ExecuTorch on Small Bare-Metal Microcontrollers

Shane Slattery
Pietra Ferreira
William Jones
Jeremy Bennett

The Cost of AI

50

The Cost of AI

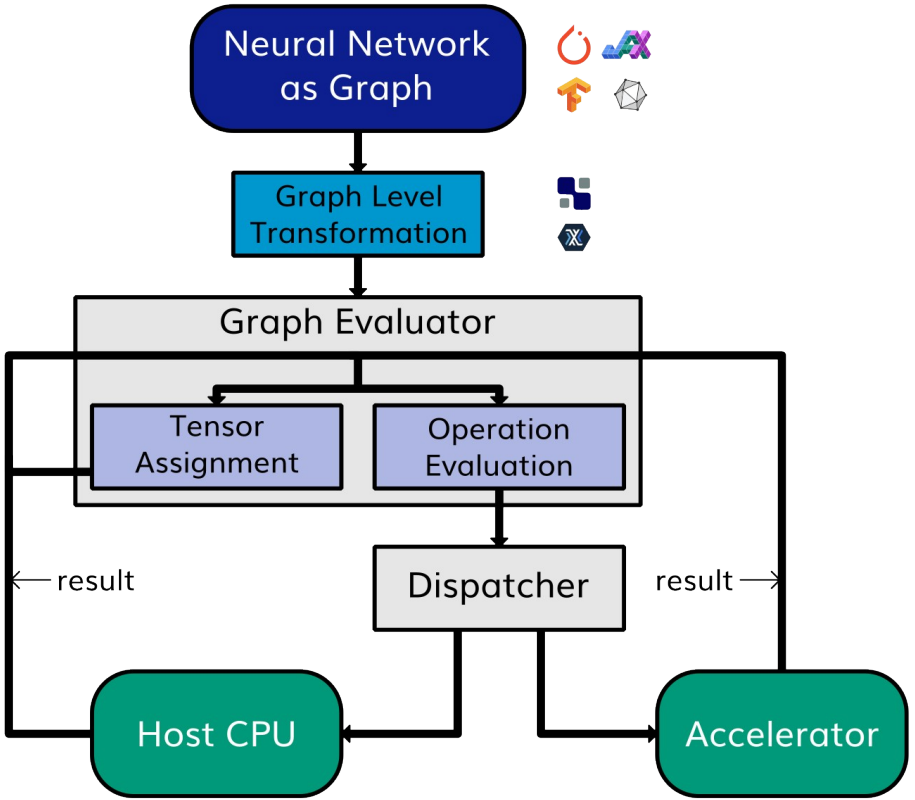
50 0.03

The Cost of AI

50 0.03 10-100+

Building AI at the Edge

What are PyTorch and ExecuTorch?



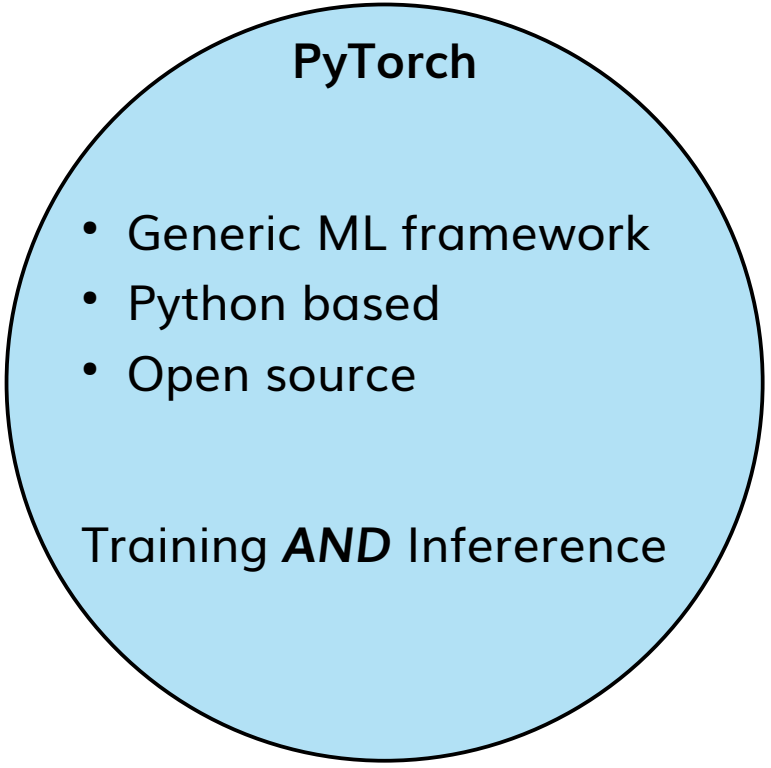
What are PyTorch and ExecuTorch?

What are PyTorch and ExecuTorch?

PyTorch

- Generic ML framework
- Python based
- Open source

What are PyTorch and ExecuTorch?



What are PyTorch and ExecuTorch?

PyTorch

- Generic ML framework
- Python based
- Open source

Training ***AND*** Infererence

ExecuTorch

- PyTorch extension
- Resource limited

What are PyTorch and ExecuTorch?

