vLLM

Release Notes

RN-11517-001_v26.05 | June 2026

Table of Contents

Chapter 1. vLLM Overview

vLLM is a high-throughput and memory-efficient inference and serving engine for Large Language Models (LLMs). It seamlessly integrates with popular models from hubs like Hugging Face and offers a simple Python-based API. At its core is PagedAttention, a novel attention algorithm that manages key-value caches with near-zero memory waste by treating GPU memory like an operating system's virtual memory. This innovation allows for significantly larger batch sizes and provides state-of-the-art serving throughput.

vLLM also implements continuous batching, highly optimized CUDA kernels, and distributed inference through tensor parallelism. Inference requests are processed dynamically in a continuous stream rather than in static batches, which maximizes GPU utilization and dramatically reduces latency for real-world workloads.

For more information about vLLM, including documentation and examples, see:

- vLLM website

Chapter 2. Pulling A Container

About this task

Using the vLLM NGC Container requires the host system to have the following installed:

- Docker Engine
- NVIDIA GPU Drivers
- NVIDIA Container Toolkit

For supported versions, see the Framework Containers Support Matrix and the NVIDIA Container Toolkit Documentation.

**No other installation, compilation, or dependency management is required. It is not necessary to install the NVIDIA CUDA Toolkit*

Chapter 3. Running vLLM

Before you begin

To run a container, issue the appropriate command as explained in the Running A Container chapter in the NVIDIA Containers For Deep Learning Frameworks User's Guide and specify the registry, repository, and tags. For more information about using NGC, refer to the NGC Container User Guide.

Before you begin

If you have Docker 19.03 or later, a typical command to launch the container is:

```
docker run --gpus all -it --rm nvcr.io/nvidia/vllm:xx.xx-py3
```

If you have Docker 19.02 or earlier, a typical command to launch the container is:

```
nvidia-docker run -it --rm -v nvcr.io/nvidia/vllm:xx.xx-py3
```

Where:

- xx.xx is the container version. For example, 25.08.

vLLM can be run by importing it as a Python module:

```
export VLLM_ATTENTION_BACKEND=FLASHINFER
python3 -c "
from vllm import LLM
from vllm.sampling_params import SamplingParams
llm = LLM(model='TinyLlama/TinyLlama-1.1B-Chat-v0.4', trust_remote_code=True,
 gpu_memory_utilization=0.70)
sampling_params = SamplingParams(max_tokens=50, temperature=0.0)
prompts = [
 '<s> NVIDIA loves vLLM \U0001F49A',
 '<s> NVIDIA loves',
]
outputs = llm.generate(prompts, sampling_params)
"
```

vLLM can be deployed in a client–server configuration. Start the HTTP inference server inside the container:

```
python3 -m vllm.entrypoints.openai.api_server --model
nvidia/Llama-3.1-8B-Instruct-FP8 --trust-remote-code
--tensor-parallel-size $num_gpus --quantization fp8
--gpu-memory-utilization 0.90
```

From a client, issue a text-generation request by POST-ing to /generate with a JSON body containing the prompt and sampling parameters:

```
curl -X 'POST' 'http://0.0.0.0:8000/v1/chat/completions' -H
'accept: application/json' -H 'Content-Type:
application/json' -d '{
"model": "nvidia/Llama-3.1-8B-Instruct-FP8",
"max_tokens": 1024,
"messages": [{"role":"user", "content": "What is NVIDIA famous for?"}]
}'
```

See `/workspace/README.md` inside the container for information on getting started and customizing your vLLM image.

You might want to pull in data and model descriptions from locations outside the container for use by vLLM. To accomplish this, the easiest method is to mount one or more host directories as Docker bind mounts. For example:

```
docker run --gpus all -it --rm -v local_dir:container_dir nvcr.io/nvidia/vllm:xx.xx-
py3
```

Chapter 4. vLLM Release 26.04

The NVIDIA vLLM Release 26.04 is made up of two container images available on NGC: vLLM.

