Искусственный интеллект глазами разработчика

Конференция
для IT-специалистов,
интересующихся темой
Artificial Intelligence

МИР

МИЛАНДР
ГРУППА КОМПАНИЙ

Module
НАУЧНО-ТЕХНИЧЕСКИЙ ЦЕНТР

НовГУ им.
Ярослава Мудрого

Искусственный интеллект глазами разработчика

29 мая 2021

Мастерская инструментов разработки
mir.dev

Компилятор для нейронных сетей на базе фреймворка Apache TVM

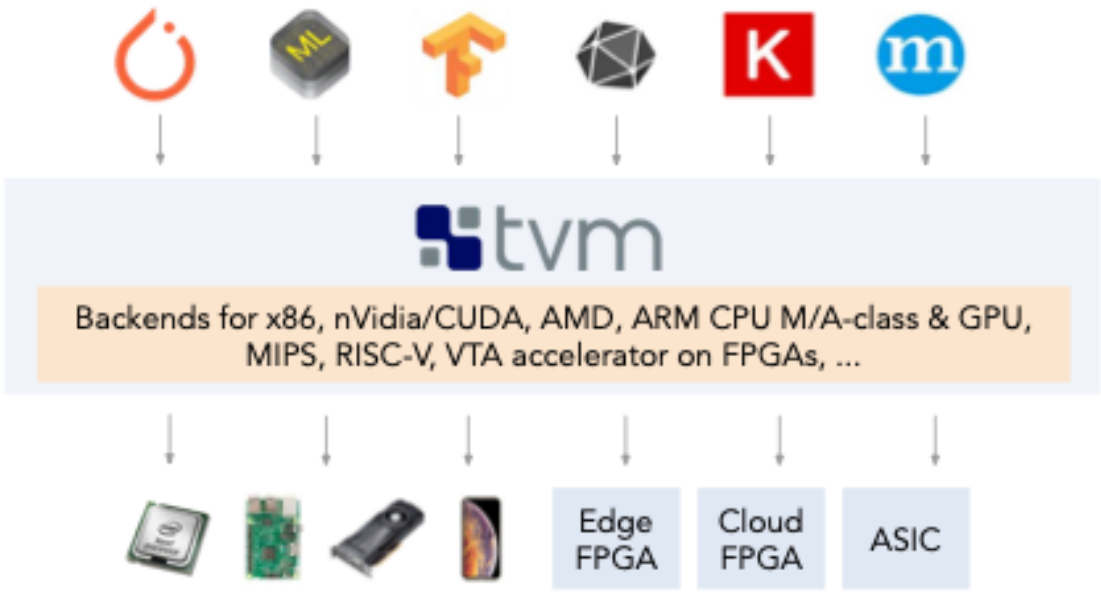
Создание бэкенда для тензорного ускорителя

Илья Гозман

Архитектор, компания МИР

Выбор TVM

- Открытый исходный код и лицензия Apache
- Активное большое комьюнити
- Участие в разработке крупных компаний (Amazon, ARM)
- Поддержка различных фронт- и бэкендов
- Python- и C- API
- Достаточная гибкость и набор инструментов для нетипичных архитектур
- Разработка под Linux и Windows, сборка через CMake



Внутренние представления

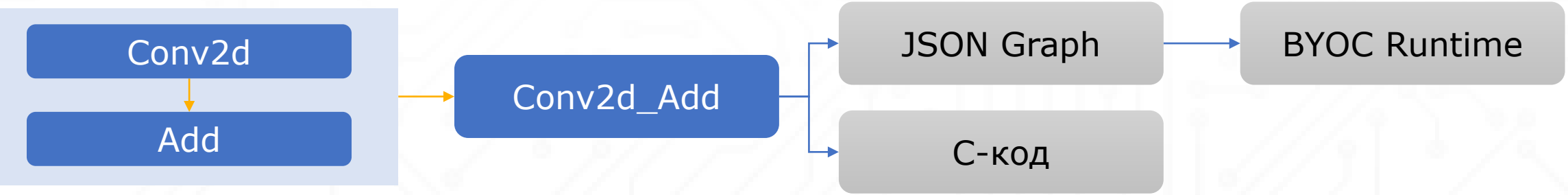
Вход – Tensorfow, ONNX, PyTorch, …

- Преобразует сеть в собственный формат Relay
- Преобразования на уровне графа, поиск и выделение сабграфов, изменение формы тензоров (layout)
- Общие оптимизации на уровне слоя, внедрение интринсик
- Генерация целевого представления для TIR-сабграфов (nodes), например LLVMIR, C-код, .so библиотека

