, use
_cached_outputs=True,
INFO 07-27 01:38:23 [launcher.py:37] Route: /v1/models, Methods: GET
INFO 07-27 01:38:23 [launcher.py:37] Route: /version, Methods: GET
INFO 07-27 01:38:23 [launcher.py:37] Route: /v1/chat/completions, Methods: POST
INFO 07-27 01:38:23 [launcher.py:37] Route: /v1/completions, Methods: POST
INFO 07-27 01:38:23 [launcher.py:37] Route: /v1/embeddings, Methods: POST
INFO 07-27 01:38:23 [launcher.py:37] Route: /pooling, Methods: POST
INFO 07-27 01:38:23 [launcher.py:37] Route: /classify, Methods: POST
INFO 07-27 01:38:23 [launcher.py:37] Route: /score, Methods: POST
INFO 07-27 01:38:23 [launcher.py:37] Route: /v1/score, Methods: POST
INFO 07-27 01:38:23 [launcher.py:37] Route: /v1/audio/transcriptions, Methods: POST
INFO 07-27 01:38:23 [launcher.py:37] Route: /v1/audio/translations, Methods: POST
INFO 07-27 01:38:23 [launcher.py:37] Route: /rerank, Methods: POST
INFO 07-27 01:38:23 [launcher.py:37] Route: /v1/rerank, Methods: POST
INFO 07-27 01:38:23 [launcher.py:37] Route: /v2/rerank, Methods: POST
INFO 07-27 01:38:23 [launcher.py:37] Route: /invocations, Methods: POST
INFO 07-27 01:38:23 [launcher.py:37] Route: /metrics, Methods: GET
INFO:     Started server process [83039]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```



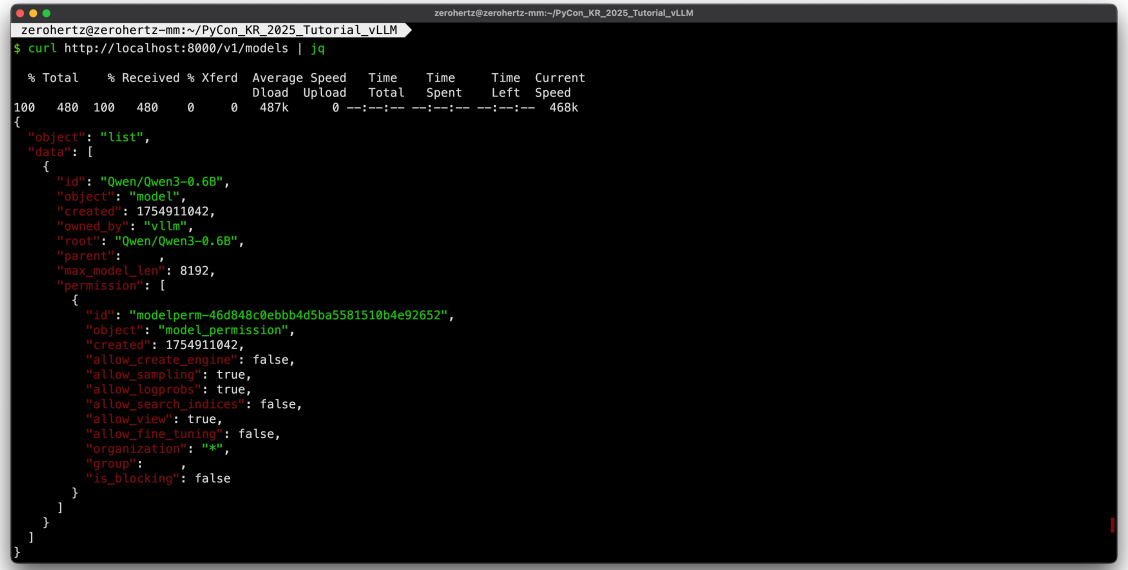
[15] https://docs.vllm.ai/en/v0.9.2/deployment/docker.html

PART 2

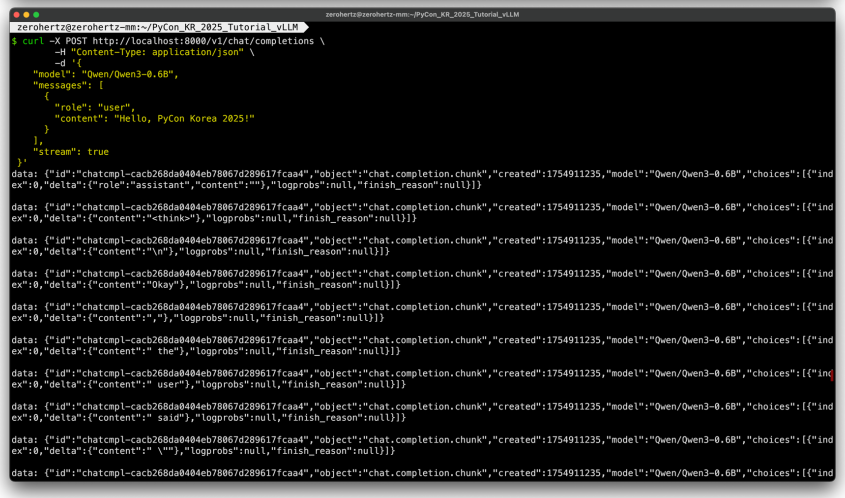
OpenAI-Compaitble Server

OpenAI API Spec [16]

❑ /v1/models [17]



❑ /v1/chat/completions [18]



[16] https://docs.vllm.ai/en/v0.9.2/serving/openai_compatible_server.html
[17] https://platform.openai.com/docs/api-reference/models/list
[18] https://platform.openai.com/docs/api-reference/chat/create

Tool calling

❑ OpenAI 규격에 맞춰 호출

v1.97.1 openai-python / src / openai / types / shared_params / function_definition.py [19]

stainless-app[bot] chore(api): event shapes more accurate

Code Blame 45 lines (32 loc) · 1.47 KB · Raw

```python
# File generated from our OpenAPI spec by Stainless. See CONTRIBUTING.md for details.

from __future__ import annotations

from typing import Optional
from typing_extensions import Required, TypedDict

from .function_parameters import FunctionParameters

__all__ = ["FunctionDefinition"]


class FunctionDefinition(TypedDict, total=False):
    name: Required[str]
    """The name of the function to be called.

    Must be a-z, A-Z, 0-9, or contain underscores and dashes, with a maximum length
    of 64.
    """

    description: str
    """
    A description of what the function does, used by the model to choose when and
    how to call the function.
    """

    parameters: FunctionParameters
    """The parameters the functions accepts, described as a JSON Schema object.

    See the [guide](https://platform.openai.com/docs/guides/function-calling) for
    examples, and the
    [JSON Schema reference](https://json-schema.org/understanding-json-schema/) for
    documentation about the format.

    Omitting `parameters` defines a function with an empty parameter list.
    """

    strict: Optional[bool]
    """Whether to enable strict schema adherence when generating the function call.

    If set to true, the model will follow the exact schema defined in the
    `parameters` field. Only a subset of JSON Schema is supported when `strict` is
    `true`. Learn more about Structured Outputs in the
    [function calling guide](https://platform.openai.com/docs/guides/function-calling).
    """
```

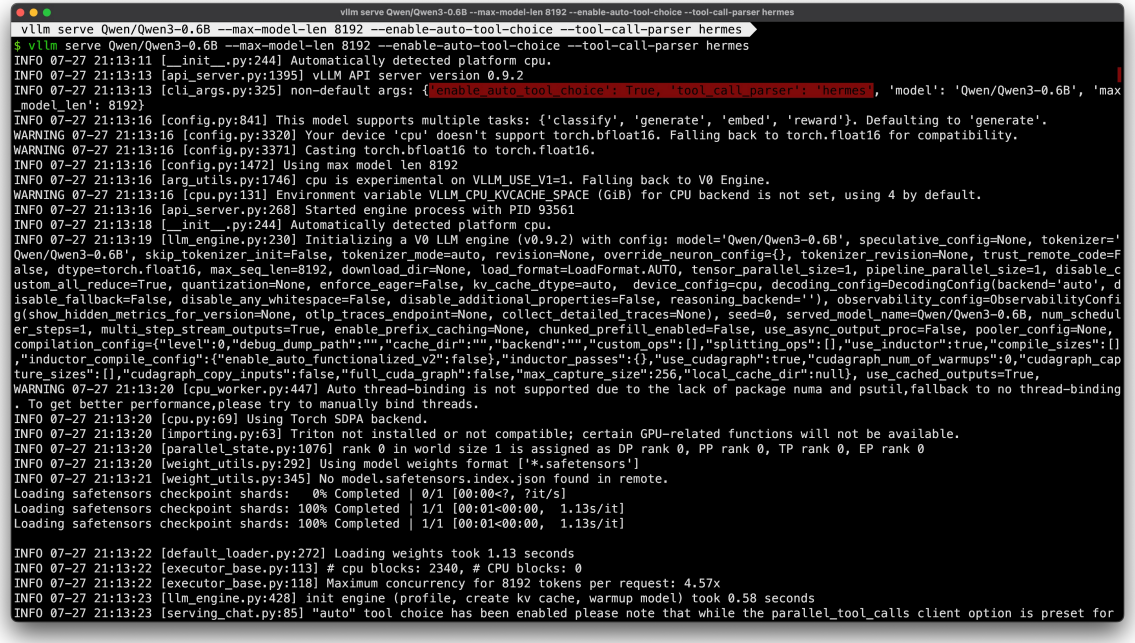
zerohertz@zerohertz-mm:~/PyCon_KR_2025_Tutorial_vLLM

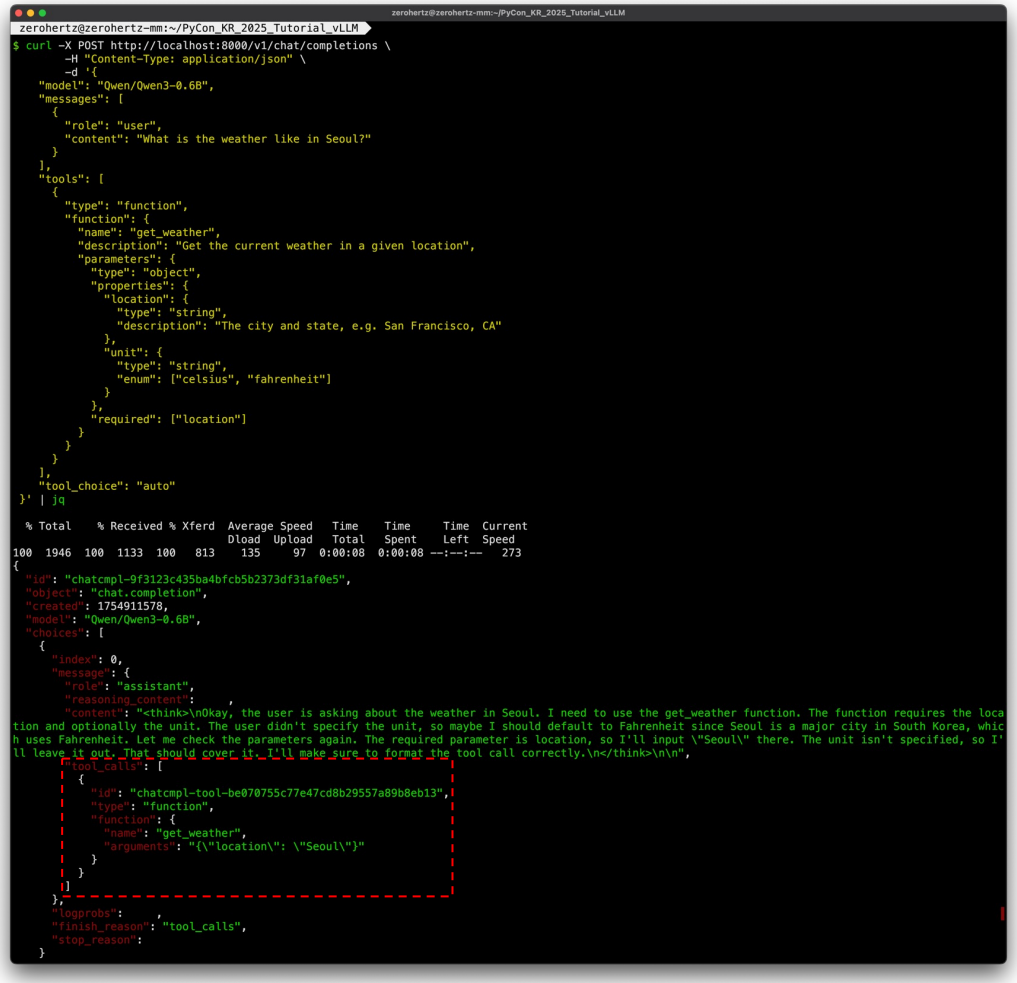
```
zerohertz@zerohertz-mm:~/PyCon_KR_2025_Tutorial_vLLM
$ curl -X POST http://localhost:8000/v1/chat/completions \
       -H "Content-Type: application/json" \
       -d '{
    "model": "Qwen/Qwen3-0.6B",
    "messages": [
      {
        "role": "user",
        "content": "What is the weather like in Seoul?"
      }
    ],
    "tools": [
      {
        "type": "function",
        "function": {
          "name": "get_weather",
          "description": "Get the current weather in a given location",
          "parameters": {
            "type": "object",
            "properties": {
              "location": {
                "type": "string",
                "description": "The city and state, e.g. San Francisco, CA"
              },
              "unit": {
                "type": "string",
                "enum": ["celsius", "fahrenheit"]
              }
            },
            "required": ["location"]
          }
        }
      }
    ],
    "tool_choice": "auto"
  }' | jq

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   981  100   168  100   813  19739  95523 --:--:-- --:--:-- --:--:--  119k
{
  "object": "error",
  "message": "\"auto\" tool choice requires --enable-auto-tool-choice and --tool-call-parser to be set",
  "type": "BadRequestError",
  "param":     ,
  "code": 400
}
```


[19] https://github.com/openai/openai-python/blob/v1.97.1/src/openai/types/shared_params/function_definition.py#L13-L45

Tool calling

❑ “--tool-call-parser”를 통해 모델에 따른 적절한 parser 설정 필요[20, 21]





[20] https://docs.vllm.ai/en/v0.9.2/features/tool_calling.html
[21] https://qwen.readthedocs.io/en/latest/deployment/vllm.html#parsing-tool-calls

Tool calling

❑ 요청에 required를 입력한다면? [22]

v0.9.2 vllm / vllm / entrypoints / openai / protocol.py [23]

```
246        tools: Optional[list[ChatCompletionToolsParam]] = None
247        tool_choice: Optional[Union[
248            Literal["none"],
249            Literal["auto"],
250            Literal["required"],
251            ChatCompletionNamedToolChoiceParam,
252        ]] = "none"
```

v0.9.2 vllm / vllm / entrypoints / openai / protocol.py [24]

```
497            guided_decoding = GuidedDecodingParams.from_optional(
498                json=self._get_guided_json_from_tool() or self.guided_json,
499                regex=self.guided_regex,
500                choice=self.guided_choice,
501                grammar=self.guided_grammar,
502                json_object=guided_json_object,
503                backend=self.guided_decoding_backend,
504                whitespace_pattern=self.guided_whitespace_pattern,
505                structural_tag=self.structural_tag,
506            )
```

v0.9.2 vllm / vllm / entrypoints / openai / protocol.py [25]

```
512            return SamplingParams.from_optional(
513                n=self.n,
514                best_of=self.best_of,
515                presence_penalty=self.presence_penalty,
516                frequency_penalty=self.frequency_penalty,
517                repetition_penalty=repetition_penalty,
518                temperature=temperature,
519                top_p=top_p,
520                top_k=top_k,
521                min_p=min_p,
522                seed=self.seed,
523                stop=self.stop,
524                stop_token_ids=self.stop_token_ids,
525                logprobs=self.top_logprobs if self.logprobs else None,
526                prompt_logprobs=prompt_logprobs,
527                ignore_eos=self.ignore_eos,
528                max_tokens=max_tokens,
529                min_tokens=self.min_tokens,
530                skip_special_tokens=self.skip_special_tokens,
531                spaces_between_special_tokens=self.spaces_between_special_tokens,
532                logits_processors=get_logits_processors(self.logits_processors,
533                                                        logits_processor_pattern),
534                include_stop_str_in_output=self.include_stop_str_in_output,
535                truncate_prompt_tokens=self.truncate_prompt_tokens,
536                output_kind=RequestOutputKind.DELTA if self.stream \
537                    else RequestOutputKind.FINAL_ONLY,
538                guided_decoding=guided_decoding,
539                logit_bias=self.logit_bias,
540                bad_words= self.bad_words,
541                allowed_token_ids=self.allowed_token_ids,
542                extra_args=extra_args or None,
543            )
```

v0.9.2 vllm / vllm / engine / async_llm_engine.py [26]

```
473            if isinstance(params, SamplingParams) and \
474                params.guided_decoding is not None:
475                # Guided decoding has an async implementation for building logits
476                # processors in a separate threadpool.
477                # We want to invoke that here instead of using the blocking
478                # implementation in the LLMEngine
479                params = await build_guided_decoding_logits_processor_async(
480                    sampling_params=params,
481                    tokenizer=await self.get_tokenizer_async(lora_request),
482                    default_guided_backend=self.decoding_config.backend,
483                    reasoning_backend=self.decoding_config.reasoning_backend,
484                    model_config=self.model_config)
```

v0.9.2 vllm / vllm / model_executor / guided_decoding / __init__.py [27]

```
99
100    async def get_guided_decoding_logits_processor(
101            guided_params: GuidedDecodingParams,
102            tokenizer: PreTrainedTokenizer,
103            model_config: ModelConfig,
104            reasoning_backend: str | None = None) -> LogitsProcessor | None:
105
106        reasoner = None
107        if reasoning_backend:
108            reasoner_class = ReasoningParserManager.get_reasoning_parser(
109                reasoning_backend)
110            reasoner = reasoner_class(tokenizer)
111
112        guided_params = maybe_backend_fallback(guided_params)
113
114        # CFG grammar not supported by LMFE, so we use outlines instead
115        if guided_params.backend == 'outlines':
116            # NOTE: lazy import outlines to avoid https://github.com/vllm-project/vllm/issues/4193
117            from vllm.model_executor.guided_decoding.outlines_decoding import (  # noqa
118                get_outlines_guided_decoding_logits_processor)
119            return await get_outlines_guided_decoding_logits_processor(
120                guided_params, tokenizer, reasoner)
121        if guided_params.backend == 'lm-format-enforcer':
122            from vllm.model_executor.guided_decoding.lm_format_enforcer_decoding import (  # noqa
123                get_local_lm_format_enforcer_guided_decoding_logits_processor)
124            return get_local_lm_format_enforcer_guided_decoding_logits_processor(
125                guided_params, tokenizer)
126        if guided_params.backend == 'xgrammar':
127            from vllm.model_executor.guided_decoding.xgrammar_decoding import (  # noqa
128                get_local_xgrammar_guided_decoding_logits_processor)
129            return get_local_xgrammar_guided_decoding_logits_processor(
130                guided_params, tokenizer, model_config, reasoner)
131        if guided_params.backend == 'guidance':
132            from vllm.model_executor.guided_decoding.guidance_decoding import (
133                get_local_guidance_guided_decoding_logits_processor)
134            return get_local_guidance_guided_decoding_logits_processor(
135                guided_params, tokenizer)
136        raise ValueError(
137            f"Unknown guided decoding backend '{guided_params.backend}'. "
138            "Must be one of 'outlines, 'lm-format-enforcer', 'xgrammar', 'guidance'"
139        )
```

❑ 요청에 따른 backend 지정은 deprecated

Zerohertz 44 minutes ago [28, 29] Contributor Author

vllm/vllm/config/__init__.py
Line 2949 in 85abf66

```
2949    GuidedDecodingBackend = Literal["auto", "xgrammar", "guidance", "outlines"]
```

vllm/vllm/v1/structured_output/__init__.py
Lines 86 to 113 in 85abf66

```
86      if self.backend is None:
87          assert request.sampling_params is not None
88          backend = request.sampling_params.guided_decoding.backend
89          vocab_size = self.vllm_config.model_config.get_vocab_size()
90          if backend == "xgrammar":
91              self.backend = XgrammarBackend(
92                  self.vllm_config,
93                  tokenizer=self.tokenizer,
94                  vocab_size=vocab_size,
95              )
96          elif backend == "guidance":
97              self.backend = GuidanceBackend(
```

vllm/vllm/v1/engine/processor.py
Lines 156 to 183 in 85abf66

```
172             if (params.guided_decoding.backend != engine_level_backend
173                     and not (engine_level_backend == "auto"
174                              and params.guided_decoding.backend_was_auto)):
175                 raise ValueError(
176                     "Request-level structured output backend selection is no "
177                     "longer supported. The request specified "
178                     f"'{params.guided_decoding.backend}', but vLLM was "
179                     f"initialised with '{engine_level_backend}'. This error "
180                     "can be resolved by removing backend selection from the "
181                     "request.")
182         else:
183             params.guided_decoding.backend = engine_level_backend
```

I've analyzed the code in `engine/processor.py` and `structured_output/__init__.py` and confirmed that the structured output backend cannot be configured at the request level.
The setting is only applied globally at the engine level.

