OZONtech

Профилирование моделей в Triton Inference Server

Perfomance/Model Analyzer

OzonTech, Самусевич Юрий,
ведущий математик-разработчик

01 {

Профилирование

Что такое профилирование в общем виде?

Профилирование — оценка того, как ведет себя код/программа при исполнении:

1. Сколько времени занимает исполнение
2. Как и на что тратятся ресурсы
3. Какие части программы выполняются чаще всего

OZONtech

Зачем профилировать?

Ресурсы ограничены

Пользовательский опыт

Нагрузка высокая

Сравнение моделей

02

Что можно профилировать в ML?

Да в общем-то всё

1. Энкодинги/декодинги данных
2. Трансфер данных
3. Ресайзы изображений и аугментации
4. Токенизацию
5. Инференс модели
6. Кастомную логику вокруг всего этого

OZONtech

03

Perfomance Analyzer

Краткий обзор

1. Позволяет проводить нагрузочное тестирование
2. Считает задержку ответа и пропускную способность
3. Поддерживает асинхронность
4. Собирает серверные метрики
5. Работает с http/grpc
6. Работает для заранее заданного конфига модели
7. Поддерживает tfserving/torchserve
8. Генерирует отчеты

OZON tech

С чем умеет работать?

1. Ансамбли
2. Последовательные модели
3. Разделённые модели

OZONtech

Что такое пропускная способность?

Пропускная способность — метрика, показывающая, сколько запросов за единицу времени сервер может успешно обработать.

OZON{ech

Задержка

В общем виде **задержка** — время от получения запроса сервером до отправления ответа.

Perf analyzer отдаёт разбивку задержки на 3 компонента:

- **Queue** — время, которое запрос находится в очереди
- **Compute** — время, потраченное на инференс моделью и ансамблем с учётом перекладывания данных на ЦПУ/ГПУ
- **Overhead** — все остальное время, включая парсинг запросов, валидацию данных, формирование ответа и прочее

Пропускная способность VS Задержка

Важность метрик определяется сценарием использования модели - онлайн или офлайн. Между ними существует компромисс.

GPU-метрики

- **GPU Utilization** — усреднённое время загрузки графического процессора
- **GPU Power Usage** — усреднённое потребление ватт
- **GPU Used Memory** — потребление памяти в пике
- **GPU Total Memory** — общее количество доступной памяти

OZONtech

Данные для профилирования

Для оценки работы модели нужны данные, а сам результат измерений крайне часто зависит от этих самых данных. Perfomance Analyzer предоставляет несколько способов отправить что-то в модель с необходимой размерностью:

- тензор из нулей
- тензор из случайных значений
- заданная строка
- реальные данные

Формат реальных данных


```
{
 "data":
   [
     {
       "INPUT0": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
       "INPUT1": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
     }
   ],
 "validation_data":
   [
     {
       "OUTPUT0": [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       "OUTPUT1": [2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]
     }
   ]
}
```

OpenAI API

```
{
  "data": [
    {
      "payload": [
        {
          "model": "facebook/opt-125m",
          "messages": [{ "role": "user",
                         "content": "Who wrote the play Romeo and Juliet?" }],
          "max_tokens": 32
        }
      ]
    }
  ]
}
```

Что получаем в итоге?

Concurrency	Inferences/Second	Client Send	Network+Server Send/Recv	Server Queue	Server Compute Input	...	Server Compute Output	Client Recv	p50 latency	p90 latency	p95 latency	p99 latency
300	141.305	32547	62271	8566161	65380	...	288	0	8782767.0	9484258.0	9611550.0	9675698.0
400	142.193	35784	66660	11211021	60868	...	308	0	11288099.0	12074667.0	12361071.0	12512672.0
450	142.193	35784	66660	11211021	60868	...	308	0	NaN	NaN	NaN	NaN
350	147.523	39574	76219	9241462	59482	...	347	0	9428468.0	10008779.0	10100556.0	10263372.0
50	166.650	32302	54667	1028645	53098	...	224	0	1147502.0	1632915.0	1890374.0	2336826.0
250	181.311	28562	57345	5288061	44981	...	214	0	5417849.0	5883504.0	5929392.0	6007273.0
150	203.977	36644	79937	2768134	37389	...	196	0	2879736.0	3535509.0	3885260.0	4059153.0
200	205.752	59545	1323839	2307545	39287	...	203	0	3821189.0	4251893.0	4316908.0	4641283.0
100	215.090	25411	51162	1712698	35021	...	194	0	1751439.0	2166428.0	2605259.0	3141438.0

Concurrency	throughput	latency
100	903.863 infer/sec	443279 usec
200	906.528 infer/sec	867465 usec
300	895.814 infer/sec	1309799 usec
400	916.291 infer/sec	1709743 usec
500	901.137 infer/sec	2172470 usec
600	918.857 infer/sec	2545185 usec
700	917.979 infer/sec	2992571 usec
800	917.125 infer/sec	3404102 usec
900	911.8 infer/sec	3852445 usec
1000	919.75 infer/sec	4262039 usec

Установка

1. SDK-контейнер
2. Pip
3. Build from Source

OZONtech

04

Model Analyzer

Ещё один инструмент?

Perfomance Analyzer — хороший инструмент, который позволяет оценить уже собранную модель, но что, если мы хотим подобрать лучшие параметры автоматически, а не заниматься правкой конфига каждый раз?

OZONtech

Ключевые особенности

1. Под капотом использует Perfomance Analyzer — поддерживает все его опции и команды
2. Позволяет задавать различные ограничения и целевые метрики
3. Умеет подбирать параметры полным и быстрым перебором по решетке
   + в бете байесовская оптимизация
4. Взвешивание моделей и нескольких метрик
5. Даёт чуть больше информации и красивые отчёты
6. И даже переваривает LLM

OZONtech

Что умеет перебирать?