Виды бэкендов

BYOC (Bring Your Own Codegen)

Бэкенд для Relay-компилятора, позволяет заменять операции или слои на составные device-функции



LLVM

Использование инфраструктуры LLVM (через API) для получения объектных файлов или биткода:

ADMGPU, ARM, Mali, X86 (_64), Hexagon, NVPTX

Виды бэкендов

Source

Генерация исходного кода для использования с внешним компилятором (например NVCC или NVRTC для CUDA):

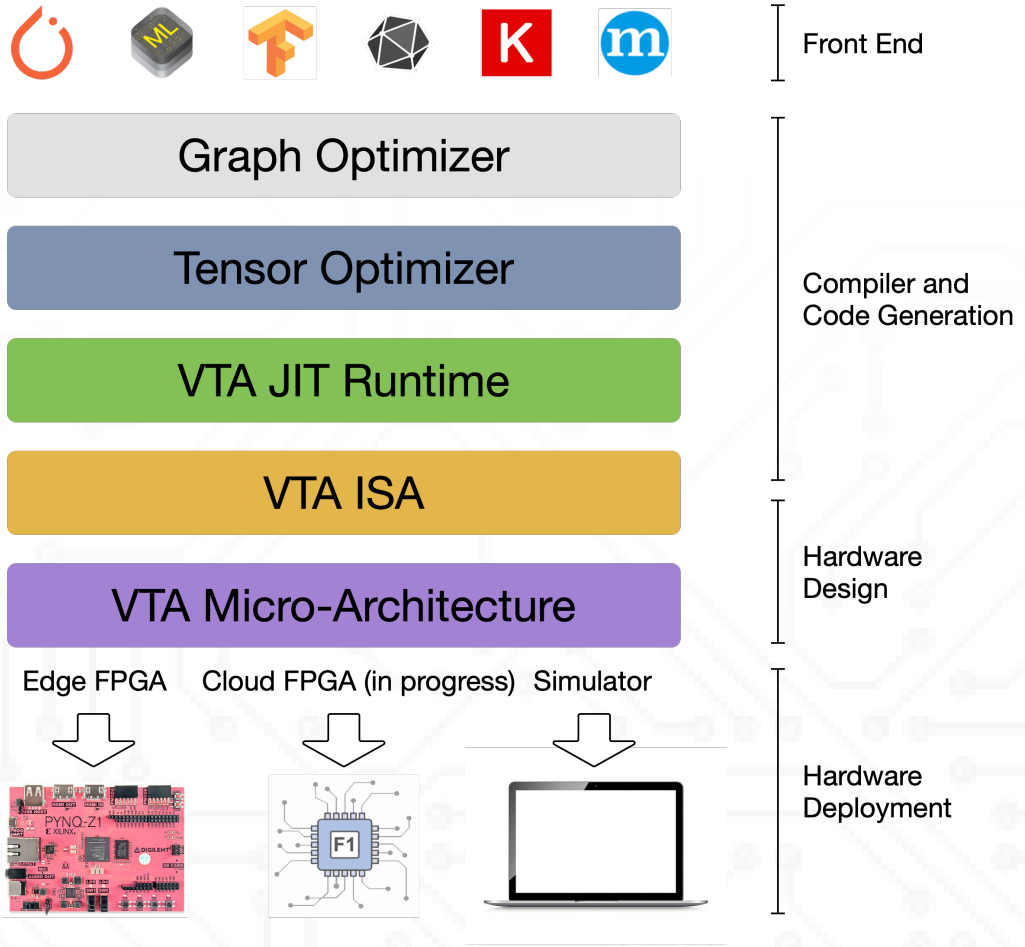
C, CUDA, METAL, OpenCL, Vivado HLS

Custom

Использование в планировщике механизма tensorize:

- Внедрение интринсик
- Использование внешних вызовов
- Вставки на целевом ассемблере или биткоде

