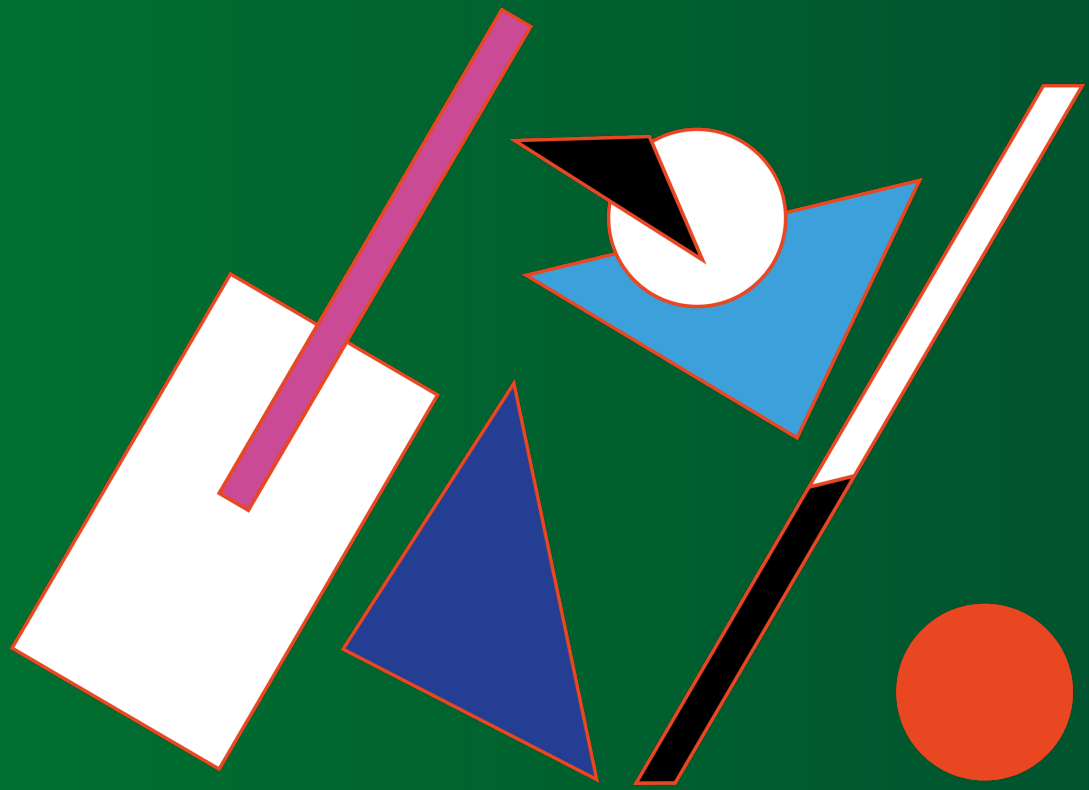
PostgresPro

C++ RUSSIA 2024

Нецелевое использование ONNX в качестве математической библиотеки

Тимур Магомедов



Postgres Professional

- Российский разработчик системы управления базами данных Postgres Pro на основе PostgreSQL.
- Активно развиваем open-source СУБД PostgreSQL.
- Второе место в мировом рейтинге контрибьюторов PostgreSQL.
- За последние три года бизнес компании вырос в 8 раз.
- Читаем лекции, сотрудничаем с вузами, выпускаем книги.
- Проводим сертификацию специалистов по PostgreSQL.

Nvidia CUDA

Плюсы:

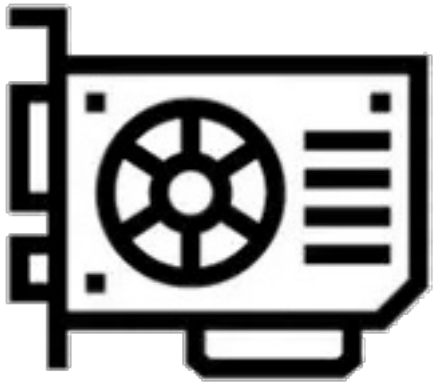
- Отличная документация.
- Много оптимизированных библиотек.
- Накоплен большой опыт эксплуатации.
- Высокая производительность.

Минусы:

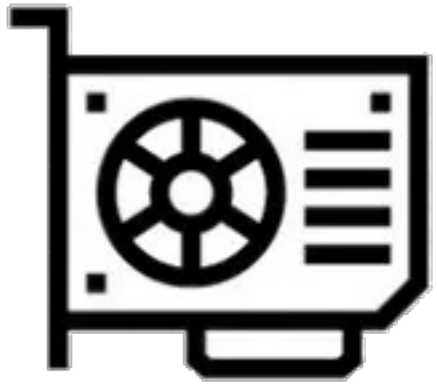
- Работает только на Nvidia.
- Нет Nvidia? Переписывай всё заново!