PyTorch

- Generic ML framework
- Python based
- Open source

Training ***AND*** Infererence

ExecuTorch

- PyTorch extension
- Resource limited

Infererence ***ONLY***

Two Steps to ExecuTorch Development

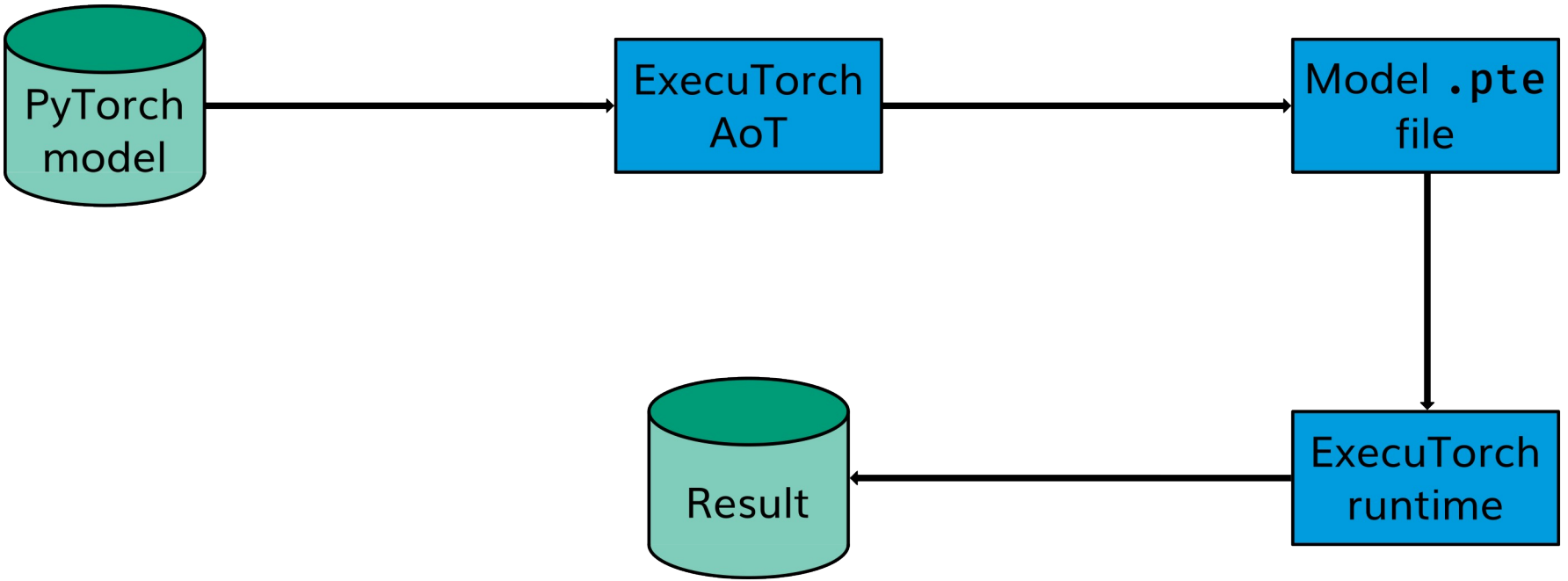
1. Build ExecuTorch

Two Steps to ExecuTorch Development

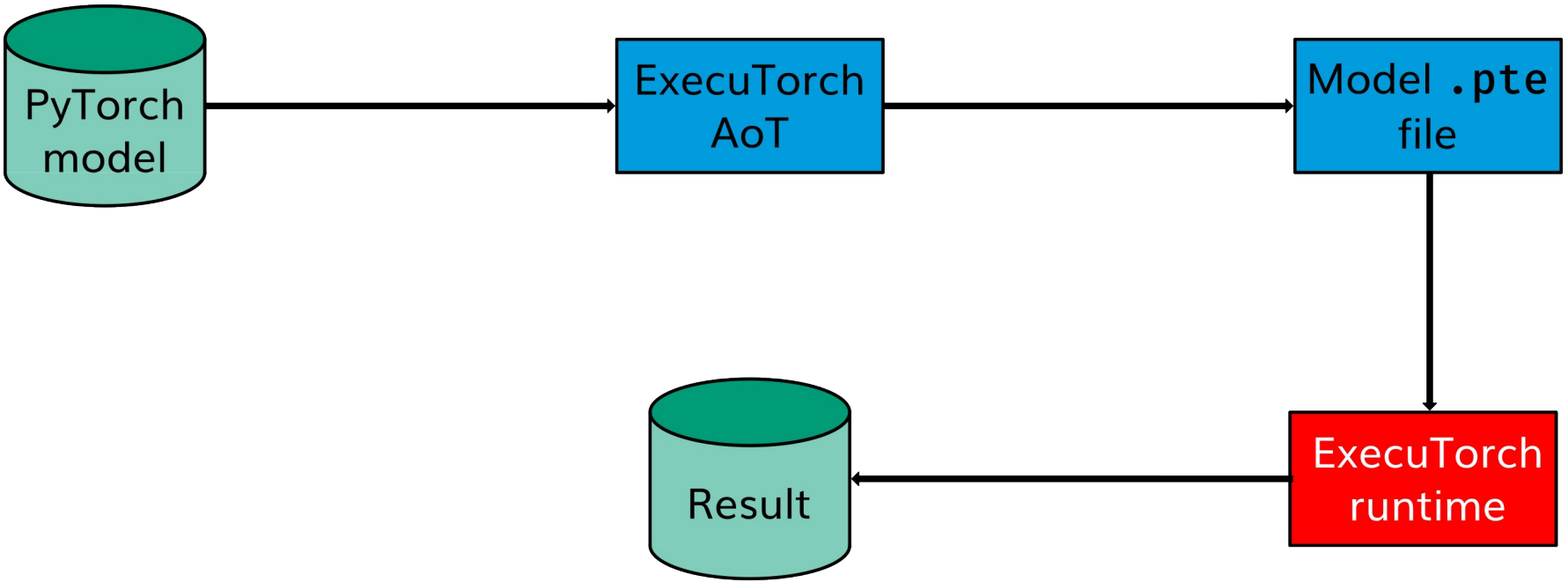
1. Build ExecuTorch

2. Customize Performance

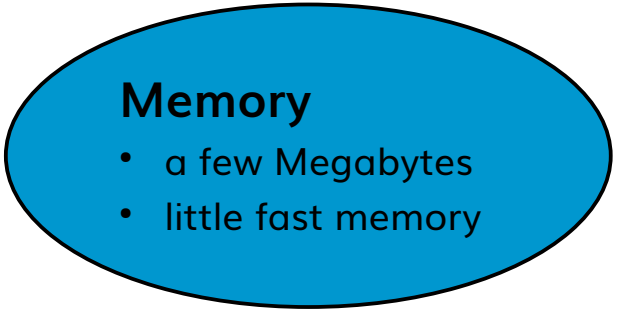