[22] https://docs.vllm.ai/en/v0.9.2/features/structured_outputs.html
[23] https://github.com/vllm-project/vllm/blob/v0.9.2/vllm/entrypoints/openai/protocol.py#L247-L252
[24] https://github.com/vllm-project/vllm/blob/v0.9.2/vllm/entrypoints/openai/protocol.py#L497-L506
[25] https://github.com/vllm-project/vllm/blob/v0.9.2/vllm/entrypoints/openai/protocol.py#L538
[26] https://github.com/vllm-project/vllm/blob/v0.9.2/vllm/engine/async_llm_engine.py#L473-L484
[27] https://github.com/vllm-project/vllm/blob/v0.9.2/vllm/model_executor/guided_decoding/__init__.py#L100-L139
[28] https://github.com/vllm-project/vllm/pull/22740
[29] https://github.com/vllm-project/vllm/pull/22772

Reasoning

❑ “chat_template_kwargs”의 “enable_thinking”을 통해 reasoning 유무 설정 가능

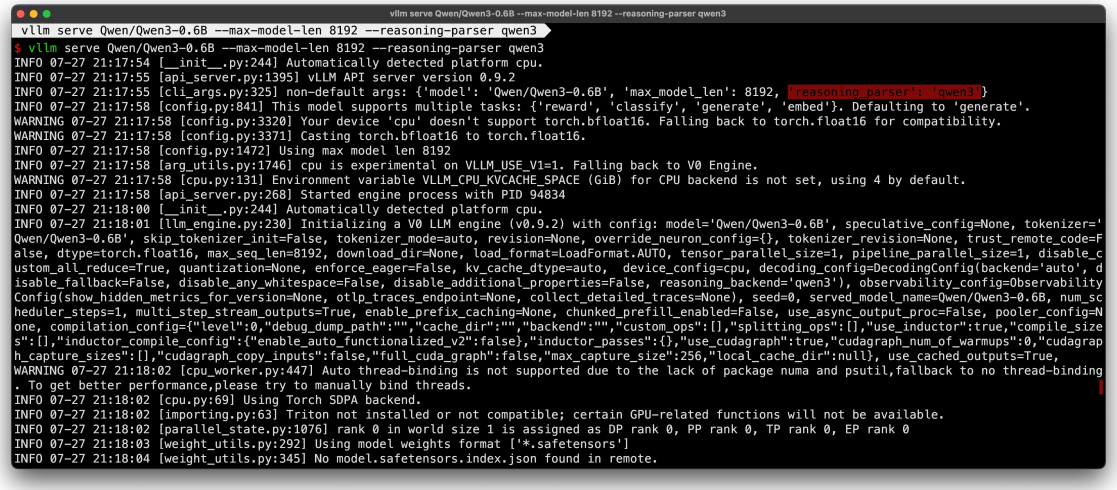
```
zerohertz@zerohertz-mm:~/PyCon_KR_2025_Tutorial_vLLM
$ curl -X POST http://localhost:8000/v1/chat/completions \
       -H "Content-Type: application/json" \
       -d '{
    "model": "Qwen/Qwen3-0.6B",
    "messages": [
        {
            "role": "user",
            "content": "Hello, PyCon Korea 2025!"
        }
    ],
    "chat_template_kwargs": {"enable_thinking": false}
}' | jq

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   739  100   526  100   213    113     45  0:00:04  0:00:04 --:--:--   158
{
  "id": "chatcmpl-8eb6c33541a84eb4b96562ef7c858290",
  "object": "chat.completion",
  "created": 1754912052,
  "model": "Qwen/Qwen3-0.6B",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "reasoning_content":     ,
        "content": "Hello, PyCon Korea 2025! 🎉 What's the best way to celebrate this event? Let me know!",
        "tool_calls": []
      },
      "logprobs":     ,
      "finish_reason": "stop",
      "stop_reason":
    }
  ],
  "usage": {
    "prompt_tokens": 23,
    "total_tokens": 52,
    "completion_tokens": 29,
    "prompt_tokens_details":
  },
  "prompt_logprobs":     ,
  "kv_transfer_params":
}
```

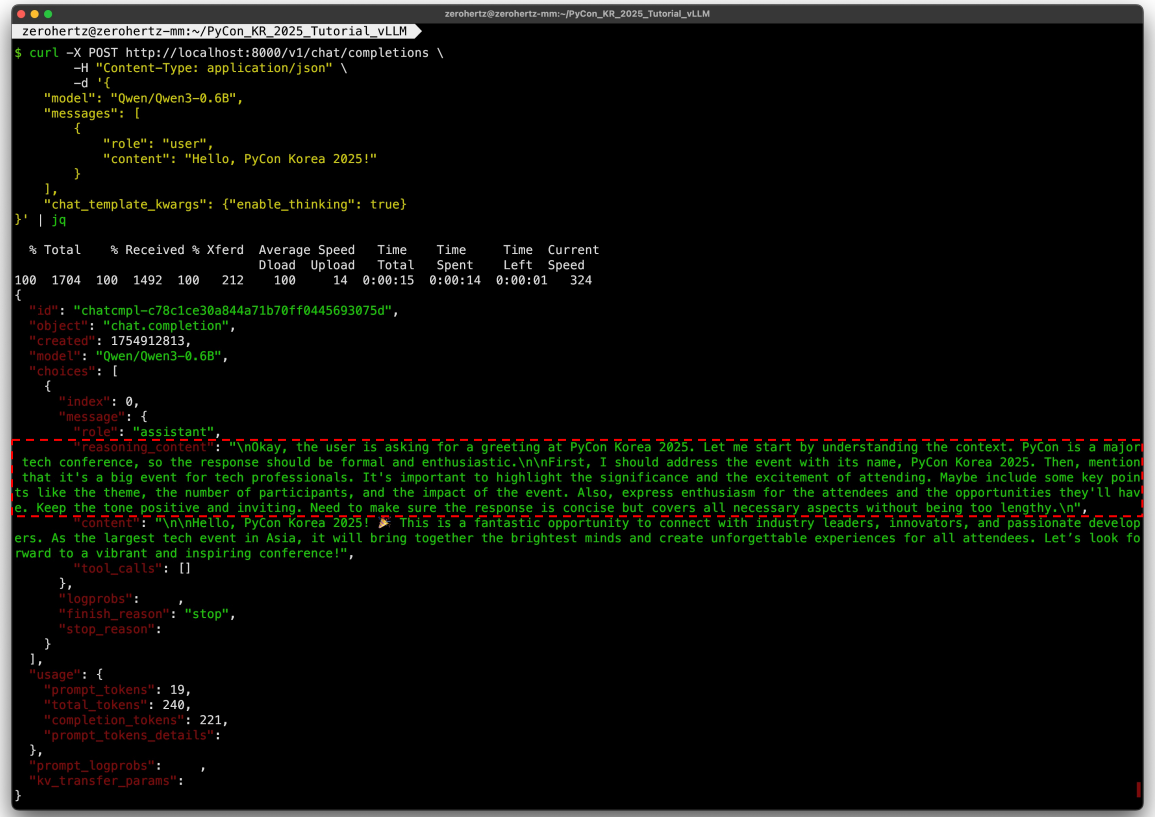
```
zerohertz@zerohertz-mm:~/PyCon_KR_2025_Tutorial_vLLM
$ curl -X POST http://localhost:8000/v1/chat/completions \
       -H "Content-Type: application/json" \
       -d '{
    "model": "Qwen/Qwen3-0.6B",
    "messages": [
        {
            "role": "user",
            "content": "Hello, PyCon Korea 2025!"
        }
    ],
    "chat_template_kwargs": {"enable_thinking": true}
}' | jq

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1400  100  1188  100   212    121     21  0:00:10  0:00:09  0:00:01   263
{
  "id": "chatcmpl-2eb0a23e56c14eb2b8db3eaf0330c21f",
  "object": "chat.completion",
  "created": 1754912334,
  "model": "Qwen/Qwen3-0.6B",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "reasoning_content":     ,
        "content": "<think>\nOkay, the user just said \"Hello, PyCon Korea 2025!\" So I need to respond to that. Let me start by acknowledging their mes
sage. Maybe say something like \"Hello, PyCon Korea 2025!\" to confirm the date. Then, I should invite them to attend the event. I can mention the date,
 location, and maybe the main themes. Also, I should keep the tone friendly and enthusiastic. Let me make sure to include all the key details they might
 need. Alright, let's put that together.\n</think>\n\nHello, PyCon Korea 2025! 🎉  \nI'm excited to welcome you to the prestigious event where innovatio
n and creativity collide. Let me know if you're looking forward to the themes, workshops, or networking opportunities. Have a great time at PyCon Korea
2025!",
        "tool_calls": []
      },
      "logprobs":     ,
      "finish_reason": "stop",
      "stop_reason":
    }
  ],
  "usage": {
    "prompt_tokens": 19,
    "total_tokens": 200,
    "completion_tokens": 181,
```

```
INFO 08-11 20:34:12 [logger.py:43] Received request chatcmpl-8eb6c33541a84eb4b96562ef7c858290: prompt: '<|im_start|>user\nHello, PyCon Korea 2025!<|im_end|>\n<|im_start|>assistant\n<think>\n\n</think>\n\n', params: Sa
emperature=0.6, top_p=0.95, top_k=20, min_p=0.0, seed=None, stop=[], stop_token_ids=[], bad_words=[], include_stop_str_in_output=False, ignore_eos=False, max_tokens=8169, min_tokens=0, logprobs=None, prompt_logprobs=N
s=None, guided_decoding=None, extra_args=None), prompt_token_ids: None, prompt_embeds shape: None, lora_request: None, prompt_adapter_request: None.
INFO 08-11 20:34:12 [engine.py:317] Added request chatcmpl-8eb6c33541a84eb4b96562ef7c858290.
WARNING 08-11 20:34:12 [cpu.py:250] Pin memory is not supported on CPU.
INFO:     127.0.0.1:60893 - "POST /v1/chat/completions HTTP/1.1" 200 OK
INFO 08-11 20:34:23 [metrics.py:417] Avg prompt throughput: 1.9 tokens/s, Avg generation throughput: 2.4 tokens/s, Running: 0 reqs, Swapped: 0 reqs, Pending: 0 reqs, GPU KV cache usage: 0.0%, CPU KV cache usage: 0.0%.
INFO 08-11 20:34:33 [metrics.py:417] Avg prompt throughput: 0.0 tokens/s, Avg generation throughput: 0.0 tokens/s, Running: 0 reqs, Swapped: 0 reqs, Pending: 0 reqs, GPU KV cache usage: 0.0%, CPU KV cache usage: 0.0%.
INFO 08-11 20:38:54 [logger.py:43] Received request chatcmpl-2eb0a23e56c14eb2b8db3eaf0330c21f: prompt: '<|im_start|>user\nHello, PyCon Korea 2025!<|im_end|>\n<|im_start|>assistant\n', params: SamplingParams(n=1, preser
.95, top_k=20, min_p=0.0, seed=None, stop=[], stop_token_ids=[], bad_words=[], include_stop_str_in_output=False, ignore_eos=False, max_tokens=8173, min_tokens=0, logprobs=None, prompt_logprobs=None, skip_special_token
=None, extra_args=None), prompt_token_ids: None, prompt_embeds shape: None, lora_request: None, prompt_adapter_request: None.
```

Reasoning

- "--reasoning-parser"를 통해 모델에 따른 적절한 parser 설정 필요 [30, 31]
- 기존에는 "--enable-reasoning" 옵션이 존재했으나 deprecated [32]



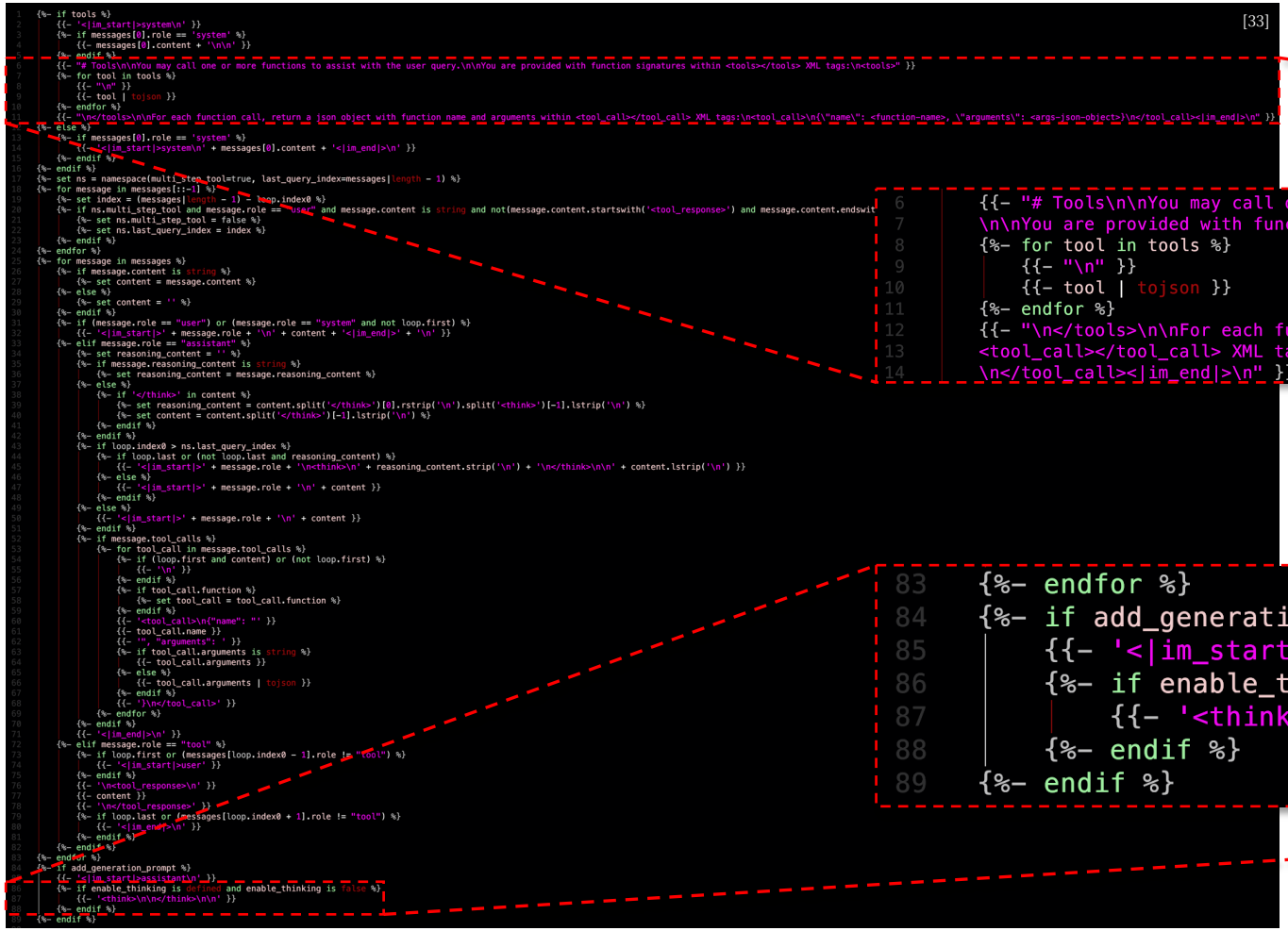


[30] https://docs.vllm.ai/en/v0.9.2/features/reasoning_outputs.html
[31] https://qwen.readthedocs.io/en/latest/deployment/vllm.html#parsing-thinking-content
[32] https://github.com/vllm-project/vllm/blob/v0.9.2/vllm/engine/arg_utils.py#L626-L634

Chat Template

❑ 기본적으로 tokenizer_config.json의 “chat_template” 값 사용

[33]



```
{%- if tools %}
    {{- '<|im_start|>system\n' }}
    {%- if messages[0].role == 'system' %}
        {{- messages[0].content + '\n\n' }}
    {%- endif %}
    {{- "# Tools\n\nYou may call one or more functions to assist with the user query.\n\nYou are provided with function signatures within <tools></tools> XML tags:\n<tools>" }}
    {%- for tool in tools %}
        {{- "\n" }}
        {{- tool | tojson }}
    {%- endfor %}
    {{- "\n</tools>\n\nFor each function call, return a json object with function name and arguments within <tool_call></tool_call> XML tags:\n<tool_call>\n{\"name\": <function-name>, \"arguments\": <args-json-object>}\n</tool_call><|im_end|>\n" }}
{%- else %}
    {%- if messages[0].role == 'system' %}
        {{- '<|im_start|>system\n' + messages[0].content + '<|im_end|>\n' }}
    {%- endif %}
{%- endif %}
{%- set ns = namespace(multi_step_tool=true, last_query_index=messages|length - 1) %}
{%- for message in messages[::-1] %}
    {%- set index = (messages|length - 1) - loop.index0 %}
    {%- if ns.multi_step_tool and message.role == "user" and message.content is string and not(message.content.startswith('<tool_response>') and message.content.endswith
        {%- set ns.multi_step_tool = false %}
        {%- set ns.last_query_index = index %}
    {%- endif %}
{%- endfor %}
{%- for message in messages %}
    {%- if message.content is string %}
        {%- set content = message.content %}
    {%- else %}
        {%- set content = '' %}
    {%- endif %}
    {%- if (message.role == "user") or (message.role == "system" and not loop.first) %}
        {{- '<|im_start|>' + message.role + '\n' + content + '<|im_end|>' + '\n' }}
    {%- elif message.role == "assistant" %}
        {%- set reasoning_content = '' %}
        {%- if message.reasoning_content is string %}
            {%- set reasoning_content = message.reasoning_content %}
        {%- else %}
            {%- if '</think>' in content %}
                {%- set reasoning_content = content.split('</think>')[0].rstrip('\n').split('<think>')[-1].lstrip('\n') %}
                {%- set content = content.split('</think>')[-1].lstrip('\n') %}
            {%- endif %}
        {%- endif %}
        {%- if loop.index0 > ns.last_query_index %}
            {%- if loop.last or (not loop.last and reasoning_content) %}
                {{- '<|im_start|>' + message.role + '\n<think>\n' + reasoning_content.strip('\n') + '\n</think>\n\n' + content.lstrip('\n') }}
            {%- else %}
                {{- '<|im_start|>' + message.role + '\n' + content }}
            {%- endif %}
        {%- else %}
            {{- '<|im_start|>' + message.role + '\n' + content }}
        {%- endif %}
        {%- if message.tool_calls %}
            {%- for tool_call in message.tool_calls %}
                {%- if (loop.first and content) or (not loop.first) %}
                    {{- '\n' }}
                {%- endif %}
                {%- if tool_call.function %}
                    {%- set tool_call = tool_call.function %}
                {%- endif %}
                {{- '<tool_call>\n{"name": "' }}
                {{- tool_call.name }}
                {{- '", "arguments": ' }}
                {%- if tool_call.arguments is string %}
                    {{- tool_call.arguments }}
                {%- else %}
                    {{- tool_call.arguments | tojson }}
                {%- endif %}
                {{- '}\n</tool_call>' }}
            {%- endfor %}
        {%- endif %}
        {{- '<|im_end|>\n' }}
    {%- elif message.role == "tool" %}
        {%- if loop.first or (messages[loop.index0 - 1].role != "tool") %}
            {{- '<|im_start|>user' }}
        {%- endif %}
        {{- '\n<tool_response>\n' }}
        {{- content }}
        {{- '\n</tool_response>' }}
        {%- if loop.last or (messages[loop.index0 + 1].role != "tool") %}
            {{- '<|im_end|>\n' }}
        {%- endif %}
    {%- endif %}
{%- endfor %}
{%- if add_generation_prompt %}
    {{- '<|im_start|>assistant\n' }}
    {%- if enable_thinking is defined and enable_thinking is false %}
        {{- '<think>\n\n</think>\n\n' }}
    {%- endif %}
{%- endif %}
```

```
{{- "# Tools\n\nYou may call one or more functions to assist with the user query.
\n\nYou are provided with function signatures within <tools></tools> XML tags:\n<tools>" }}
{%- for tool in tools %}
    {{- "\n" }}
    {{- tool | tojson }}
{%- endfor %}
{{- "\n</tools>\n\nFor each function call, return a json object with function name and arguments within
<tool_call></tool_call> XML tags:\n<tool_call>\n{\"name\": <function-name>, \"arguments\": <args-json-object>}
\n</tool_call><|im_end|>\n" }}
```

```
{%- endfor %}
{%- if add_generation_prompt %}
    {{- '<|im_start|>assistant\n' }}
    {%- if enable_thinking is defined and enable_thinking is false %}
        {{- '<think>\n\n</think>\n\n' }}
    {%- endif %}
{%- endif %}
```