Устройства для вычислений

CPU

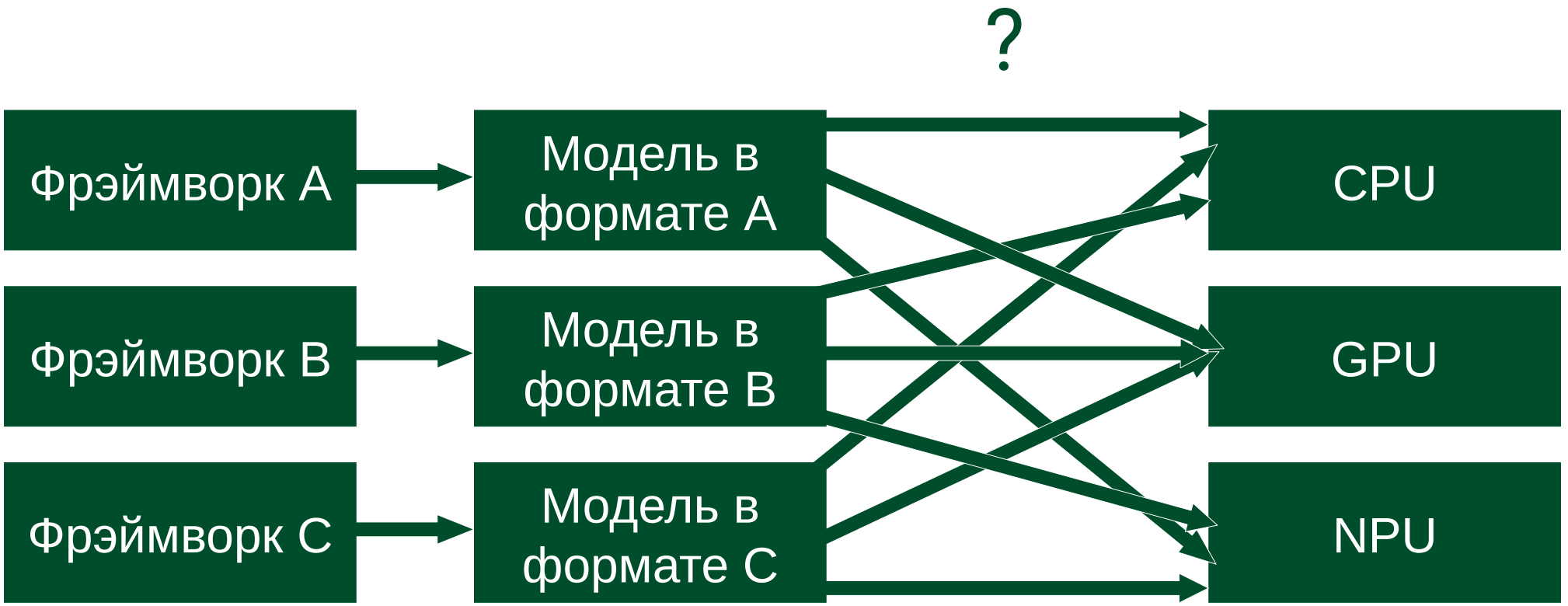


GPU

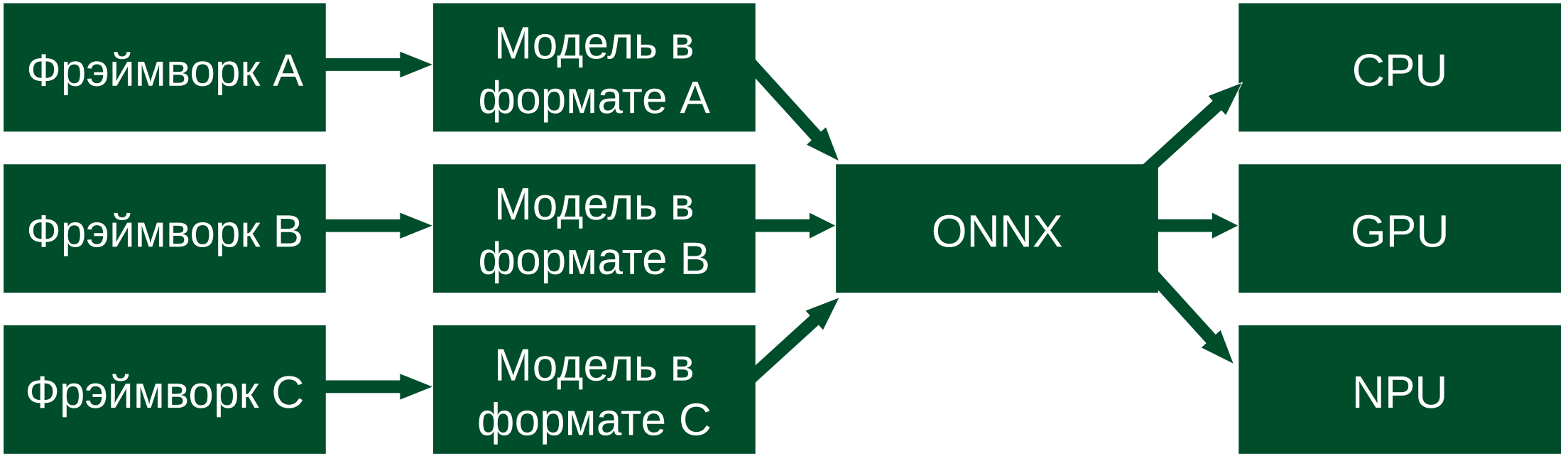


NPU

Как подружить машинное обучение с железом?



Open Neural Network Exchange



Сообщество ONNX

Поддерживают экспорт моделей в ONNX

Caffe2
Yandex CatBoost
Chainer
Cognitive Toolkit
CoreML
Optimum
Keras
LibSVM
MATLAB
MindSpore
composer
mxnet
MyCaffe
NCNN
NeoML
Neural Network Libraries
PaddlePaddle
PyTorch
sas
SIEMENS
Simio Forward Thinking
SINGA
SciKit Learn
Tengine
TensorFlow
Tribuo
dmlc XGBoost
WOLFRAM
ZAMA

Может ли NPU перемножить матрицы?

Умеет считать свёртки — значит сумеет и матрицы.