Building With ExecuTorch



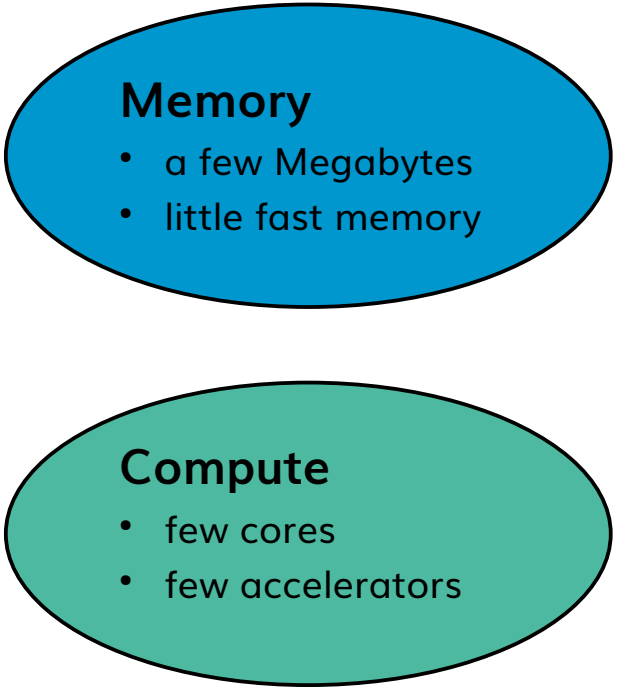
Building With ExecuTorch



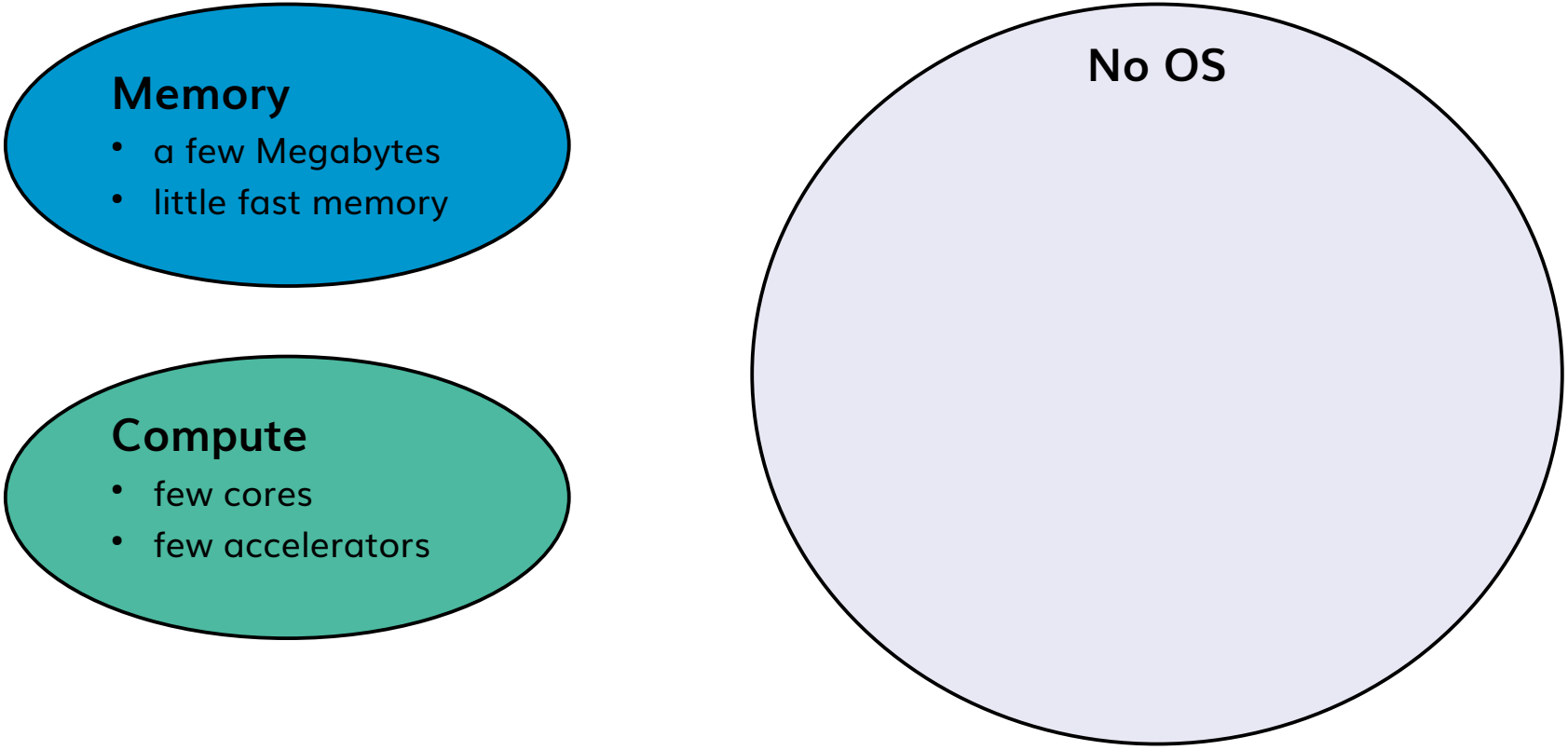
Our Experience: A RISC-V Platform



Our Experience: A RISC-V Platform



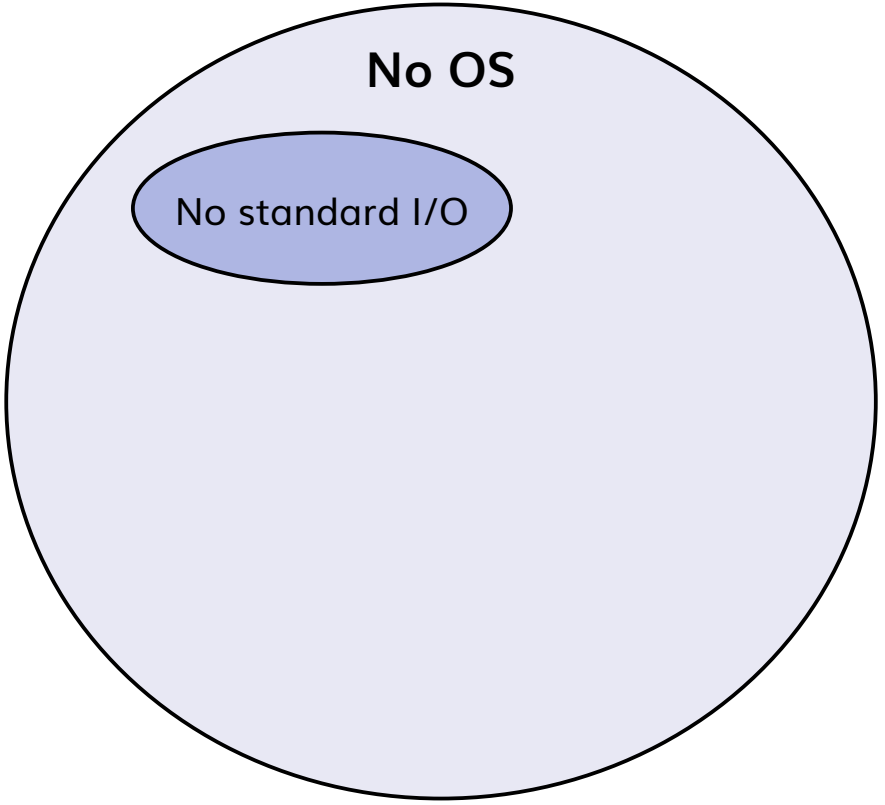
Our Experience: A RISC-V Platform



Our Experience: A RISC-V Platform

Memory
- a few Megabytes
- little fast memory

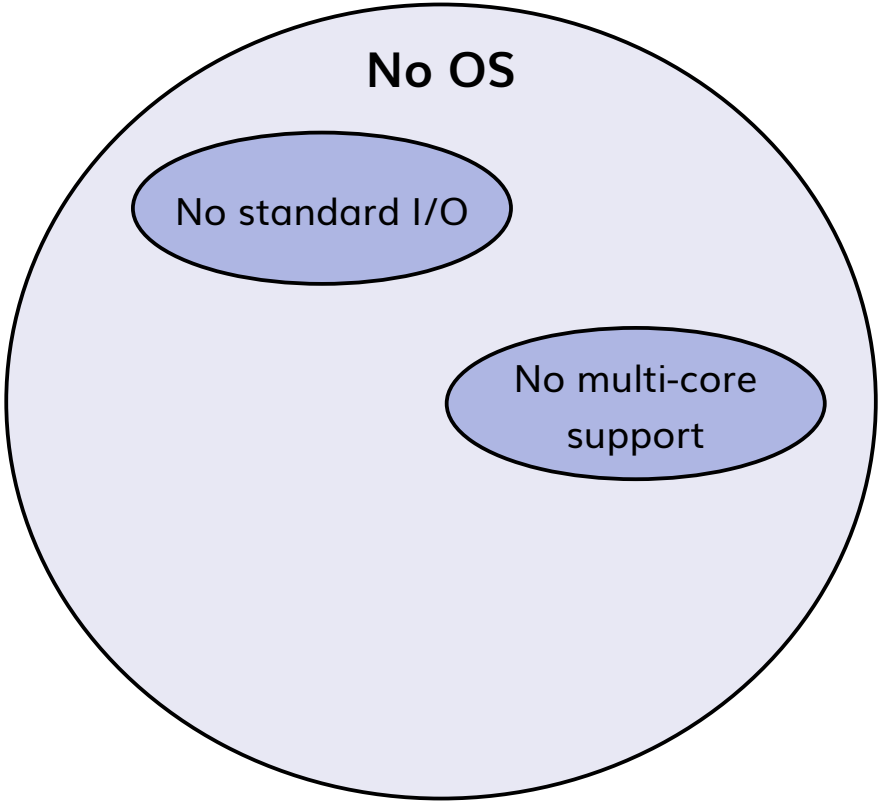
Compute
- few cores