Contents of the vLLM container

This container image contains the complete source of the version of vLLM in /opt/vllm. It is pre-built and installed in the default system Python environment (/usr/local/lib/python3.12/dist-packages/vllm) in the container image. Visit vLLM Docs to learn more about vLLM.

The NVIDIA vLLM Container is optimized for use with NVIDIA GPUs, and contains the following software for GPU acceleration

- Please see to the CUDA section for the list of libraries inherited from the CUDA container.
- vLLM: 0.19.0
- flashinfer 0.6.7 post3
- transformers 4.57.6
- flash-attention 2.7.4.post1
- xgrammar 0.1.33
- Torch 2.12.0a0+0291f960b6.nv26.04.48445190

Driver Requirements

Release 26.04 is based on CUDA 13.2.1 For comprehensive and up-to-date driver compatibility information, please refer to the following documentation:

- NVIDIA CUDA Compatibility Guide - Compatibility information between CUDA versions and driver releases
- CUDA Toolkit Release Notes - Driver version requirements and compatibility matrices
- NVIDIA Drivers Download - Latest NVIDIA drivers

Key Features and Enhancements

This vLLM release includes the following key features and enhancements.

- Support Nemotron Super V3
- Support Nemotron 3 Nano Omni

Announcements

- None.

Known Issues

- `vLLM serve` uses aggressive GPU memory allocation by default (effectively --gpu-memory-utilization#1.0). On systems with shared/unified GPU memory (e.g. DGX Spark or Jetson platforms), this can lead to out-of-memory errors. If you encounter OOM, start vllm serve with a lower utilization value, for example: `vllm serve <model> --gpu-memory-utilization 0.7`.
- When running Nemotron Nano V3 or Nemotron Super V3 NVFP4 models on Spark it is required to limit the number of sequences to 4:
  - vllm serve <model> --max-num-seqs 4&
- When running Nemotron 3 Nano Omni you are required to override the model architecture reference:
  - Vllm serve <model> --hf-overrides='{"architectures":["NemotronH_Nano_VL_V2"]}'

Chapter 5. vLLM Release 26.03

The NVIDIA vLLM Release 26.03 is made up of two container images available on NGC: vLLM.

Contents of the vLLM container

This container image contains the complete source of the version of vLLM in /opt/vllm. It is pre-built and installed in the default system Python environment (/usr/local/lib/python3.12/dist-packages/vllm) in the container image. Visit vLLM Docs to learn more about vLLM.

The NVIDIA vLLM Container is optimized for use with NVIDIA GPUs, and contains the following software for GPU acceleration

- Please see to the CUDA section for the list of libraries inherited from the CUDA container.
- vLLM: 0.17.1
- flashinfer 0.6.7
- transformers 4.57.5
- flash-attention 2.7.4.post1
- xgrammar 0.1.32
- 2.11.0a0+a6c236b9fd1

Driver Requirements

Release 26.03 is based on CUDA 13.2.0 For comprehensive and up-to-date driver compatibility information, please refer to the following documentation:

- NVIDIA CUDA Compatibility Guide - Compatibility information between CUDA versions and driver releases
- CUDA Toolkit Release Notes - Driver version requirements and compatibility matrices
- NVIDIA Drivers Download - Latest NVIDIA drivers

Key Features and Enhancements

This vLLM release includes the following key features and enhancements.

- Support Nemotron Super V3

Announcements

- None.

Known Issues

- `vLLM serve` uses aggressive GPU memory allocation by default (effectively --gpu-memory-utilization#1.0). On systems with shared/unified GPU memory (e.g. DGX Spark or Jetson platforms), this can lead to out-of-memory errors. If you encounter OOM, start vllm serve with a lower utilization value, for example: `vllm serve <model> --gpu-memory-utilization 0.7`.

Chapter 6. vLLM Release 26.02

The NVIDIA vLLM Release 26.02 is made up of two container images available on NGC: vLLM.

Contents of the vLLM container

This container image contains the complete source of the version of vLLM in /opt/vllm. It is pre-built and installed in the default system Python environment (/usr/local/lib/python3.12/dist-packages/vllm) in the container image. Visit vLLM Docs to learn more about vLLM.

