OZONtech

Развертывание ML-моделей в высоконагруженных системах

Дядюнов Андрей, TeamLead ML OzonTech

Горшков Виктор, ML Engineer OzonTech

Организационные моменты

- Формула оценки: 0.2 x ДЗ-1 + 0.25 x ДЗ-2 + 0.25 x ДЗ-3 + 0.3 x Проект
- 8 лекций, 3 ДЗ, 1 финальный проект
- Дополнительные вопросы в чате TG
- Все материалы: GitHub + Яндекс.Диск + Wiki курса - в пятницу вечером
- Задания сдаем в anytask
- На досдачу просроченных домашек 4 дня

Краткая история курса

- Накапливали знания о высоких нагрузках Ozon на ML с 2021го года
- В начале 2024 сделали курс на внутреннюю аудиторию
- В середине 2024 выкатили курс на Ozon Route256
- Курс продолжает дополняться и улучшаться

О чем поговорим

1. Вводная лекция
2. Triton Inference Server
    1. Семинар
    2. Домашнее задание
3. Backend: python, dali, tensorrt, onnx, llm
    1. Семинар
    2. Домашнее задание
4. Конвертация моделей
    1. Семинар
5. Ансамбли
    1. Семинар
    2. Домашнее задание
6. Perf-analyzer
7. Сравнение подходов к развертыванию на практике
8. Итоговый проект, ответы на вопросы
    1. Проект

Вводная лекция

Что такое deployment и почему это так важно?

Какие сейчас есть методы развернуть ML модель на проде?

Что мы рассмотрим на курсе?

Немного про Docker

00

Deployment

Deployment

В широком смысле слова — это процесс, при котором разработанное программное обеспечение или приложение переносится из среды разработки в рабочую среду, где оно становится доступным для конечных пользователей. Это важный этап жизненного цикла разработки программного обеспечения, который включает в себя подготовку, настройку и запуск приложения на серверах или в облачной инфраструктуре.

В контексте ML — это процесс, при котором обученная модель и вся дополнительная логика пред- и постобработки данных попадает из тестового окружения (вашего компьютера или сервера) в продакшн для конечного потребителя

Deployment

Чтобы что-то раскатить на плодовый сервер в условиях высокой нагрузки нам нужно несколько дополнительных инструментов:

Docker

Kubernetes

Как подружить модель с инфрой

- Часто основной ЯП не Python
- Важно оставлять ML-решение и всю логику отдельно
- Нужно собирать много разных метрик, чтобы все проверять

Какие есть варианты?

- FastAPI
- TorchServe
- TensorFlow Serving
- Triton Inference Server
- MLFlow
- BentoML
- Seldon Core
- Flask
- Ray Serve
- …

Triton Inference Server

Для своих просто Triton обладает большим количеством плюсов

Triton выделяется благодаря своей производительности, гибкости и поддержке различных фреймворков. Он особенно хорош для высоконагруженных систем, где важны низкая задержка, высокая пропускная способность и эффективное использование ресурсов. Если ваш проект требует масштабируемости, производительности и универсальности, Triton — один из лучших вариантов.

Сначала о простом. Docker

Docker — это платформа для разработки, доставки и запуска приложений в изолированных средах, называемых контейнерами. Контейнеры позволяют упаковать приложение со всеми его зависимостями (библиотеками, настройками, кодом) в единый образ, который можно легко развернуть на любой системе.

Почему Docker важен для ML?

- Изоляция: ML-модели часто требуют специфичных версий библиотек, которые могут конфликтовать с другими приложениями. Docker решает эту проблему.
- Портативность: Образ Docker можно запустить на любой системе или в облаке.
- Масштабируемость: Docker интегрируется с Kubernetes и другими оркестраторами, что критично для высоконагруженных систем.

Основные концепции Docker

- Образ (Image)
- Контейнер (Container)
- Dockerfile
- Docker Compose

```dockerfile
FROM python:3.9-slim  # Базовый образ
WORKDIR /app          # Рабочая директория
COPY requirements.txt .  # Копируем зависимости
RUN pip install -r requirements.txt  # Устанавливаем зависимости
COPY . .              # Копируем код модели
CMD ["python", "app.py"]  # Команда для запуска
```

Пример Dockerfile

Посмотрим на важные команды

docker build -t ml-model:1.0 . — создает образ с именем ml-model и версией 1.0

docker run -p 5000:5000 ml-model:1.0 — запускаем контейнер

docker ps

docker inspect

docker kill

…

Важные преимущества

1. Изоляция и стабильность — каждая модель работает в своем контейнере, что предотвращает конфликты.

2. Масштабируемость — Docker легко интегрируется с Kubernetes для автоматического масштабирования.

3. Повторяемость — образ Docker гарантирует, что модель будет работать одинаково на всех этапах (разработка, тестирование, продакшн).

4. Эффективность — контейнеры легче виртуальных машин и быстрее запускаются.

Помимо прочего

5. Многомодельное развертывание
6. Интеграция с CI/CD
7. Мониторинг и логирование
8. Безопасность

Вопросы