# vLLM: Easy, Fast, and Cheap LLM Serving for Everyone

vLLM Team

# vLLM 's Goal

Build the fastest and

easiest-to-use open-source

LLM inference & serving engine

# Why vLLM For Performance?

vLLM implements the key optimizations for **fast inference**

**Inference Optimizations**

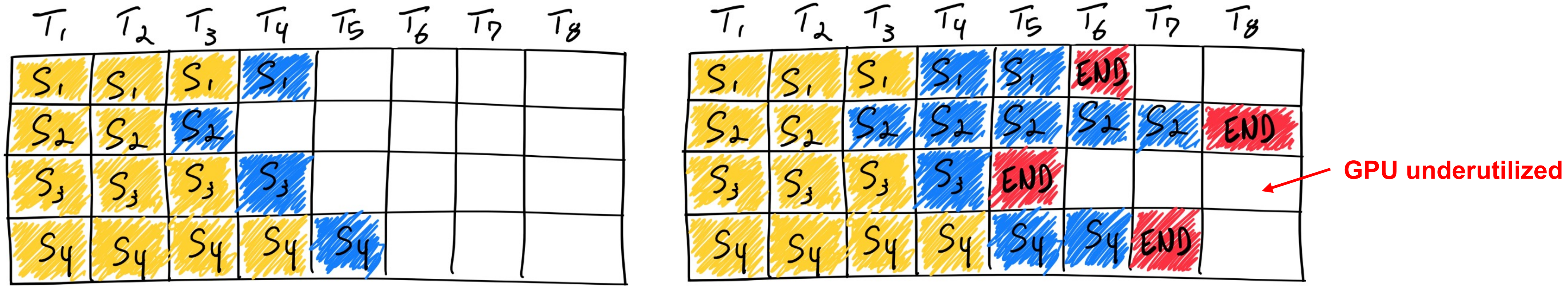
To make your models faster

**Distributed Parallel Inference**

To deploy large models efficiently

# Batching

## ❌Naive batching

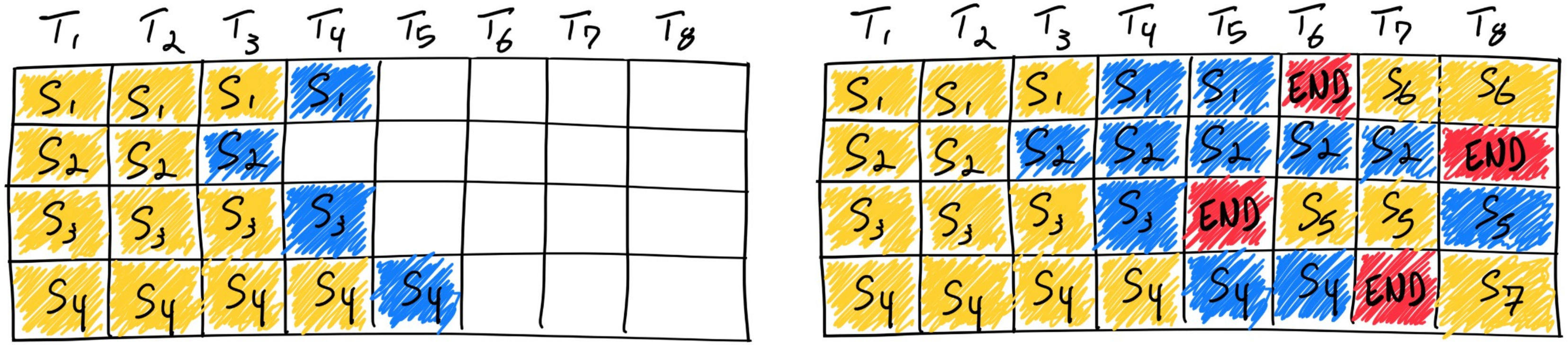


| $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ | $T_8$ |
|---|---|---|---|---|---|---|---|
| $S_1$ | $S_1$ | $S_1$ | $S_1$ | | | | |
| $S_2$ | $S_2$ | $S_2$ | | | | | |
| $S_3$ | $S_3$ | $S_3$ | $S_3$ | | | | |
| $S_4$ | $S_4$ | $S_4$ | $S_4$ | $S_4$ | | | |

| $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ | $T_8$ |
|---|---|---|---|---|---|---|---|
| $S_1$ | $S_1$ | $S_1$ | $S_1$ | $S_1$ | END | | |
| $S_2$ | $S_2$ | $S_2$ | $S_2$ | $S_2$ | $S_2$ | $S_2$ | END |
| $S_3$ | $S_3$ | $S_3$ | $S_3$ | END | | | |
| $S_4$ | $S_4$ | $S_4$ | $S_4$ | $S_4$ | $S_4$ | END | |

GPU underutilized

## ✅Continuous Batching

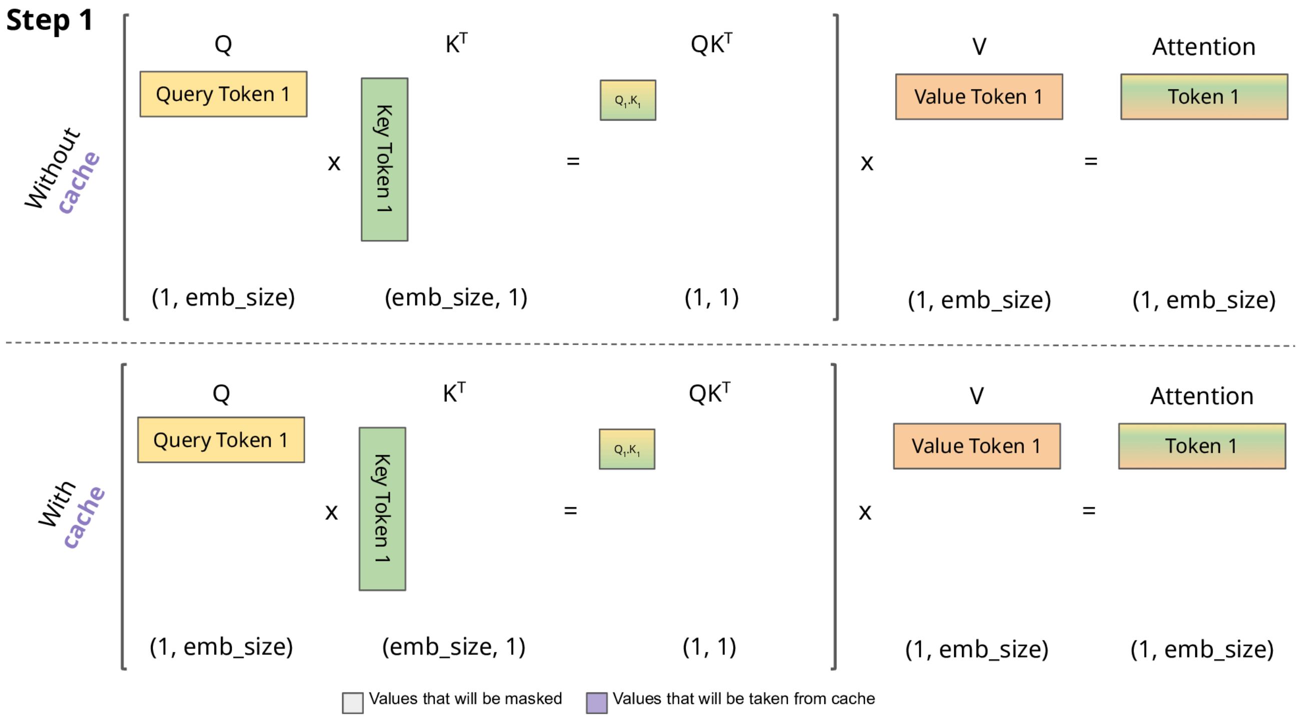


| $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ | $T_8$ |
|---|---|---|---|---|---|---|---|
| $S_1$ | $S_1$ | $S_1$ | $S_1$ | | | | |
| $S_2$ | $S_2$ | $S_2$ | | | | | |
| $S_3$ | $S_3$ | $S_3$ | $S_3$ | | | | |
| $S_4$ | $S_4$ | $S_4$ | $S_4$ | $S_4$ | | | |

| $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ | $T_8$ |
|---|---|---|---|---|---|---|---|
| $S_1$ | $S_1$ | $S_1$ | $S_1$ | $S_1$ | END | $S_6$ | $S_6$ |
| $S_2$ | $S_2$ | $S_2$ | $S_2$ | $S_2$ | $S_2$ | $S_2$ | END |
| $S_3$ | $S_3$ | $S_3$ | $S_3$ | END | $S_5$ | $S_5$ | $S_5$ |
| $S_4$ | $S_4$ | $S_4$ | $S_4$ | $S_4$ | $S_4$ | END | $S_7$ |

# KV Cache

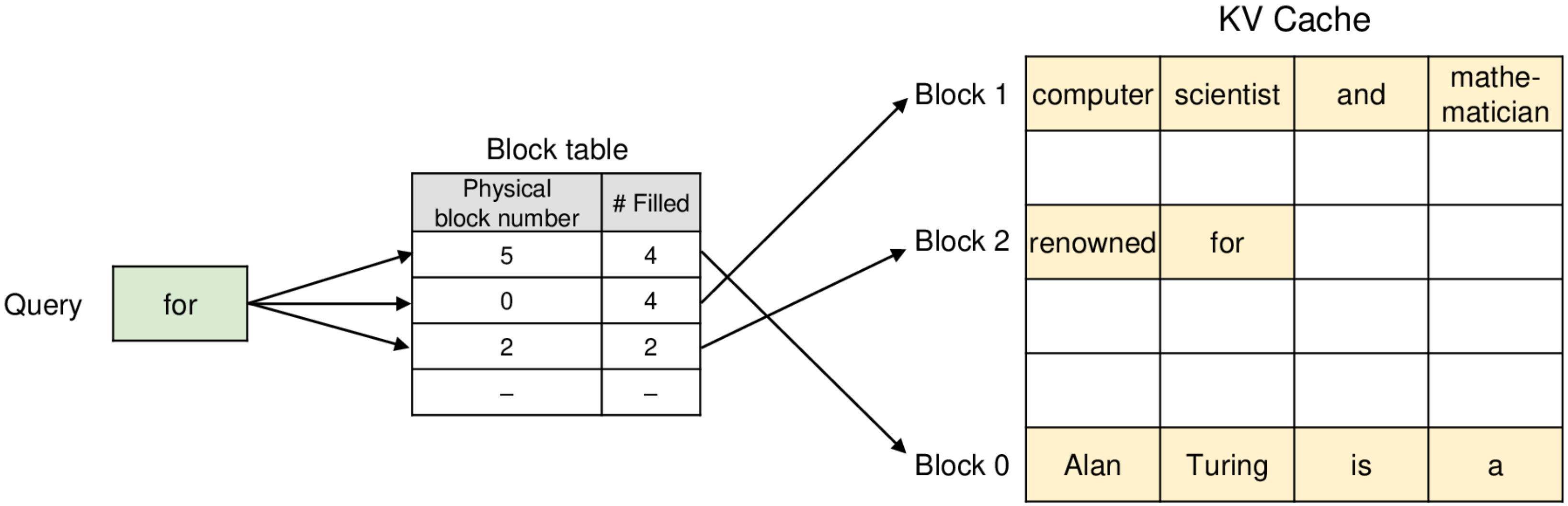
## Caching Key and Value vectors in self-attention saves redundant computation and accelerates decoding



https://medium.com/@joaolages/kv-caching-explained-276520203249

# PagedAttention

An attention algorithm that allows for storing continuous keys and values in non-contiguous memory space.



Block table

| Physical block number | # Filled |
|---|---|
| 5 | 4 |
| 0 | 4 |
| 2 | 2 |
| – | – |

KV Cache

| | | | | |
|---|---|---|---|---|
| Block 1 | computer | scientist | and | mathe-matician |
| | | | | |
| Block 2 | renowned | for | | |
| | | | | |
| | | | | |
| Block 0 | Alan | Turing | is | a |

Woosuk Kwon, etc. Efficient Memory Management for Large Language Model Serving with PagedAttention, SOSP 2023

# vLLM Combines All Optimizations Together

## Without Optimizations

Prompt

<SYSTEM> You are a helpful assistant. ...
Keep your answers precise and concise.
<USER> Generate a description for this item: ...

Output

## vLLM

Prompt

<SYSTEM> You are a helpful assistant. ...
Keep your answers precise and concise.
<USER> Generate a description for this item: ...

