GenAI-Perf for NIM LLM: a Quickstart

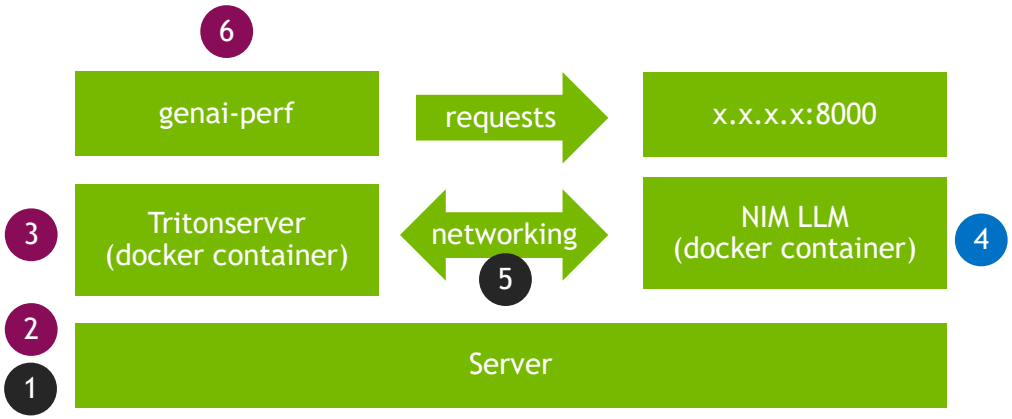
Pre-Requisites

- You'll need multiple terminals.
- For simplicity, this guide simply uses terminal A, B, C, D, and so on.

- For copy-paste-ready materials, see `genai-perf-exercise.md` and `docker-run-llama-3.1-8b-instruct.sh`
  - **IMPORTANT**: review the `*.sh` file first. Make changes as you like. For e.g., the `*.sh` script sources a `.env` file to set the `NGC_API_KEY` env var, so either you follow this style (which requires you to create that `.env` file first), or modify the script to your liking.

- So, let's get started.

TL/DR



Legend:

⌛ First attempt may take time

⌛ 1. [**Terminal A**] docker pull nim llm & tritonserver images

2. [**Terminal B**] Start tritonserver container

3. [**Terminal B**] Cache llama3.1 tokenizer in container

⌛ 4. [**Terminal C**] Start nim-llm container

5. [**Terminal A**] Connect both containers. Both containers to lose internet connection.

6. [**Terminal C**] Run `genai-perf profile`

Clean up: stop all containers, then delete network.

Step 1 | Terminal A

Pull containers

```
user1@ubuntu-28-r760xa-l40s-n4:~/verdimrc$ /usr/bin/time docker pull nvcr.io/nvidia/tritonserver:24.08-py3-sdk
24.08-py3-sdk: Pulling from nvidia/tritonserver
857cc8cb19c0: Pull complete
0b41952b72ac: Pull complete
4f4fb700ef54: Pull complete
07ba40811e97: Pull complete
5225d47729a1: Pull complete
a6736266741e: Pull complete
1d4f5bd6b322: Pull complete
31b0ebce44a6: Pull complete
80331719ab01: Pull complete
89611129266d: Pull complete
a6fb8d7f959b: Pull complete
92d994dcc9cf: Pull complete
5bb995e55ec8: Pull complete
9a2596b7b726: Pull complete
056fe255e556: Pull complete
25629fcc6fd9: Pull complete
e1115ebe8651: Pull complete
8ef815f89e82: Pull complete
ee92c7497919: Pull complete
a26c9e13c78f: Pull complete
4b0be3ca4e1c: Pull complete
6872ea0b4efa: Pull complete
0c5f6dae767f: Pull complete
497025ee1b79: Pull complete
926819ac7d1c: Pull complete
70704ce87a34: Pull complete
43be65e19e9d: Pull complete
7115a3a94730: Pull complete
90ecbf691a23: Pull complete
09b1170405d2: Pull complete
2b380b08e9bc: Pull complete
d67385517b0f: Pull complete
a38b60381eb9: Pull complete
d53f46034214: Pull complete
Digest: sha256:af34153227000b64d1ed4faf9612570a44d414ab8aa0e1dc143f18c19d71a5a7
Status: Downloaded newer image for nvcr.io/nvidia/tritonserver:24.08-py3-sdk
nvcr.io/nvidia/tritonserver:24.08-py3-sdk
0.33user 0.25system 6:36.35elapsed 0%CPU (0avgtext+0avgdata 26624maxresident)k
0inputs+0outputs (0major+3924minor)pagefaults 0swaps
user1@ubuntu-28-r760xa-l40s-n4:~/verdimrc$ docker images | grep 'nvcr.io/nvidia/tritonserver'
nvcr.io/nvidia/tritonserver              24.08-py3-sdk                  8f810f2f8b66   3 months ago   14.2GB
```