Есть библиотека AscendCL CBLAS для NPU Huawei, но...

Перемножение матриц с AscendCL CBLAS


```
aclError aclblasGemmEx(
    aclTransType transA, aclTransType transB, aclTransType transC,
    int m, int n, int k,
    const void *alpha,
    const void *matrixA, int lda, aclDataType dataTypeA,
    const void *matrixB, int ldb, aclDataType dataTypeB,
    const void *beta,
    void *matrixC, int ldc, aclDataType dataTypeC,
    aclComputeType type,
    aclrtStream stream)
```

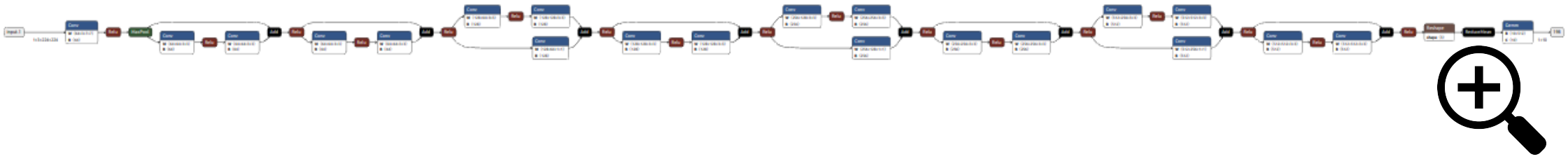
Но это не всё!

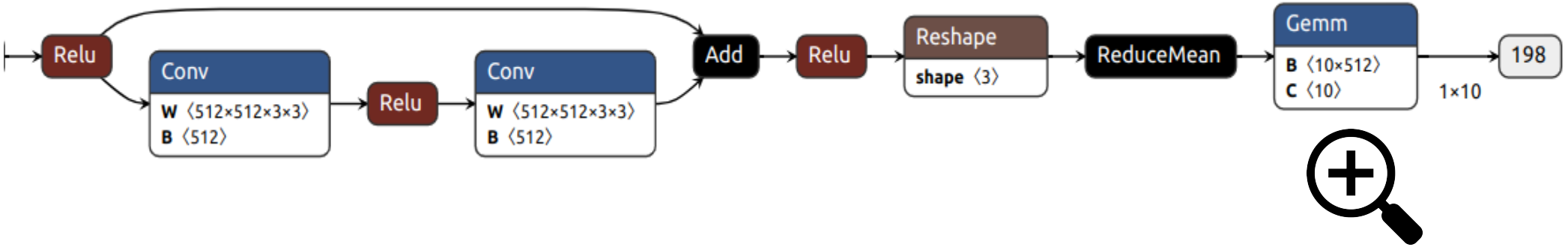
1) Создать специальный gemm.json с описанием всех аргументов.