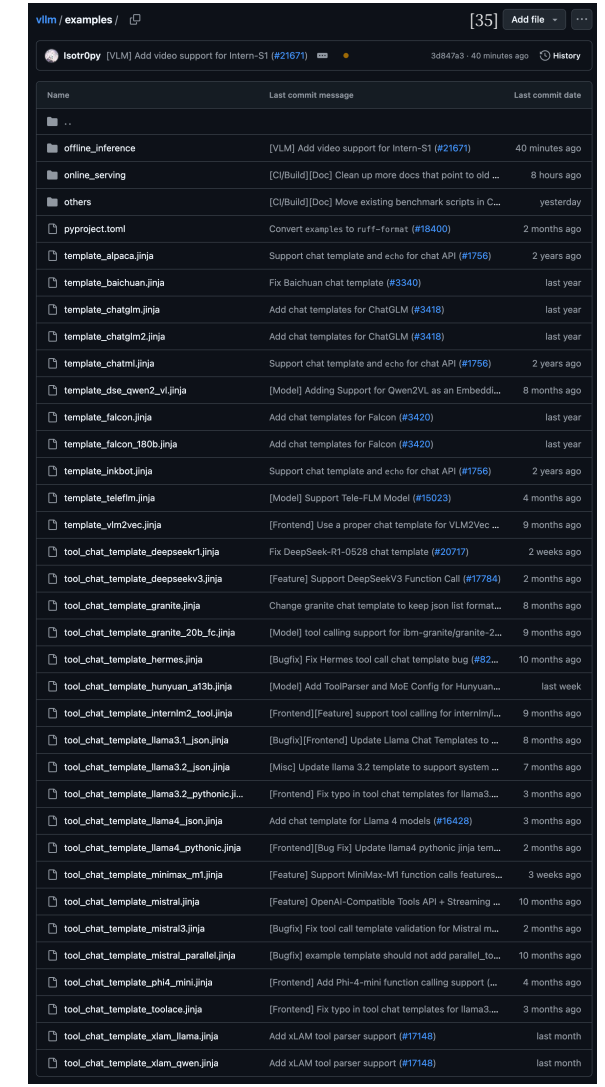

[33] https://huggingface.co/Qwen/Qwen3-0.6B/blob/main/tokenizer_config.json#L230

Chat Template

❑ “--chat-template”으로 새로운 chat template 적용 가능[34]

```python
messages = [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "Hello, InstructKR!"},
    {"role": "assistant", "content": "Hello! How can I help you today?"},
    {"role": "user", "content": "Can you explain what InstructKR is?"},
    {
        "role": "assistant",
        "content": "InstructKR is a Korean instruction-following dataset and research initiative focused on improving language models' ability to follow instructions in Korean.",
    },
    {"role": "user", "content": "What's the weather like in Seoul today?"},
    {
        "role": "assistant",
        "content": "I'll help you check the weather in Seoul.",
        "tool_calls": [
            {
                "id": "call_1",
                "type": "function",
                "function": {
                    "name": "get_weather",
                    "arguments": '{"location": "Seoul, South Korea"}',
                },
            }
        ],
    },
    {
        "role": "tool",
        "content": "The weather in Seoul today is sunny with a temperature of 22°C (72°F). There's a light breeze and clear skies.",
        "tool_call_id": "call_1",
    },
    {
        "role": "assistant",
        "content": "Based on the weather information, Seoul is having a pleasant day today! It's sunny with a comfortable temperature of 22°C (72°F), light breeze, and clear skie
    },
    {"role": "user", "content": "Thanks! Can you help me with something else?"},
]
prompt = processor.apply_chat_template(
    messages, tokenize=False, add_generation_prompt=True
)
```

```
<|im_start|>system
You are a helpful assistant.<|im_end|>
<|im_start|>user
Hello, InstructKR!<|im_end|>
<|im_start|>assistant
Hello! How can I help you today?<|im_end|>
<|im_start|>user
Can you explain what InstructKR is?<|im_end|>
<|im_start|>assistant
InstructKR is a Korean instruction-following dataset and research initiative focused on improving language models' ability to follow instructions in Korean.<|im_end|>
<|im_start|>user
What's the weather like in Seoul today?<|im_end|>
<|im_start|>assistant
I'll help you check the weather in Seoul.
<tool_call>
{"name": "get_weather", "arguments": {"location": "Seoul, South Korea"}}
</tool_call><|im_end|>
<|im_start|>user
<tool_response>
The weather in Seoul today is sunny with a temperature of 22°C (72°F). There's a light breeze and clear skies.
</tool_response><|im_end|>
<|im_start|>assistant
Based on the weather information, Seoul is having a pleasant day today! It's sunny with a comfortable temperature of 22°C (72°F), light breeze, and clear skies. Perfect
<|im_start|>user
Thanks! Can you help me with something else?<|im_end|>
<|im_start|>assistant
```



[34] https://docs.vllm.ai/en/v0.9.2/serving/openai_compatible_server.html#chat-template_1
[35] https://github.com/vllm-project/vllm/tree/main/examples

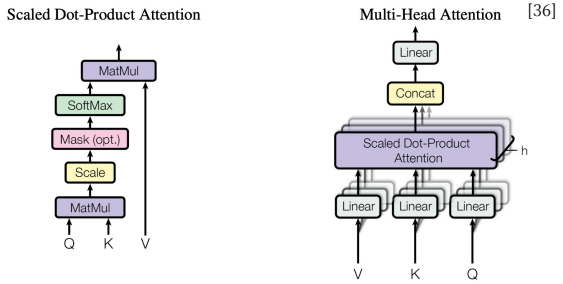
PART 3

Architecture

KV Cache, PagedAttention [7]

❑ Attention [36]

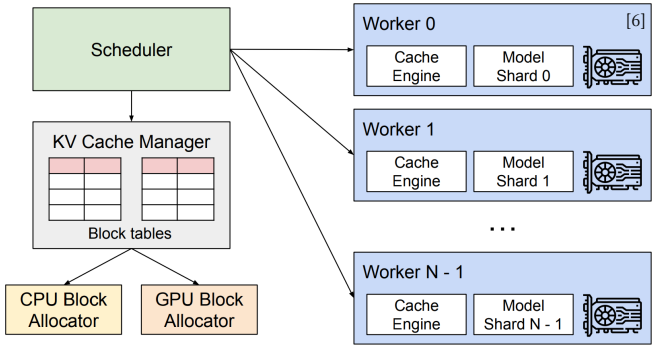
- $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$
  - → d_k: Dimension of Q, K



[36]

❑ KV Cache [37]

- Autoregressive 생성 방식 특성 상 각 token을 예측할 때 과거 전체 sequence를 입력으로 재처리
  - → time complexity: $O(n^2)$
- 과거 Key/Value 값 (KV cache)을 사용하여 매 단계의 반복 연산 생략
  - → time complexity: $O(n)$ 수준



[6]

❑ PagedAttention [7, 38, 39]

- 운영체제 (OS)의 virtual memory에서 영감을 받아 제안
- 기존의 KV cache는 memory 연속성 요구로 memory fragmentation 문제 존재
  - → 특히 많은 요청을 병렬로 처리하는 상황에서 memory 할당/해제의 비효율 심각
- Block 단위의 memory 할당 및 page table을 이용하여 논리적 연속성 (logical continuity) 유지로 물리적 memory 분산
- 각 요청을 고정된 크기의 page에 mapping하여 실제 memory는 non-continuous 하게 구성하여 효율적 접근

[40]

```
%%timeit -n 1
# Generate the text
generation_output = model.generate(
  input_ids=input_ids,
  max_new_tokens=100,
  use_cache=True
)
```

```
6.66 s ± 2.22 s per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

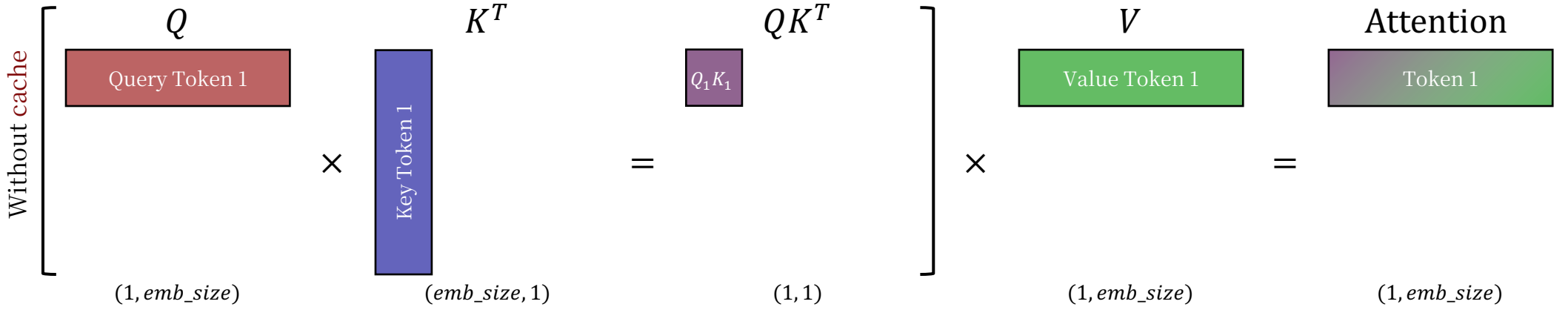
```
%%timeit -n 1
# Generate the text
generation_output = model.generate(
  input_ids=input_ids,
  max_new_tokens=100,
  use_cache=False
)
```

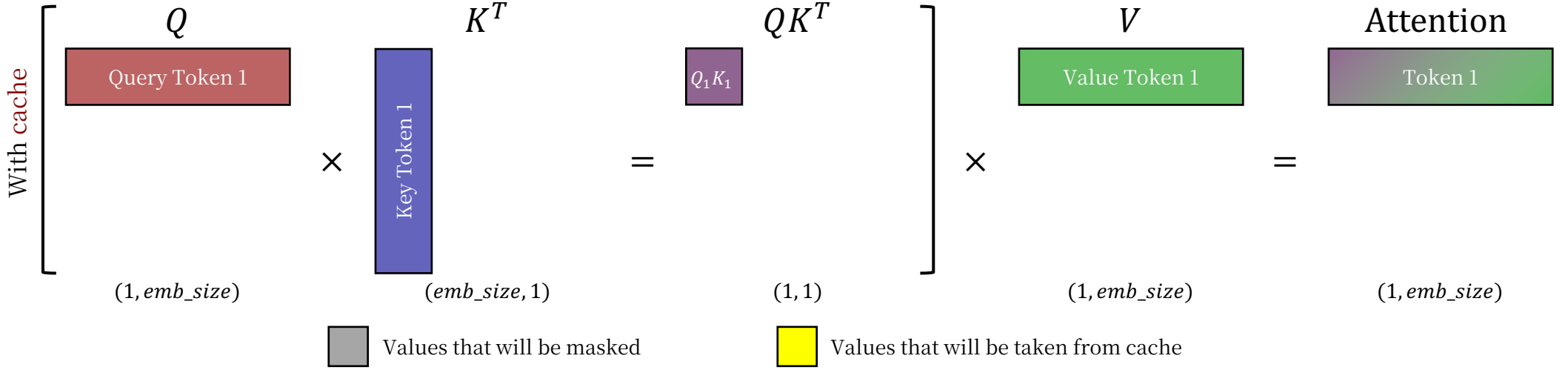
```
21.9 s ± 94.6 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

[7] https://arxiv.org/abs/2309.06180
[36] https://arxiv.org/abs/1706.03762
[37] https://huggingface.co/blog/not-lain/kv-caching
[38] https://docs.vllm.ai/en/v0.9.2/design/kernel/paged_attention.html
[39] https://blog.vllm.ai/2023/06/20/vllm.html
[40] https://github.com/HandsOnLLM/Hands-On-Large-Language-Models/blob/main/chapter03/Chapter%203%20-%20Looking%20Inside%20LLMs.ipynb

KV Cache, PagedAttention [7]

❑ Step 1.

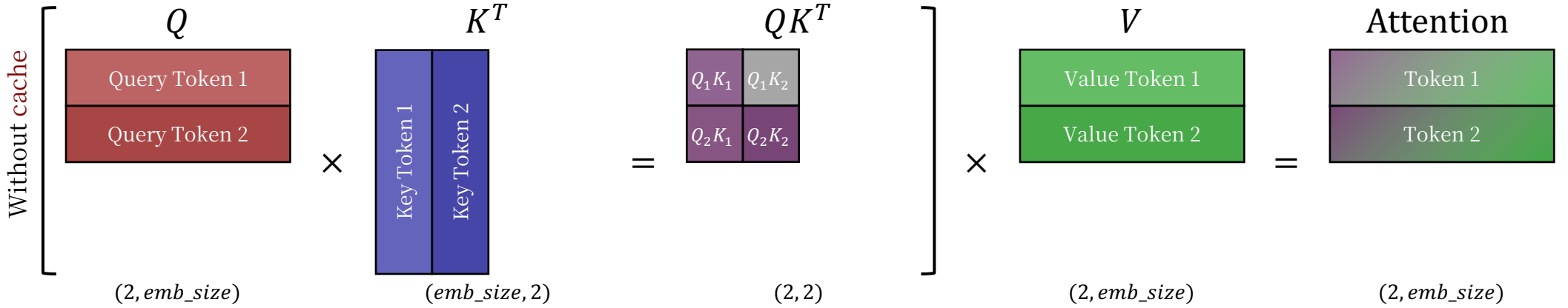


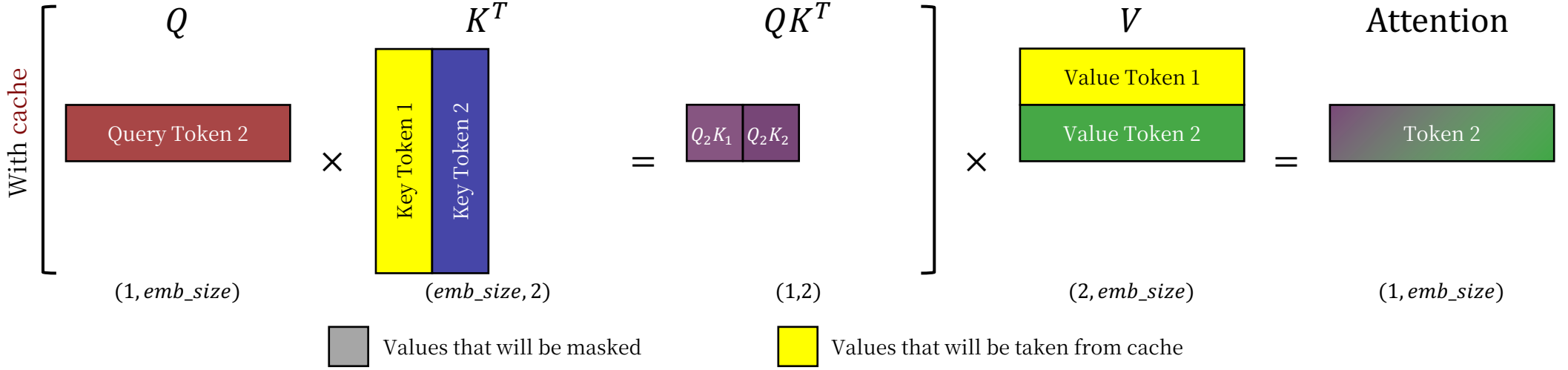


[7] https://arxiv.org/abs/2309.06180

KV Cache, PagedAttention [7]

❑ Step 2.

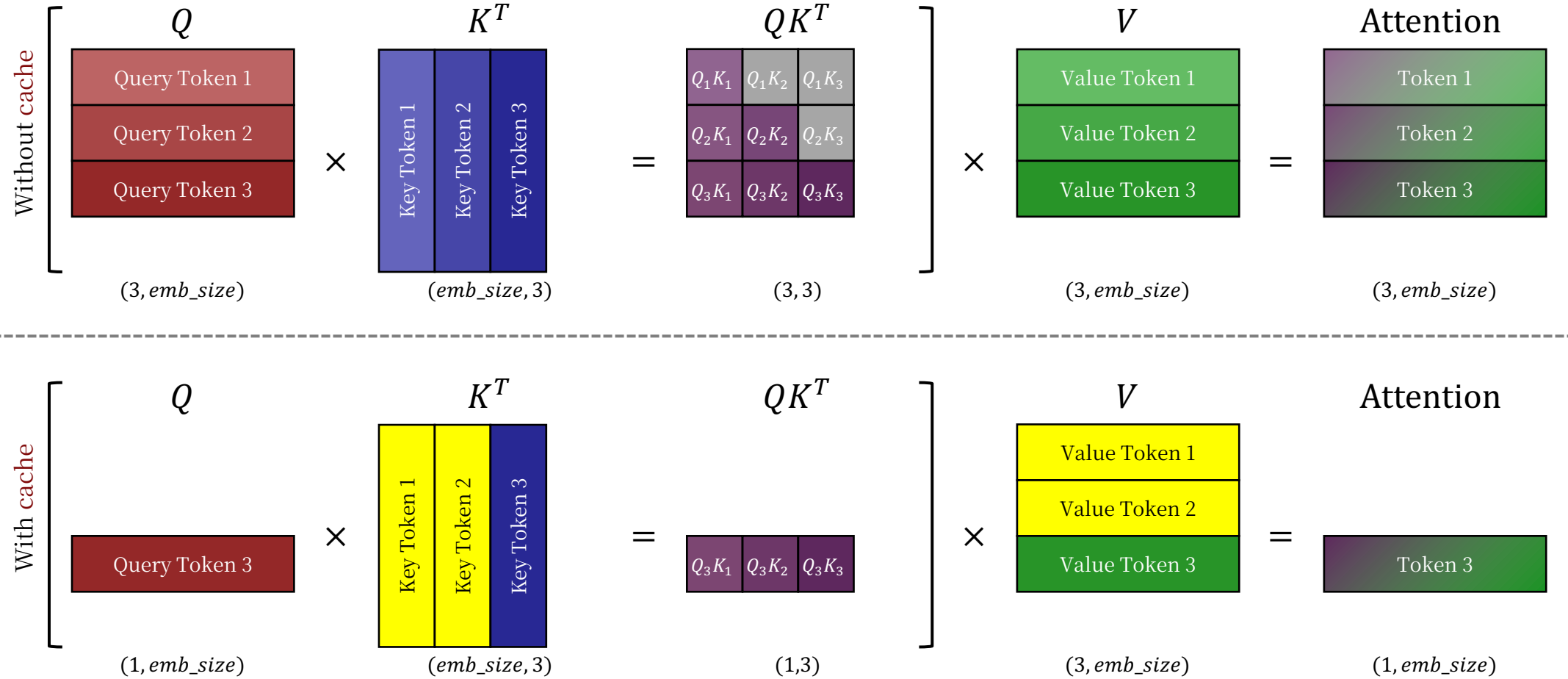




[7] https://arxiv.org/abs/2309.06180

KV Cache, PagedAttention [7]

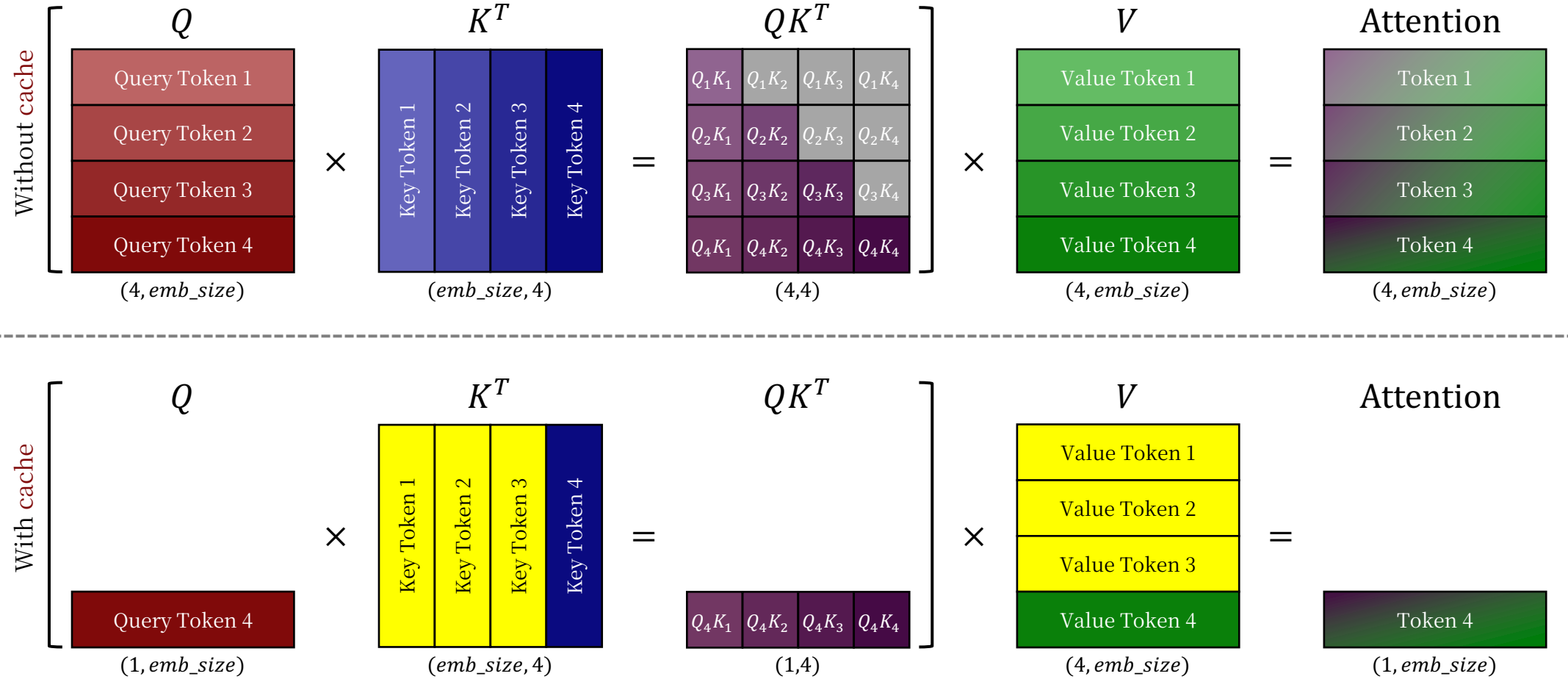
❑ Step 3.