VTA (Versatile Tensor Accelerator) бэкенд

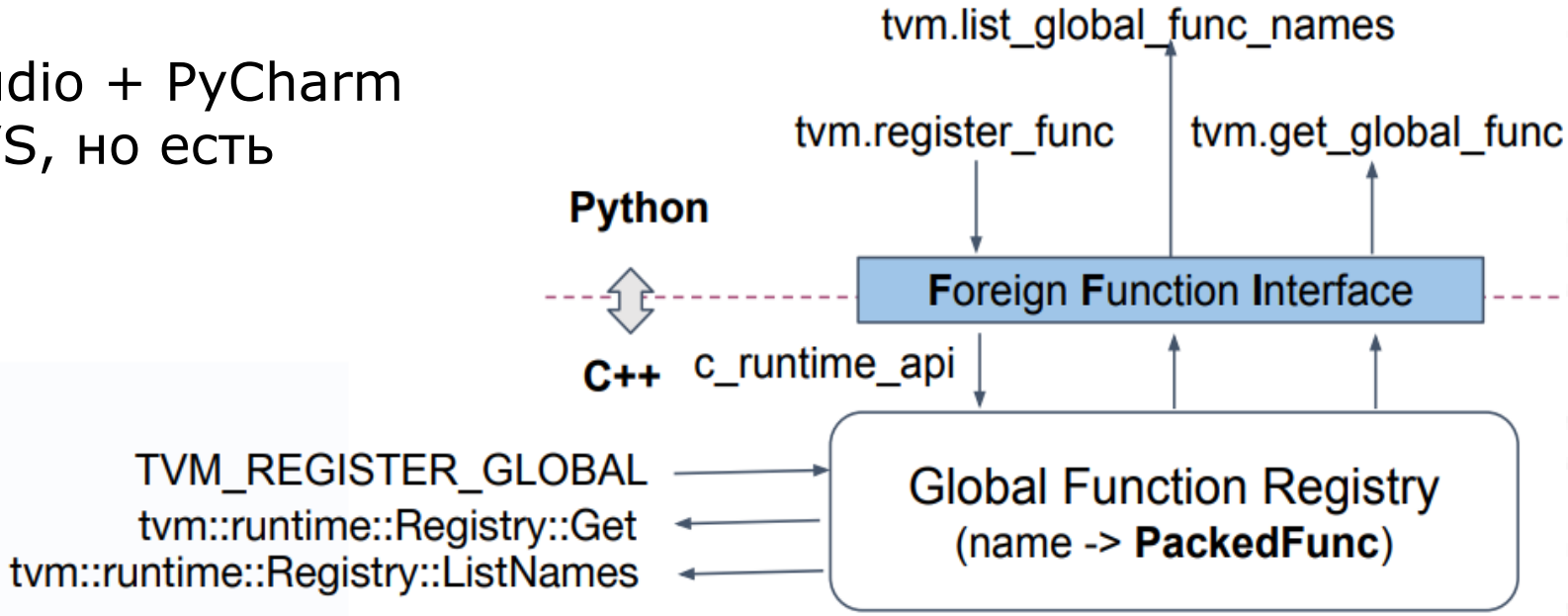


- Дизайн ускорителя операций линейной алгебры (модули GEMM, ALU)
- Реализации на Vivado HLS C++, Chisel, OpenCL
- Симулятор TSIM на базе Verilator
- Готовая интеграция в TVM