2) Скомпилировать его с помощью Ascend Tensor Compiler в gemm.om.

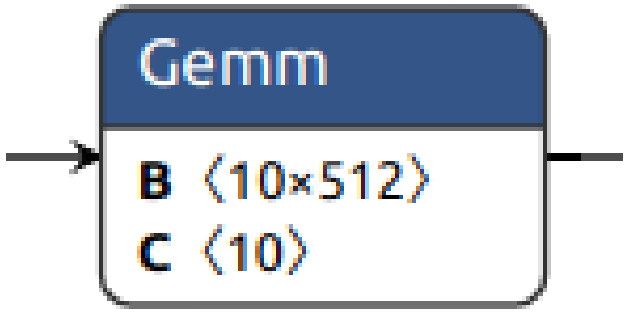
3) Перед вызовом aclblasGemmEx() вызвать
   aclopSetModelDir(<каталог с gemm.om>)

Визуализация ONNX модели ResNet18

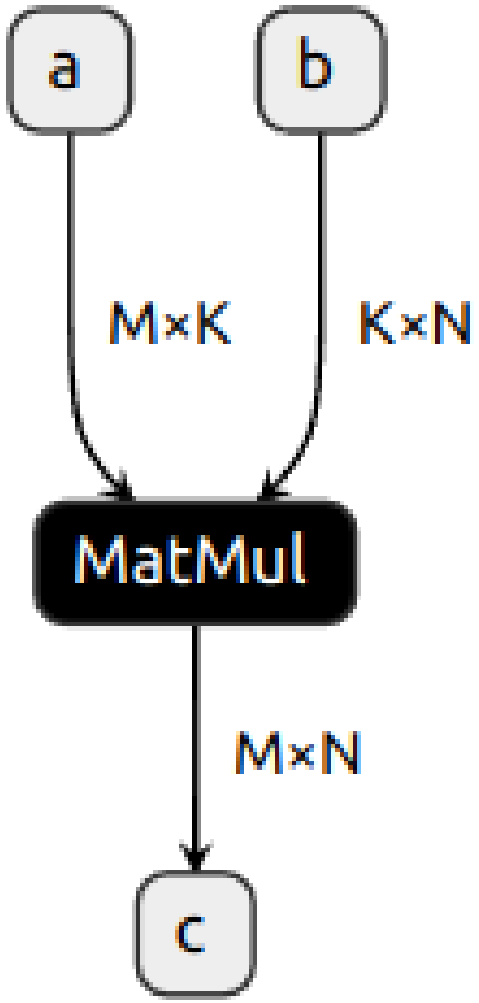




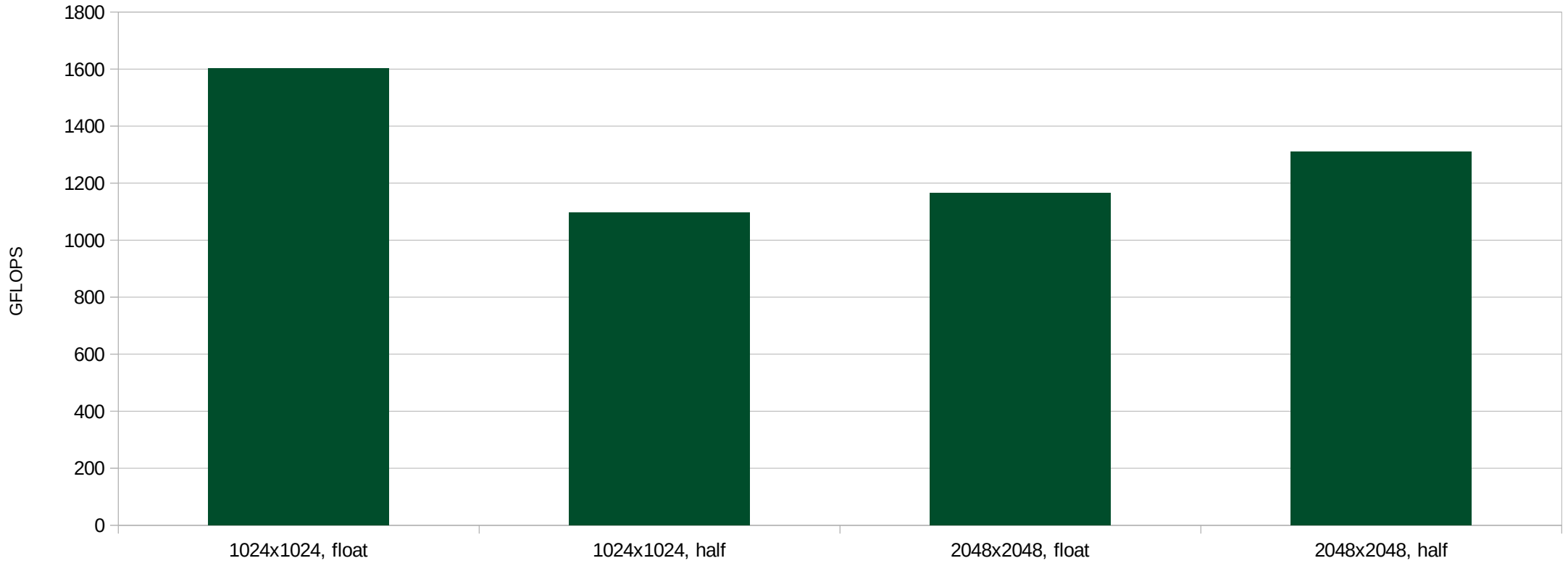
Да это же GEMM,
произведение матриц!