[7] https://arxiv.org/abs/2309.06180

KV Cache, PagedAttention [7]

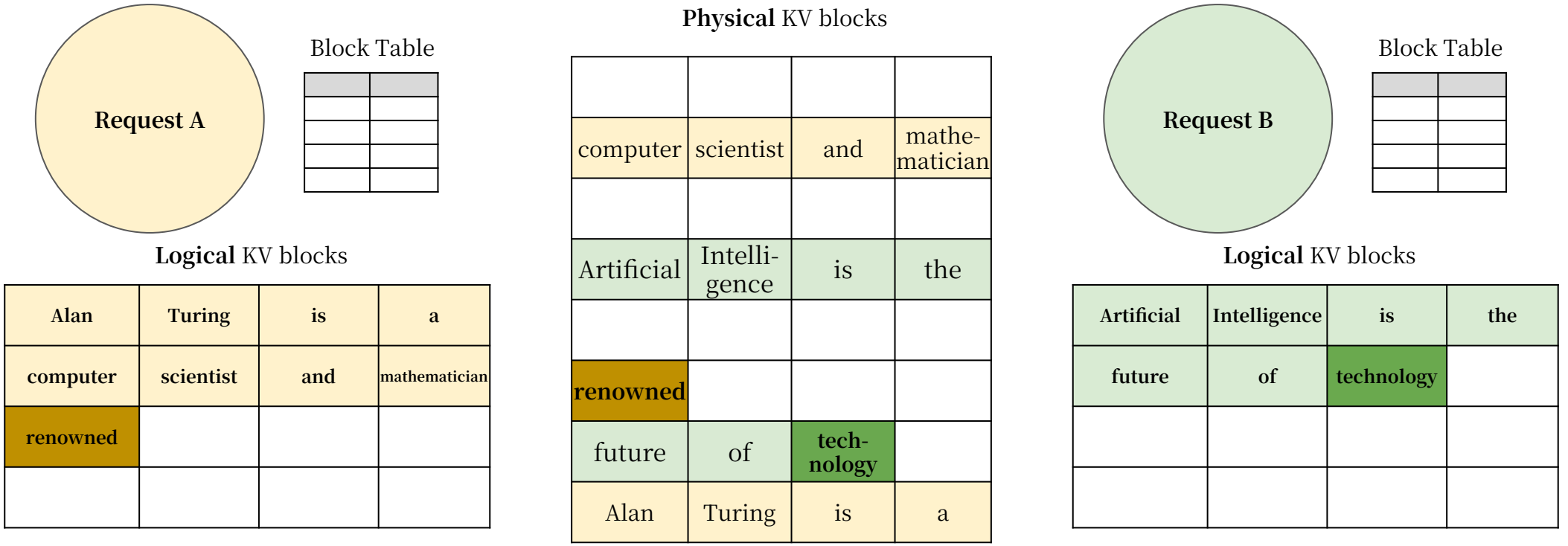
❑ Step 4.



[7] https://arxiv.org/abs/2309.06180

KV Cache, PagedAttention [7]

- Logical KV Blocks: 하나의 request 관점의 연속적 memory view
- Physical Blocks: GPU memory 상의 실제 저장 위치 (비연속적, memory fragmentation 방지)
- Block Table: Logical block과 physical block을 연결하는 mapping 정보



[7] https://arxiv.org/abs/2309.06180

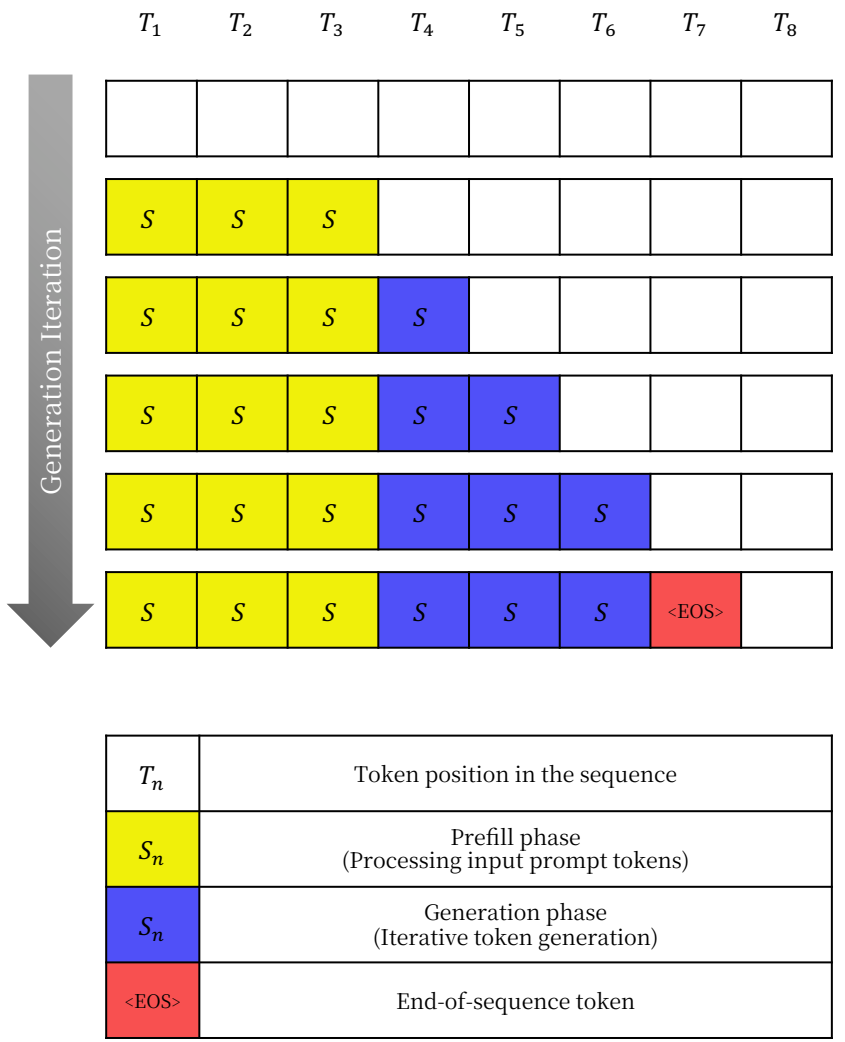
Continuous Batching [41]

❑ Batching
- 여러 개의 요청을 하나로 묶어 한 번의 kernel 실행 (큰 행렬곱)으로 동시에 처리하는 기술
- 큰 GEMM (General Matrix Multiplication)으로 연산 집약도 (Arithmetic Intensity) 상승

❑ Static Batching
- Fixed batch size의 요청이 모일 때까지 대기 후 추론
- 요청량이 적은 경우 무한정 대기 → GPU 낭비

❑ Dynamic Batching [42]
- 짧은 수집 window에 대한 요청의 batch 구성 후, 시간 제한 도달 시 batch size 미달이어도 추론
- Batch 내 하나라도 step이 완료되지 않으면 step 종료까지 batch 고정 → GPU 낭비

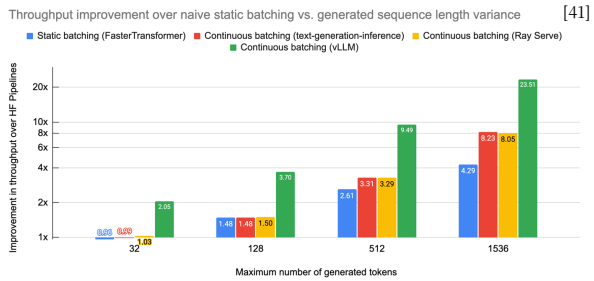
❑ Continuous Batching
- Batch를 요청 단위가 아닌 하나의 token 생성 단계 (iteration) 단위로 운영
- 새 요청은 다음 단계에 즉시 합류, 완료된 sequence는 즉시 제거



[41] https://www.anyscale.com/blog/continuous-batching-llm-inference
[42] https://docs.nvidia.com/deeplearning/triton-inference-server/user-guide/docs/user_guide/batcher.html

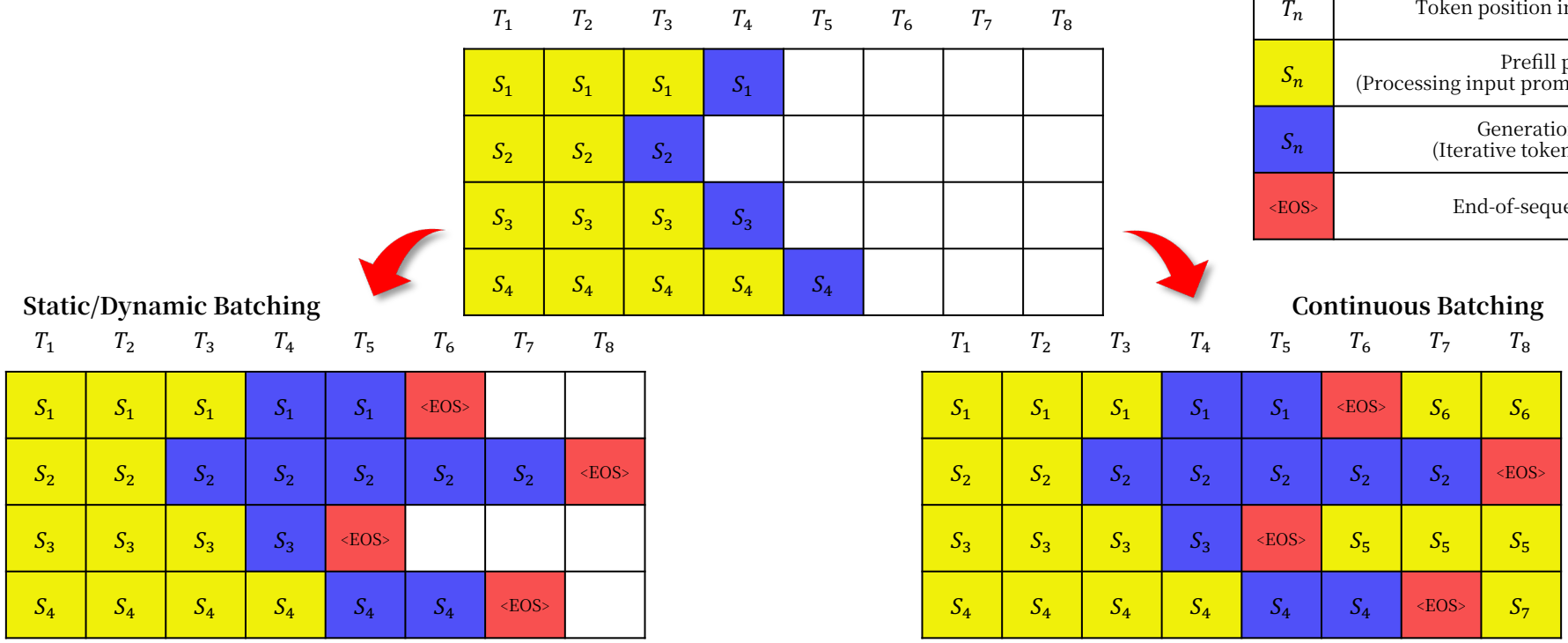
Continuous Batching [41]

- Limitations of Continuous Batching
  - Memory Fragmentation: 각 sequence의 KV cache가 연속적 memory 공간 요구
  - Memory Pre-allocation Overhead: 각 요청에 대해 최대 sequence 길이를 기준으로 memory 사전 할당
- vLLM + PagedAttention
  - Block-wise Storage: KV cache를 물리적으로 비연속적인 block 단위로 저장
  - Dynamic Allocation: 필요한 만큼의 page만 동적 할당하여 불필요한 memory overhead 방지
  - Naive static batching에 비해 2배 이상 성능 차이



[41]

T_n	Token position in the sequence
S_n	Prefill phase (Processing input prompt tokens in parallel)
S_n	Generation phase (Iterative token generation)
<EOS>	End-of-sequence token



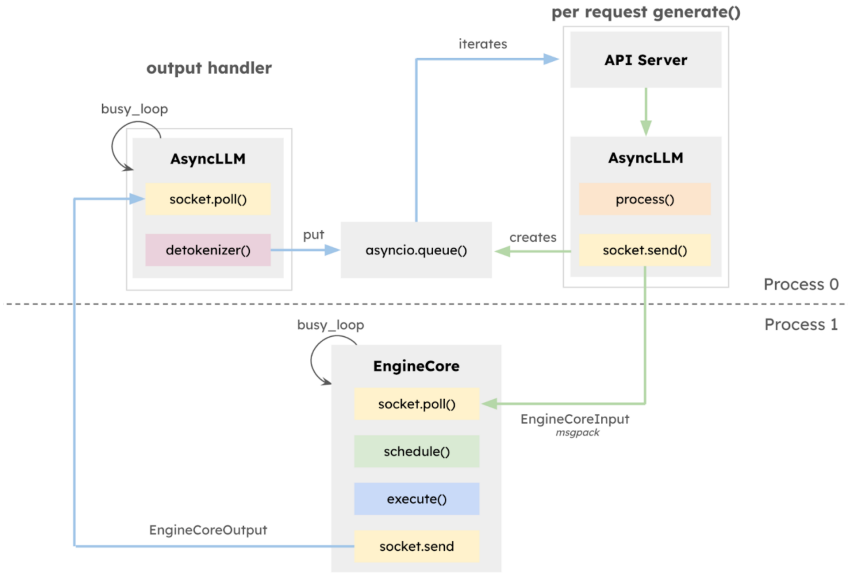
[41] https://www.anyscale.com/blog/continuous-batching-llm-inference

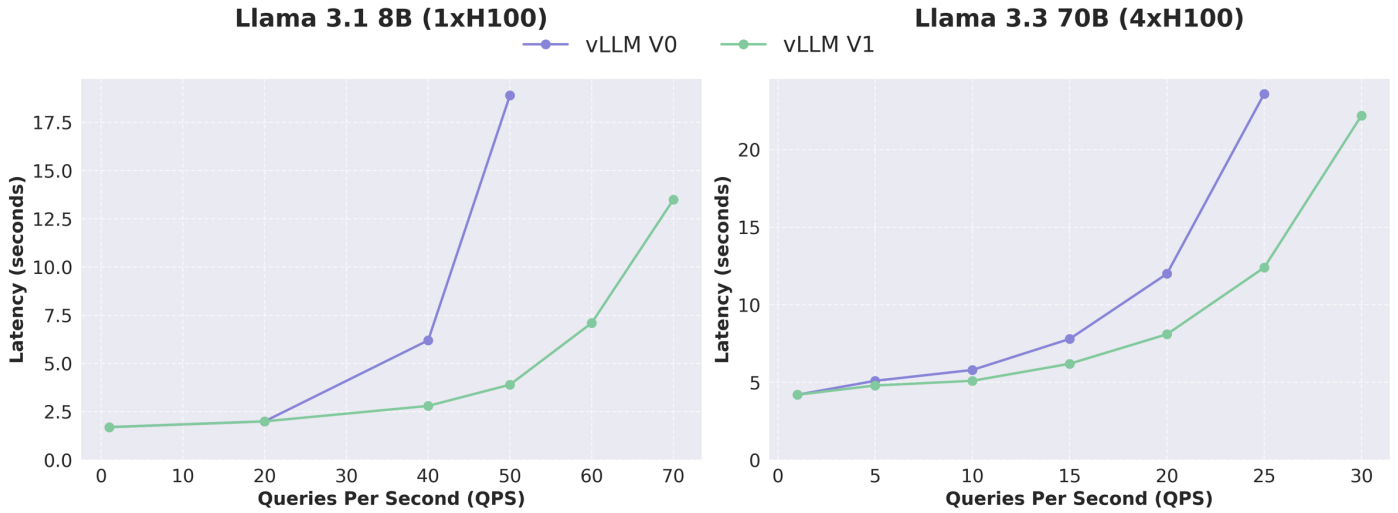
V0 Engine[43] vs. V1 Engine[44, 45]

- Optimized Execution Loop & API Server: EngineCore와 AsyncLLM 분리로 API server, token화 등 CPU 작업과 GPU model 실행을 완전 비동기 병렬화
- Simple & Flexible Scheduler: “prefill”, “decode” 구분 제거, “{request_id: num_tokens}” 기반 동적 token 할당으로 chunked-prefill, 사전 caching, 추측 decoding 지원
- Zero-Overhead Prefix Caching: Hash+LRU cache 구조 최적화로 cache hit rate 0%여도 1% 미만 성능 저하
- Clean Architecture for Tensor-Parallel Inference: Worker 상태 caching 후 diff만 전송, scheduler, worker 분리로 IPC overhead 대폭 감소
- Efficient Input Preparation: Persistent batch 기법으로 입력 tensor 재생성 없이 diff만 적용, Numpy 활용으로 CPU overhead 최소화
- torch.compile & Piecewise CUDA Graphs: Model 최적화 자동화 및 유연한 CUDA graph 통합으로 kernel customizing 최소화
- Enhanced Support for Multimodal LLMs: 비차단 image 전처리, image hash 기반 KV cache, encoder cache를 활용한 chunked-prefill 구현
- FlashAttention 3: 동적 batch 환경에서 최적화된 고성능 attention kernel 제공
- 환경 변수 **VLLM_USE_V1=1** 를 통해 V1 Engine 활성화

Versioning Scheme Explained [46]

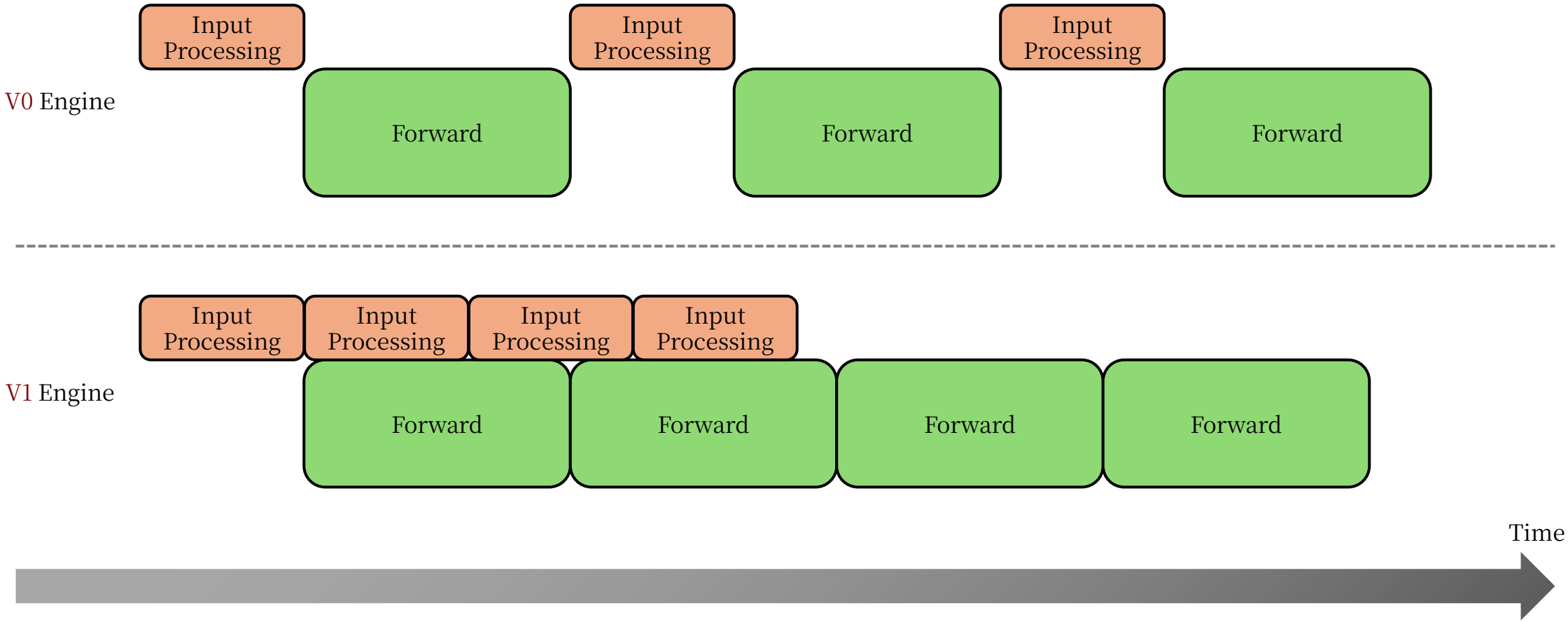
- **v0.9**: Last supported version of the V0 engine, with frozen features and continued testing.
- **v0.10**: Version marking the start of V0 code removal.
- **v0.11**: First version without any V0 components.