Output

# Forms of Parallelism in vLLM

**Tensor Parallelism (TP)**

**Pipeline Parallelism (PP)**

**Expert Parallelism (EP)**

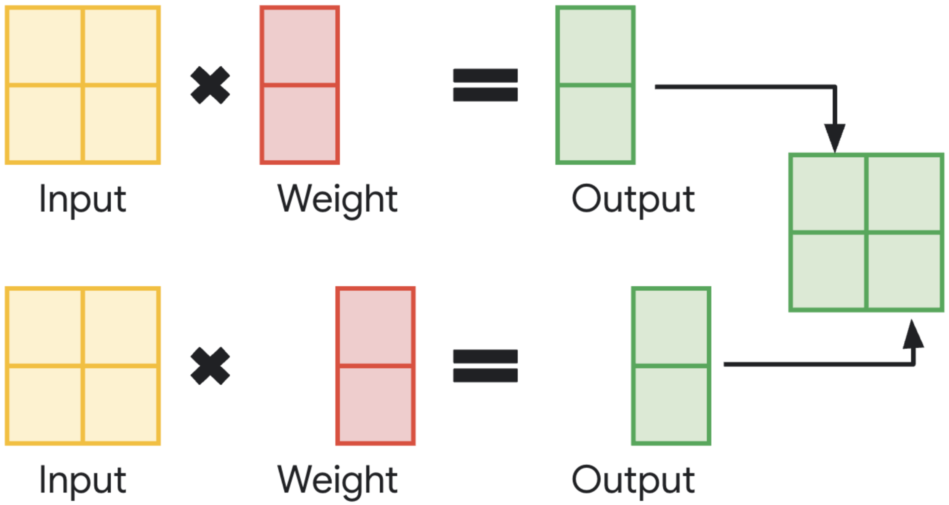
**Data Parallelism (DP)**

**Disaggregated Serving**

# Tensor Parallelism

Partition the model’s hidden dimension → All-reduce to aggregate the outputs

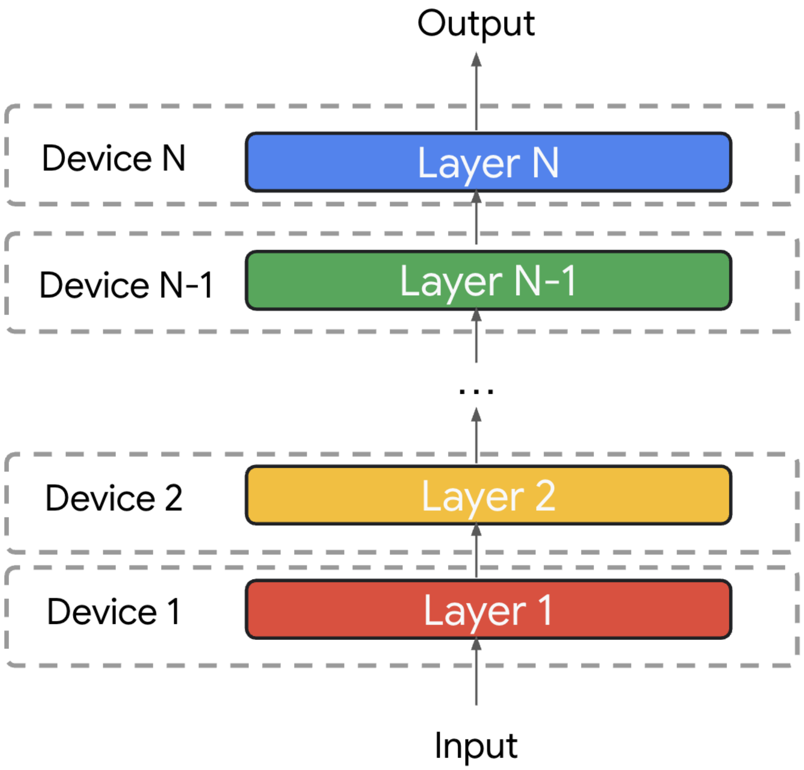
- Works well for ≤ 8 devices
- vLLM provides optimized all-reduce implementation
- Limited scalability
- Communication overhead can be critical for small model



# Pipeline Parallelism

Distribute layers to different devices → execute in a pipelined fashion

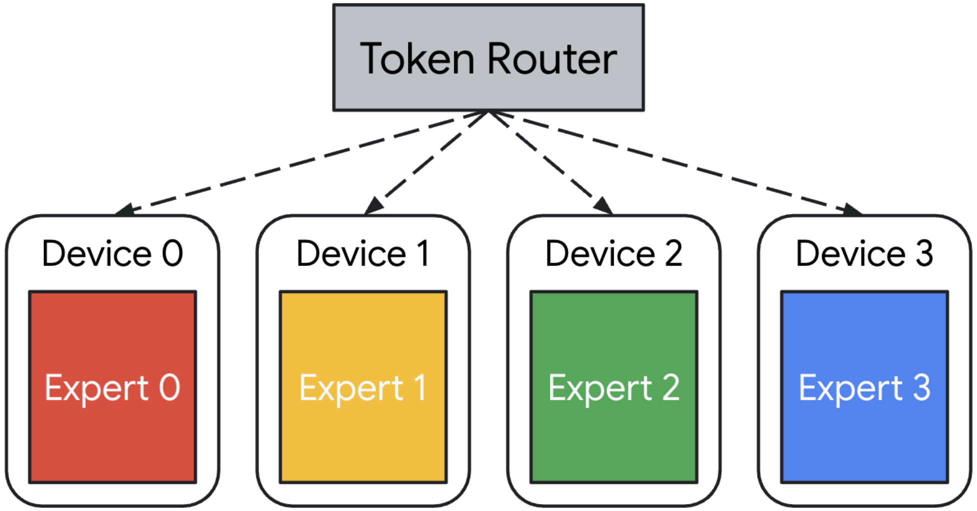
- Point-to-point communication instead of expensive all-reduce
- Load imbalance b/t stages
- Doesn’t reduce latency



# Expert Parallelism

Place experts to different devices
→ All-to-all to exchange tokens

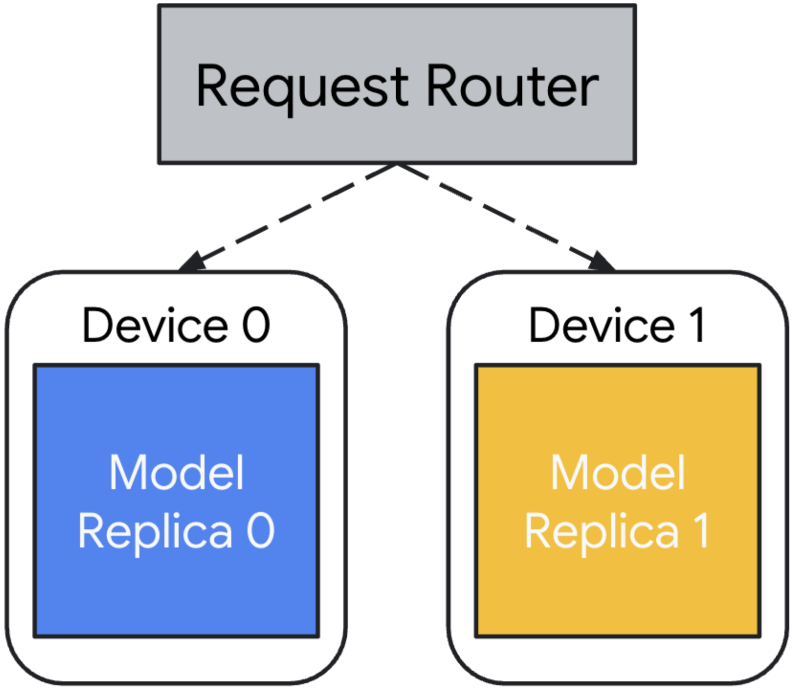
- Lower communication overheads than tensor parallelism
- Load imbalance between experts



# Data Parallelism

Partition the inputs instead of the model → Model weights are replicated

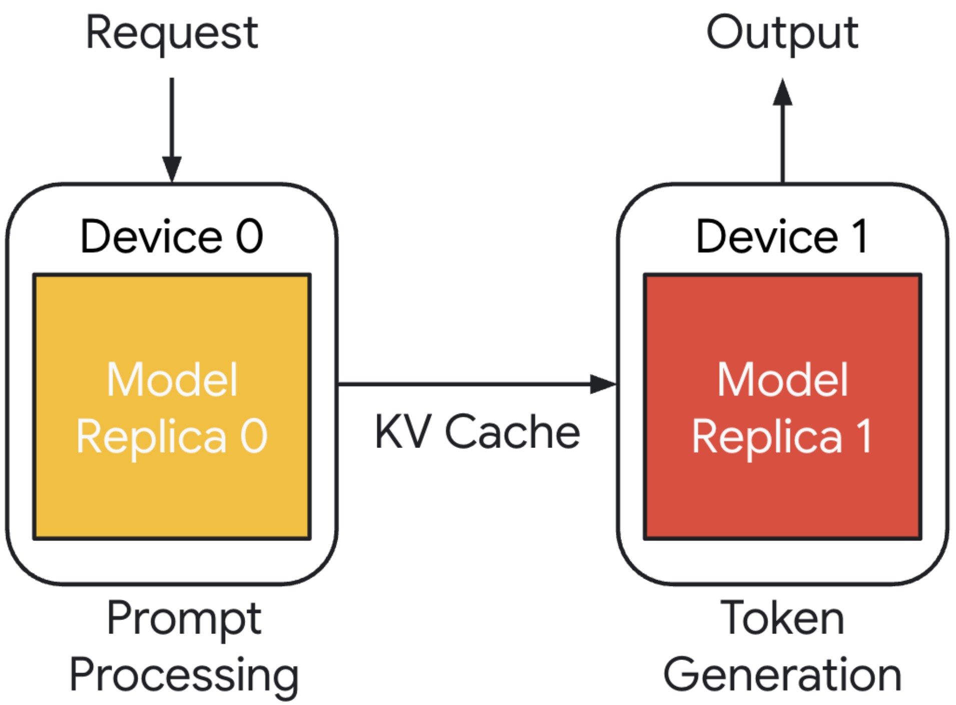
- Lower communication overheads
- Load imbalance between replicas
- Increased memory consumption for model weights



# Disaggregated Serving

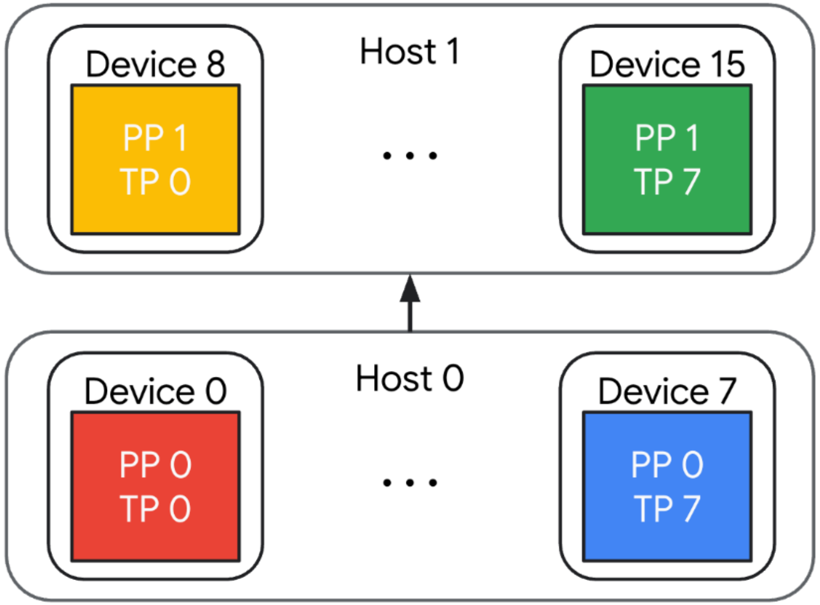
Partition the “time” dimension
→ Separate instances for prompt processing & token generation

- Separation of concern
- Better control over latency
- KV cache transfer overheads
- Lower device utilization



# vLLM Supports Mixed Parallelism

**Tensor + Pipeline Parallelism**
(e.g., Llama 3 405B)



Host 1

Device 8
PP 1
TP 0

. . .