Наша нейросеть будет выглядеть так

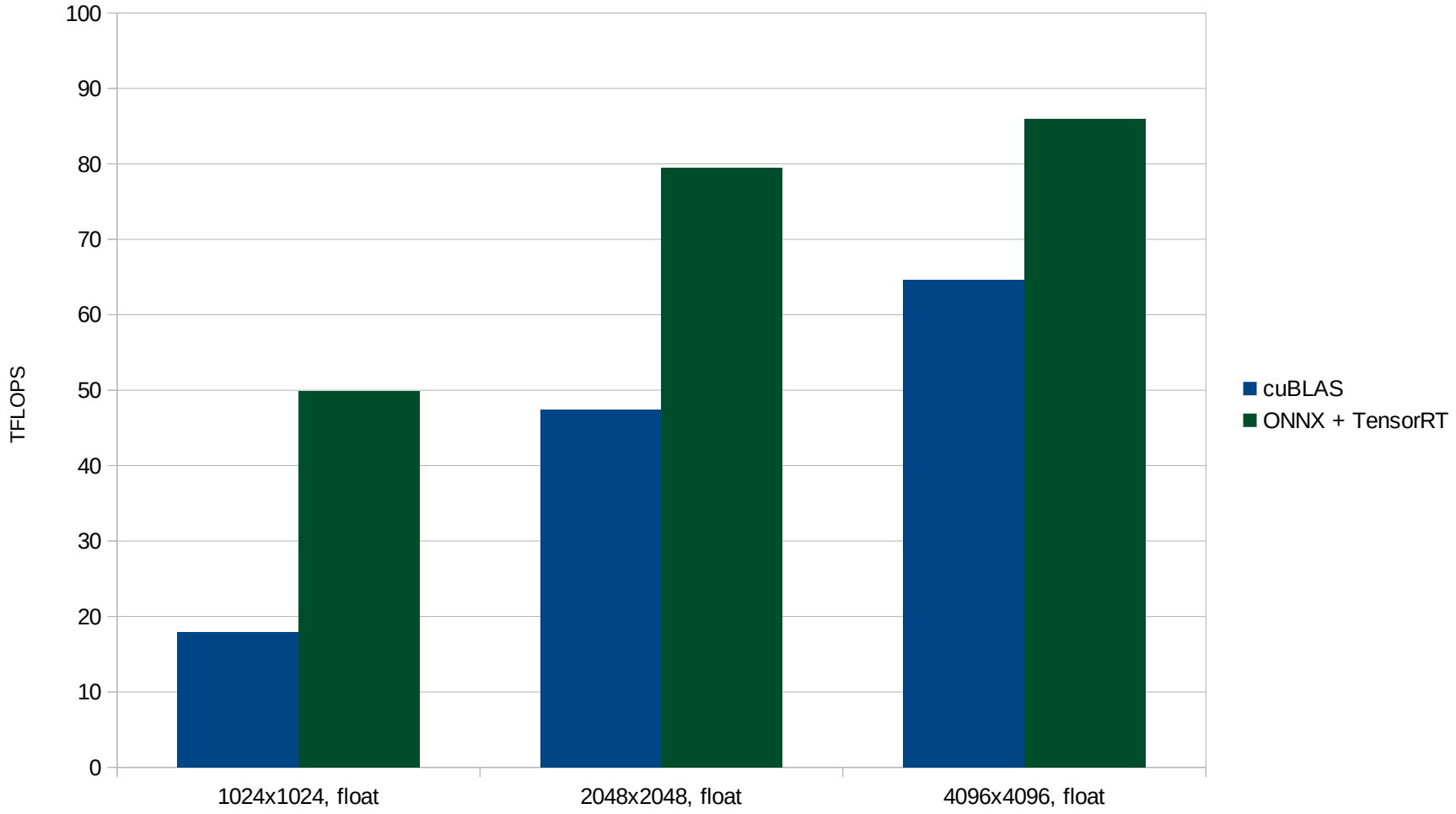


NPU Huawei Ascend 310, 8 Вт



164-200 GFLOPS/W

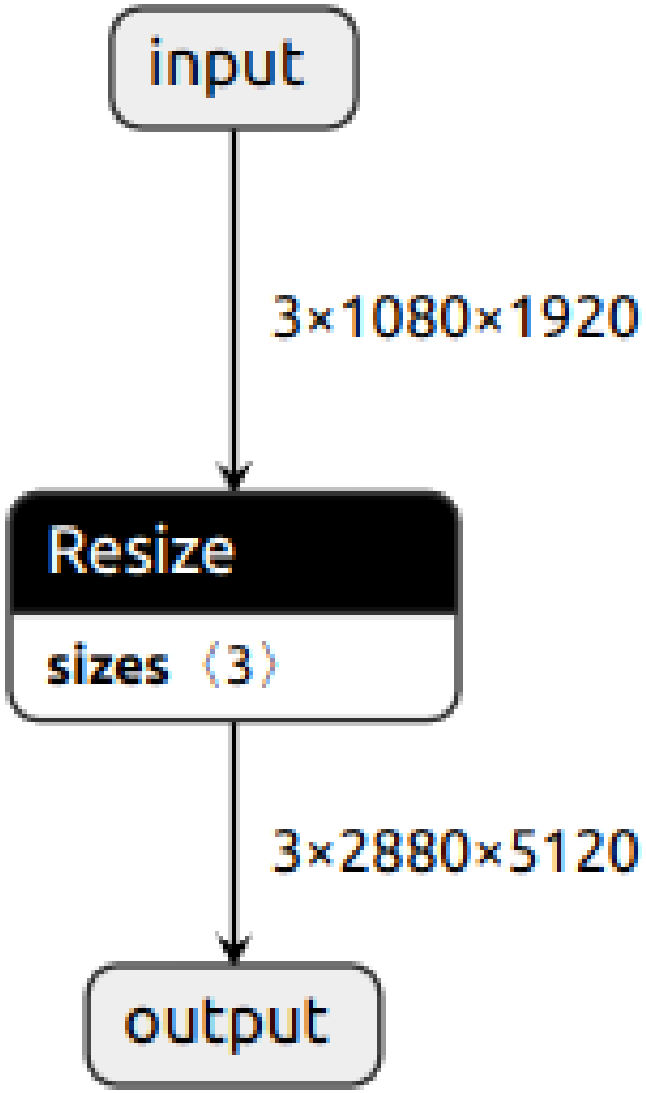
Запуск на Nvidia GeForce RTX 4090



111-191 GFLOPS/W

Производительность не хуже математических библиотек.

Upscale из FullHD в 5K



Что может ONNX?

- Всё, что могут нейросети.
- Поддержка вещественных и целых типов.
- Линейная алгебра — перемножение тензоров.
- Криптография — побитовые операции.
- Обработка изображений — декодер, ресэмплинг, свёртки.
- Обработка сигналов — свёртки, FFT.

А код для запуска модели для каждой платформы свой?

Можно использовать «родные» API фрэймворков:

- Huawei CANN
- Nvidia TensorRT
- oneDNN
- ...

А можно...

Один ONNX Runtime для всего

CPU	GPU	IoT/Edge/Mobile	Other
Default CPU	NVIDIA CUDA	Intel OpenVINO	Rockchip NPU (*preview*)
Intel DNNL	NVIDIA TensorRT	ARM Compute Library (*preview*)	Xilinx Vitis-AI (*preview*)
TVM (*preview*)	DirectML	Android Neural Networks API	Huawei CANN (*preview*)
Intel OpenVINO	AMD MIGraphX	ARM-NN (*preview*)	AZURE (*preview*)
XNNPACK	Intel OpenVINO	CoreML (*preview*)	
	AMD ROCm	TVM (*preview*)	
	TVM (*preview*)	Qualcomm QNN	
		XNNPACK	

Запуск ONNX через ONNX Runtime

- Создаём объект **Ort::Env env**.
- Создаём объект сессию из ONNX файла **Ort::Session session**.
- Создаём входные тензоры с помощью **Ort::Value::CreateTensor()**.
- Запускаем инференс и получаем выходные тензоры с помощью **session.Run()**.