The NVIDIA vLLM Container is optimized for use with NVIDIA GPUs, and contains the following software for GPU acceleration

- Please see to the CUDA section for the list of libraries inherited from the CUDA container.
- vLLM: 0.15.1
- flashinfer 0.6.1
- transformers 4.57.5
- flash-attention 2.7.4.post1
- xgrammar 0.1.27
- torch 2.11.0a0+eb65b36914

Driver Requirements

Release 26.02 is based on CUDA 13.1.1 For comprehensive and up-to-date driver compatibility information, please refer to the following documentation:

- NVIDIA CUDA Compatibility Guide - Compatibility information between CUDA versions and driver releases
- CUDA Toolkit Release Notes - Driver version requirements and compatibility matrices
- NVIDIA Drivers Download - Latest NVIDIA drivers

Key Features and Enhancements

This vLLM release includes the following key features and enhancements.

- Support for `openai/gpt-oss-20b` and `openai/gpt-oss-120b`
- Support Nemotron-Nano-V2

Announcements

- None.

Known Issues

- vLLM serve uses aggressive GPU memory allocation by default (effectively --gpu-memory-utilization#1.0). On systems with shared/unified GPU memory (e.g. DGX Spark or Jetson platforms), this can lead to out-of-memory errors. If you encounter OOM, start vllm serve with a lower utilization value, for example: `vllm serve <model> --gpu-memory-utilization 0.7`.
- On DGX Spark, workloads utilizing FP8 models may fail with CUDA stream capture errors due to illegal synchronization operations in FlashInfer kernels. A fix is available in FlashInfer.
- When running Nemotron Super V3 model with FP8 it is required to use the Triton back end for attention, for example `vllm serve <model> --attention-backend triton_attn&`

Chapter 7. vLLM Release 26.01

The NVIDIA vLLM Release 26.01 is made up of two container images available on NGC: vLLM.

Contents of the vLLM container

This container image contains the complete source of the version of vLLM in /opt/vllm. It is pre-built and installed in the default system Python environment (/usr/local/lib/python3.12/dist-packages/vllm) in the container image. Visit vLLM Docs to learn more about vLLM.

The NVIDIA vLLM Container is optimized for use with NVIDIA GPUs, and contains the following software for GPU acceleration

- Please see to the CUDA section for the list of libraries inherited from the CUDA container.
- vLLM: 0.11.1
- flashinfer 0.5.2
- transformers 4.57.1
- flash-attention 2.7.4.post1
- xgrammer 0.1.25
- PyTorch 2.10.0a0+a36e1d39eb

Driver Requirements

Release 26.01 is based on CUDA 13.1.1 For comprehensive and up-to-date driver compatibility information, please refer to the following documentation:

- NVIDIA CUDA Compatibility Guide - Compatibility information between CUDA versions and driver releases
- CUDA Toolkit Release Notes - Driver version requirements and compatibility matrices
- NVIDIA Drivers Download - Latest NVIDIA drivers

Key Features and Enhancements

This vLLM release includes the following key features and enhancements.

- Support for `openai/gpt-oss-20b` and `openai/gpt-oss-120b`
- Support Nemotron-Nano-V2

Announcements

- None.

Known Issues

- vLLM serve uses aggressive GPU memory allocation by default (effectively --gpu-memory-utilization#1.0). On systems with shared/unified GPU memory (e.g. DGX Spark or Jetson platforms), this can lead to out-of-memory errors. If you encounter OOM, start vllm serve with a lower utilization value, for example: `vllm serve <model> --gpu-memory-utilization 0.7`.
- On DGX Spark, workloads utilizing FP8 models may fail with CUDA stream capture errors due to illegal synchronization operations in FlashInfer kernels. A fix is available in FlashInfer.

Chapter 8. vLLM Release 25.12

The NVIDIA vLLM Release 25.12 is made up of two container images available on NGC: vLLM.