Device 15
PP 1
TP 7

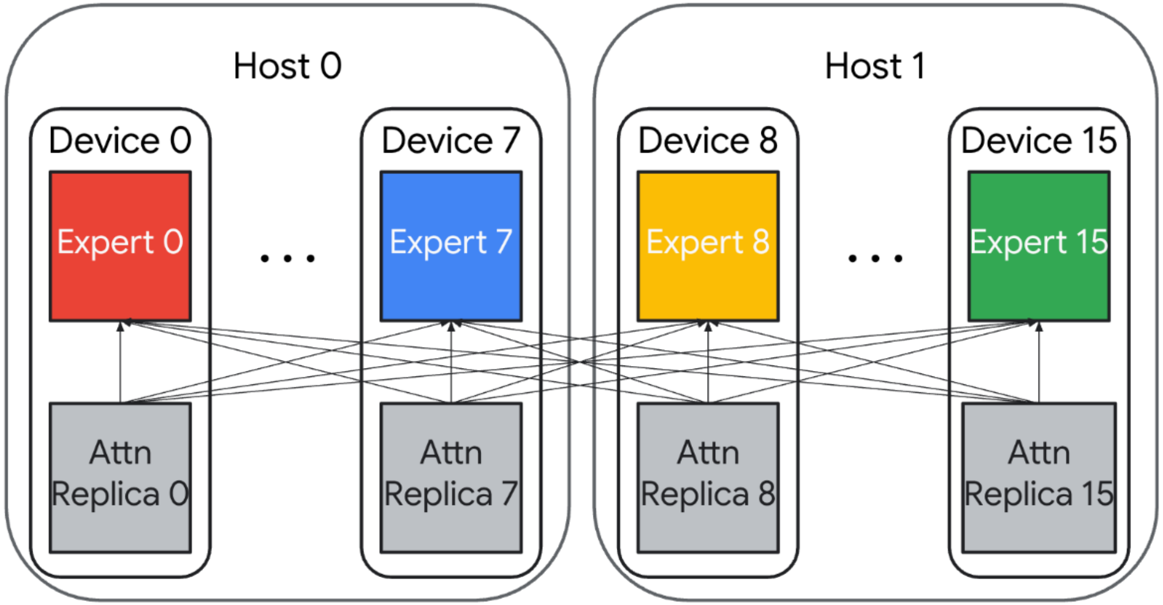
Host 0

Device 0
PP 0
TP 0

. . .

Device 7
PP 0
TP 7

**Data + Expert Parallelism**
(e.g., DeepSeek V3)



Host 0

Device 0
Expert 0
Attn Replica 0

. . .

Device 7
Expert 7
Attn Replica 7

Host 1

Device 8
Expert 8
Attn Replica 8

. . .

Device 15
Expert 15
Attn Replica 15

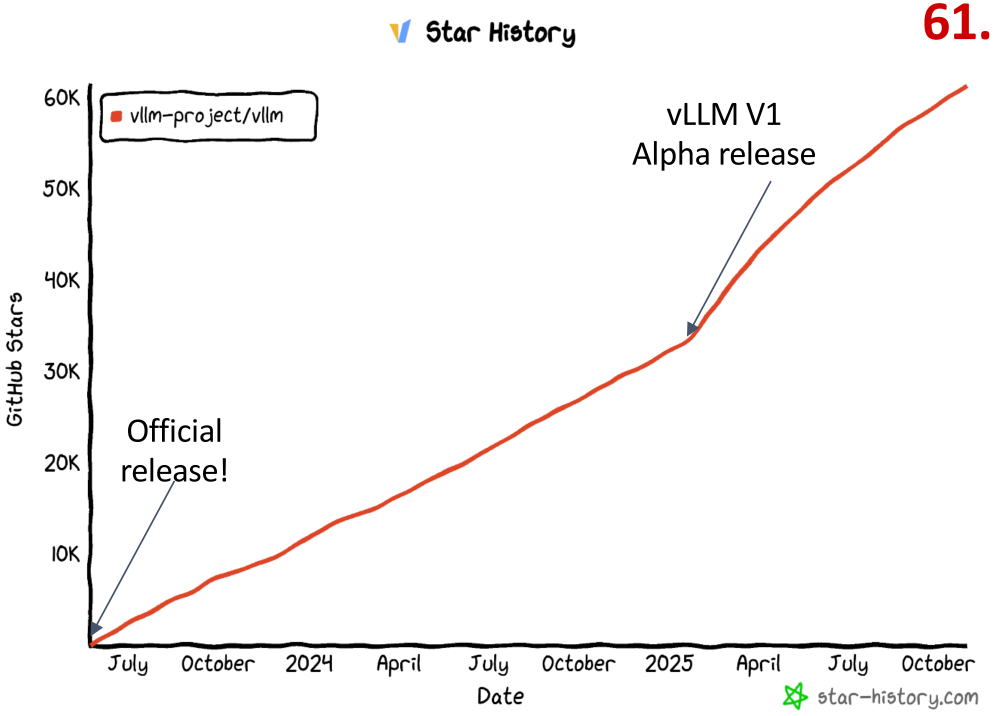
**How do I run model X on hardware Y with task Z?**

See our out-of-box recipes: **https://docs.vllm.ai/projects/recipes/en/latest/index.html**

```
$ pip install vllm
```

vLLM

61.3K Stars



vLLM V1 Alpha release

Official release!

# vLLM Adopters

## Open-Source Projects

LLaMA-Factory

Easy and Efficient LLM Fine-Tuning

LangChain

unsloth

verl

Dify

...

## Companies

### Hardware

NVIDIA

AMD

intel

Apple

...

### Cloud

aws

Microsoft Azure

Google Cloud

Alibaba Cloud

...

### Model Vendor

Qwen

KIMI

deepseek

Mistral AI

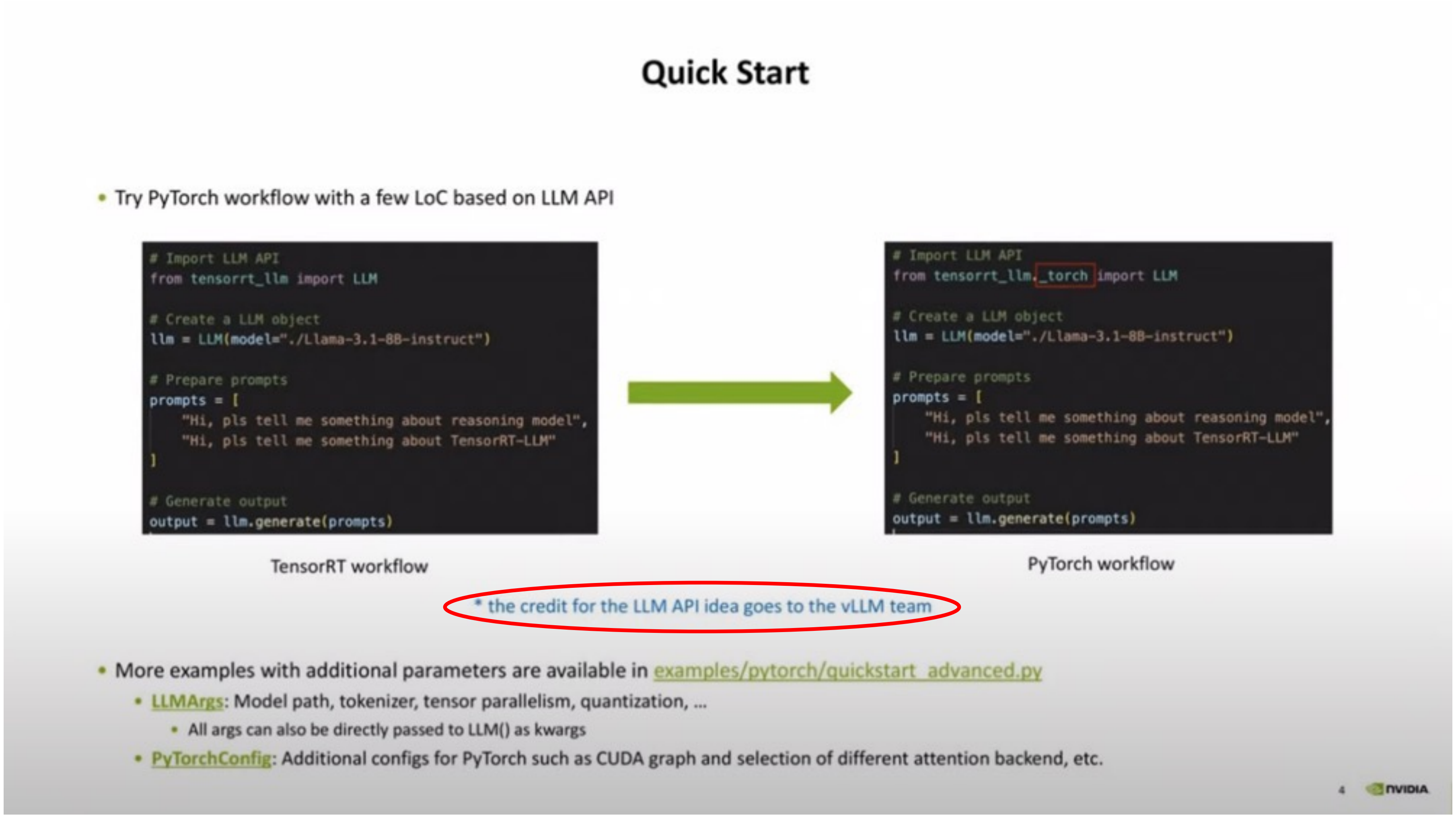
...

# vLLM API (1): LLM class

A Python interface for offline batched inference

```python
from vllm import LLM

# Example prompts.
prompts = ["Hello, my name is", "The capital of France is"]
# Create an LLM with HF model name.
llm = LLM(model="meta-llama/Meta-Llama-3.1-8B")
# Generate texts from the prompts.
outputs = llm.generate(prompts)
```

# vLLM API (1): LLM class



## Quick Start

- Try PyTorch workflow with a few LoC based on LLM API

```
# Import LLM API
from tensorrt_llm import LLM

# Create a LLM object
llm = LLM(model="./Llama-3.1-8B-instruct")

# Prepare prompts
prompts = [
    "Hi, pls tell me something about reasoning model",
    "Hi, pls tell me something about TensorRT-LLM"
]

# Generate output
output = llm.generate(prompts)
```

TensorRT workflow

```
# Import LLM API
from tensorrt_llm._torch import LLM

# Create a LLM object
llm = LLM(model="./Llama-3.1-8B-instruct")

# Prepare prompts
prompts = [
    "Hi, pls tell me something about reasoning model",
    "Hi, pls tell me something about TensorRT-LLM"
]

# Generate output
output = llm.generate(prompts)
```

PyTorch workflow

* the credit for the LLM API idea goes to the vLLM team

- More examples with additional parameters are available in examples/pytorch/quickstart_advanced.py
  - LLMArgs: Model path, tokenizer, tensor parallelism, quantization, …
    - All args can also be directly passed to LLM() as kwargs
  - PyTorchConfig: Additional configs for PyTorch such as CUDA graph and selection of different attention backend, etc.

# vLLM API (2): OpenAI-compatible server

A FastAPI-based server for online serving

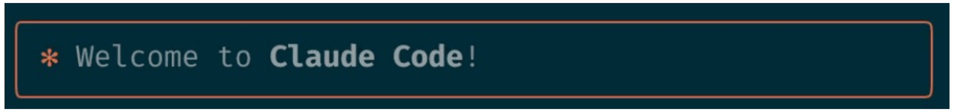
Server

```
$ vllm serve meta-llama/Meta-Llama-3.1-8B
```

Client

```
$ curl http://localhost:8000/v1/completions \
    -H "Content-Type: application/json" \
    -d '{
        "model": "meta-llama/Meta-Llama-3.1-8B",
        "prompt": "San Francisco is a",
        "max_tokens": 7,
        "temperature": 0
    }'
```