Полный пример запуска ONNX:

https://github.com/tmagomedov/onnx-misuse/blob/main/ort_runner/ort_runner.cpp

ONNX Runtime — запрос Execution Provider’ов


```cpp
// ONNX Runtime environment.
Ort::Env env;

std::cout << "Available execution providers:" << std::endl;
std::vector<std::string> providers = Ort::GetAvailableProviders();
bool has_cuda = false;
bool has_tensorrt = false;
for (size_t i = 0; i < providers.size(); i++)
{
    std::cout << "  " << i << ". " << providers[i] << std::endl;
    if ("CUDAExecutionProvider" == providers[i])
    {
        has_cuda = true;
    }
    if ("TensorrtExecutionProvider" == providers[i])
    {
        has_tensorrt = true;
    }
}
```

ONNX Runtime — специфичные опции сессии


```
Ort::SessionOptions session_options;
if (has_tensorrt)
{
    OrtTensorRTProviderOptions trt_options = {};
    trt_options.device_id = 0;
    // TensorRT builds own models from ONNX,
    // caching them increases the performance of first session.Run().
    trt_options.trt_engine_cache_enable = 1;
    trt_options.trt_engine_cache_path = "./";
    session_options.AppendExecutionProvider_TensorRT(trt_options);
}
else if (has_cuda)
{
    OrtCUDAProviderOptions cuda_options = {};
    cuda_options.device_id = 0;
    session_options.AppendExecutionProvider_CUDA(cuda_options);
}
```

ONNX Runtime — создание сессии


```
// Enable built-in ONNX Runtime profiler information dump to json file.
// Json can be inspected using about:tracing tab in chrome
// based browsers.
session_options.EnableProfiling(argv[0]);

Ort::Session session{env, onnx_fname, session_options};
```

ONNX Runtime — создание входных тензоров


```
// Where to allocate the tensors.
auto memory_info = Ort::MemoryInfo::CreateCpu(OrtDeviceAllocator, OrtMemTypeCPU);

// Allocate input tensors.
std::vector<Ort::Value> input_tensors;
std::vector<std::vector<char>> input_data;
for (size_t i = 0; i < inputs_count; i++)
{
    ONNXTensorElementDataType t =
                        session.GetInputTypeInfo(i).GetTensorTypeAndShapeInfo().GetElementType();
    size_t input_size = get_type_size(t);
    for (auto x : input_shapes[i])
    {
        input_size *= x;
    }
    input_data.emplace_back(input_size);
    // Creates tensor with supplied buffer, need to keep input_data in memory while it is alive.
    input_tensors.push_back(Ort::Value::CreateTensor(memory_info, input_data.back().data(),
                                                     input_data.back().size(), input_shapes[i].data(),
                                                     input_shapes[i].size(), t));
}
```

ONNX Runtime — запуск модели


```
// Run the inference.
std::cout << "Running inference ... " << std::flush;
std::vector<Ort::Value> output_tensors = session.Run(
    Ort::RunOptions{nullptr},
    input_names_c.data(),
    input_tensors.data(),
    input_tensors.size(),
    output_names_c.data(),
    output_names_c.size());
```

Запуск ONNX на данных из базы данных

Расширение для PostgreSQL — pg_onnx

Интеграция ONNX Runtime в Postgres.

https://github.com/kibae/pg_onnx

```
SELECT pg_onnx_import_model(
               'sample_model', ---------------- model name
               'v20230101', ------------------- model version
               PG_READ_BINARY_FILE('/your_model_path/model.onnx')::bytea, -- model binary data
               '{"cuda": true}'::jsonb, ------ options
               'sample model' ----------------- description
       );

SELECT pg_onnx_execute_session(
               'sample_model', -- model name
               'v20230101', ----- model version
               '{
                 "x": [[1], [2], [3]],
                 "y": [[3], [4], [5]],
                 "z": [[5], [6], [7]]
               }' --------------- inputs
       );
```

А как создавать ONNX файлы?

- С помощью визуальных редакторов (в разработке).
- Пакеты onnx.helper, ONNX GraphSurgeon (Python).
- Модель «нейросети» в любимом ML фрэймворке + экспорт в ONNX.
- На любом языке, поддерживающем Protocol Buffers (в том числе C++).

Пример создания ONNX:

https://github.com/tmagomedov/onnx-misuse/tree/main/matmul

А минусы есть?

- Нужных операций может не быть в ONNX.
- Нет полноты по Тьюрингу — решаемо кодом на C++.
- Конкретные ONNX операции могут плохо поддерживаться бэкендом.
- У каждого бэкенда свои баги.
- Производительность бывает непредсказуема.