- few accelerators



Our Experience: A RISC-V Platform

Memory
- a few Megabytes
- little fast memory

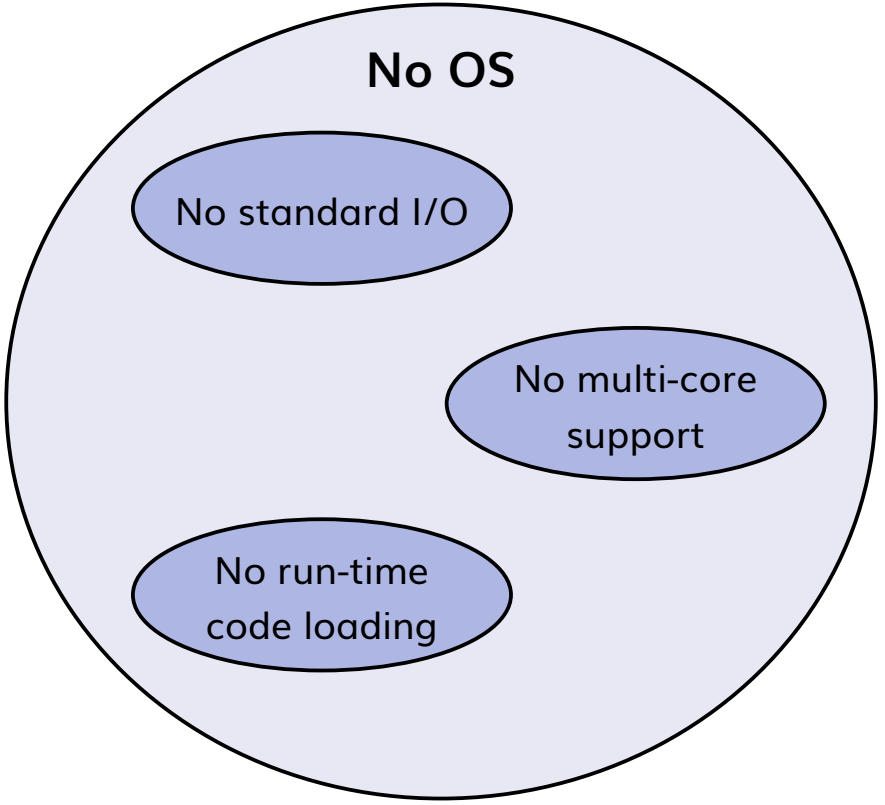
Compute
- few cores
- few accelerators



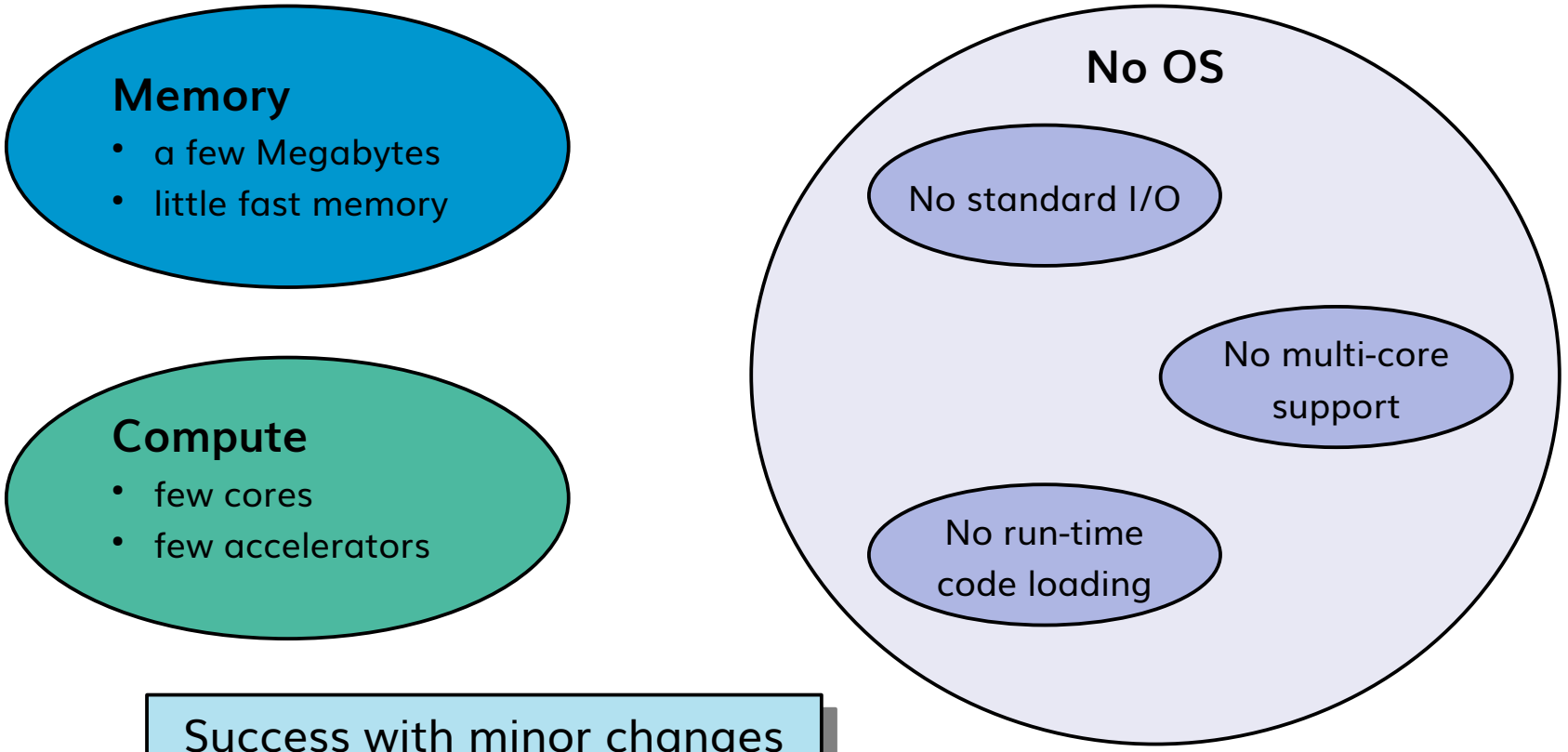
Our Experience: A RISC-V Platform

Memory
- a few Megabytes
- little fast memory

Compute
- few cores
- few accelerators

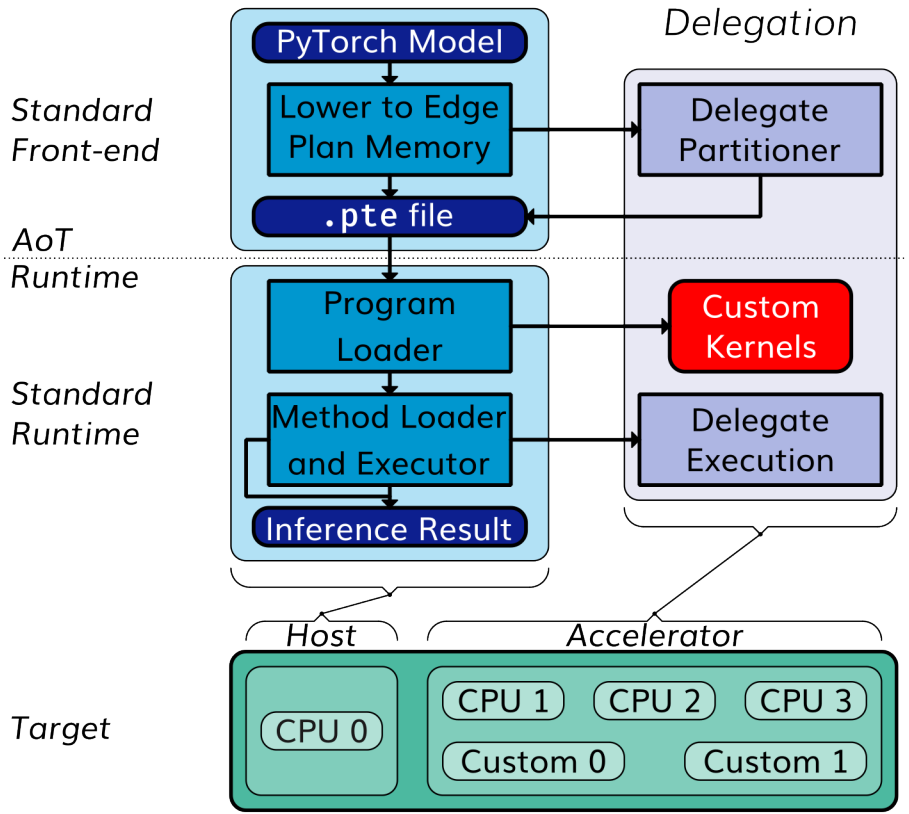


Our Experience: A RISC-V Platform