Contents of the vLLM container

This container image contains the complete source of the version of vLLM in /opt/vllm. It is pre-built and installed in the default system Python environment (/usr/local/lib/python3.12/dist-packages/vllm) in the container image. Visit vLLM Docs to learn more about vLLM.

The NVIDIA vLLM Container is optimized for use with NVIDIA GPUs, and contains the following software for GPU acceleration

- Please see to the CUDA section for the list of libraries inherited from the CUDA container.
- NVIDIA CUDA 13.1.0.36
- vLLM: 0.11.1
- flashinfer 0.5.2
- transformers 4.57.1
- flash-attention 2.7.4.post1
- xgrammer 0.1.25
- torch-2.10.0a0+b4e4ee81d3

Driver Requirements

Release 25.12 is based on CUDA 13.1.0. For comprehensive and up-to-date driver compatibility information, please refer to the following documentation:

- NVIDIA CUDA Compatibility Guide - Compatibility information between CUDA versions and driver releases
- CUDA Toolkit Release Notes - Driver version requirements and compatibility matrices
- NVIDIA Drivers Download - Latest NVIDIA drivers

Key Features and Enhancements

This vLLM release includes the following key features and enhancements.

- Support for `openai/gpt-oss-20b` and `openai/gpt-oss-120b`
- Support Nemotron-Nano-V2

Announcements

- None.

Known Issues

- vLLM serve uses aggressive GPU memory allocation by default (effectively --gpu-memory-utilization#1.0). On systems with shared/unified GPU memory (e.g. DGX Spark or Jetson platforms), this can lead to out-of-memory errors. If you encounter OOM, start vllm serve with a lower utilization value, for example: `vllm serve <model> --gpu-memory-utilization 0.7`.
- On DGX Spark, workloads utilizing FP8 models may fail with CUDA stream capture errors due to illegal synchronization operations in FlashInfer kernels. A fix is available in FlashInfer.

Chapter 9. vLLM Release 25.11

The NVIDIA vLLM Release 25.11 is made up of two container images available on NGC: vLLM.

Contents of the vLLM container

This container image contains the complete source of the version of vLLM in /opt/vllm. It is pre-built and installed in the default system Python environment (/usr/local/lib/python3.12/dist-packages/vllm) in the container image. Visit vLLM Docs to learn more about vLLM.

The NVIDIA vLLM Container is optimized for use with NVIDIA GPUs, and contains the following software for GPU acceleration

- Please see to the CUDA section for the list of libraries inherited from the CUDA container.
- NVIDIA CUDA 13.0.2.006
- vLLM: 0.11.0
- flashinfer 0.5.0
- transformers 4.57.1
- flash-attention 2.7.4.post1
- xgrammer 0.1.25
- torch2.10.0a0+b558c986e8

Driver Requirements

Release 25.11 is based on CUDA 13.0.2. For comprehensive and up-to-date driver compatibility information, please refer to the following documentation:

- NVIDIA CUDA Compatibility Guide - Compatibility information between CUDA versions and driver releases
- CUDA Toolkit Release Notes - Driver version requirements and compatibility matrices
- NVIDIA Drivers Download - Latest NVIDIA drivers

Key Features and Enhancements

This vLLM release includes the following key features and enhancements.

- Support for `openai/gpt-oss-20b` and `openai/gpt-oss-120b`

Announcements

- None.

Known Issues

- vllm serve uses aggressive GPU memory allocation by default (effectively --gpu-memory-utilization#1.0). On systems with shared/unified GPU memory (e.g. DGX Spark or Jetson platforms), this can lead to out-of-memory errors. If you encounter OOM, start vllm serve with a lower utilization value, for example: `vllm serve <model> --gpu-memory-utilization 0.7`.

Chapter 10. vLLM Release 25.10

The NVIDIA vLLM Release 25.10 is made up of two container images available on NGC: vLLM.

Contents of the vLLM container

This container image contains the complete source of the version of vLLM in /opt/vllm. It is pre-built and installed in the default system Python environment (/usr/local/lib/python3.12/dist-packages/vllm) in the container image. Visit vLLM Docs to learn more about vLLM.