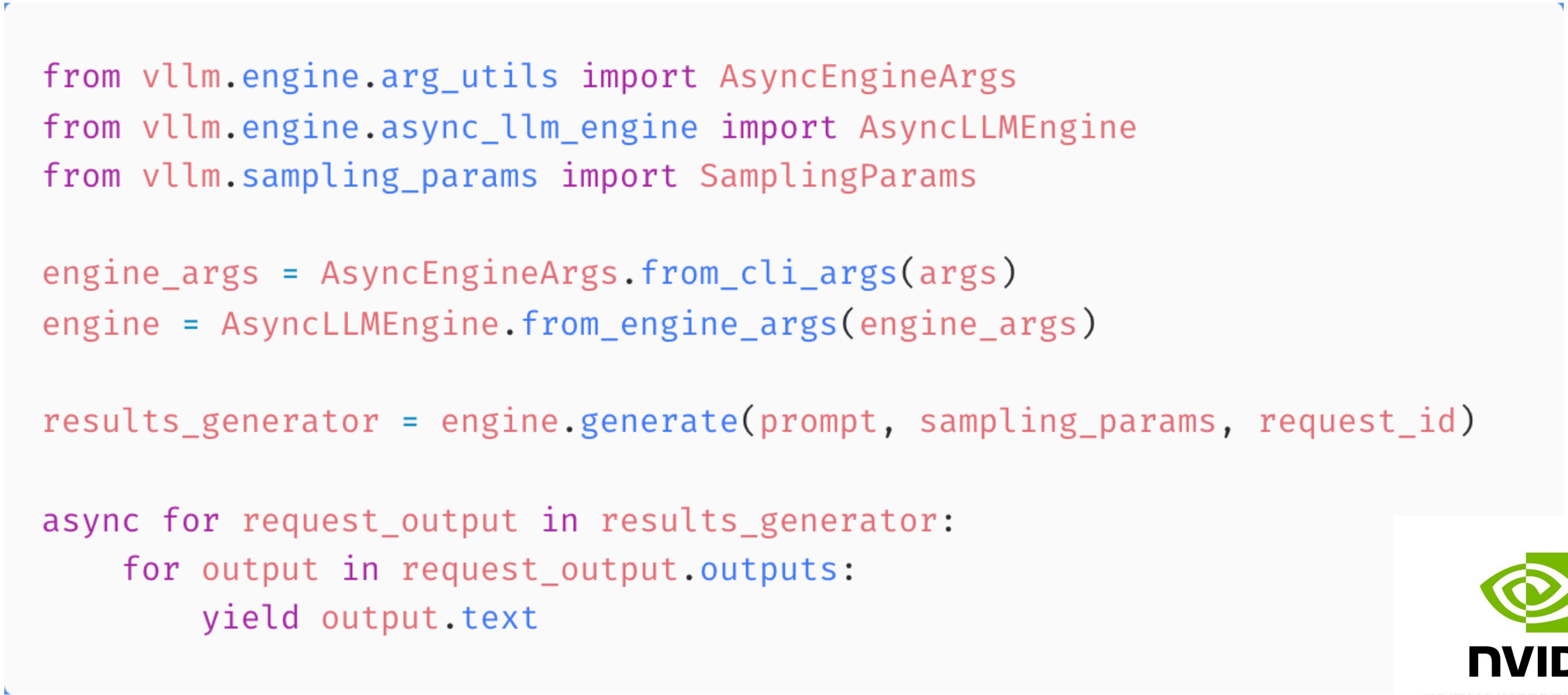
MinerU



```
* Welcome to Claude Code!
```

# vLLM API (3): Embeddable LLMEngine

A Python library with the full power of vLLM in your framework



```
from vllm.engine.arg_utils import AsyncEngineArgs
from vllm.engine.async_llm_engine import AsyncLLMEngine
from vllm.sampling_params import SamplingParams

engine_args = AsyncEngineArgs.from_cli_args(args)
engine = AsyncLLMEngine.from_engine_args(engine_args)

results_generator = engine.generate(prompt, sampling_params, request_id)

async for request_output in results_generator:
    for output in request_output.outputs:
        yield output.text
```

# Broad Model Support

vLLM supports almost all popular **text-generation models**!

StepFun Step-3

KIMI Kimi-K2

Deepseek-3.2

ERNIE

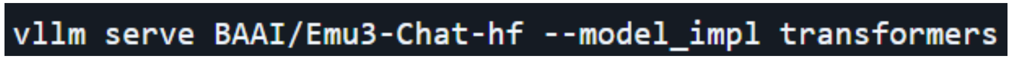
Tencent Hunyuan

GLM-4.6

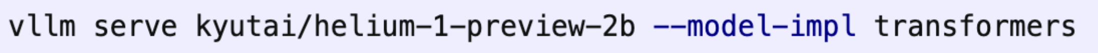
# Transformers backend

## Supports models not in vLLM, but runnable by Transformers

https://blog.vllm.ai/2025/04/11/transformers-backend.html



```
vllm serve BAAI/Emu3-Chat-hf --model_impl transformers
```



```
vllm serve kyutai/helium-1-preview-2b --model-impl transformers
```

- Write model implementation in Transformers
- Leverage vLLM’s kv cache management and continuous batching
- Work for both vision and language models
- Performance on par with pure vLLM implementation

For model vendors who develop new models, it is also recommended to use https://docs.vllm.ai/en/latest/contributing/model/registration.html#out-of-tree-models

# Broad Model Support

## Architecture

- Transformer-like LLMs
  - Llama
- Mixture-of-Expert LLMs
  - DeepSeekV3
- State-Space Models
  - Jamba
- Linear-Attention Models
  - Minimax-M2
- ……

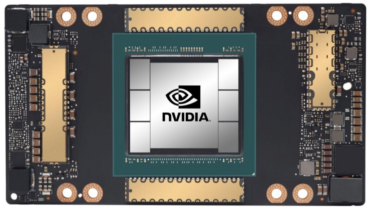
## Modality

- Text-input LLMs
  - Llama
- Image-text input LLMs
  - Deepseek-OCR
- Video-input LLMs
  - Qwen3-VL
- Audio-input LLMs
  - Whisper
- ……

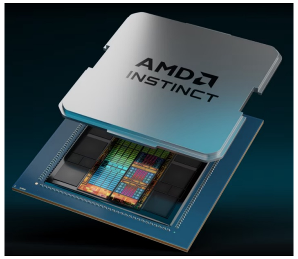
## Task

- Chat/Completion
  - Llama
- Embedding
  - E5-Mistral
- Reward
  - Qwen2.5-Math-RM
- Rerank
  - Qwen3-Reranker
- ……

# Diverse Hardware Support



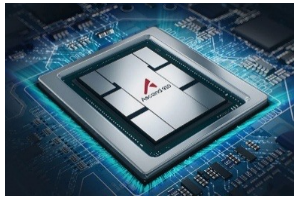
NVIDIA GPU



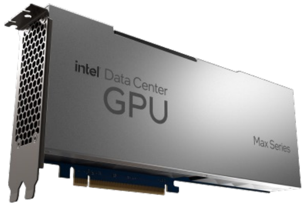
AMD GPU

x86 CPU

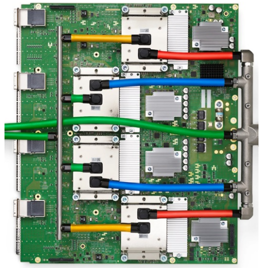
AWS Neuron (WIP)