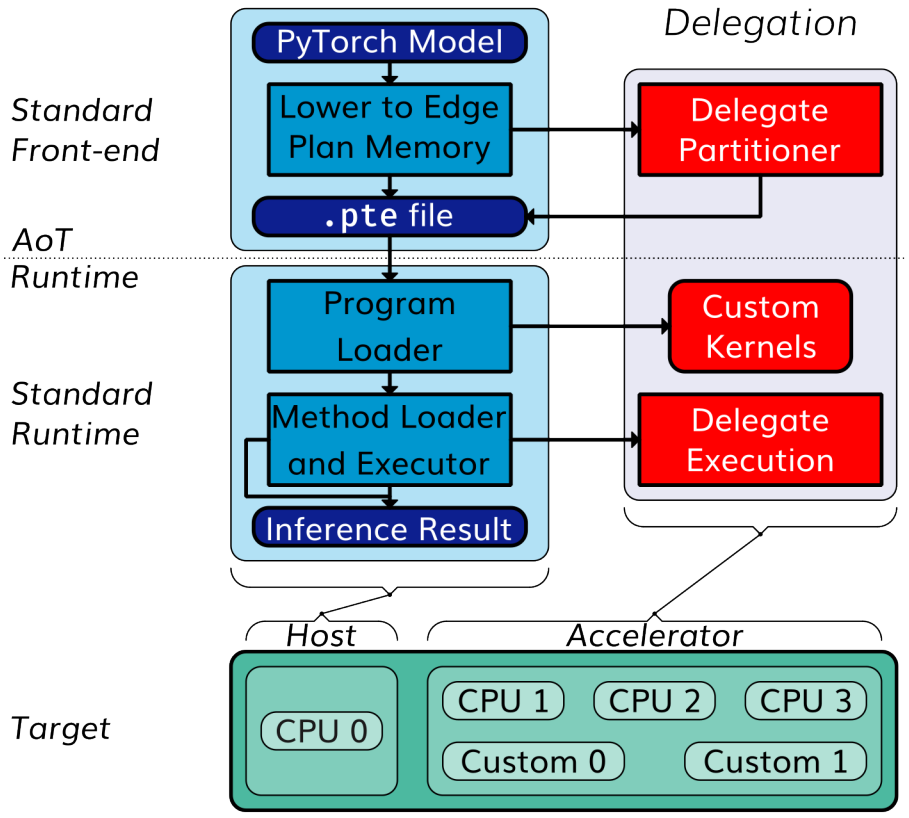
Success with minor changes

Customize Performance



Existing functions

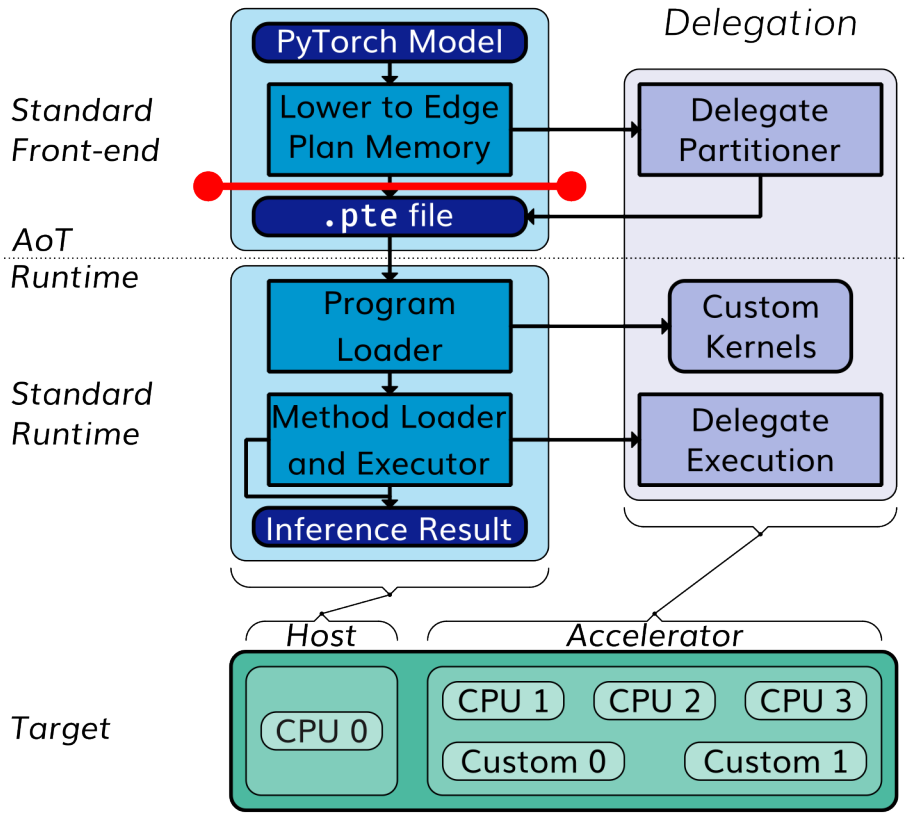
Customize Performance



Existing functions

New capabilities

Customize Performance



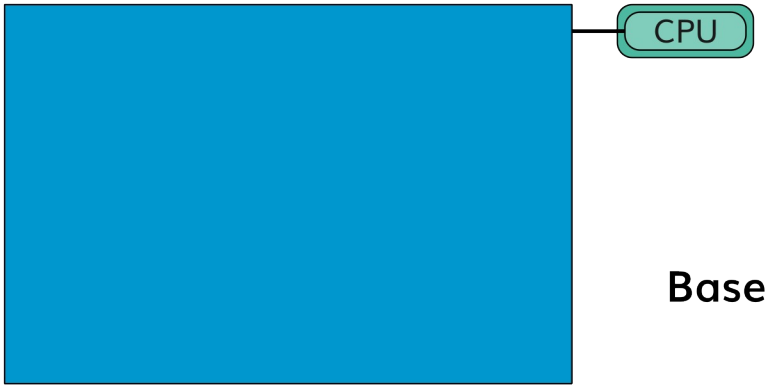
Existing functions

New capabilities

Graph level optimization

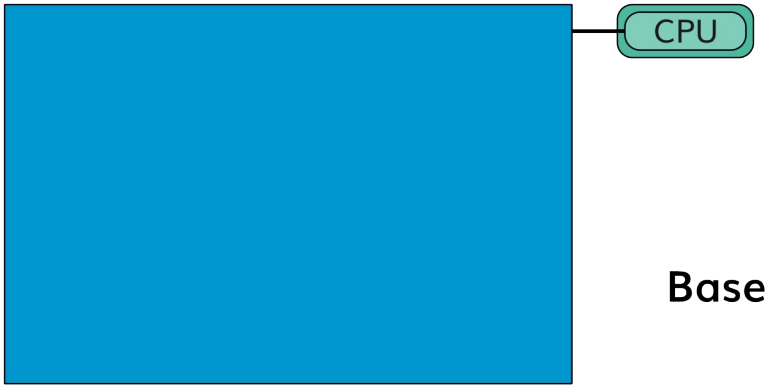
Case Study: A RISC-V Processor with Custom NPU