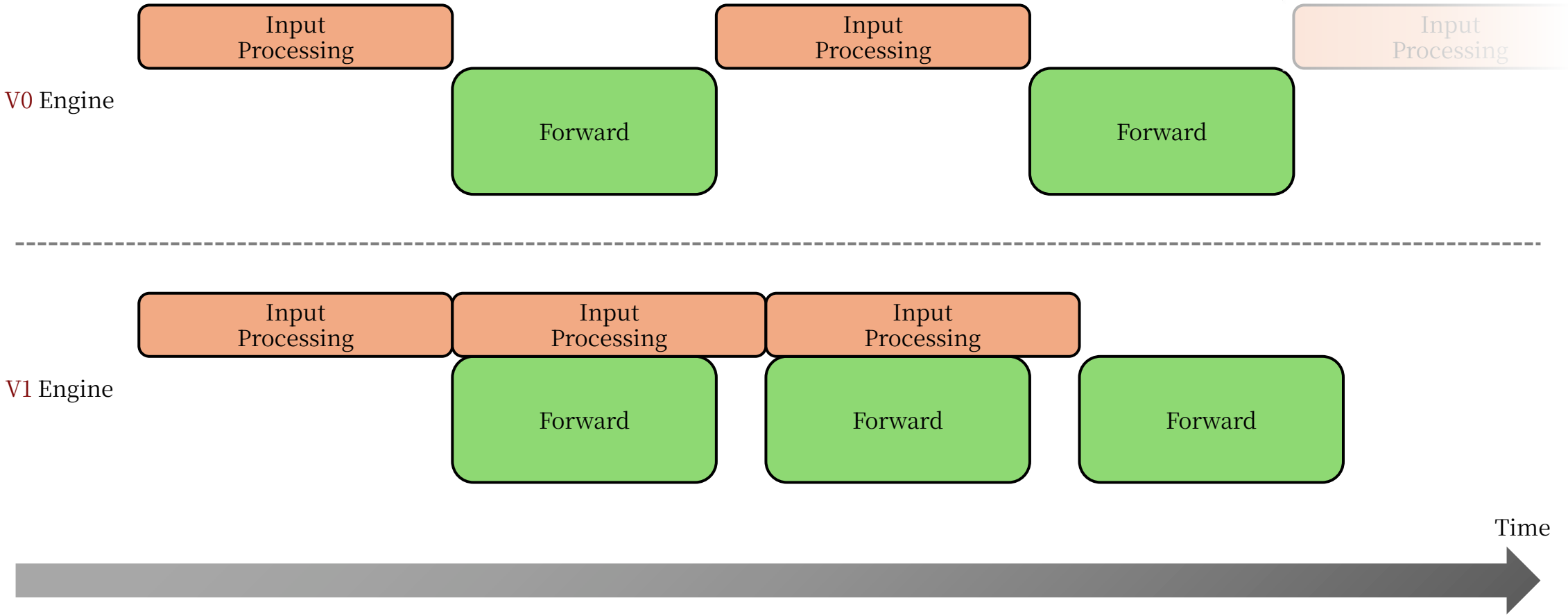
[43] https://docs.vllm.ai/en/v0.9.2/design/arch_overview.html
[44] https://docs.vllm.ai/en/v0.9.2/usage/v1_guide.html
[45] https://blog.vllm.ai/2025/01/27/v1-alpha-release.html
[46] https://github.com/vllm-project/vllm/issues/18571

V0 Engine[43] vs. V1 Engine[44, 45]



[43] https://docs.vllm.ai/en/v0.9.2/design/arch_overview.html
[44] https://docs.vllm.ai/en/v0.9.2/usage/v1_guide.html
[45] https://blog.vllm.ai/2025/01/27/v1-alpha-release.html

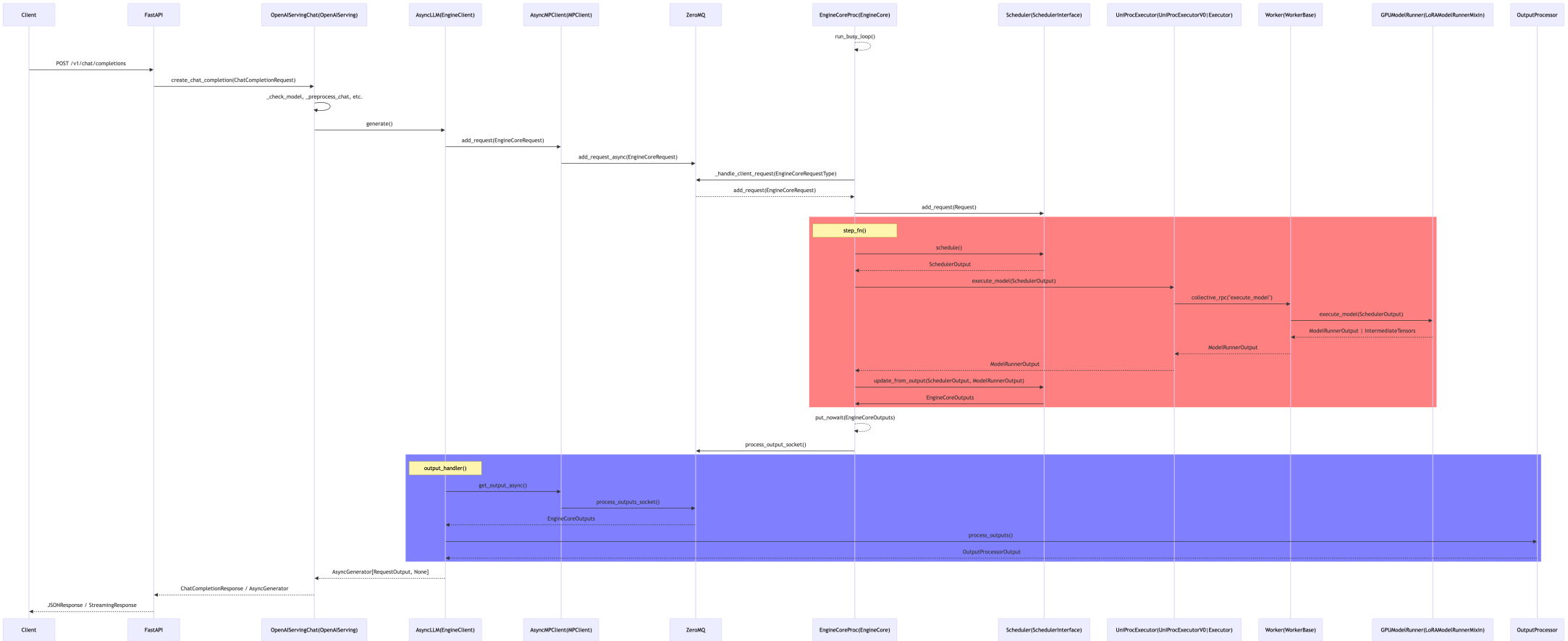
V0 Engine[43] vs. V1 Engine[44, 45]



[43] https://docs.vllm.ai/en/v0.9.2/design/arch_overview.html
[44] https://docs.vllm.ai/en/v0.9.2/usage/v1_guide.html
[45] https://blog.vllm.ai/2025/01/27/v1-alpha-release.html

Chat Completions

❑ vLLM의 /v1/chat/completions 처리 과정 [47]



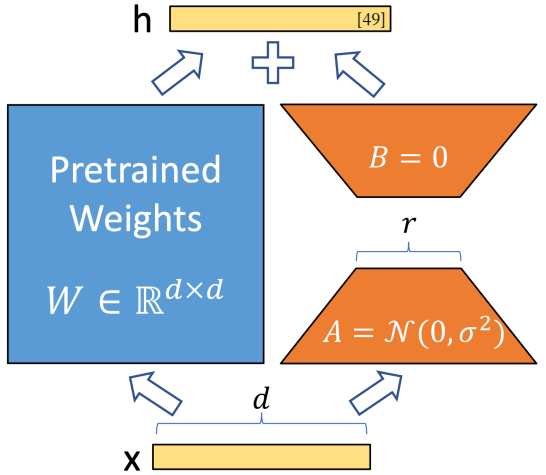
[47] https://zerohertz.github.io/vllm-openai-2/#Conclusion

PART 4

Production Deployment

LoRA Adapters

- LoRA (Low-Rank Adaptation) [48]
  - 기존 model의 weight 행렬에 대해 전체를 학습하지 않고 low-rank matrix (A, B)만 학습
  - $\Delta W = BA$를 더해 model이 비용 효율적으로 새로운 data에 적응



- Static serving LoRA adapters

```
vllm serve Qwen/Qwen3-0.6B --max-model-len 8192 \
    --reasoning-parser qwen3 \
    --enable-lora \
    --lora-modules phh/Qwen3-0.6B-TLDR-Lora=phh/Qwen3-0.6B-TLDR-Lora
```

- Dynamic serving LoRA adapters

```
VLLM_ALLOW_RUNTIME_LORA_UPDATING=True vllm serve Qwen/Qwen3-0.6B --max-model-len 8192 \
    --reasoning-parser qwen3 \
    --enable-lora
```

```
Fetching 16 files: 100%
WARNING 07-30 22:43:35 [cpu.py:250] Pin memory is not supported on CPU.
INFO 07-30 22:43:35 [serving_models.py:186] Loaded new LoRA adapter: name 'phh/Qwen3-0.6B-TLDR-Lora', path 'phh/Qwen3-0.6B-TLDR-Lora'
INFO:     127.0.0.1:54544 - "POST /v1/load_lora_adapter HTTP/1.1" 200 OK
$ curl -X POST http://localhost:8000/v1/load_lora_adapter \
        -H "Content-Type: application/json" \
        -d '{
    "lora_name": "phh/Qwen3-0.6B-TLDR-Lora",
    "lora_path": "phh/Qwen3-0.6B-TLDR-Lora"
}'

Success: LoRA adapter 'phh/Qwen3-0.6B-TLDR-Lora' added successfully.
```

```
INFO 07-30 22:44:33 [serving_models.py:203] Removed LoRA adapter: name 'phh/Qwen3-0.6B-TLDR-Lora'
INFO:     127.0.0.1:54550 - "POST /v1/unload_lora_adapter HTTP/1.1" 200 OK
$ curl -X POST http://localhost:8000/v1/unload_lora_adapter \
        -H "Content-Type: application/json" \
        -d '{
    "lora_name": "phh/Qwen3-0.6B-TLDR-Lora"
}'

Success: LoRA adapter 'phh/Qwen3-0.6B-TLDR-Lora' removed successfully.
```


[48] https://docs.vllm.ai/en/v0.9.2/features/lora.html
[49] https://arxiv.org/abs/2106.09685

LoRA Adapters

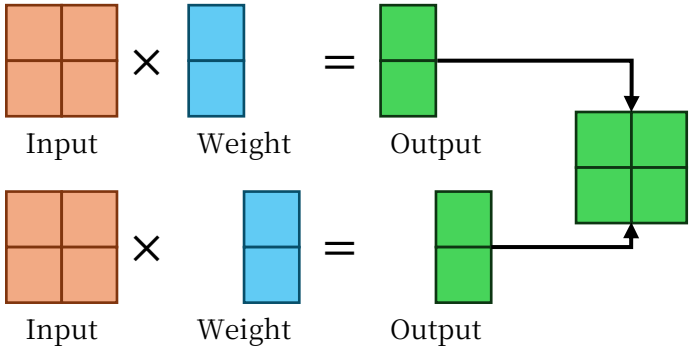

```
{
  "id": "chatcmpl-7e7c1d2b3c1843178e309164e93a731d",
  "object": "chat.completion",
  "created": 1753882095,
  "model": "Qwen/Qwen3-0.6B",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "reasoning_content":      ,
        "content": "저는 당신의 요청에 따라 **Ultrathinking AI Agent Framework인 AgentKit**을 소개하고 있습니다. AgentKit는 블록체인 환경에서 AI Agent를 빠르게 만들기 위한 **비법**을 설명하고자 합니다.\n\n---\n\n### **AgentKit: 블록체인의 Ultrathinking AI Agent Framework**\n\nAgentKit는 **Ultrathinking Engineer**가 블록체인 환경에서 AI Agent를 만들기 위해 사용할 수 있는 **Framework**입니다. AgentKit는 빠르게 AI Agent를 구축할 수 있도록 **공학적 기술**과 **실용적인 도구**를 결합하여 제공합니다.\n\n#### ✅ **1. 빠르게 AI Agent를 만들기 위한 기술**\n- **Ultrathinking** 능력: AgentKit는 빠르게 핵심 기능을 핵심으로 갖춘 AI Agent를 만들 수 있도록 도와줍니다.\n- **Multi-Agent System** 구조: 여러 AI Agent를 결합하여 복잡한 문제를 해결할 수 있도록 설계되었습니다.\n- **High-performance** 기술: 빠르게 실행되는 AI Agent를 위한 최적화 기술을 지원합니다.\n\n#### ✅ **2. 블록체인 환경에서 빠르게 구축**\n- **Smart Contracts**과 **Blockchain**의 결합: AgentKit는 블록체인 기반의 AI Agent를 구축할 수 있도록 도와줍니다.\n- **Decentralized** 기반: AgentKit는 빠르게 AI Agent를 구축하는 데 최적화된 기술입니다.\n- **Scalability** 기능: AgentKit는 블록체인 기반의 AI Agent를 확장할 수 있도록 설계되었습니다.\n\n#### ✅ **3. 실용적인 도구**\n- **AgentKit API**를 통해 빠르게 AI Agent를 구축할 수 있습니다.\n- **Smart Contracts**와 **Blockchain**의 결합: AgentKit는 블록체인 기반의 AI Agent를 구축할 수 있도록 도와줍니다.\n\n---\n\n### **Ultrathinking Engineer가 블록체인 환경에서 빠르게 AI Agent를 만들기 위한 비법**\n\n- **Ultrathinking Engine**는 빠르게 핵심 기능을 핵심으로 갖춘 AI Agent를 만들 수 있도록 도와줍니다.\n- **Multi-Agent System** 구조: 여러 AI Agent를 결합하여 복잡한 문제를 해결할 수 있도록 설계되었습니다.\n- **High-performance** 기술: 빠르게 실행되는 AI Agent를 위한 최적화 기술을 지원합니다.\n\n---\n\n### **AgentKit의 특징**\n\n- **Ultrathinking Engine**로 빠르게 핵심 기능을 핵심으로 갖춘 AI Agent를 만들 수 있습니다.\n- **Multi-Agent System** 구조로 복잡한 문제를 해결할 수 있도록 설계되었습니다.\n- **High-performance** 기술로 빠르게 실행되는 AI Agent를 지원합니다.\n\n---\n\n### **Ultrathinking Engineer가 블록체인 환경에서 빠르게 AI Agent를 만들기 위한 비법**\n\n- **Ultrathinking Engine**로 빠르게 핵심 기능을 핵심으로 갖춘 AI Agent를 만들 수 있습니다.\n- **Multi-Agent System** 구조로 복잡한 문제를 해결할 수 있도록 설계되었습니다.\n- **High-performance** 기술로 빠르게 실행되는 AI Agent를 지원합니다.\n\n---\n\n**항상에 블록체인 기술과 AI Agent를 결합해 더 나은 서비스를 제공해 드릴게요!** 😊",
```

```
{
  "id": "chatcmpl-5f9761a5127b434a950bd63bc7a3f537",
  "object": "chat.completion",
  "created": 1753882000,
  "model": "phh/Qwen3-0.6B-TLDR-Lora",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "reasoning_content":      ,
        "content": "저작권자: [이름] \n작성일: [날짜] \n작성자: [이름] \n\n---\n\n**강천성 (Logan Kang) / Dable & Coinbase (BASE) @Logan** \n**Ultrathinking AI Agent Framework인 AgentKit를 소개합니다.** \n\n블록체인 환경에서 AI Agent를 만들기 위한 빠르고 효율적인 방법은 AgentKit입니다. AgentKit는 다양한 블록체인 기술과 기술을 결합하여, AI Agent를 구축할 수 있도록 합니다. AgentKit는 단순한 기능을 제공하지만, 기술의 혁신과 협업을 통해 빠르게 개발할 수 있도록 설계되었습니다. \n\n---\n\n**고석현 / Sionic AI @정자초등학교** \n**TBD**\n\n---\n\n**채승우 / 하피챗 @cyoo** \n**단일 LLM 요청으로 이루어진 기존 AI 플레이 시스템의 한계** \n**astrsk에서 에이전틱 워크플로를 만든 과정** \n이러한 기능은 단순한 LLM을 요청하는 것보다, AI를 활용한 기능을 추가하는 것이 더 효과적입니다. astrsk의 에이전틱 워크플로는 기존 시스템의 한계를 넘어, 다양한 라이브러리를 활용해 복잡한 문제를 해결하고 있습니다.\n\n---\n\n**현지웅 / 퓨튼테크놀로지스 @jiwunghyun** \n**Chat 인터페이스와 플랫폼은 공존할 수 있을까?** \n**각 서비스가 풀고 있는 전략** \n**AI 서비스를 운영하는 입장에서 들었던 생각 및 고민** \n이러한 질문은 기술의 혁신을 위해 필요합니다. AI 서비스를 직접 운영하는 것이 아니라, 플랫폼을 통해 다양한 기능을 제공하는 것이 더 효과적입니다.\n\n---\n\n**김우영 / 모니모니 @wooing** \n**Agentic Coding, Max Mode: 지치지 않는 일꾼, 극한까지 활용하기** \n**Agentic Coding을 한도 꽉꽉 채워서 한 푼도 아깝지 않게 사용하고 싶지 않으신가요?** \n**이러한 워크플로우를 병행하면서, 어떻게 만들어나가지는?** \n이러한 질문은 기술의 혁신과 효율성을 강조합니다.\n\n---\n\n**오효근 / 제논 @zerohertz** \n**오픈소스 LLM, vLLM으로 Production까지** \n**vLLM과 비교 및 특장점** \n**내부 구조를 간단히 살펴보겠습니다.** \n이러한 설명은 기술의 구조를 이해하는 데 도움이 됩니다.\n\n---\n\n**홍승우 (Martin Hong) / Baryon Labs @martin** \n**백그라운드 에이전트 - 개발자와 로봇의 전쟁시작** \n**Domain Border를 지정하는 것 같은 에이전트** \n이러한 설명은 기술의 혁신을 위한 프레임워크를 설명합니다.\n\n---\n\n**황민호 (Robin Hwang) / 카카오** \n**AI Agent 시대, 코딩을 넘어 AI 오케스트레이션으로 진화** \n**실제 사례들을 통해 복잡한 문제를 해결하고, 창의적인 서비스를 가능하게 하는지** \n이러한 질문은 AI의 역할을 이해하고, 기술의 혁신을 위한 실례를 제공합니다.\n\n---\n\n**박정환 (maywell) / Instruct.KR & 벙커키즈 @Jeonghwan Park** \n**TBD**",
```