Huawei Ascend

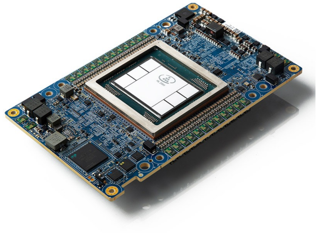


Intel GPU



Google TPU

ARM CPU

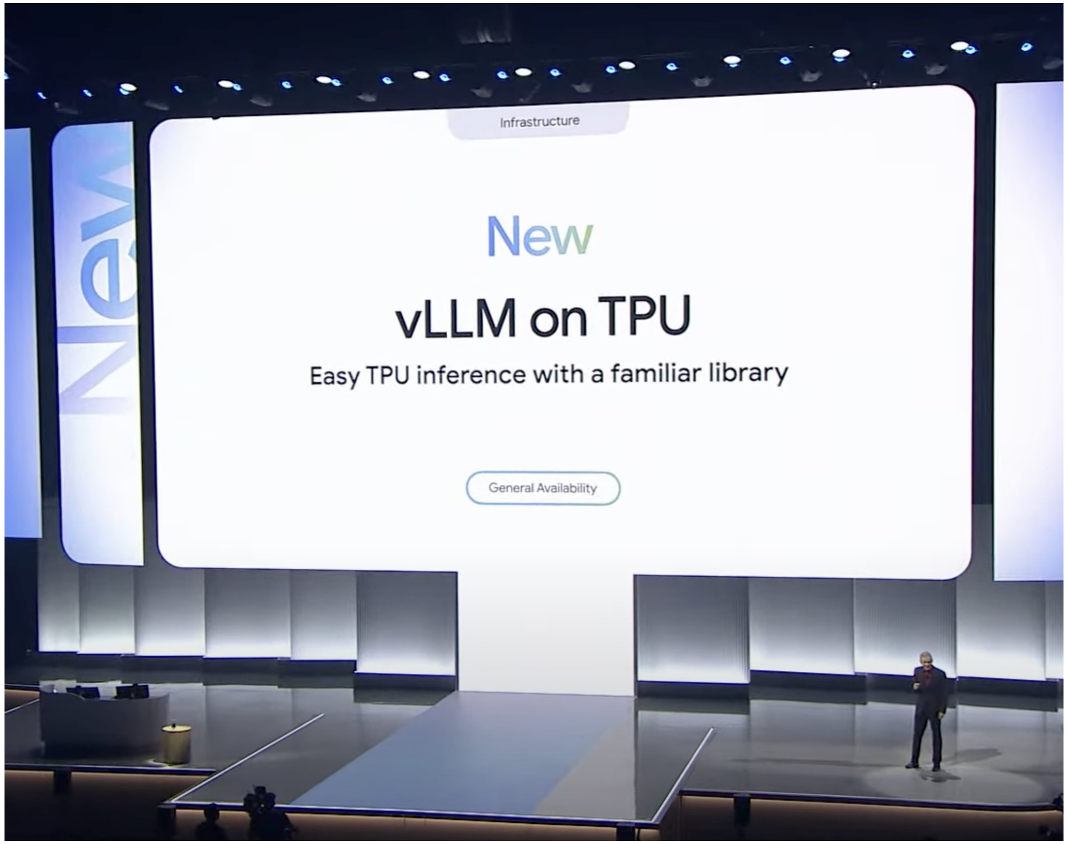


Intel Gaudi

IBM Spyre

support in vllm repo

hardware plugin ⭐



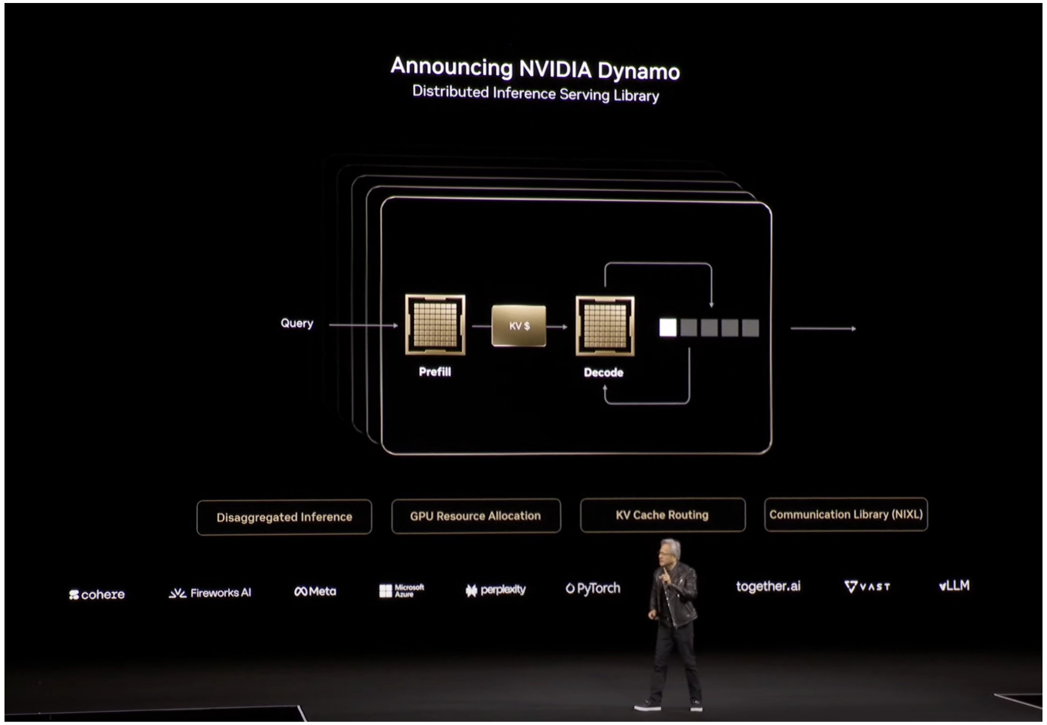
Infrastructure

New

vLLM on TPU

Easy TPU inference with a familiar library

General Availability



Announcing NVIDIA Dynamo

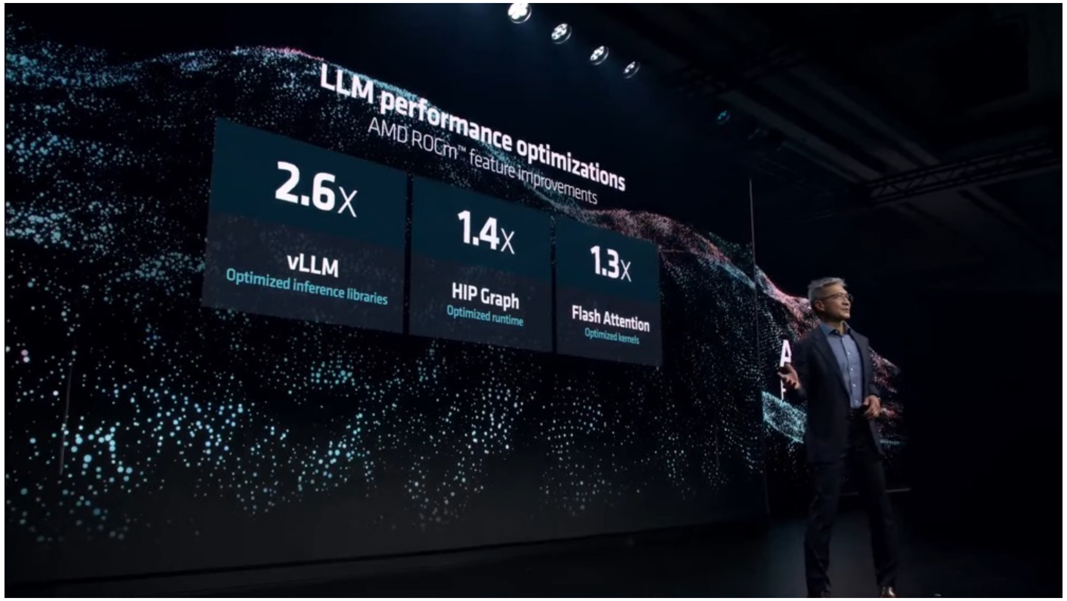
Distributed Inference Serving Library

Disaggregated Inference

GPU Resource Allocation

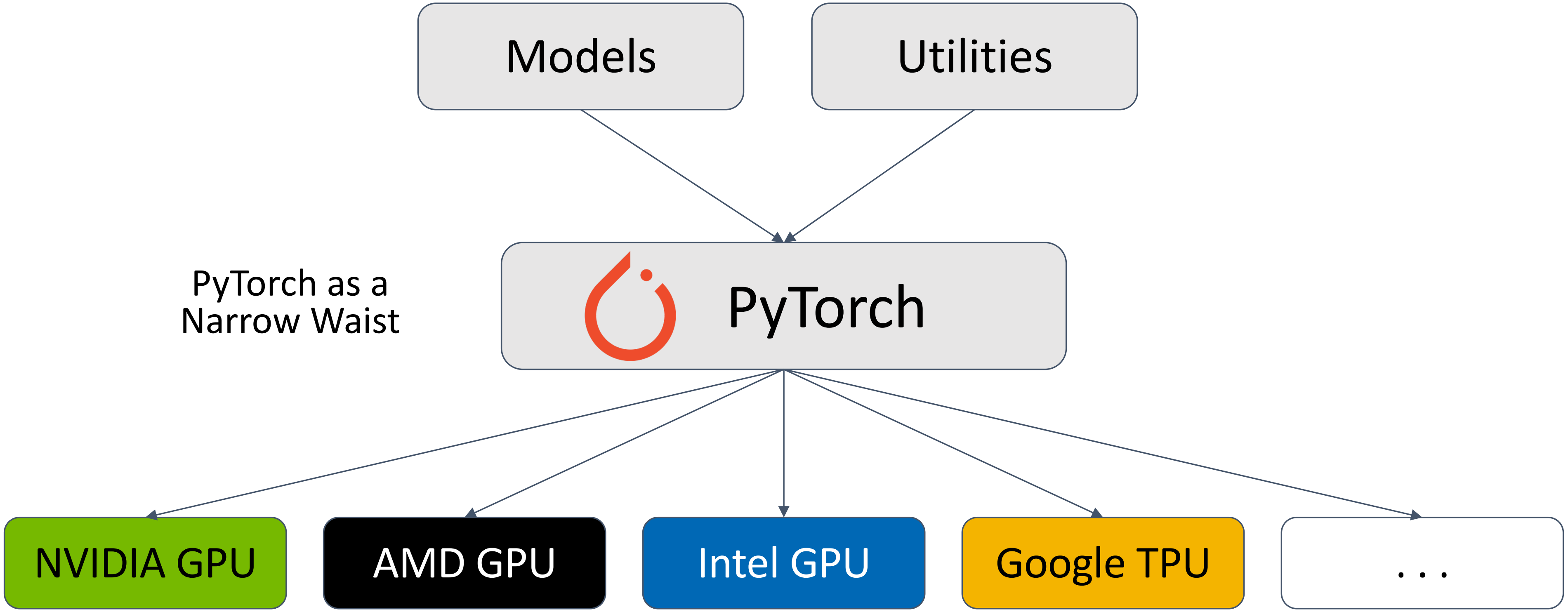
KV Cache Routing

Communication Library (NIXL)


LLM performance optimizations
AMD ROCm™ feature improvements
2.6x
vLLM
Optimized inference libraries
1.4x
HIP Graph
Optimized runtime
1.3x
Flash Attention
Optimized kernels

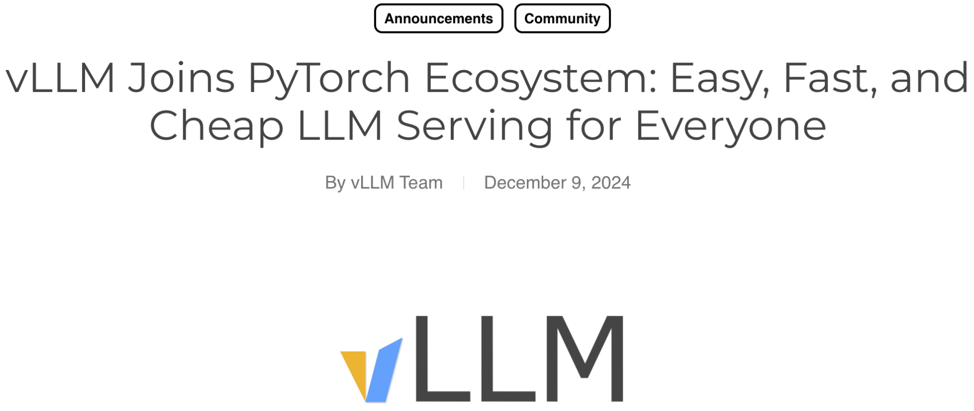
kubernetes
llm-d
vLLM

# Diverse Hardware Support



PyTorch as a
Narrow Waist

# Close Collaboration with PyTorch



Announcements Community

vLLM Joins PyTorch Ecosystem: Easy, Fast, and Cheap LLM Serving for Everyone

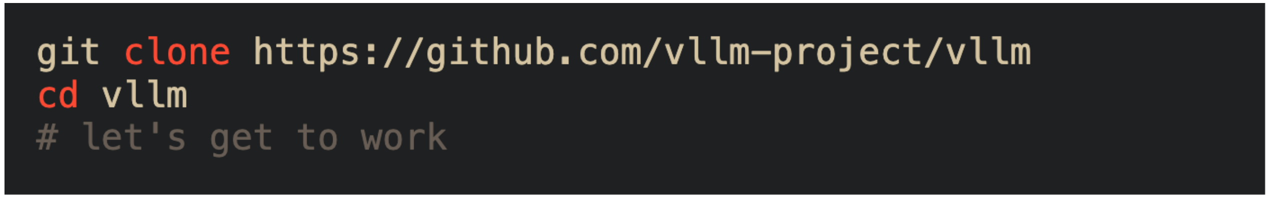
By vLLM Team | December 9, 2024

vLLM has joined PyTorch Foundation!

# vLLM: Building, Testing and Contributing

# So how does I get started with vLLM?

- Source: github.com/vllm-project/vllm



```
git clone https://github.com/vllm-project/vllm
cd vllm
# let's get to work
```

- Docs: docs.vllm.ai/en/latest Yes! Docs are useful!

- Issue tracker is a good source of information vllm-project/vllm/issues

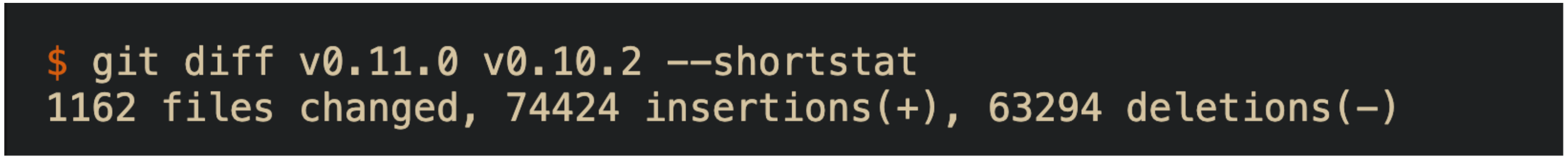
# Some vLLM facts

- Current vllm version v0.11.0 (released Oct 10th), with v0.11.1 brewing (rc1 available)

- One release every three weeks

- 1716 (!) contributors as of today, 207 in the last release (65 new!). This is a 200+ contributor increase from just a couple months ago!

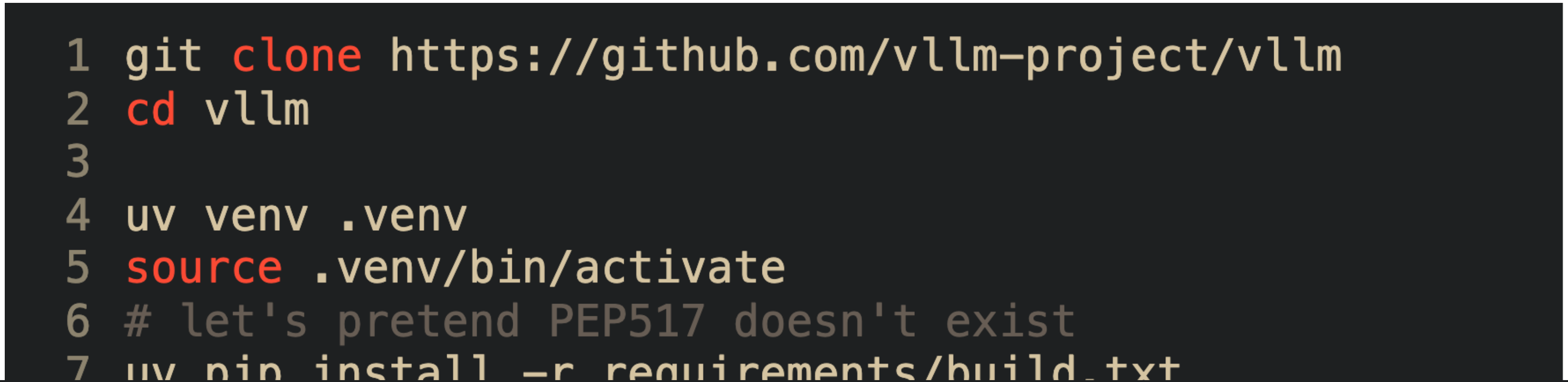
- A lot of code is being pushed out each release. For the last release:



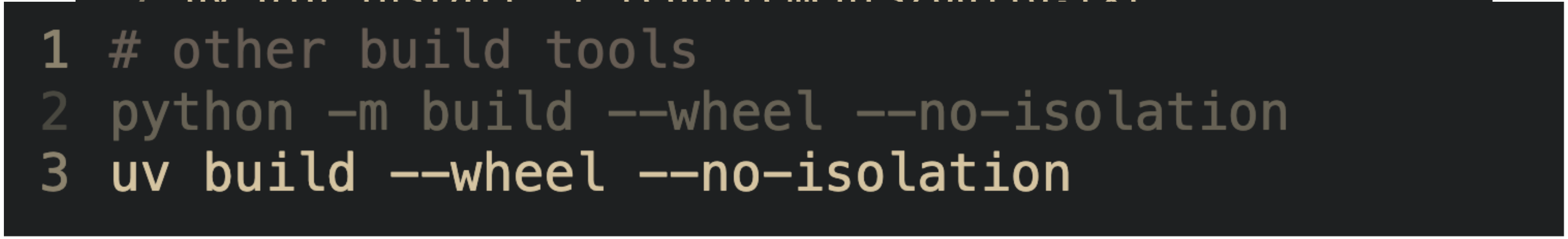
```
$ git diff v0.11.0 v0.10.2 --shortstat
1162 files changed, 74424 insertions(+), 63294 deletions(-)
```

# BUILDING

Here's how easy it is to build vLLM using uv



```
git clone https://github.com/vllm-project/vllm
cd vllm

uv venv .venv
source .venv/bin/activate
# let's pretend PEP517 doesn't exist
uv pip install -r requirements/build.txt
```



```
# other build tools
python -m build --wheel --no-isolation
uv build --wheel --no-isolation
```