The NVIDIA vLLM Container is optimized for use with NVIDIA GPUs, and contains the following software for GPU acceleration

- vLLM: 0.10.2
- flashinfer 0.4.0
- transformers 4.56.1
- flash-attention 2.7.4
- xgrammer 0.1.24
- NVIDIA PyTorch 25.09

Driver Requirements

Release 25.10 is based on CUDA 13.0.2. For comprehensive and up-to-date driver compatibility information, please refer to the following documentation:

- NVIDIA CUDA Compatibility Guide - Compatibility information between CUDA versions and driver releases
- CUDA Toolkit Release Notes - Driver version requirements and compatibility matrices
- NVIDIA Drivers Download - Latest NVIDIA drivers

Key Features and Enhancements

This vLLM release includes the following key features and enhancements.

- Support for `openai/gpt-oss-20b` and `openai/gpt-oss-120b`

Announcements

- None.

Known Issues

- vllm serve uses aggressive GPU memory allocation by default (effectively --gpu-memory-utilization#1.0). On systems with shared/unified GPU memory (e.g. DGX Spark or Jetson platforms), this can lead to out-of-memory errors. If you encounter OOM, start vllm serve with a lower utilization value, for example: `vllm serve <model> --gpu-memory-utilization 0.7`.

Chapter 11. vLLM Release 25.09

The NVIDIA vLLM Release 25.09 is made up of two container images available on NGC: vLLM.

Contents of the vLLM container

This container image contains the complete source of the version of vLLM in /opt/vllm. It is pre-built and installed in the default system Python environment (/usr/local/lib/python3.12/dist-packages/vllm) in the container image. Visit vLLM Docs to learn more about vLLM.

The NVIDIA vLLM Container is optimized for use with NVIDIA GPUs, and contains the following software for GPU acceleration

- vLLM: 0.10.1.1
- flashinfer 0.4.0
- transformers 4.55.2
- flash-attention 2.7.4
- xgrammer 0.1.22
- NVIDIA PyTorch 25.09

Driver Requirements

Release 25.09 is based on CUDA 13.0. For comprehensive and up-to-date driver compatibility information, please refer to the following documentation:

- NVIDIA CUDA Compatibility Guide - Compatibility information between CUDA versions and driver releases
- CUDA Toolkit Release Notes - Driver version requirements and compatibility matrices
- NVIDIA Drivers Download - Latest NVIDIA drivers

Key Features and Enhancements

This vLLM release includes the following key features and enhancements.

- Compatibility with CUDA 13.0.

- Support for multi-node configurations.
- RTX PRO™ 6000 Blackwell Server Edition functional support.
- DGX Spark functional support.
- Jetson support.
- Support for 8-bit floating point (FP8) precision on Hopper GPUs and above.
- Support NVIDIA innovative 4-bit floating point NVFP4 format on Blackwell GPUs (including Jetson Thor and DGX Spark), which provides better training and inference performance with lower memory utilization.
- Support for DeepSeek-R1, Llama-3.1-8B-Instruct

Announcements

- 25.09 is the first NVIDIA vLLM container release that brings optimizations for NVIDIA GPUs.

Known Issues

- None

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation ("NVIDIA") makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer's own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

HDMI

HDMI, the HDMI logo, and High-Definition Multimedia Interface are trademarks or registered trademarks of HDMI Licensing LLC.

OpenCL

OpenCL is a trademark of Apple Inc. used under license to the Khronos Group Inc.

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
https://www.nvidia.com

Trademarks

NVIDIA, the NVIDIA logo, and cuBLAS, CUDA, DALI, DGX, DGX-1, DGX-2, DGX Station, DLProf, Jetson, Kepler, Maxwell, NCCL, Nsight Compute, Nsight Systems, NvCaffe, PerfWorks, Pascal, SDK Manager, Tegra, TensorRT, Triton Inference Server, Tesla, TF-TRT, and Volta are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
https://www.nvidia.com