- Do the same for the NIM container: `docker pull nvcr.io/nim/meta/llama-3.1-8b-instruct:1.1.2`

Step 2-3 | Terminal B

Start tritonserver container, and cache tokenizers

Step 2: start tritonserver container

```
docker run -it --rm nvcr.io/nvidia/tritonserver:24.08-py3-sdk /bin/bash
```

Step 2: inside container: cache tokenizers

```
root@c5cd18efc017:/workspace# huggingface-cli login

    To login, `huggingface_hub` requires a token generated from https://huggingface.co/settings/tokens .
Enter your token (input will not be visible):
Add token as git credential? (Y/n) n
Token is valid (permission: read).
Your token has been saved to /root/.cache/huggingface/token
Login successful
```

```
root@c5cd18efc017:/workspace# python3 -c 'from transformers import AutoTokenizer
tokenizer = AutoTokenizer.from_pretrained("meta-llama/Meta-Llama-3.1-8B-Instruct")
'
None of PyTorch, TensorFlow >= 2.0, or Flax have been found. Models won't be available and only tokenizers, configuration and file/data ut
ilities can be used.
tokenizer_config.json: 100%|                                                                                    | 55.4k/55.4k [00:00<00:00, 281kB/s]
tokenizer.json: 100%|                                                                                           | 9.09M/9.09M [00:01<00:00, 5.91MB/s]
special_tokens_map.json: 100%|                                                                                  | 296/296 [00:00<00:00, 1.20MB/s]
```