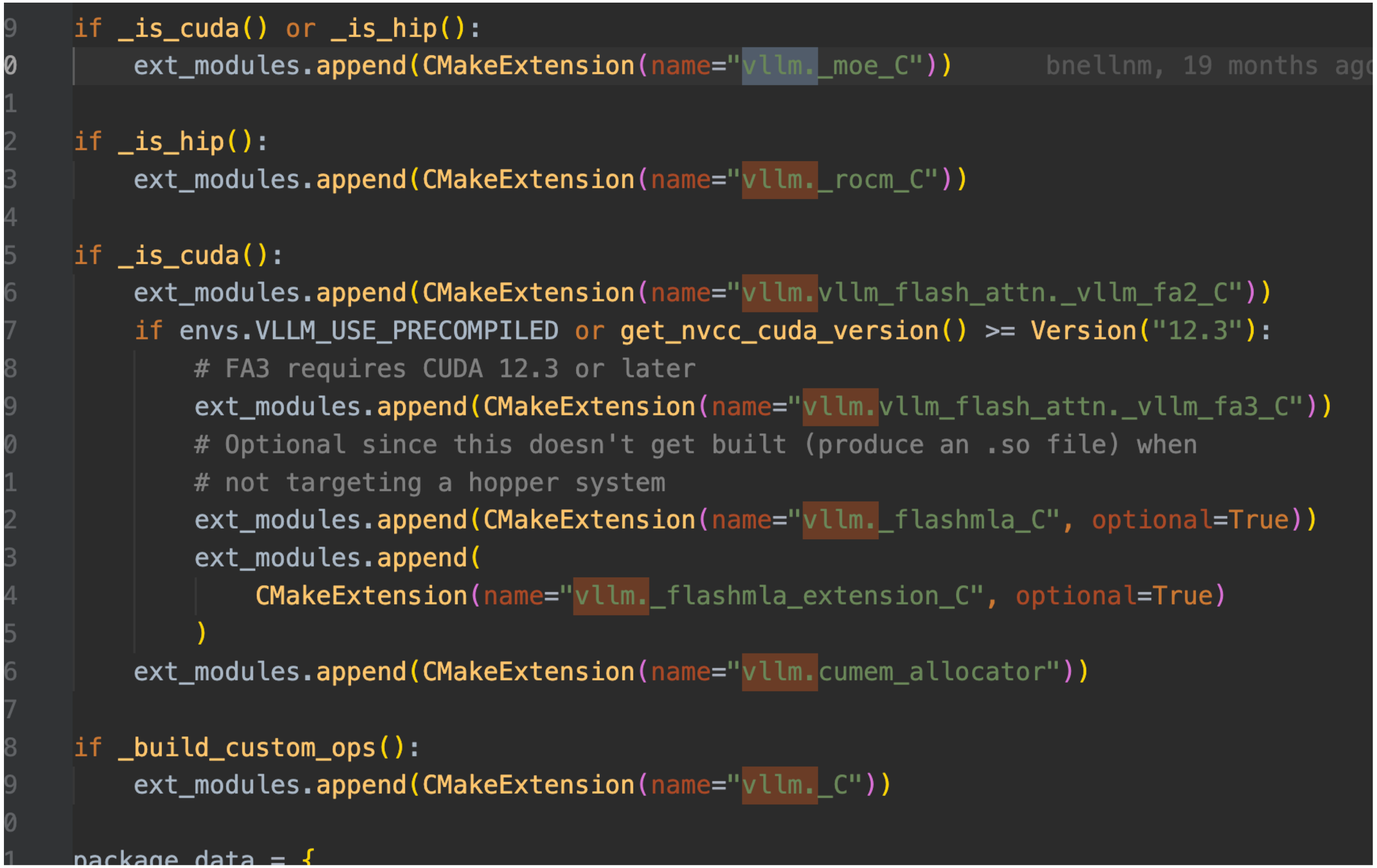
# What's not simple?

- There are several available targets: Nvidia CUDA, AMD ROCm, Google TPU, Intel HPU, CPU, ...

- Different toolchains required, depending on target
- Hardware requirements

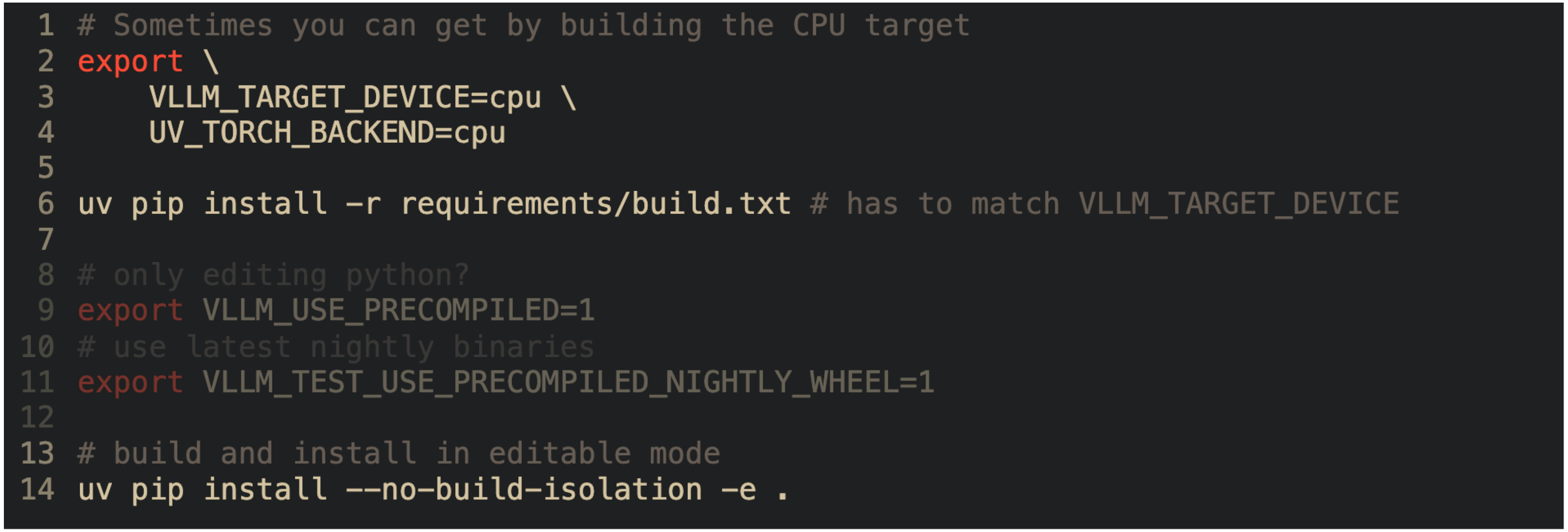
- Build can be very hungry for memory and CPU (several GBs of memory required per process)

# vLLM extensions



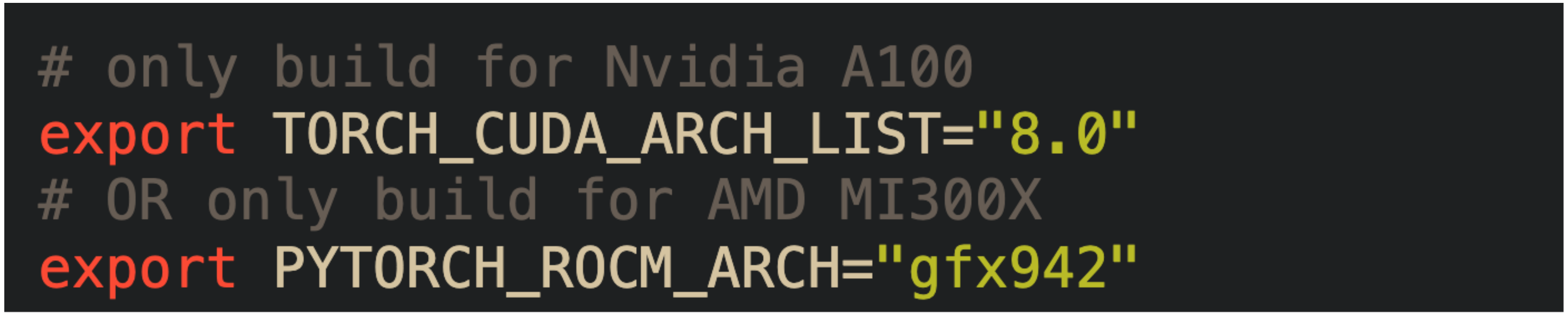
```
if _is_cuda() or _is_hip():
    ext_modules.append(CMakeExtension(name="vllm._moe_C"))      bnellnm, 19 months ago

if _is_hip():
    ext_modules.append(CMakeExtension(name="vllm._rocm_C"))

if _is_cuda():
    ext_modules.append(CMakeExtension(name="vllm.vllm_flash_attn._vllm_fa2_C"))
    if envs.VLLM_USE_PRECOMPILED or get_nvcc_cuda_version() >= Version("12.3"):
        # FA3 requires CUDA 12.3 or later
        ext_modules.append(CMakeExtension(name="vllm.vllm_flash_attn._vllm_fa3_C"))
        # Optional since this doesn't get built (produce an .so file) when
        # not targeting a hopper system
        ext_modules.append(CMakeExtension(name="vllm._flashmla_C", optional=True))
        ext_modules.append(
            CMakeExtension(name="vllm._flashmla_extension_C", optional=True)
        )
    ext_modules.append(CMakeExtension(name="vllm.cumem_allocator"))

if _build_custom_ops():
    ext_modules.append(CMakeExtension(name="vllm._C"))

package_data = {
```

# Local Development

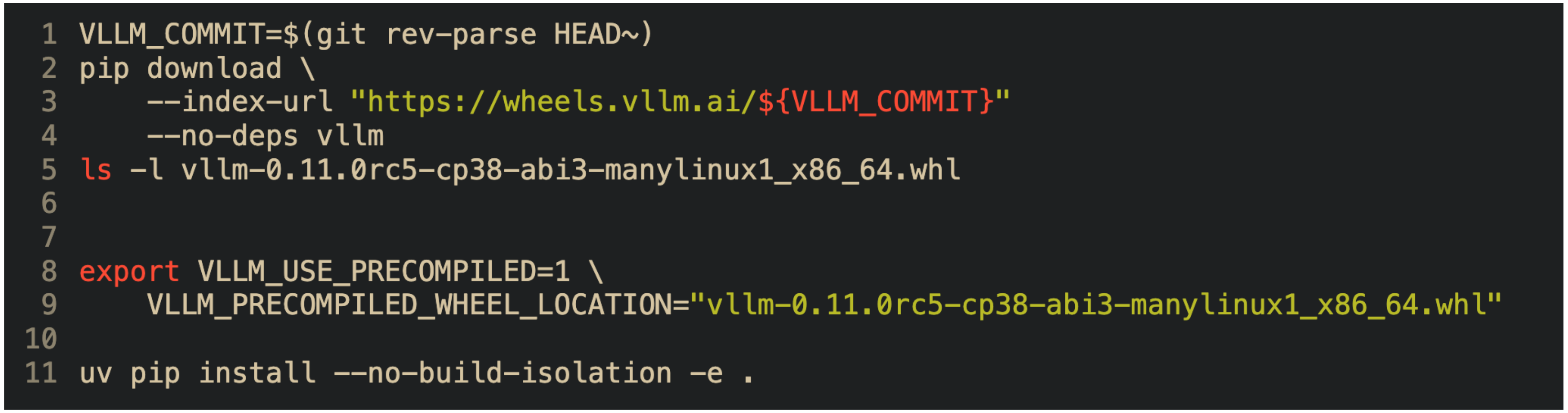


```
# Sometimes you can get by building the CPU target
export \
    VLLM_TARGET_DEVICE=cpu \
    UV_TORCH_BACKEND=cpu

uv pip install -r requirements/build.txt # has to match VLLM_TARGET_DEVICE

# only editing python?
export VLLM_USE_PRECOMPILED=1
# use latest nightly binaries
export VLLM_TEST_USE_PRECOMPILED_NIGHTLY_WHEEL=1

# build and install in editable mode
uv pip install --no-build-isolation -e .
```

Using a precompiled wheel from a specific commit



```
# only build for Nvidia A100
export TORCH_CUDA_ARCH_LIST="8.0"
# OR only build for AMD MI300X
export PYTORCH_ROCM_ARCH="gfx942"
```