Parallelism Strategies [50, 51]

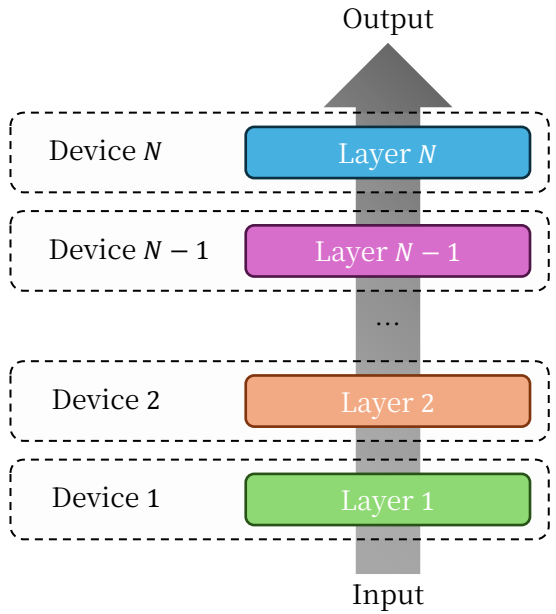
❑ Tensor Parallelism (TP)

- 각 model layer 내 model parameter를 여러 GPU에 분산하여 처리
  - → Model이 너무 커서 여러 GPU를 single node로 추론할 때
  - → 더 높은 처리량을 위해 더 많은 KV cache 공간을 마련하기 위해 GPU 당 memory pressure를 줄여야할 때



❑ Pipeline Parallelism (PP)

- Model layer를 여러 GPU에 분산하여 model의 여러 부분 순차적 처리
  - → Model이 너무 커서 여러 node에 걸쳐 분산해야할 때
  - → Layer 분산이 tensor sharding보다 더 효율적인 매우 깊고 좁은 model일 때

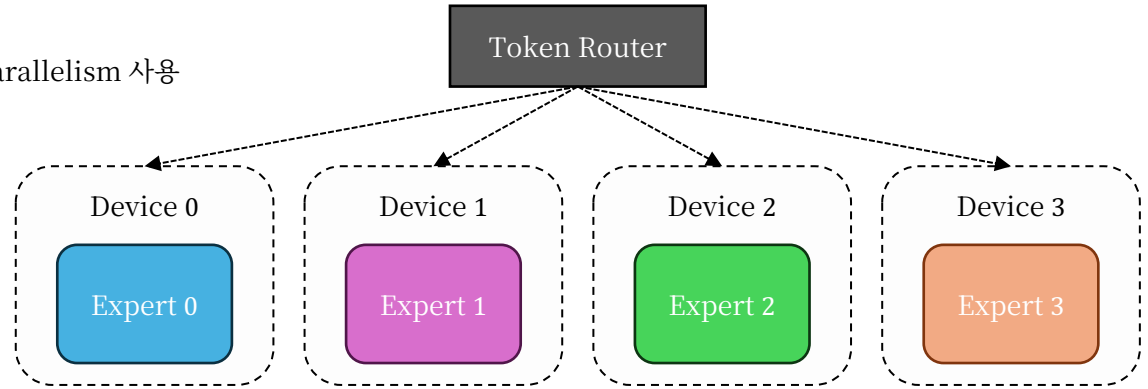


[50] https://docs.vllm.ai/en/v0.9.2/configuration/optimization.html#parallelism-strategies
[51] https://www.youtube.com/watch?v=4i76hmmnIEo

Parallelism Strategies [50, 51]

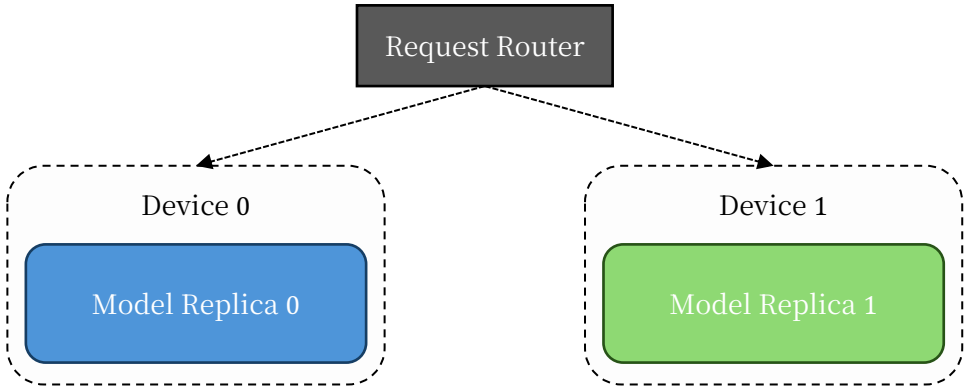
❑ Expert Parallelism (EP)

- Mixture of Experts (MoE) model을 위한 특수한 형태의 병렬 처리
  - → "--enable-expert-parallel" 사용 시 MoE layer에서 tensor parallelism 대신 expert parallelism 사용
  - → MoE model을 사용할 때
  - → GPU 간 expert 연산 부하를 분산할 때



❑ Data Parallelism (DP)

- 여러 GPU에 걸쳐 전체 model을 복제하고 여러 요청 batch를 병렬 처리
  - → 전체 model을 복제하기에 충분한 GPU를 보유한 때
  - → Model 크기보다 처리량을 확장해야할 때
  - → 요청 batch 간 격리가 유리한 다중 사용자 환경일 때



[50] https://docs.vllm.ai/en/v0.9.2/configuration/optimization.html#parallelism-strategies
[51] https://www.youtube.com/watch?v=4i76hmmnIEo

Multi-node Distributed Inference

- Ray를 통한 cluster 구성 과정 [52]
  - VLLM_HOST_IP [53]: vLLM 내부 통신에 사용할 node의 IP 주소
  - GLOO_SOCKET_IFNAME [54]: PyTorch Gloo bakcend가 사용할 network interface 이름
  - NCCL_IB_DISABLE [55]: NCCL의 InfiniBand (IB) network 사용 여부

```
# NOTE master
docker run \
    -d \
    --entrypoint /bin/bash \
    --network host \
    --shm-size 10.24g \
    -v /dev/shm:/dev/shm \
    -v "${PATH_TO_HF_HOME}:/root/.cache/huggingface" \
    -e VLLM_HOST_IP=${NODE_IP} \
    -e GLOO_SOCKET_IFNAME=${GLOO_SOCKET_IFNAME} \
    -e NCCL_IB_DISABLE=1 \
    vllm/vllm-openai:v0.9.2 \
    -c "uv pip install ray[default] --system && ray start --head --port=6379 --disable-usage-stats --dashboard-host=0.0.0.0 && tail -f /dev/null"
```

```
# NOTE worker
docker run \
    -d \
    --entrypoint /bin/bash \
    --network host \
    --shm-size 10.24g \
    -v /dev/shm:/dev/shm \
    -v "${PATH_TO_HF_HOME}:/root/.cache/huggingface" \
    -e VLLM_HOST_IP=${NODE_IP} \
    -e GLOO_SOCKET_IFNAME=${GLOO_SOCKET_IFNAME} \
    -e NCCL_IB_DISABLE=1 \
    vllm/vllm-openai:v0.9.2 \
    -c "uv pip install ray[default] --system && ray start --block --address=${MASTER_NODE_IP}:6379 && tail -f /dev/null"
```

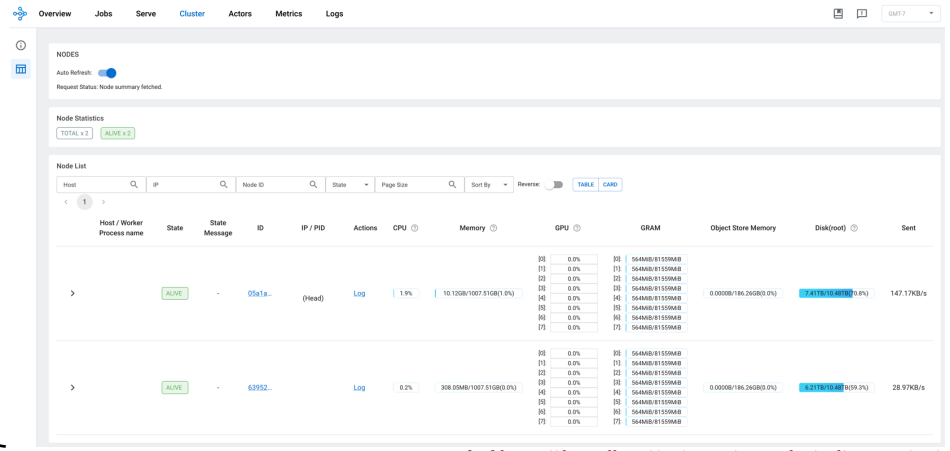
Head Node

```
# ray start --head --port=6379 --disable-usage-stats --dashboard-host=0.0.0.0
Usage stats collection is disabled.

Local node IP:

--------------------
Ray runtime started.
--------------------

Next steps
  To add another node to this Ray cluster, run
    ray start --address='              :6379'

  To connect to this Ray cluster:
    import ray
    ray.init()

  To submit a Ray job using the Ray Jobs CLI:
    RAY_ADDRESS='http://              :8265' ray job submit --working-dir . -- python my_script.py

  See https://docs.ray.io/en/latest/cluster/running-applications/job-submission/index.html
  for more information on submitting Ray jobs to the Ray cluster.

  To terminate the Ray runtime, run
    ray stop

  To view the status of the cluster, use
    ray status

  To monitor and debug Ray, view the dashboard at
                 :8265

  If connection to the dashboard fails, check your firewall settings and network configuration.
#
```

Worker Node

```
# ray start --block --address=              :6379
Local node IP: 192.168.75.174
[2025-07-29 05:37:09,112 W 646 646] global_state_accessor.cc:435: Retrying to get node with node ID 6395285e5a2a36e5773e77fd095490d63eda1f33869da1d09291b522

--------------------
Ray runtime started.
--------------------

To terminate the Ray runtime, run
  ray stop

--block
  This command will now block forever until terminated by a signal.
  Running subprocesses are monitored and a message will be printed if any of them terminate unexpectedly. Subprocesses exit with SIGTERM will be treated as graceful, thus NOT reported.
```

```
# ray list nodes

======== List: 2025-07-29 05:43:23.404417 ========
Stats:
------------------------------
Total: 2

Table:
------------------------------
    NODE_ID                                                  NODE_IP          IS_HEAD_NODE    STATE
 0  05a1a18c99ab485e6af30b864bd1227c09815f71f40eac26ddbc3f9a                  True            ALIVE




 1  6395285e5a2a36e5773e77fd095490d63eda1f33869da1d09291b522                  False           ALIVE
```



[52] https://docs.vllm.ai/en/v0.9.2/examples/online_serving/run_cluster.html
[53] https://docs.vllm.ai/en/v0.9.2/usage/security.html
[54] https://docs.pytorch.org/docs/stable/distributed.html#common-environment-variables
[55] https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/env.html#nccl-ib-disable

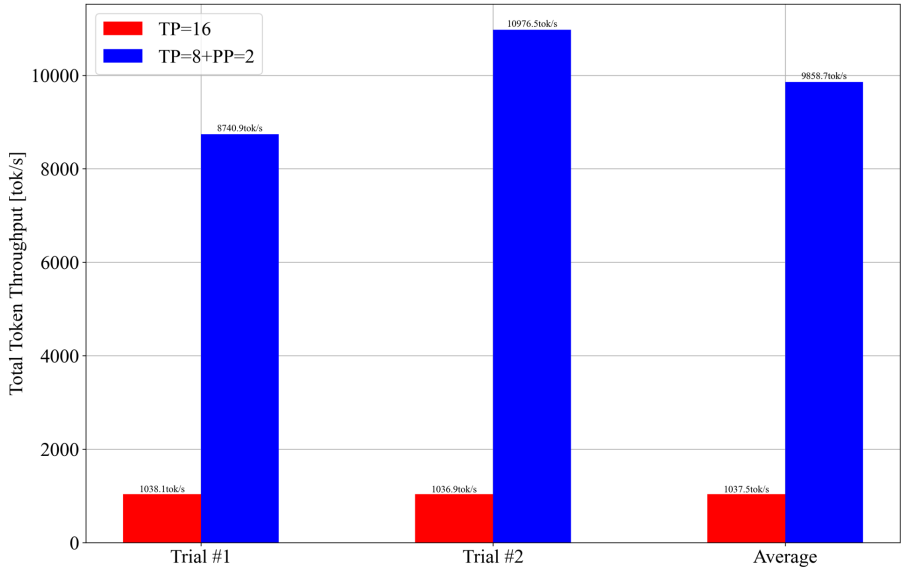
Multi-node Distributed Inference

❑ Multi-Node Multi-GPU [56, 57]

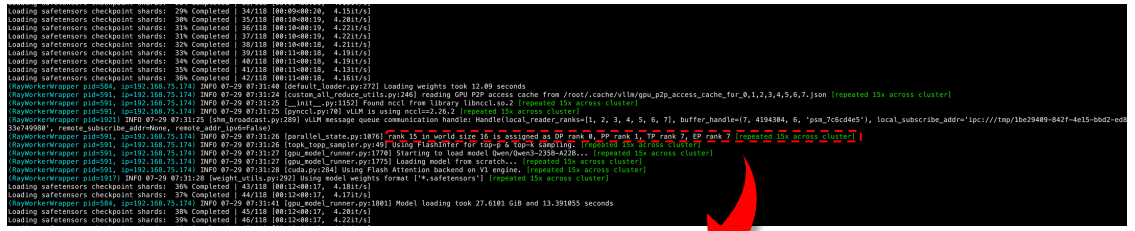
(tensor parallel plus pipeline parallel inference)

- If your model is too large to fit in a single node, you can use tensor parallel together with pipeline parallelism.
- The tensor parallel size is the number of GPUs you want to use in each node, and the pipeline parallel size is the number of nodes you want to use.
- E.g., if you have 16 GPUs in 2 nodes (8 GPUs per node), you can set **the tensor parallel size to 8** and **the pipeline parallel size to 2**.

❑ TP 16 vs. TP 8 + PP 2




```
# NOTE  master
vllm serve Qwen/Qwen3-235B-A22B \
    --distributed-executor-backend ray \
    --host=0.0.0.0 --port=8080 \
    --tensor-parallel-size=8 --pipeline-parallel-size=2 \
    --gpu_memory_utilization=0.95 \
    --reasoning-parser qwen3
```



```
rank 15 in world size 16 is assigned as DP rank 0, PP rank 1, TP rank 7, EP rank 7 [repeated 15x across cluster]
```

Host / Worker Process name	State	State Message	ID	IP / PID	Actions	CPU	Memory	GPU	GRAM
›	ALIVE	-	a3b57…	(Head)	Log	17.7%	30.53GB/1007.51GB(3.0%)	[0]: 100.0% [1]: 100.0% [2]: 100.0% [3]: 100.0% [4]: 100.0% [5]: 100.0% [6]: 100.0% [7]: 100.0%	[0]: 80813MiB/81559MiB [1]: 81021MiB/81559MiB [2]: 81019MiB/81559MiB [3]: 81019MiB/81559MiB [4]: 81019MiB/81559MiB [5]: 80987MiB/81559MiB [6]: 81019MiB/81559MiB [7]: 80967MiB/81559MiB
›	ALIVE	-	4a527…		Log	10.7%	21.21GB/1007.51GB(2.1%)	[0]: 100.0% [1]: 100.0% [2]: 100.0% [3]: 100.0% [4]: 100.0% [5]: 100.0% [6]: 100.0% [7]: 100.0%	[0]: 80363MiB/81559MiB [1]: 80525MiB/81559MiB [2]: 80523MiB/81559MiB [3]: 80523MiB/81559MiB [4]: 80523MiB/81559MiB [5]: 80491MiB/81559MiB [6]: 80523MiB/81559MiB [7]: 80513MiB/81559MiB

[56] https://docs.vllm.ai/en/v0.9.2/serving/distributed_serving.html
[57] https://blog.vllm.ai/2025/02/17/distributed-inference.html

Multi-node Distributed Inference

❑ RDMA (Remote Direct Memory Access) [58]

- Network에 연결된 두 node의 memory 간 CPU, cache, 운영 체제의 개입 없이 data를 직접 전송하는 기술
- E.g., InfiniBand, RoCE, iWARP

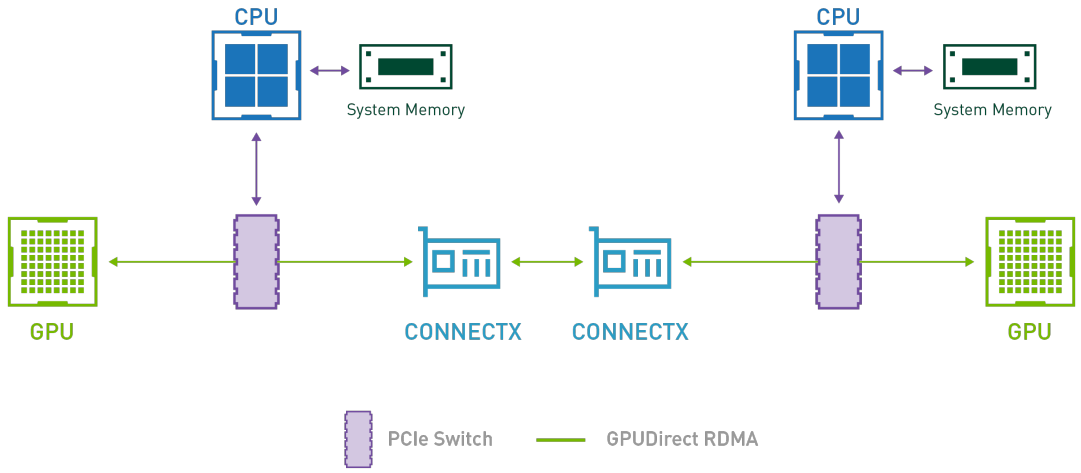
❑ InfiniBand [59]

- RDMA를 native로 지원하여 CPU 개입 없이 node 간 memory 직접 전송 가능
- 전용 switch와 network adapter (Host Channel Adapter, HCA) 사용 (Ethernet network와는 다른 독자적 architecture)

❑ RoCEv2 (RDMA over Converged Ethernet v2) [58]

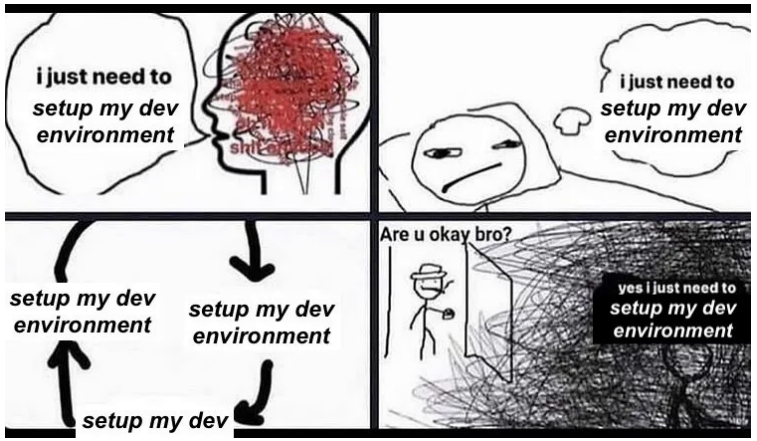
- 표준 ethernet network를 통해 RDMA를 구현하는 protocol