Step 4 | Terminal C

Start NIM endpoint

```
user1@ubuntu-28-r760xa-l40s-n4:~/verdimrc$ ./docker-run-llama-3.1-8b-instruct.sh                                    [10/10]

==========================================
== NVIDIA Inference Microservice LLM NIM ==
==========================================

NVIDIA Inference Microservice LLM NIM Version 1.1.2
Model: nim/meta/llama-3.1-8b-instruct

Container image Copyright (c) 2016-2024, NVIDIA CORPORATION & AFFILIATES. All rights reserved.

The use of this model is governed by the NVIDIA AI Foundation Models Community License Agreement (found at https://www.nvidia.com/en-us/ag
reements/enterprise-software/nvidia-ai-foundation-models-community-license-agreement/#:~:text=This%20license%20agreement%20(%E2%80%9CAgree
ment%E2%80%9D,algorithms%2C%20parameters%2C%20configuration%20files%2C).

ADDITIONAL INFORMATION: Llama 3.1 Community License Agreement, Built with Llama.

INFO 12-03 12:50:33.517 ngc_profile.py:222] Running NIM without LoRA. Only looking for compatible profiles that do not support LoRA.
INFO 12-03 12:50:33.517 ngc_profile.py:224] Detected 6 compatible profile(s).
INFO 12-03 12:50:33.517 ngc_injector.py:132] Valid profile: 0494aafce0df9eeaea49bbca6b25fc3013d0e8a752ebcf191a2ddeaab19481ee (tensorrt_llm
-l40s-bf16-tp2-latency) on GPUs [0, 1]
INFO 12-03 12:50:33.517 ngc_injector.py:132] Valid profile: a534b0f5e885d747e819fa8b1ad7dc1396f935425a6e0539cb29b0e0ecf1e669 (tensorrt_llm
-l40s-bf16-tp2-throughput) on GPUs [0, 1]
INFO 12-03 12:50:33.517 ngc_injector.py:132] Valid profile: 3807be802a8ab1d999bf280c96dcd8cf77ac44c0a4d72edb9083f0abb89b6a19 (tensorrt_llm
-l40s-bf16-tp1-throughput) on GPUs [0]
INFO 12-03 12:50:33.517 ngc_injector.py:132] Valid profile: 407c6c5d1e29be9929f41b9a2e3193359b8ebfa512353de88cefbf1e0f0b194e (vllm-fp16-tp
4) on GPUs [0, 1, 2, 3]
INFO 12-03 12:50:33.517 ngc_injector.py:132] Valid profile: 6a3ba475d3215ca28f1a8c8886ab4a56b5626d1c98adbfe751025e8ff3d9886d (vllm-fp16-tp
2) on GPUs [0, 1, 2, 3]
INFO 12-03 12:50:33.517 ngc_injector.py:132] Valid profile: 3bb4e8fe78e5037b05dd618cebb1053347325ad6a1e709e0eb18bb8558362ac5 (vllm-fp16-tp
1) on GPUs [0, 1, 2, 3]
INFO 12-03 12:50:33.517 ngc_injector.py:190] Selected profile: 0494aafce0df9eeaea49bbca6b25fc3013d0e8a752ebcf191a2ddeaab19481ee (tensorrt_
llm-l40s-bf16-tp2-latency)
INFO 12-03 12:50:33.519 ngc_injector.py:198] Profile metadata: feat_lora: false
INFO 12-03 12:50:33.519 ngc_injector.py:198] Profile metadata: gpu: L40S
INFO 12-03 12:50:33.519 ngc_injector.py:198] Profile metadata: gpu_device: 26b5:10de
INFO 12-03 12:50:33.519 ngc_injector.py:198] Profile metadata: llm_engine: tensorrt_llm
INFO 12-03 12:50:33.519 ngc_injector.py:198] Profile metadata: pp: 1
INFO 12-03 12:50:33.519 ngc_injector.py:198] Profile metadata: precision: bf16
INFO 12-03 12:50:33.519 ngc_injector.py:198] Profile metadata: profile: latency
INFO 12-03 12:50:33.519 ngc_injector.py:198] Profile metadata: tp: 2
INFO 12-03 12:50:33.519 ngc_injector.py:218] Preparing model workspace. This step might download additional files to run the model.
metadata.json [00:00:00] [                                                                                  ] 231 B/231 B 970 B/s (0s)
NOTICE.txt [00:00:00] [                                                                                     ] 609 B/609 B 2.50 KiB/s (0s)
LICENSE.txt [00:00:00] [                                                                                    ] 429 B/429 B 1.45 KiB/s (0s)
checksums.blake3 [00:00:00] [                                                                               ] 471 B/471 B 1.57 KiB/s (0s)
rank1.engine [00:08:57] [
                                                                                  ] 6.99 GiB/8.10 GiB 13.14 MiB/s
rank1.engine [00:09:44] [                                                                                   ] 8.10 GiB/8.10 GiB 14.19 MiB/s (0s)
rank0.engine [00:00:00] [                                                                                   ] 0 B/8.09 GiB 0 B/s (0s)
```