- Количество экземпляров модели
- Максимальный размер батча
- Количество параллельных запросов

OZONtech

Оптимизируемые метрики

Option Name	Description
perf_throughput	Use throughput as the objective.
perf_latency_p99	Use latency as the objective.
gpu_used_memory	Use GPU memory used by the model as the objective.
gpu_free_memory	Use GPU memory not used by the model as the objective.
gpu_utilization	Use the GPU utilization as the objective.
cpu_used_ram	Use RAM used by the model as the objective.
cpu_free_ram	Use RAM not used by the model as the objective.
output_token_throughput	Use output token throughput as the objective.
inter_token_latency_p99	Use inter token latency as the objective.
time_to_first_token_p99	Use time to first token latency as the objective.

Взвешивание моделей

В случае, когда у нас несколько моделей, мы можем указать важность каждой при подборе параметров — это может быть полезно, когда мы точно знаем, где у нас боттлнек.

```
model_repository: /path/to/model/repository/

run_config_profile_models_concurrently_enable: true

profile_models:
 model_A:
   weighting: 3
   objectives:
     perf_latency_p99: 1
 model_B:
   weighting: 1
   objectives:
     perf_throughput: 1
```

Как собрать конфиг?

```yaml
model_repository: models

run_config_search_mode: quick
run_config_search_min_instance_count: 1
run_config_search_max_instance_count: 4
run_config_search_min_model_batch_size: 4
run_config_search_max_model_batch_size: 16

profile_models:
  - recreate_card_img_encoder

perf_analyzer_flags:
  percentile: 95
  shape:
    - norm_resize_old_photo:3,448,448
    - norm_resize_current_photo:3,448,448
  protocol: grpc
  concurency-range: 500

objectives:
  - perf_throughput
```

Запуск

```
docker run --runtime nvidia --gpus '"device=2,5"' --network host \
--volume '/var/run/docker.sock:/var/run/docker.sock' \
--volume '/data/example/models:/data/example/models' \
--volume '/data/example/output:/data/example/output' \
--volume '/data/example/configs:/data/example/configs' \
--volume '/data/example/commands/real_data.json:/data/example/commands/real_data.json' \
--rm nvcr.io/nvidia/tritonserver:24.01-py3-sdk \
model-analyzer profile -f /data/example/configs/config.yaml \
--model-repository /data/example/models \
--override-output-model-repository \
--triton-launch-mode docker \
--triton-docker-image nvcr.io/nvidia/tritonserver:24.01-py3 \
--output-model-repository-path /data/example/output/res \
--export-path /data/example/output \
--async
```

Пример результата

Online Result Summary

Model: recreate_card_img_encoder

GPU(s): 2 x NVIDIA A100-PCIE-40GB

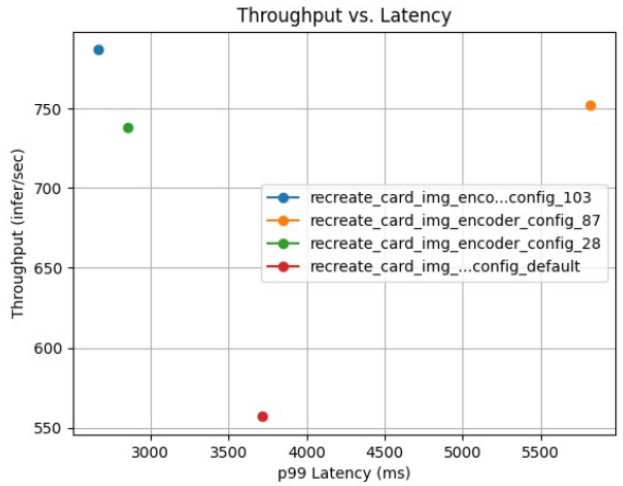
Total Available GPU Memory: 78.8 GB

Constraint targets: None

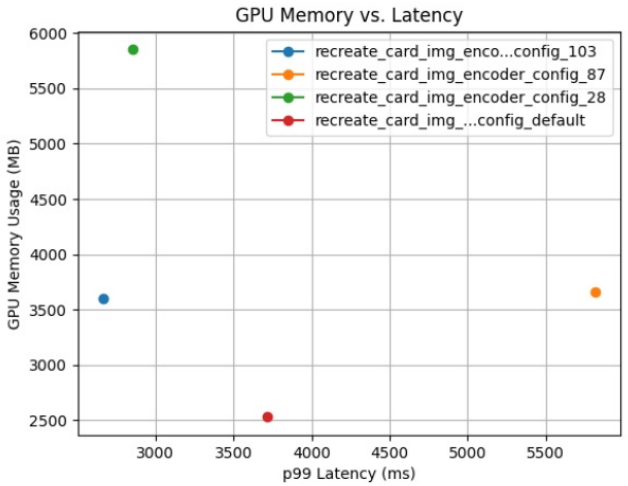
In 151 measurements across 151 configurations, **recreate_card_img_encoder_config_103** is **41%** better than the default configuration at meeting the objectives, under the given constraints, on GPU(s) 2 x NVIDIA A100-PCIE-40GB.

- **recreate_card_img_encoder_config_103**: 2 GPU instances with a max batch size of 12 on platform onnxruntime_onnx

Curves corresponding to the 3 best model configuration(s) out of a total of 151 are shown in the plots.



Throughput vs. Latency curves for 3 best configurations.



GPU Memory vs. Latency curves for 3 best configurations.

OZONtech

05 {

Время практики

Ссылки

1. GenAI-Perf
2. Perf Analyzer
3. Model Analyzer