Краткий итог

- ONNX — формат хранения обученных нейросетей.
- Ациклический граф вычислений, сериализованный в Protobuf.
- Есть эффективные рантаймы для запуска на CPU, GPU, NPU.
- Операции ONNX можно использовать как библиотеку оптимизированных вычислительных примитивов.

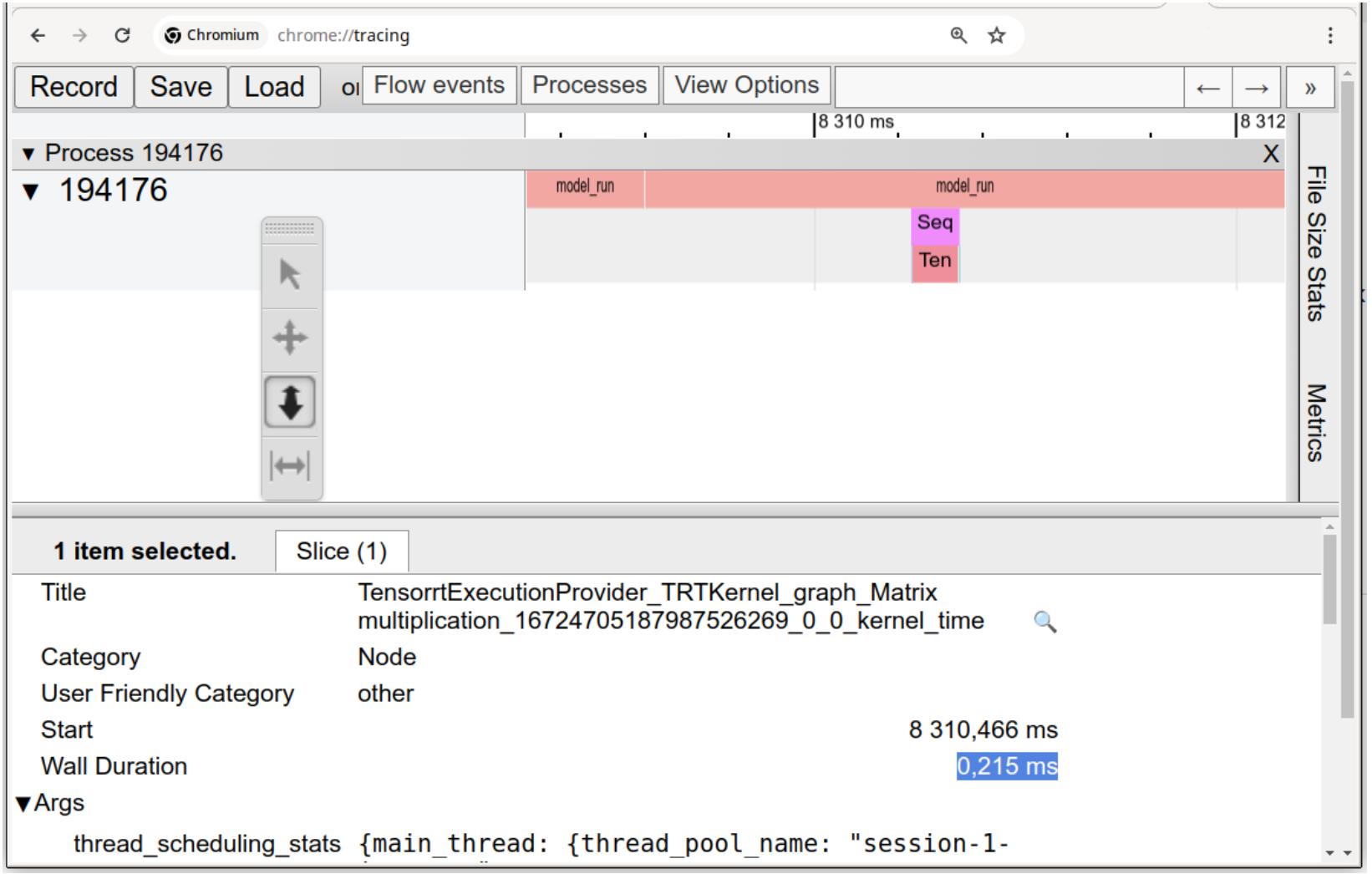
Список ссылок

- https://onnx.ai/onnx/operators/
- https://onnxruntime.ai/
- https://github.com/tmagomedov/onnx-misuse
- https://t.me/gpgpu_ru_chat

PostgresPro

Вопросы?

Как это профилировать?



В опциях ONNX Runtime сессии можно включить профилирование.