Особенности разработки и отладки

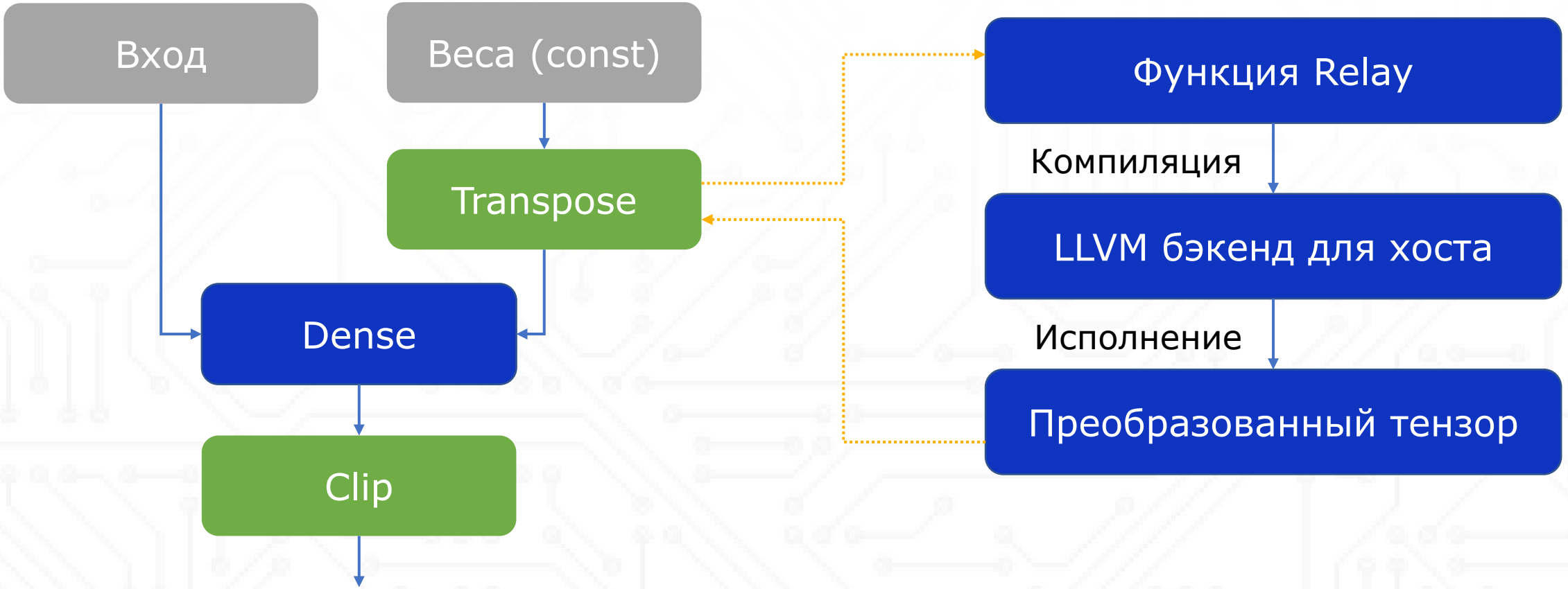
TVM PackedFunc (FFI) – универсальный механизм функций, для реализации и вызова из Python/C++/C кода
Используется в компиляторе и в runtime

Для Windows – Visual Studio + PyCharm
Можно использовать PTVS, но есть проблемы с отладкой



JIT-компилятор

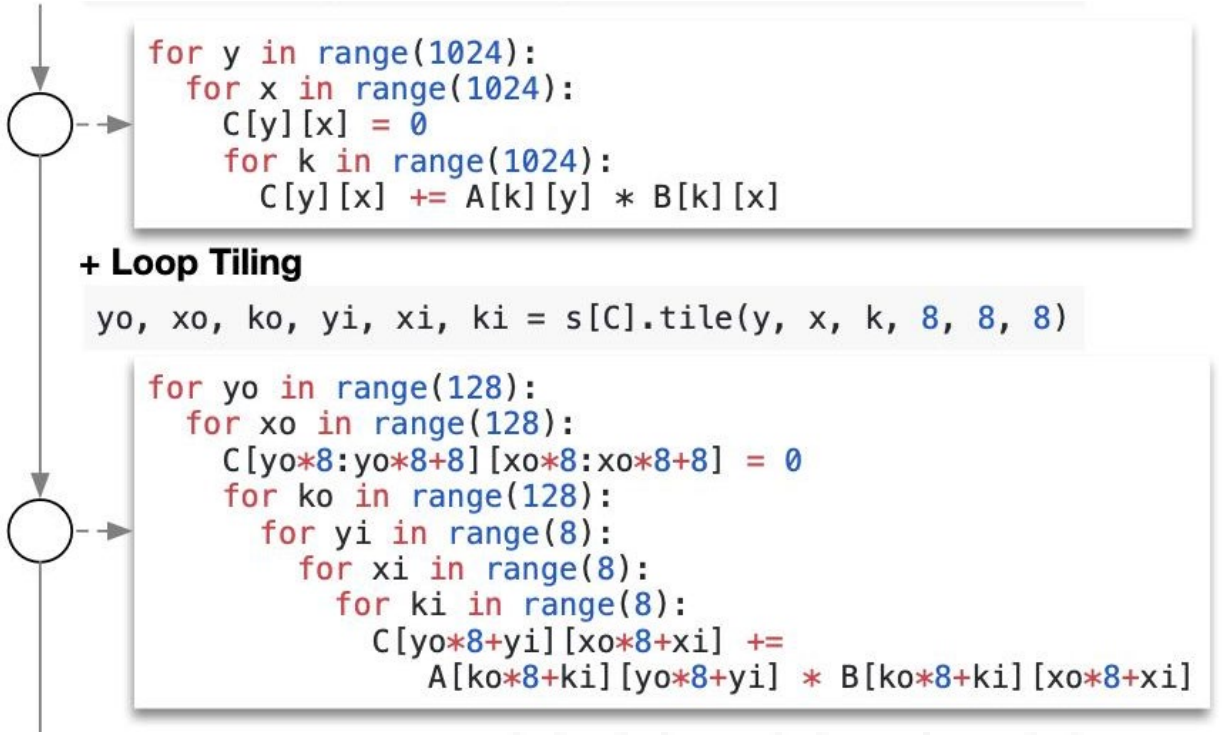
Преобразование констант или исполнение неподдерживаемых операций на хосте