Optimization Strategies

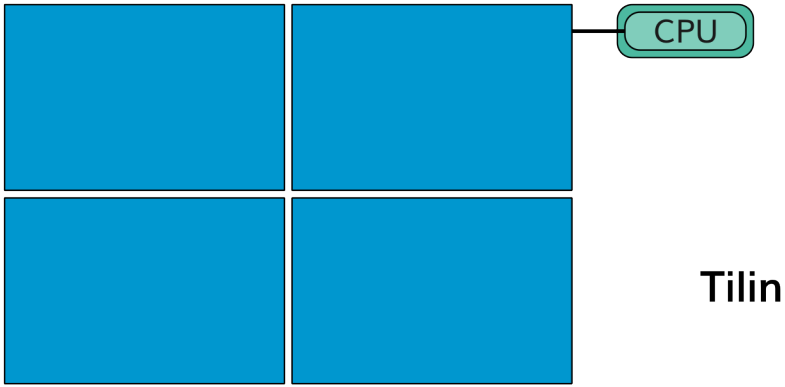


Baseline

Optimization Strategies

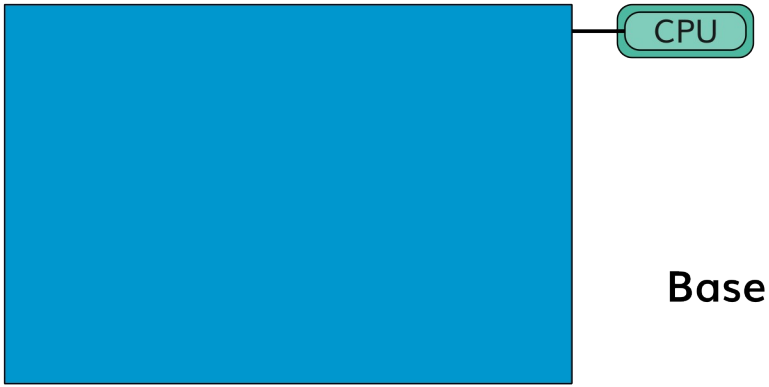


Baseline

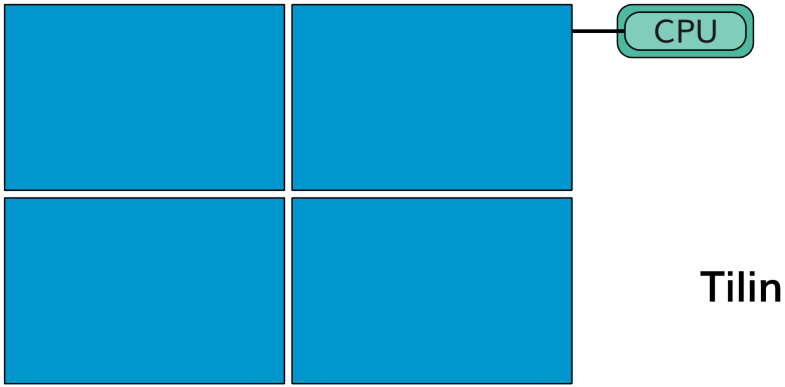


Tiling

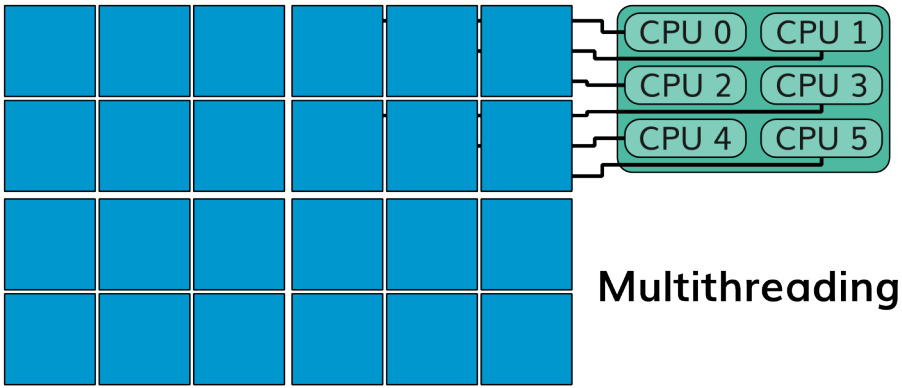
Optimization Strategies



Baseline

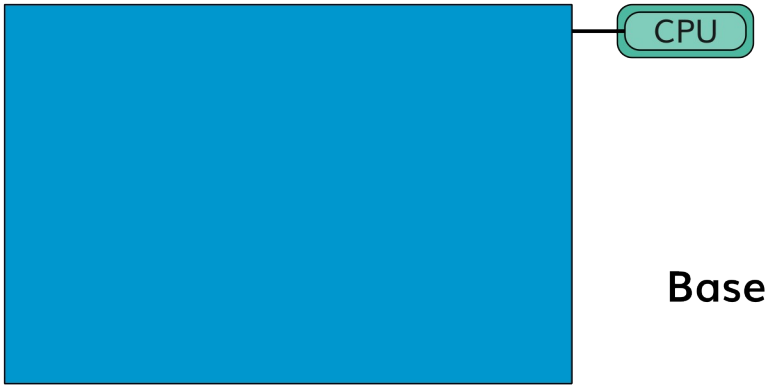


Tiling

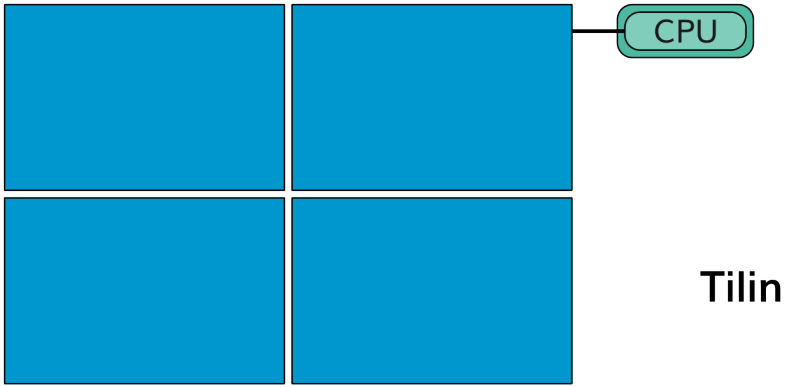


Multithreading

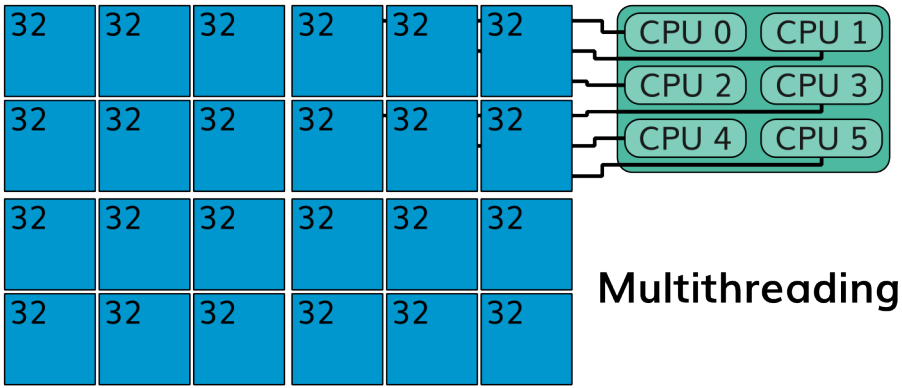
Optimization Strategies



Baseline

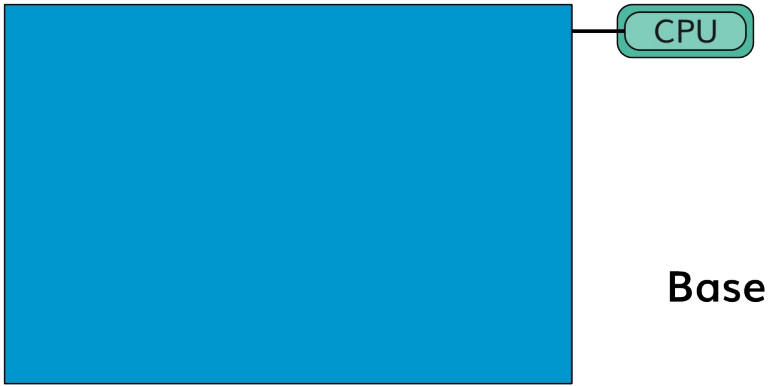


Tiling

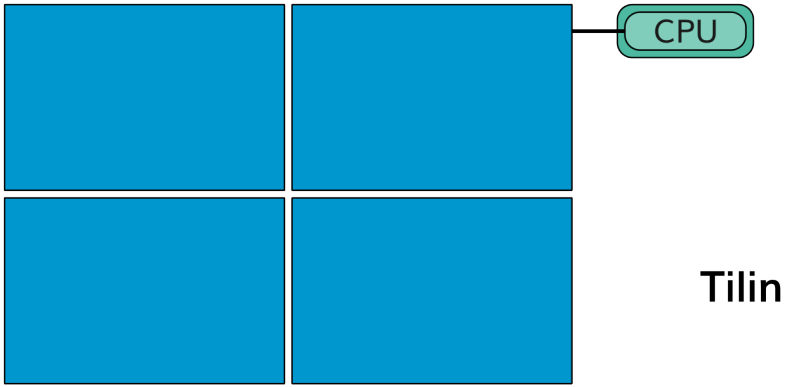