• • •

```
INFO 12-03 13:10:58.998 api_server.py:577] Serving endpoints:
  0.0.0.0:8000/openapi.json
  0.0.0.0:8000/docs
  0.0.0.0:8000/docs/oauth2-redirect
  0.0.0.0:8000/metrics
  0.0.0.0:8000/v1/health/ready
  0.0.0.0:8000/v1/health/live
  0.0.0.0:8000/v1/models
  0.0.0.0:8000/v1/license
  0.0.0.0:8000/v1/metadata
  0.0.0.0:8000/v1/version
  0.0.0.0:8000/v1/chat/completions
  0.0.0.0:8000/v1/completions
  0.0.0.0:8000/experimental/ls/inference/chat_completion
  0.0.0.0:8000/experimental/ls/inference/completion
INFO 12-03 13:10:58.998 api_server.py:581] An example cURL request:
curl -X 'POST' \
  'http://0.0.0.0:8000/v1/chat/completions' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "model": "meta/llama-3.1-8b-instruct",
    "messages": [
      {
        "role":"user",
        "content":"Hello! How are you?"
      },
      {
        "role":"assistant",
        "content":"Hi! I am quite well, how can I help you today?"
      },
      {
        "role":"user",
        "content":"Can you write me a song?"
      }
    ],
    "top_p": 1,
    "n": 1,
    "max_tokens": 15,
    "stream": true,
    "frequency_penalty": 1.0,
    "stop": ["hello"]
  }'

INFO 12-03 13:10:59.43 server.py:82] Started server process [131]
INFO 12-03 13:10:59.43 on.py:48] Waiting for application startup.
INFO 12-03 13:10:59.51 on.py:62] Application startup complete.
INFO 12-03 13:10:59.52 server.py:214] Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
```

Step 5 | Terminal A

Connect tritonserver and NIM endpoint containers

```
user1@ubuntu-28-r760xa-l40s-n4:~/verdimrc$ docker ps -a
CONTAINER ID   IMAGE                                          COMMAND
                          NAMES
ef5d43ed0421   nvcr.io/nim/meta/llama-3.1-8b-instruct:1.1.2   "/opt/nvidia/nvidia_…"
/tcp, :::8000->8000/tcp   busy_bartik
c5cd18efc017   nvcr.io/nvidia/tritonserver:24.08-py3-sdk      "/opt/nvidia/nvidia_…"
                          dazzling_panini
user1@ubuntu-28-r760xa-l40s-n4:~/verdimrc$ docker network create hahanet
035e4addc71cb108e6f4a6ca23ba6346727959ab2f90d21c6aec5d8bccbbd896
user1@ubuntu-28-r760xa-l40s-n4:~/verdimrc$ docker network connect hahanet c5cd18efc017
user1@ubuntu-28-r760xa-l40s-n4:~/verdimrc$ docker network connect hahanet ef5d43ed0421
```

This part is just a syntax highlighter. Change to `jq` if you like or drop altogether if you don't have them installed.

```
user1@ubuntu-28-r760xa-l40s-n4:~/verdimrc$ docker network inspect hahanet | bat -l json -pp
[
    {
        "Name": "hahanet",
        "Id": "035e4addc71cb108e6f4a6ca23ba6346727959ab2f90d21c6aec5d8bccbbd896",
        "Created": "2024-12-03T13:19:49.749594Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "c5cd18efc017bcc82be03c2a1d73697a8ffab1c621870a4cf0a686334b528fc9": {
                "Name": "dazzling_panini",
                "EndpointID": "52a39d91f7ab971486de589d167c3b5aeff4232a6ee72d955b6c32311aaea02d",
                "MacAddress": "02:42:ac:12:00:02",
                "IPv4Address": "172.18.0.2/16",
                "IPv6Address": ""
            },
            "ef5d43ed04217c68e3af7b20ad962d5300ee828f27f1b3a14128a6ed641e054b": {
                "Name": "busy_bartik",
                "EndpointID": "fe73e15e4bf36c1851e2cf2a2bc198e5ffd3e67cad3622732e7a058a13cb4559",
                "MacAddress": "02:42:ac:12:00:03",
                "IPv4Address": "172.18.0.3/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
```