Настройка планировщика

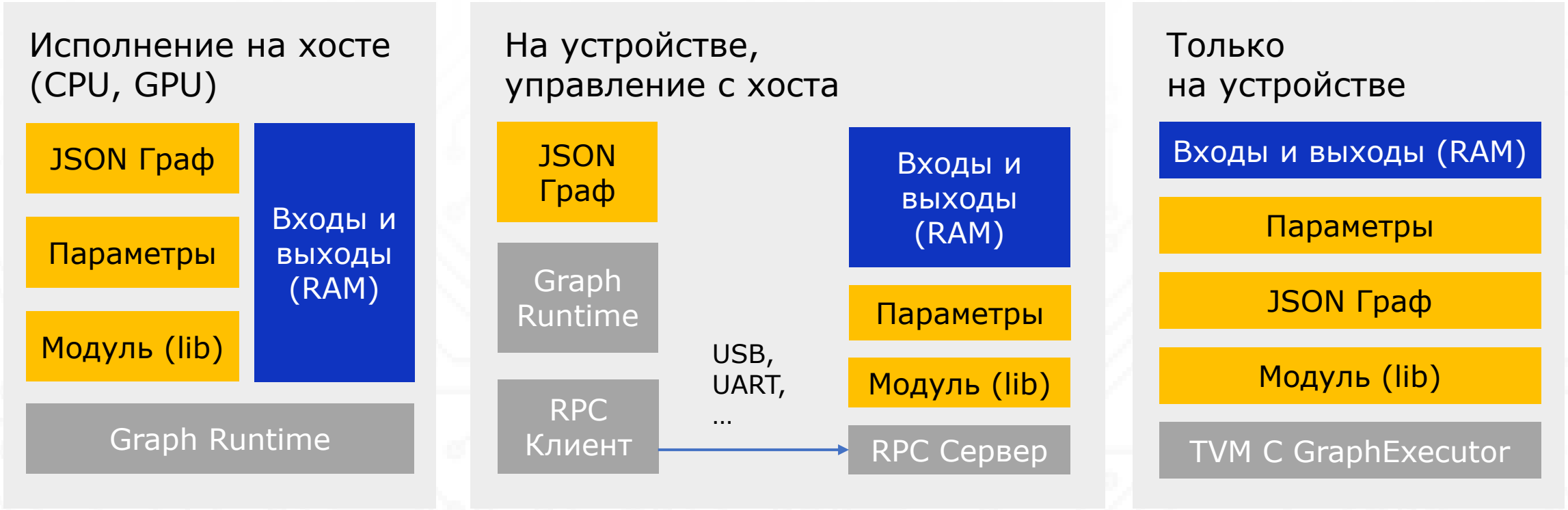
Возможности:

- **split, tile** – разделение по осям
- **fuse** – объединение осей
- **reorder** – изменение порядка вычислений по осям
- **bind** – распределение по виртуальным потокам
- **compute_at, compute_inline, compute_root** – перенос/ встраивание вычислительного цикла на другой уровень
- **tensorize** – замена фрагмента вычислений на вызов внешней функции



Runtime

- GraphRuntime (Graph Executor)
- uTVM (microTVM)



Искусственный интеллект глазами разработчика

29 мая 2021

Мастерская инструментов разработки

mir.dev

Вопросы?