Multithreading

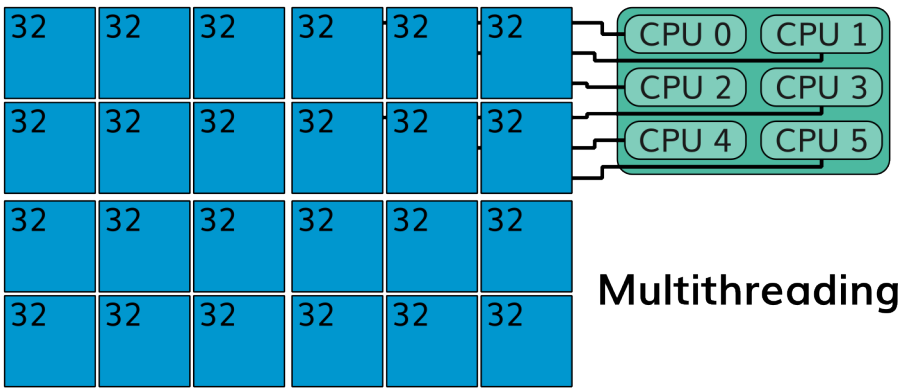
Optimization Strategies



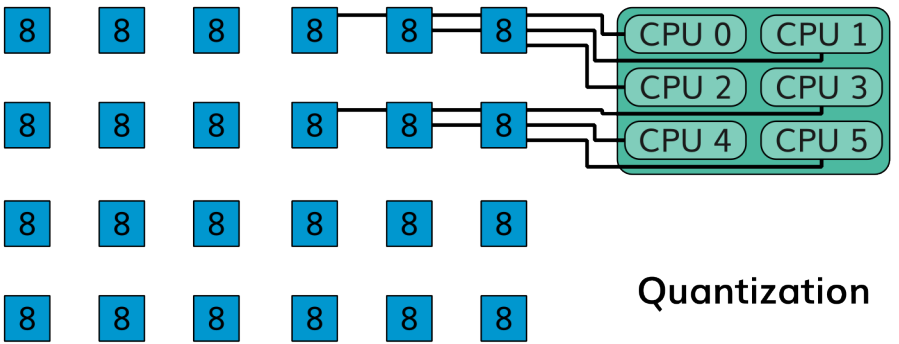
Baseline



Tiling

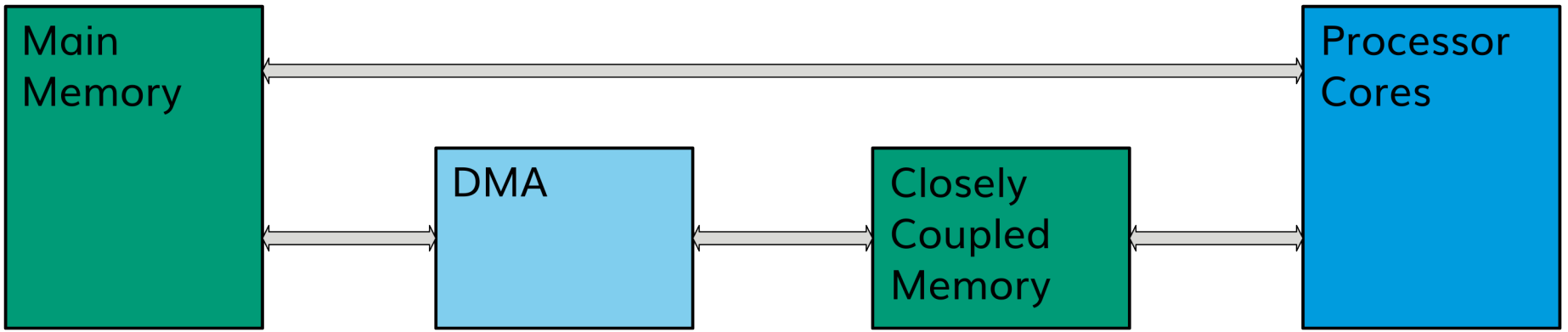


Multithreading



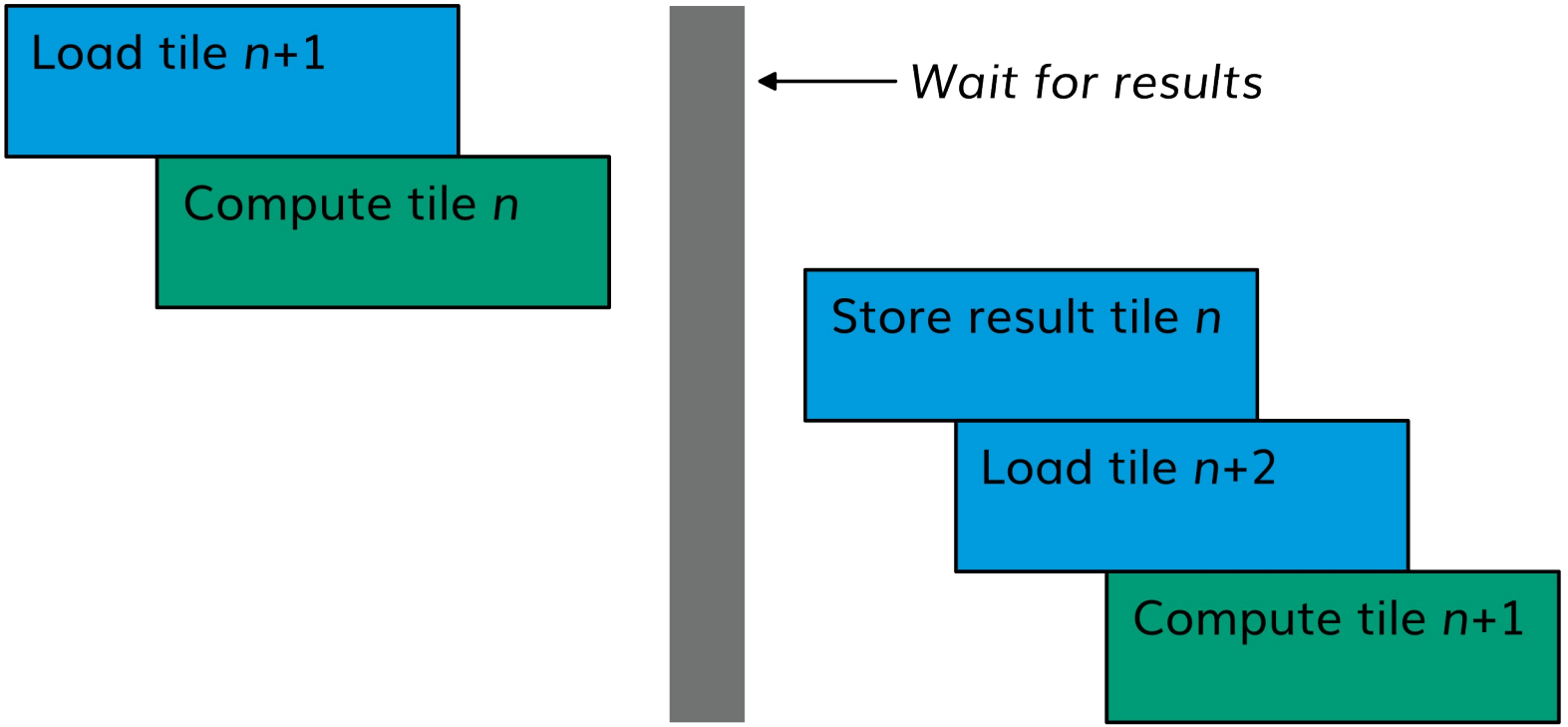
Quantization

Optimization in Depth: Memory

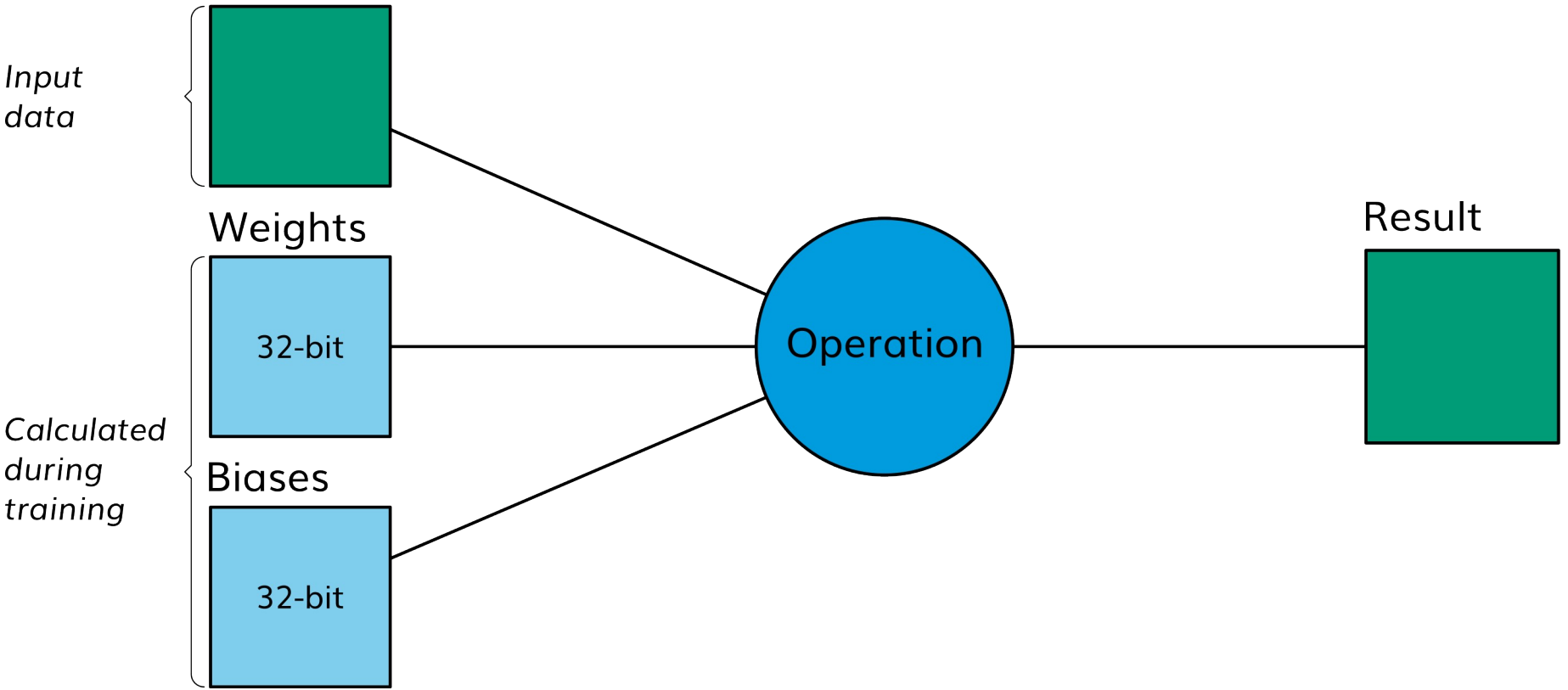


- Cache frequently used data
  - e.g. initial layers
- Break data into “tiles”
  - e.g. sub-areas of images
- Algorithms are predictable
  - regular loops
  - pre-fetch data
  - e.g. image convolution