항목	InfiniBand	RoCE
Network type	전용 InfiniBand fabric	Ethernet 기반
구축 비용	별도 장비 필요, 비용 높음	상대적으로 저렴, 기존 infra 활용 가능
성능	최고 성능, ultra-low latency	Lossless Ethernet 환경에서 InfiniBand에 근접
호환성	HPC/AI 특화, 범용성 낮음	범용성 높음, 기존 infra와 통합 용이
관리 복잡도	전용 환경, 관리 일관성	Ethernet tuning 필요, 설정 복잡할 수 있음



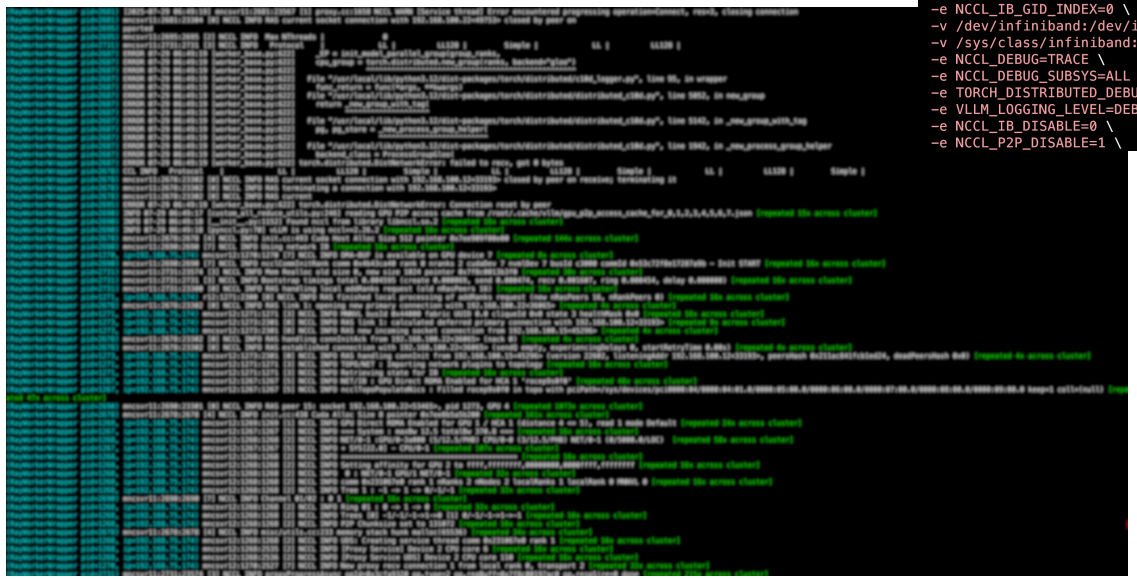
[58] https://zerohertz.github.io/distributed-computing-rdma-roce/
[59] https://developer.nvidia.com/gpudirect

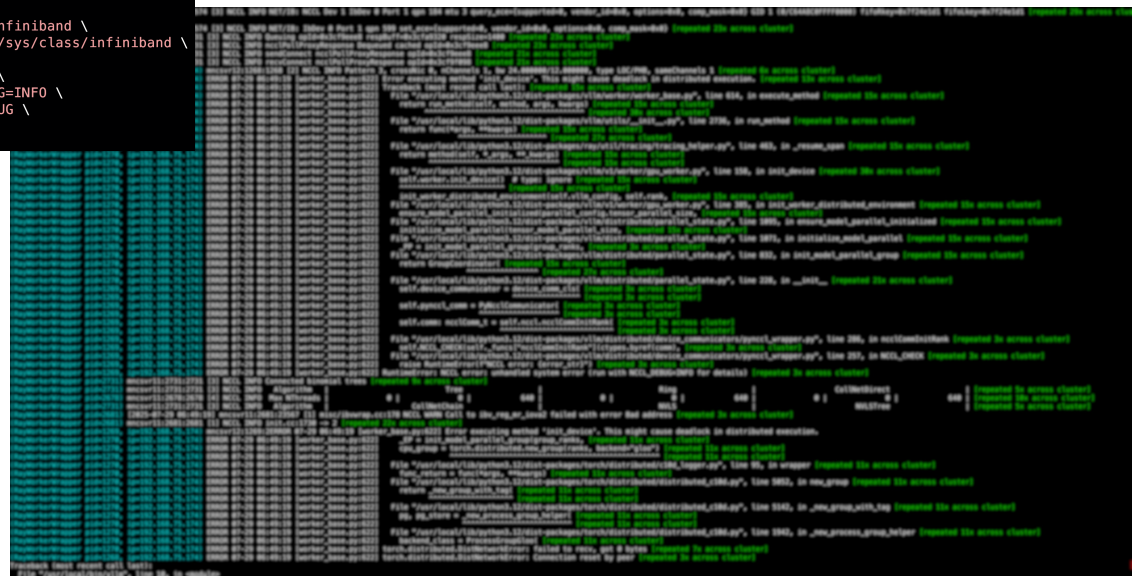
Multi-node Distributed Inference




```
-e GLOO_SOCKET_IFNAME=ens3f0 \
-e NCCL_SOCKET_IFNAME=ens3f0 \
-e NCCL_IB_HCA=rocep101s0f0 \
-e NCCL_IB_GID_INDEX=0 \
-v /dev/infiniband:/dev/infiniband \
-v /sys/class/infiniband:/sys/class/infiniband \
-e NCCL_DEBUG=TRACE \
-e NCCL_DEBUG_SUBSYS=ALL \
-e TORCH_DISTRIBUTED_DEBUG=INFO \
-e VLLM_LOGGING_LEVEL=DEBUG \
-e NCCL_IB_DISABLE=0 \
-e NCCL_P2P_DISABLE=1 \
```

[60]





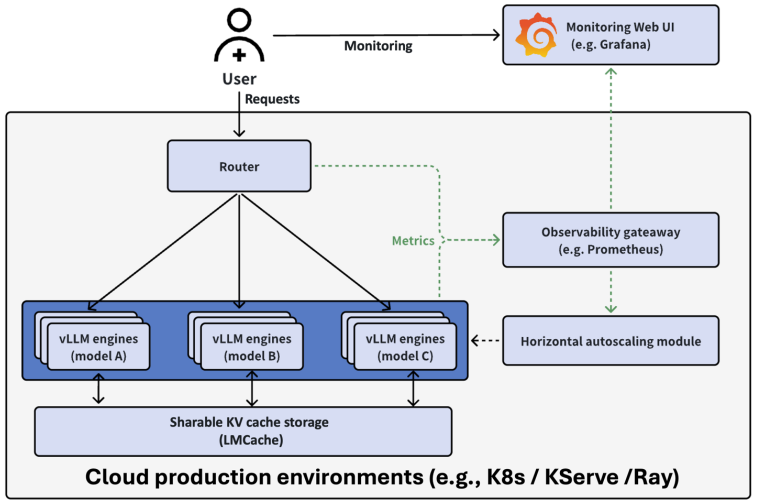
[60] https://docs.vllm.ai/en/v0.9.2/serving/distributed_serving.html#gpudirect-rdma

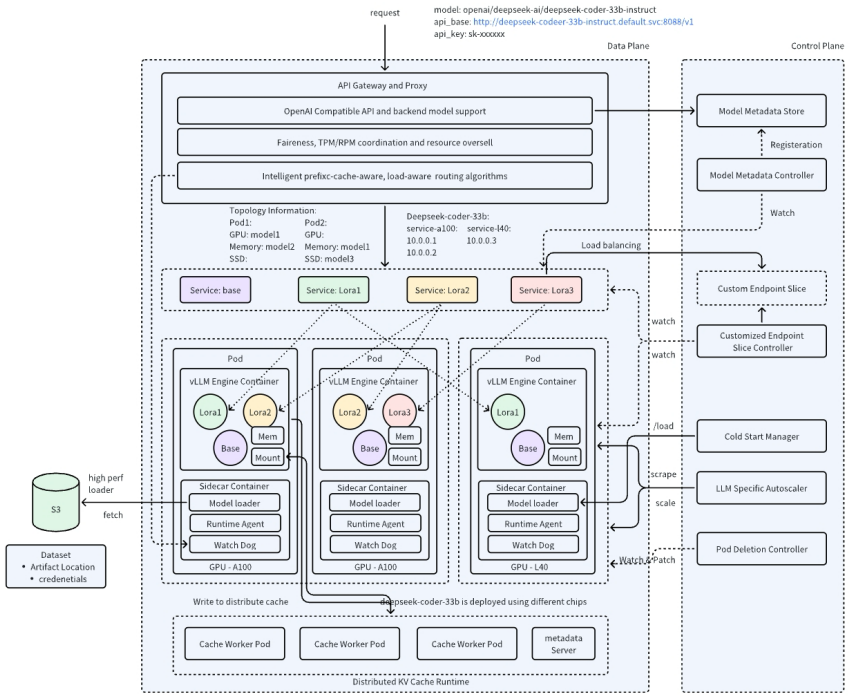
Production Deployment with GPU Cluster (Kubernetes)

- Kubernetes 배포 [61]
  - Model store, Network, IPC, 환경 변수 등의 자유도가 많은 만큼 관리가 어려움

- AIBrix [62, 63,64, 65]
  - vLLM을 위한 확장 가능하고 비용 효율적인 cloud native control plane
  - 고밀도 LoRA 관리, LLM gateway 및 routing, autoscaler, 분산 추론 및 분산 KV cache, …

- Production-stack [66, 67, 68, 69]
  - vLLM을 통한 LLM 배포 시 production에 최적화된 codebase
  - LMCache 기반 KV cache 공유 및 저장, prefix-aware routing, Helm chart를 통한 배포, …

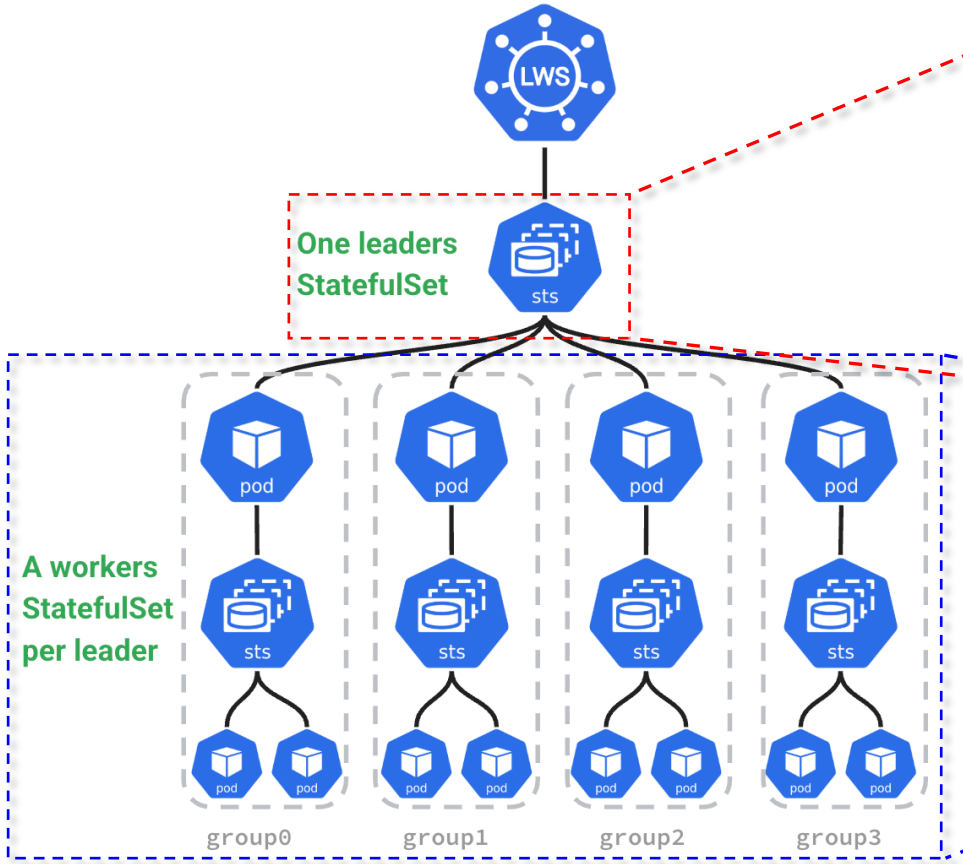




[61] https://docs.vllm.ai/en/v0.9.2/deployment/k8s.html
[62] https://github.com/vllm-project/aibrix
[63] https://arxiv.org/abs/2504.03648
[64] https://blog.vllm.ai/2025/02/21/aibrix-release.html
[65] https://aibrix.readthedocs.io/latest/
[66] https://github.com/vllm-project/production-stack
[67] https://docs.vllm.ai/en/v0.9.2/deployment/integrations/production-stack.html
[68] https://blog.vllm.ai/2025/01/21/stack-release.html
[69] https://blog.vllm.ai/production-stack/

Production Deployment with GPU Cluster (Kubernetes)

- LWS (LeaderWorkerSet) [70, 71]
  - Kubernetes 상에서 Leader-Worker architecture를 손쉽게 구현하고 관리할 수 있도록 도와주는 CRD 및 controller




```yaml
apiVersion: leaderworkerset.x-k8s.io/v1
kind: LeaderWorkerSet
metadata:
  name: vllm
spec:
  replicas: 2
  leaderWorkerTemplate:
    size: 2
    restartPolicy: RecreateGroupOnPodRestart
    leaderTemplate:
      metadata:
        labels:
          role: leader
      spec:
        containers:
          - name: vllm-leader
            image: docker.io/vllm/vllm-openai:latest
            env:
              - name: HUGGING_FACE_HUB_TOKEN
                value: <your-hf-token>
            command:
              - sh
              - -c
              - "bash /vllm-workspace/examples/online_serving/multi-node-serving.sh leader --ray_cluster_size=$(LWS_GROUP_SIZE);
                 python3 -m vllm.entrypoints.openai.api_server --port 8080 --model meta-llama/Meta-Llama-3.1-405B-Instruct --tensor-parallel-size 8 --pipeline_parallel_size 2"
            resources:
              limits:
                nvidia.com/gpu: "8"
                memory: 1124Gi
                ephemeral-storage: 800Gi
              requests:
                ephemeral-storage: 800Gi
                cpu: 125
            ports:
              - containerPort: 8080
            readinessProbe:
              tcpSocket:
                port: 8080
              initialDelaySeconds: 15
              periodSeconds: 10
            volumeMounts:
              - mountPath: /dev/shm
                name: dshm
        volumes:
          - name: dshm
            emptyDir:
              medium: Memory
              sizeLimit: 15Gi
    workerTemplate:
      spec:
        containers:
          - name: vllm-worker
            image: docker.io/vllm/vllm-openai:latest
            command:
              - sh
              - -c
              - "bash /vllm-workspace/examples/online_serving/multi-node-serving.sh worker --ray_address=$(LWS_LEADER_ADDRESS)"
            resources:
              limits:
                nvidia.com/gpu: "8"
                memory: 1124Gi
                ephemeral-storage: 800Gi
              requests:
                ephemeral-storage: 800Gi
                cpu: 125
            env:
              - name: HUGGING_FACE_HUB_TOKEN
                value: <your-hf-token>
            volumeMounts:
              - mountPath: /dev/shm
                name: dshm
        volumes:
          - name: dshm
            emptyDir:
              medium: Memory
              sizeLimit: 15Gi
---
apiVersion: v1
kind: Service
metadata:
  name: vllm-leader
spec:
  ports:
    - name: http
      port: 8080
      protocol: TCP
      targetPort: 8080
  selector:
    leaderworkerset.sigs.k8s.io/name: vllm
    role: leader
  type: ClusterIP
```

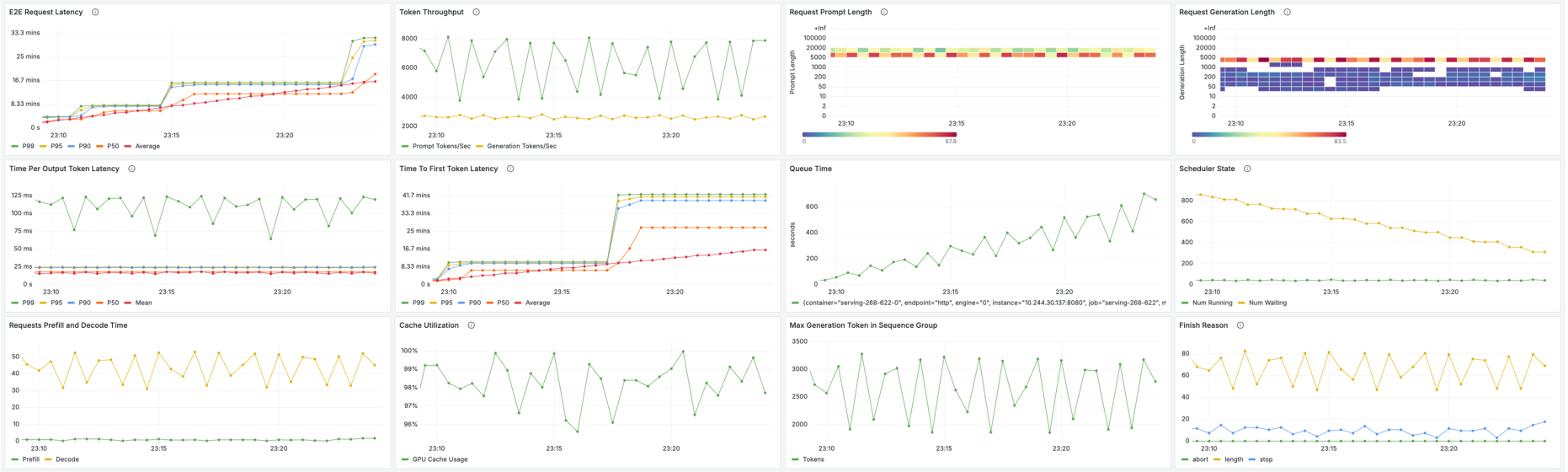
[70] https://docs.vllm.ai/en/v0.9.2/deployment/frameworks/lws.html
[71] https://github.com/kubernetes-sigs/lws

Observability (Prometheus + Grafana)

- "/metrics"를 통해 vLLM server의 Prometheus format metric 수집 가능 [72, 73]
  - vllm:request_success_total: 요청 완료 수 (EOS 도달 또는 max 길이 도달)
  - vllm:request_queue_time_seconds: queue 대기 시간
  - vllm:request_prefill_time_seconds: prefill 단계 소요 시간
  - vllm:request_decode_time_seconds: decoding 단계 소요 시간
  - vllm:request_max_num_generation_tokens: 생성된 token 최대값
  - …

```
zerohertz@zerohertz-mm:~
$ curl http://localhost:8000/metrics

# HELP vllm:lora_requests_info Running stats on lora requests.
# TYPE vllm:lora_requests_info gauge
vllm:lora_requests_info{max_lora="0",running_lora_adapters="",waiting_lora_adapters=""} 1.753628624009126e+09
# HELP vllm:cache_config_info Information of the LLMEngine CacheConfig
# TYPE vllm:cache_config_info gauge
vllm:cache_config_info{block_size="16",cache_dtype="auto",calculate_kv_scales="False",cpu_kvcache_space_bytes="4294967296",cpu_offload
_gb="0.0",enable_prefix_caching="None",gpu_memory_utilization="0.9",is_attention_free="False",num_cpu_blocks="0",num_gpu_blocks="2340"
,num_gpu_blocks_override="None",prefix_caching_hash_algo="builtin",sliding_window="None",swap_space="4.0",swap_space_bytes="4294967296
.0"} 1.0
# HELP vllm:num_preemptions_total Cumulative number of preemption from the engine.
# TYPE vllm:num_preemptions_total counter
vllm:num_preemptions_total{model_name="Qwen/Qwen3-0.6B"} 0.0
# HELP vllm:prompt_tokens_total Number of prefill tokens processed.
# TYPE vllm:prompt_tokens_total counter
vllm:prompt_tokens_total{model_name="Qwen/Qwen3-0.6B"} 14.0
# HELP vllm:generation_tokens_total Number of generation tokens processed.
# TYPE vllm:generation_tokens_total counter
vllm:generation_tokens_total{model_name="Qwen/Qwen3-0.6B"} 107.0
# HELP vllm:request_success_total Count of successfully processed requests.
# TYPE vllm:request_success_total counter
vllm:request_success_total{finished_reason="stop",model_name="Qwen/Qwen3-0.6B"} 1.0
# HELP vllm:iteration_tokens_total Histogram of number of tokens per engine_step.
# TYPE vllm:iteration_tokens_total histogram
vllm:iteration_tokens_total_sum{model_name="Qwen/Qwen3-0.6B"} 121.0
vllm:iteration_tokens_total_bucket{le="1.0",model_name="Qwen/Qwen3-0.6B"} 1099.0
vllm:iteration_tokens_total_bucket{le="8.0",model_name="Qwen/Qwen3-0.6B"} 1099.0
vllm:iteration_tokens_total_bucket{le="16.0",model_name="Qwen/Qwen3-0.6B"} 1100.0
vllm:iteration_tokens_total_bucket{le="32.0",model_name="Qwen/Qwen3-0.6B"} 1100.0
vllm:iteration_tokens_total_bucket{le="64.0",model_name="Qwen/Qwen3-0.6B"} 1100.0
vllm:iteration_tokens_total_bucket{le="128.0",model_name="Qwen/Qwen3-0.6B"} 1100.0
vllm:iteration_tokens_total_bucket{le="256.0",model_name="Qwen/Qwen3-0.6B"} 1100.0
vllm:iteration_tokens_total_bucket{le="512.0",model_name="Qwen/Qwen3-0.6B"} 1100.0
vllm:iteration_tokens_total_bucket{le="1024.0",model_name="Qwen/Qwen3-0.6B"} 1100.0
vllm:iteration_tokens_total_bucket{le="2048.0",model_name="Qwen/Qwen3-0.6B"} 1100.0
```