Pay attention to:

- Container IDs
- IP address of the NIM endpoint

Step 6 | Terminal B

Within tritonserver container, start the genai-perf

IP address of the NIM endpoint

```
root@c5cd18efc017:/workspace# export INPUT_SEQUENCE_LENGTH=200
export INPUT_SEQUENCE_STD=10
export OUTPUT_SEQUENCE_LENGTH=200
export CONCURRENCY=10
export MODEL=meta/llama-3.1-8b-instruct

# IMPORTANT: -u <IP_ADDRESS_OF_NIM_CONTAINER>, so change that line to match
#            your actual ip address
genai-perf \
    profile \
    -m $MODEL \
    --endpoint-type chat \
    --service-kind openai \
    --streaming \
    -u 172.18.0.3:8000 \
    --synthetic-input-tokens-mean $INPUT_SEQUENCE_LENGTH \
    --synthetic-input-tokens-stddev $INPUT_SEQUENCE_STD \
    --concurrency $CONCURRENCY \
    --output-tokens-mean $OUTPUT_SEQUENCE_LENGTH \
    --extra-inputs max_tokens:$OUTPUT_SEQUENCE_LENGTH \
    --extra-inputs min_tokens:$OUTPUT_SEQUENCE_LENGTH \
    --extra-inputs ignore_eos:true \
    --tokenizer meta-llama/Meta-Llama-3-8B-Instruct \
    -- \
    -v \
    --max-threads=256
2024-12-03 13:29 [INFO] genai_perf.parser:803 - Detected passthrough args: ['-v', '--max-threads=256']
2024-12-03 13:29 [INFO] genai_perf.parser:90 - Profiling these models: meta/llama-3.1-8b-instruct
2024-12-03 13:29 [INFO] genai_perf.parser:262 - Model name 'meta/llama-3.1-8b-instruct' cannot be used to create artifact directory. Inste
ad, 'meta_llama-3.1-8b-instruct' will be used.
2024-12-03 13:29 [INFO] genai_perf.wrapper:147 - Running Perf Analyzer : 'perf_analyzer -m meta/llama-3.1-8b-instruct --async --input-data
 artifacts/meta_llama-3.1-8b-instruct-openai-chat-concurrency10/llm_inputs.json -i http --concurrency-range 10 --endpoint v1/chat/completi
ons --service-kind openai -u 172.18.0.3:8000 --measurement-interval 10000 --stability-percentage 999 --profile-export-file artifacts/meta_
llama-3.1-8b-instruct-openai-chat-concurrency10/profile_export.json -v --max-threads=256'
                                    LLM Metrics
```

Statistic	avg	min	max	p99	p90	p75
Time to first token (ms)	484.13	38.50	3,958.13	3,958.11	534.85	132.85
Inter token latency (ms)	15.38	15.14	15.78	15.71	15.62	15.45
Request latency (ms)	3,544.80	3,159.64	6,971.52	6,971.50	3,612.87	3,166.23
Output sequence length	199.97	198.00	200.00	200.00	200.00	200.00
Input sequence length	200.14	178.00	219.00	218.01	214.00	207.50

```
Output token throughput (per sec): 562.90
Request throughput (per sec): 2.81
2024-12-03 13:30 [INFO] genai_perf.export_data.json_exporter:58 - Generating artifacts/meta_llama-3.1-8b-instruct-openai-chat-concurrency1
0/profile_export_genai_perf.json
2024-12-03 13:30 [INFO] genai_perf.export_data.csv_exporter:69 - Generating artifacts/meta_llama-3.1-8b-instruct-openai-chat-concurrency10
/profile_export_genai_perf.csv
```

Clean up

- Stop or exit both containers (on **their respective terminal**)
- On **terminal A**: `docker network rm -f hahanet`

Thank you!

And as always, happy coding & experimenting!