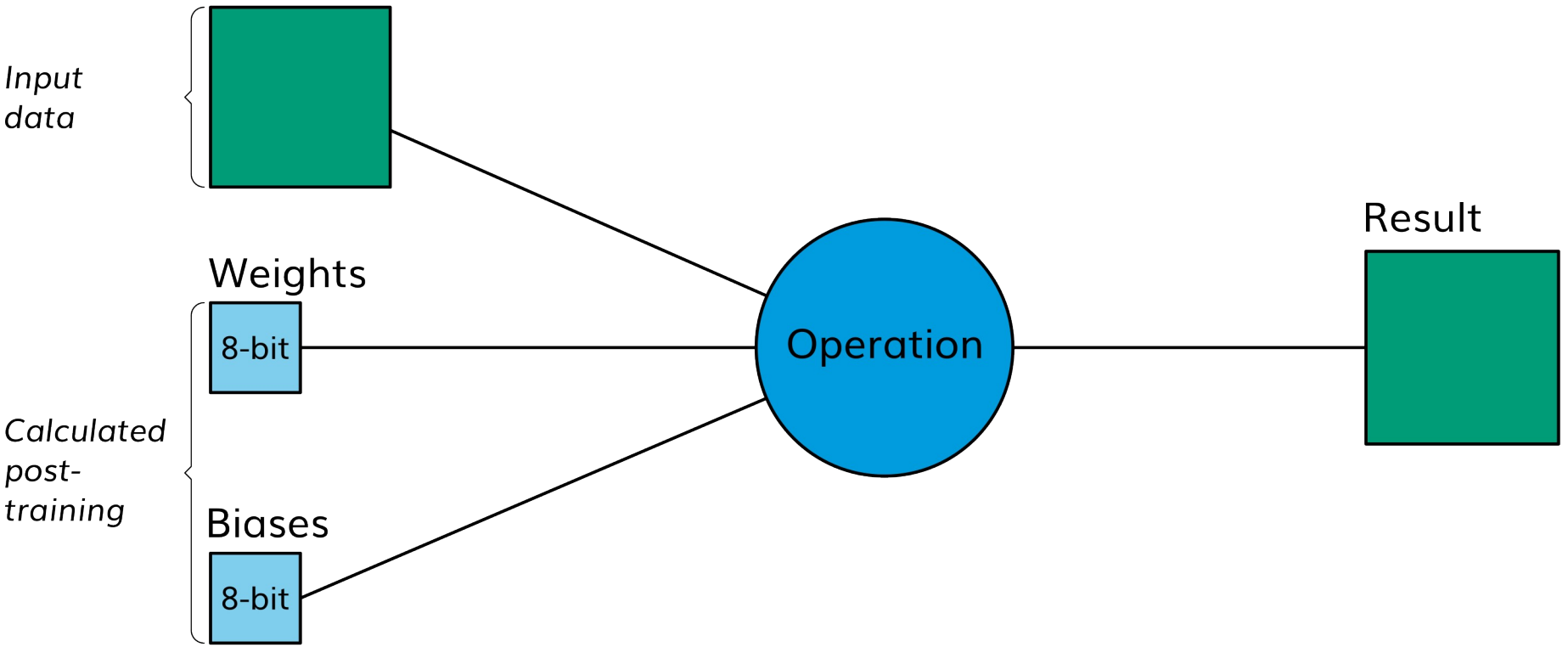
Optimization in Depth: DMA



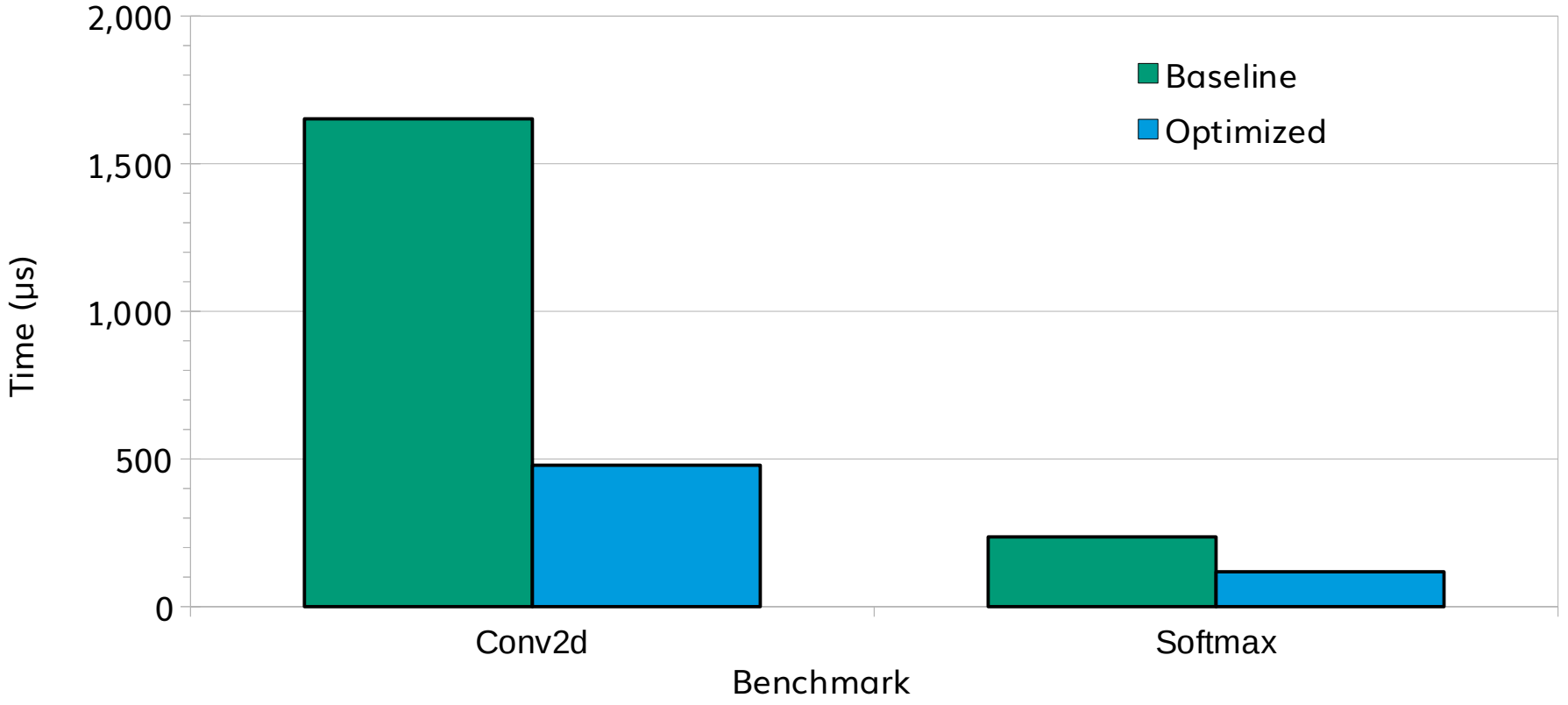
Optimization in Depth: Quantization



Optimization in Depth: Quantization



The Performance Benefit



Acknowlegement

The Embecosm team would like to thank our colleagues at Mosaic SoC in Switzerland who have supported the work presented here

Thank You

shane.slattery@embecosm.com
pietra.ferreira@embecosm.com
william.jones@embecosm.com
jeremy.bennett@embecosm.com

Shane Slattery
Pietra Ferreira
William Jones
Jeremy Bennett