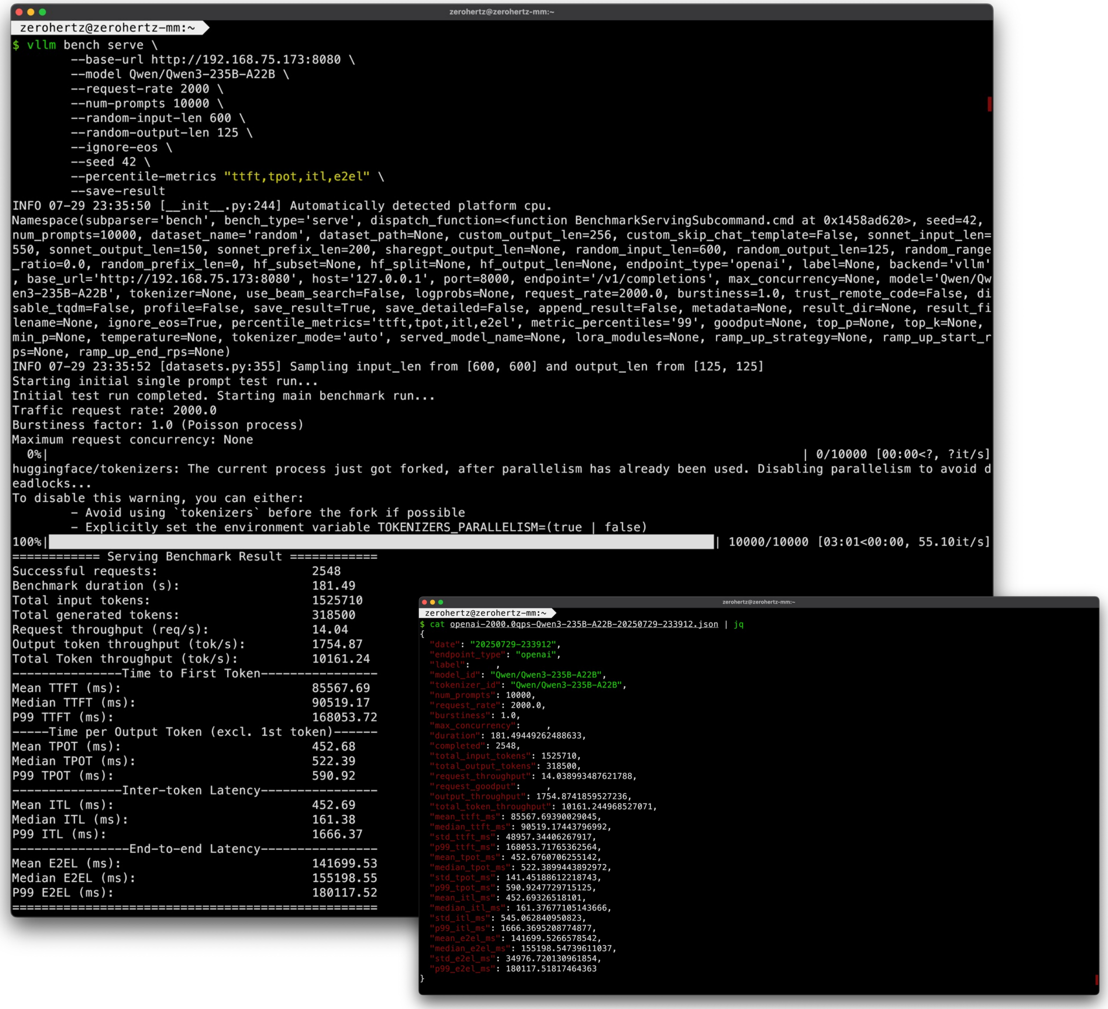
- Grafana와 연결하여 dashboard 구성 가능



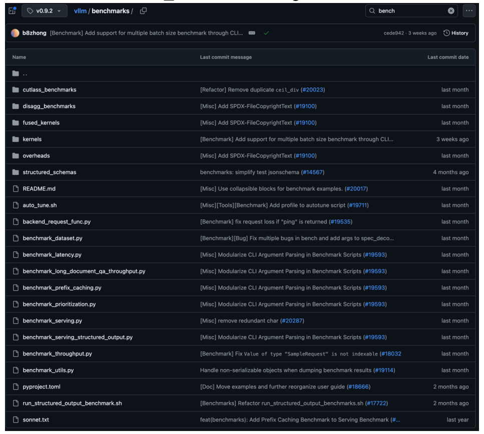
[72] https://docs.vllm.ai/en/v0.9.2/usage/metrics.html
[73] https://docs.vllm.ai/en/v0.9.2/examples/online_serving/prometheus_grafana.html

Benchmark

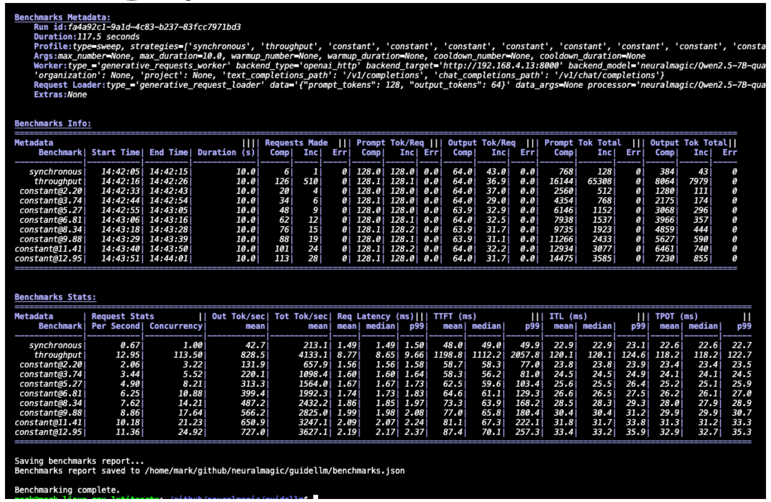
❑ “vllm bench” CLI 명령을 통해 간단한 benchmark 가능 [74]



❑ vLLM repository 내 benchmarks 내 코드 사용 [75, 76]



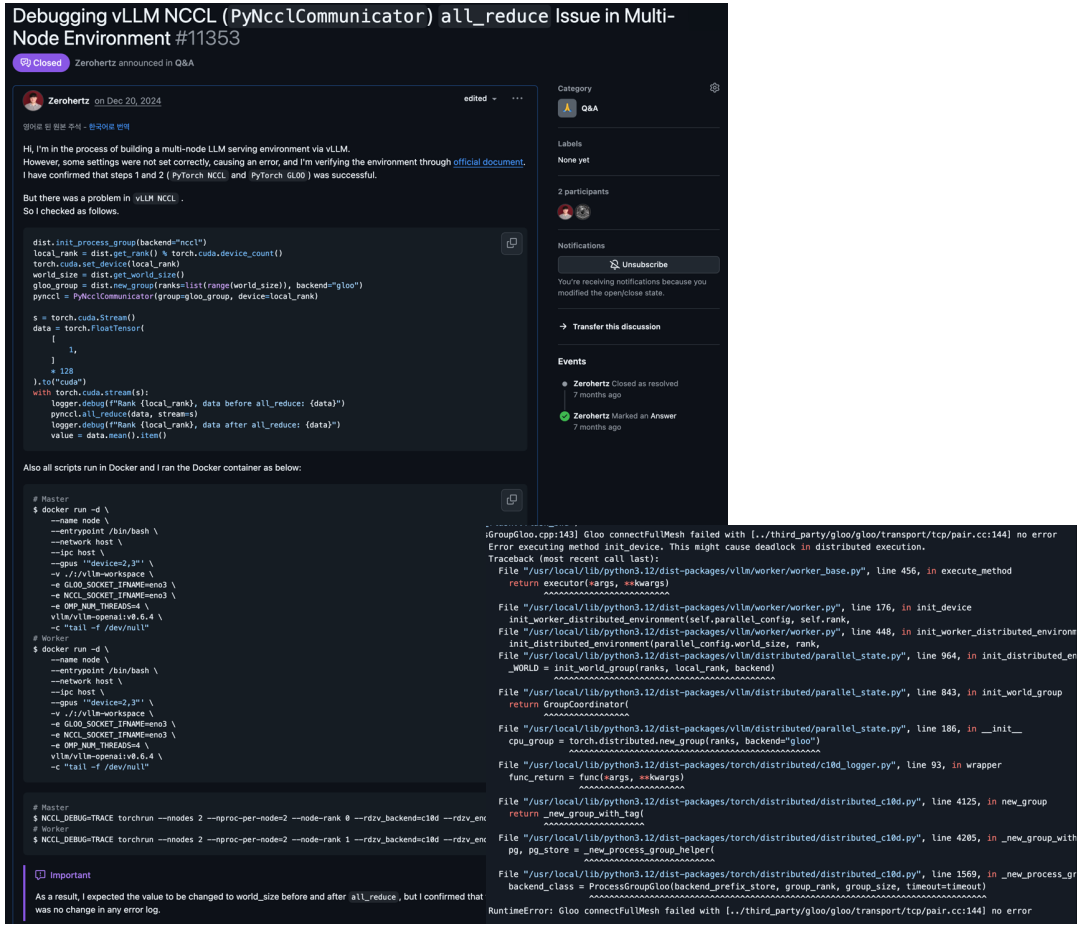
❑ vLLM project인 Guidellm 사용 [77, 78]



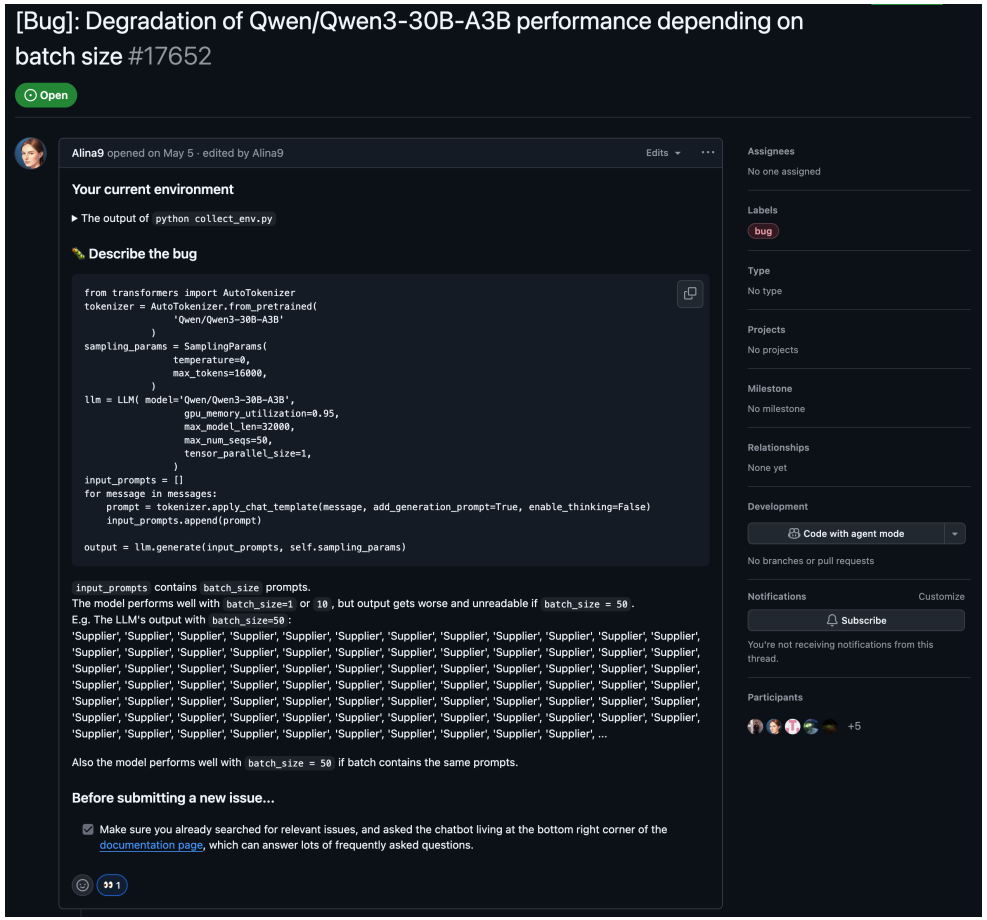
[74] https://docs.vllm.ai/en/v0.9.2/cli/index.html#bench
[75] https://docs.vllm.ai/en/v0.9.2/contributing/profiling.html
[76] https://github.com/vllm-project/vllm/tree/v0.9.2/benchmarks
[77] https://github.com/vllm-project/guidellm
[78] https://arxiv.org/pdf/2502.06494

Production Issues

- Gloo connectFullMesh failed with… [79]
  - Multi node로 process 연결 시 GLOO 연결 실패
    - → "GLOO_SOCKET_IFNAME" 환경 변수를 사용할 interface 값으로 지정하여 해결



- Qwen 계열 model의 무한 token 생성 [80, 81]
  - FlashInfer kernel 내 cascade inference 사용으로 인해 발생
    - → "--disable-cascade-attn"를 사용하여 해결



[79] https://github.com/vllm-project/vllm/discussions/11353
[80] https://github.com/vllm-project/vllm/issues/17652#issuecomment-2867891239
[81] https://flashinfer.ai/2024/02/02/cascade-inference.html

PART 5

Wrap-up

Roadmap Q3 2025[82]

- ❑ V1 Engine
  - ▪ V0 Engine 완전 제거
  - ▪ Core scheduler 최적화 및 확장
  - ▪ Async scheduling, multi-modal 처리 등 기능 구현
- ❑ Large Scale Serving
  - ▪ Mixture-of-Experts (MoE) model의 안정적 scale-out serving
  - ▪ 분산 서빙 표준화 및 autoscaling
- ❑ Models
  - ▪ 다양한 framework (training, authoring)의 tokenizer, configuration, processor 지원
  - ▪ Sparse attention mechanism
  - ▪ 1B 이하의 소형 모델 성능 향상
- ❑ Use Cases
  - ▪ RLHF
    - → 동기화 및 resharding을 위한 가중치 로딩 최적화
    - → Multi-turn scheduling
  - ▪ Evaluation
    - → Batching order에 영향받지 않는 full determinism 지원 (with/without prefix cache)
  - ▪ Batch Inference
    - → Prefix caching과 함께 scale-out을 위한 data parallel router
    - → CPU KV cache offloading

[82] https://github.com/vllm-project/vllm/issues/20336

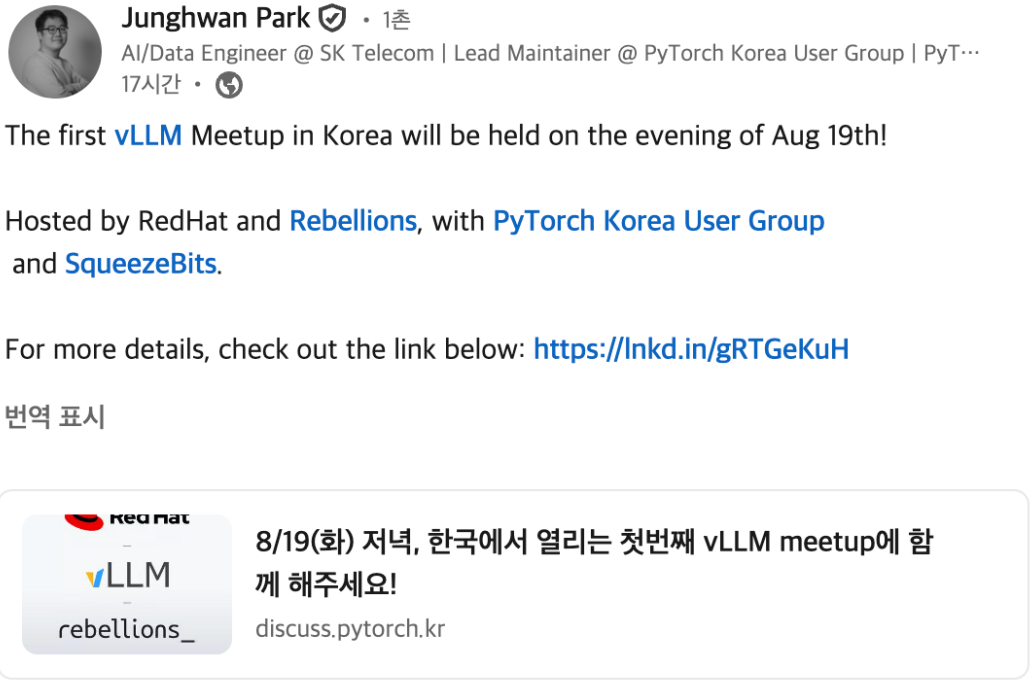
Conclusion

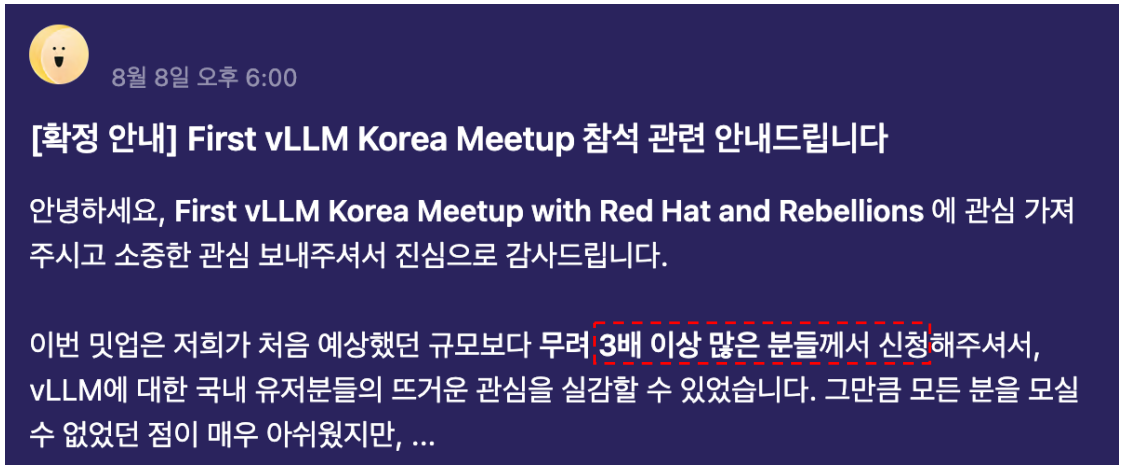
❑ OpenAI-Compatible Server
- LangChain, Gemini CLI 등 기존 생태계와의 호환성 유지
- Tool Calling, Reasoning 등 확장된 기능 활용 가능

❑ Architecture
- KV cache와 PagedAttention 기반의 효율적 메모리 관리
- V1 Engine 도입으로 단순한 고성능 구조 구현

❑ Production Deployment
- TP/PP/DP/EP 등 다양한 병렬 처리 전략 적용
- Kubernetes 기반 multi-node 분산 추론
- LoRA adapter를 활용한 사용자 맞춤형 경량화 serving
- Prometheus + Grafana를 활용한 observability 확보
- vllm bench 및 benchmark script를 통한 성능 평가

vLLM Meetup in Korea [83, 84]





[83] https://discuss.pytorch.kr/t/8-19-vllm-meetup/7401
[84] https://lu.ma/cgcgprmh

EoD

GitHub

LinkedIn

Coffee Chat