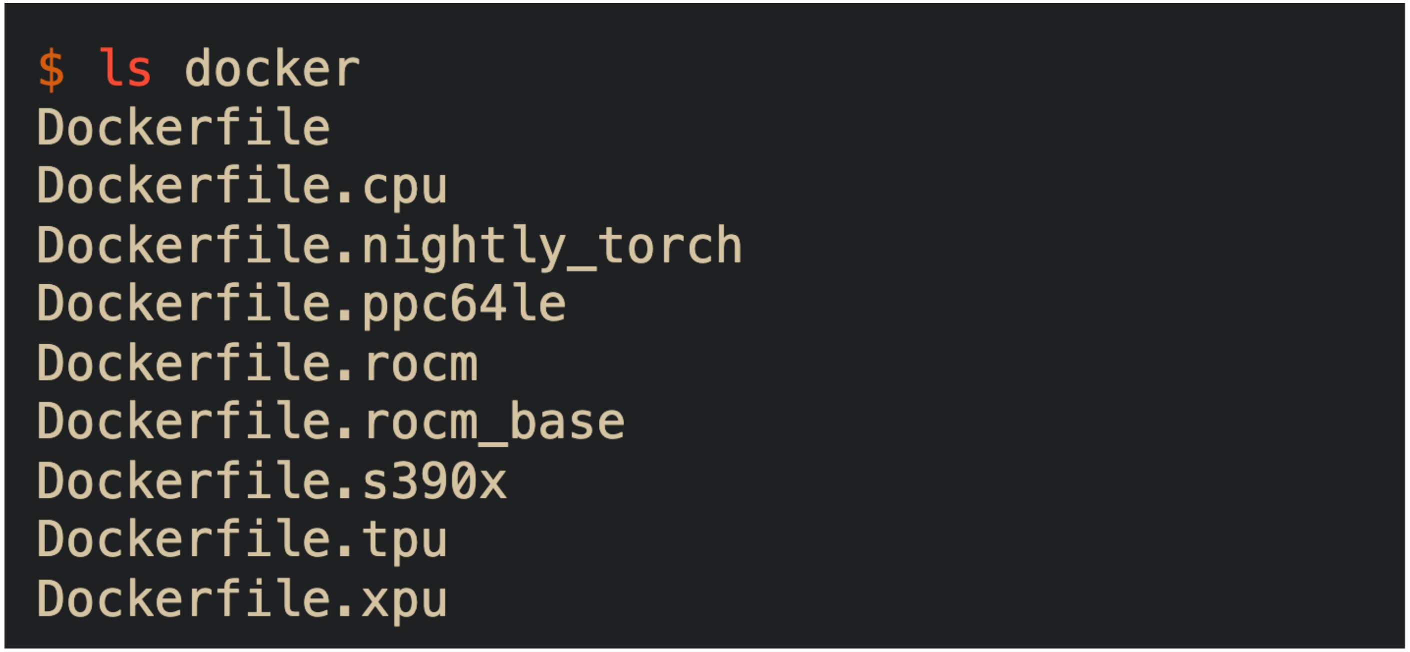


```
VLLM_COMMIT=$(git rev-parse HEAD~)
pip download \
    --index-url "https://wheels.vllm.ai/${VLLM_COMMIT}"
    --no-deps vllm
ls -l vllm-0.11.0rc5-cp38-abi3-manylinux1_x86_64.whl


export VLLM_USE_PRECOMPILED=1 \
    VLLM_PRECOMPILED_WHEEL_LOCATION="vllm-0.11.0rc5-cp38-abi3-manylinux1_x86_64.whl"

uv pip install --no-build-isolation -e .
```

# Docker

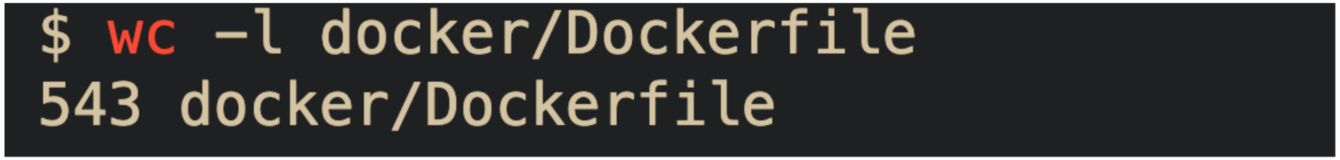
The vLLM wheel is built in a Dockerfile step.

There are different dockerfiles for each target:



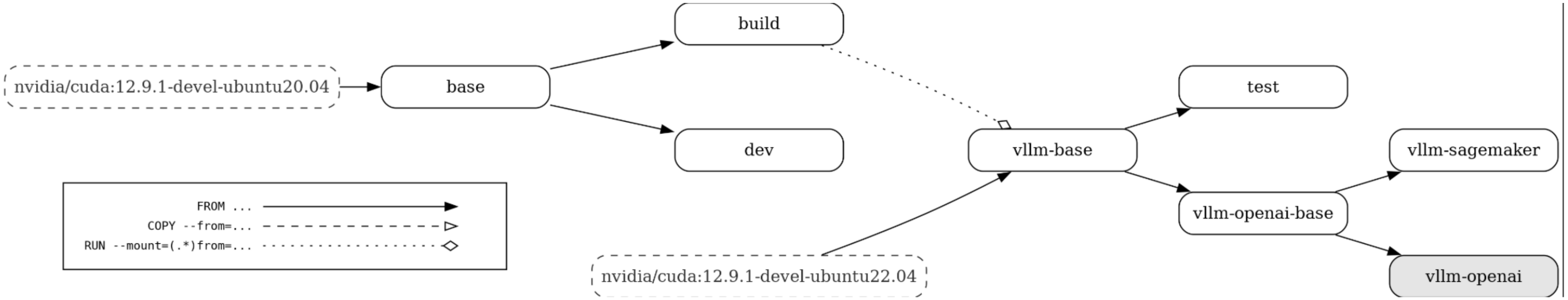
```
$ ls docker
Dockerfile
Dockerfile.cpu
Dockerfile.nightly_torch
Dockerfile.ppc64le
Dockerfile.rocm
Dockerfile.rocm_base
Dockerfile.s390x
Dockerfile.tpu
Dockerfile.xpu
```

## Docker



```
$ wc -l docker/Dockerfile
543 docker/Dockerfile
```

## Dependency graph

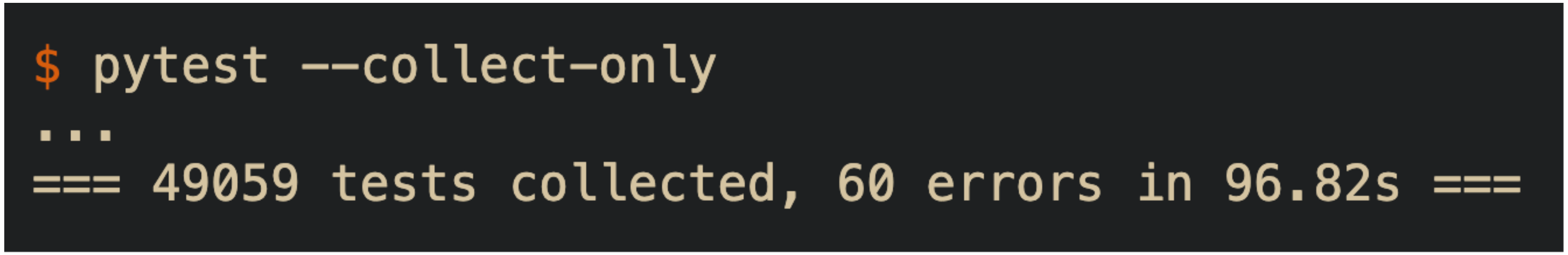


- base: toolchain setup (CUDA, nvcc, ...), python build/runtime deps
- build: vLLM wheel build
- vllm-base: installs vLLM wheel
- vllm-openai*: final stage for REST API (extra deps)
- test: stage used by CI

# Testing

Unit tests: "just" run pytest

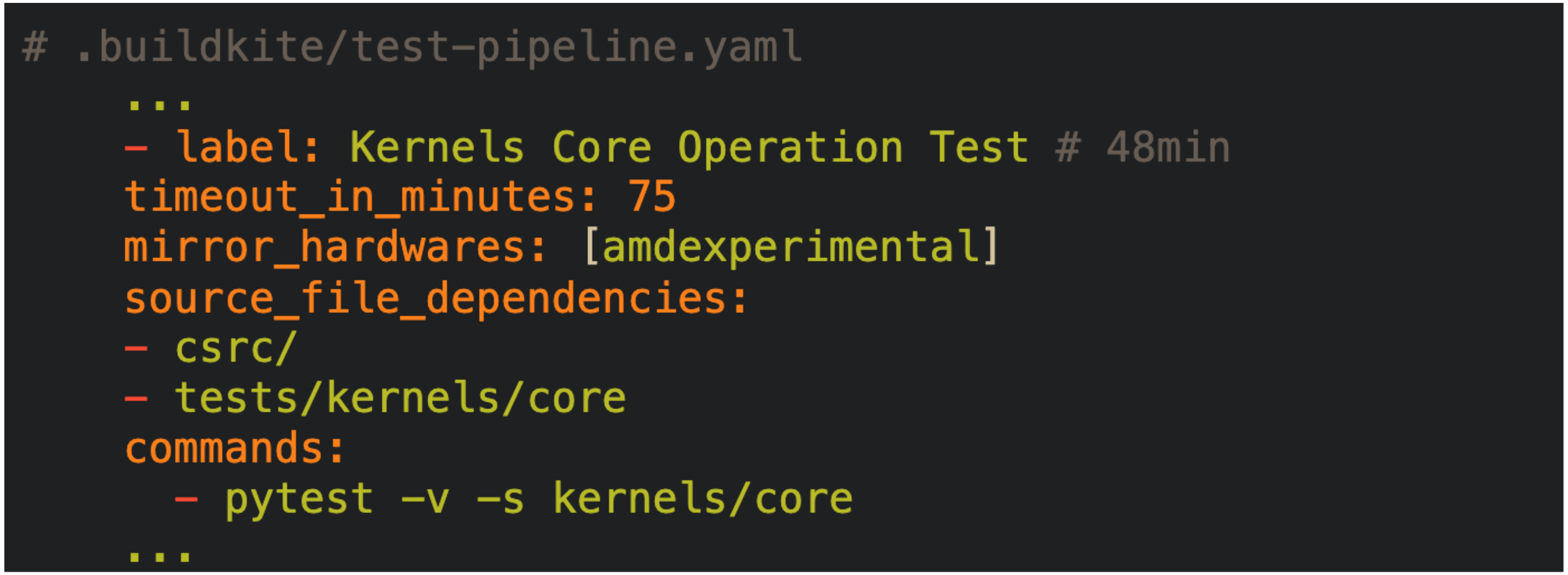
Example:



```
$ pytest --collect-only
...
=== 49059 tests collected, 60 errors in 96.82s ===
```

# **Testing** Tips

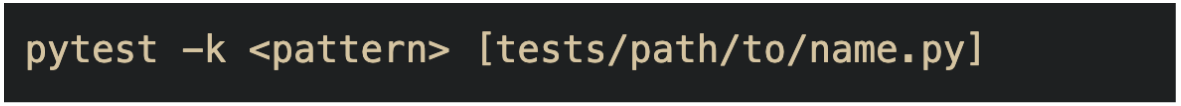
Buildkite (buildkite.com) is used for CI, check the pipeline definition to see which tests are run and how:



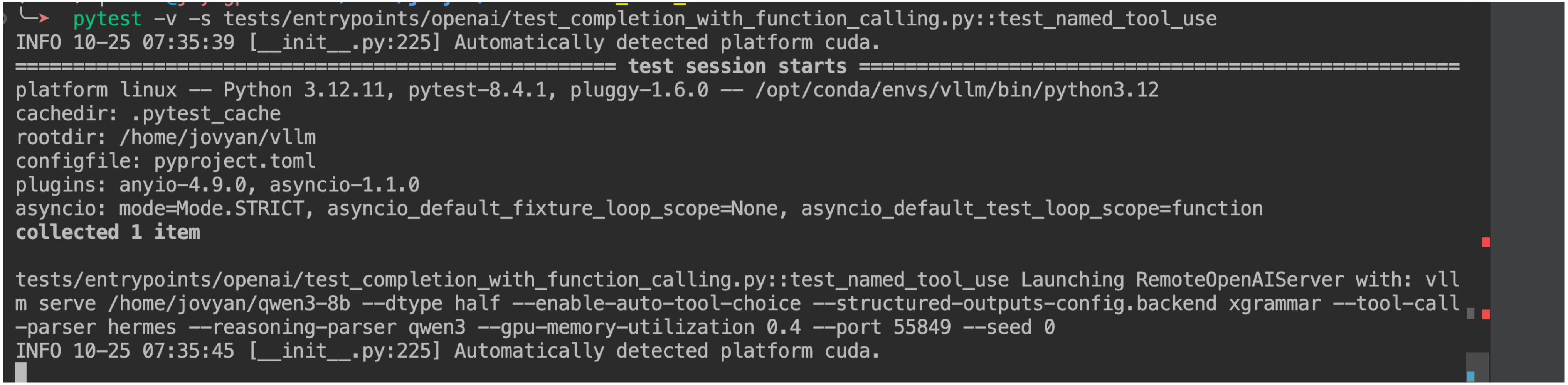
```yaml
# .buildkite/test-pipeline.yaml
    ...
    - label: Kernels Core Operation Test # 48min
    timeout_in_minutes: 75
    mirror_hardwares: [amdexperimental]
    source_file_dependencies:
    - csrc/
    - tests/kernels/core
    commands:
      - pytest -v -s kernels/core
    ...
```

## Testing

Use pytest -k to select tests by name/pattern



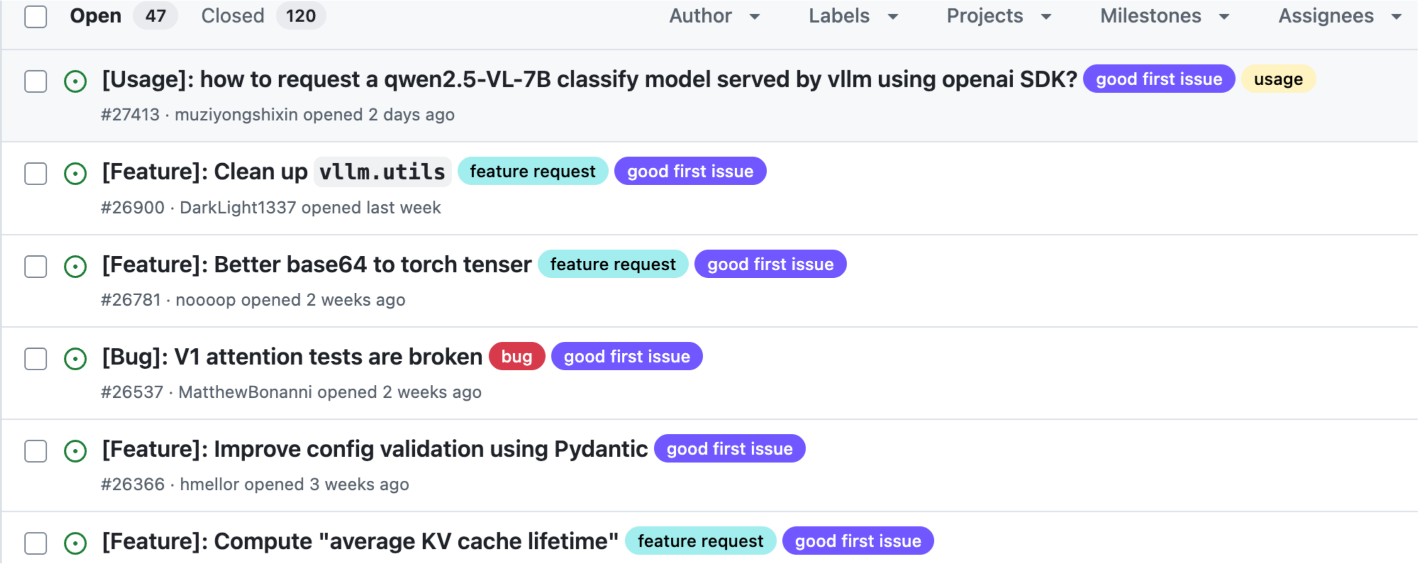
```
pytest -k <pattern> [tests/path/to/name.py]
```



```
└─➤  pytest -v -s tests/entrypoints/openai/test_completion_with_function_calling.py::test_named_tool_use
INFO 10-25 07:35:39 [__init__.py:225] Automatically detected platform cuda.
==================================================== test session starts =====================================================
platform linux -- Python 3.12.11, pytest-8.4.1, pluggy-1.6.0 -- /opt/conda/envs/vllm/bin/python3.12
cachedir: .pytest_cache
rootdir: /home/jovyan/vllm
configfile: pyproject.toml
plugins: anyio-4.9.0, asyncio-1.1.0
asyncio: mode=Mode.STRICT, asyncio_default_fixture_loop_scope=None, asyncio_default_test_loop_scope=function
collected 1 item

tests/entrypoints/openai/test_completion_with_function_calling.py::test_named_tool_use Launching RemoteOpenAIServer with: vll
m serve /home/jovyan/qwen3-8b --dtype half --enable-auto-tool-choice --structured-outputs-config.backend xgrammar --tool-call
-parser hermes --reasoning-parser qwen3 --gpu-memory-utilization 0.4 --port 55849 --seed 0
INFO 10-25 07:35:45 [__init__.py:225] Automatically detected platform cuda.
```

# Contributing

Search for "good first issue" labeled issues



□ **Open** 47   Closed 120   Author ▾   Labels ▾   Projects ▾   Milestones ▾   Assignees ▾

□ ⊙ **[Usage]: how to request a qwen2.5-VL-7B classify model served by vllm using openai SDK?** good first issue usage
#27413 · muziyongshixin opened 2 days ago

□ ⊙ **[Feature]: Clean up `vllm.utils`** feature request good first issue
#26900 · DarkLight1337 opened last week

□ ⊙ **[Feature]: Better base64 to torch tenser** feature request good first issue
#26781 · noooop opened 2 weeks ago

□ ⊙ **[Bug]: V1 attention tests are broken** bug good first issue
#26537 · MatthewBonanni opened 2 weeks ago

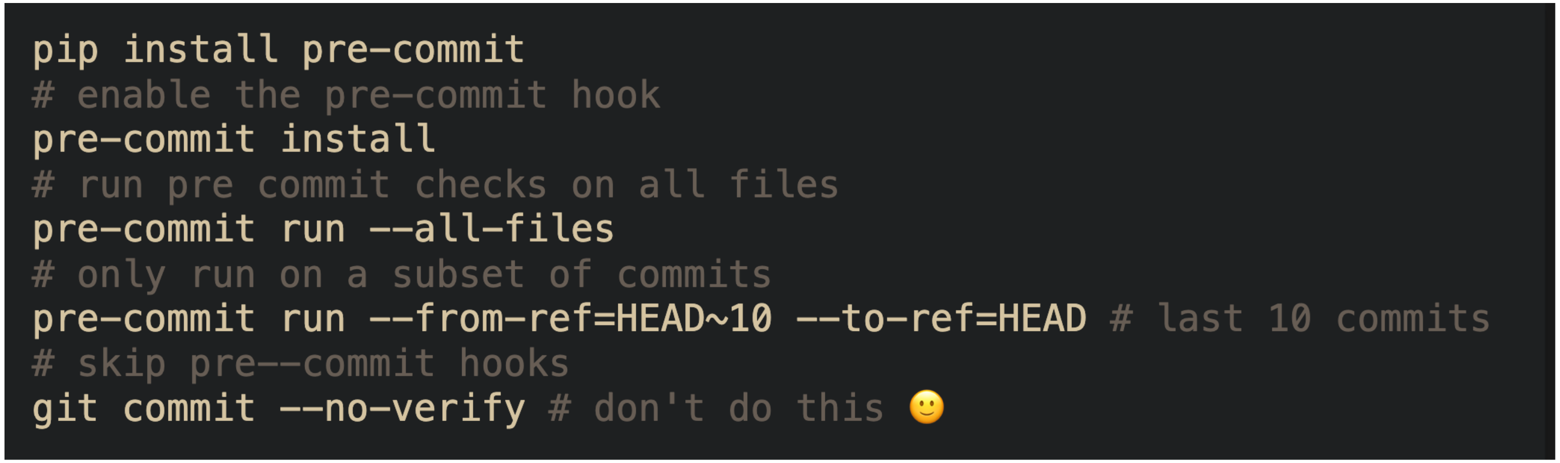
□ ⊙ **[Feature]: Improve config validation using Pydantic** good first issue
#26366 · hmellor opened 3 weeks ago

□ ⊙ **[Feature]: Compute "average KV cache lifetime"** feature request good first issue

# Contributing

- Signoff commits with git commit --signoff since Developer Certificate of Origin signoff is a required check

- Run pre-commit: pre-commit/pre-commit is used to enforce coding standards (formatting, linting)



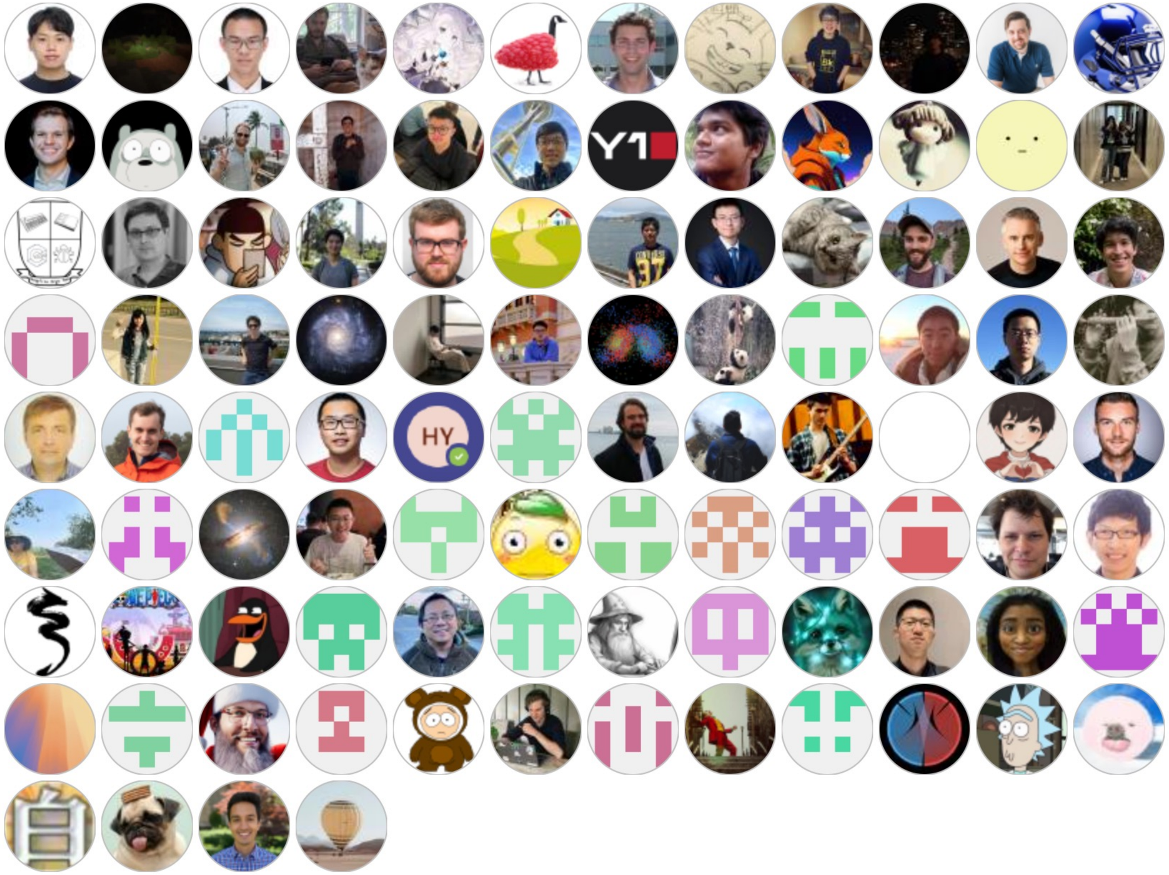
```
pip install pre-commit
# enable the pre-commit hook
pre-commit install
# run pre commit checks on all files
pre-commit run --all-files
# only run on a subset of commits
pre-commit run --from-ref=HEAD~10 --to-ref=HEAD # last 10 commits
# skip pre--commit hooks
git commit --no-verify # don't do this 🙂
```

# Contributing

- Write tests (unit/integration)
- Add documentation to docs/
- Keep changes small/incremental when possible

# Thank you contributors!

Thanks to 1700+ contributors who raised issues, participated in discussions, and submitted PRs!



Thank you sponsors (funding compute!)

a16z
Alibaba Cloud
AMD
Anyscale
aws
Crusoe
databricks
deepinfra
Dropbox
Google Cloud
Lambda
Meta
NEBIUS
novita.ai
NVIDIA
SEQUOIA
天工 Skywork
replicate
ROBLOX
RunPod
Trainy
THE UNIVERSITY OF CALIFORNIA BERKELEY 1868 LET THERE BE LIGHT
UNIVERSITY OF CALIFORNIA SAN DIEGO LET THERE BE LIGHT
ZhenFund 真格基金

# vLLM

Building the fastest and easiest-to-use open-source LLM inference & serving engine!

https://github.com/vllm-project/vllm

https://docs.vllm.ai

https://blog.vllm.ai

Ask AI

https://discuss.vllm.ai/

知 https://www.zhihu.com/people/vllm-team

https://www.linkedin.com/company/vllm-project

https://slack.vllm.ai

https://twitter.com/vllm_project

https://opencollective.com/vllm

小红书 vllm_project

✉ vllm-questions@lists.berkeley.edu