PyTorch CONFERENCE EUROPE 2026

Torch-Spyre

Compiling to a multi-core dataflow accelerator with Inductor

Dave Grove & Olivier Tardieu

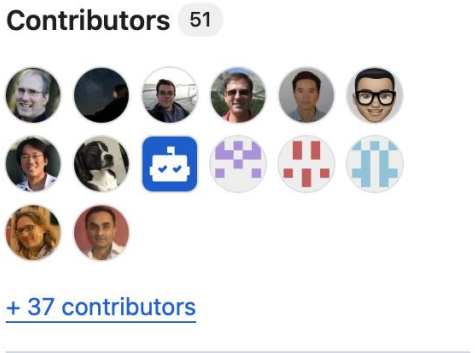
IBM

Torch-Spyre

- Open source out-of-tree PyTorch device for IBM Spyre Accelerator
- Using architecture recommended by torch.accelerator and OpenReg
- Enabling the PyTorch ecosystem + development experience for Spyre
- Leverage upstream projects & contribute back enhancements

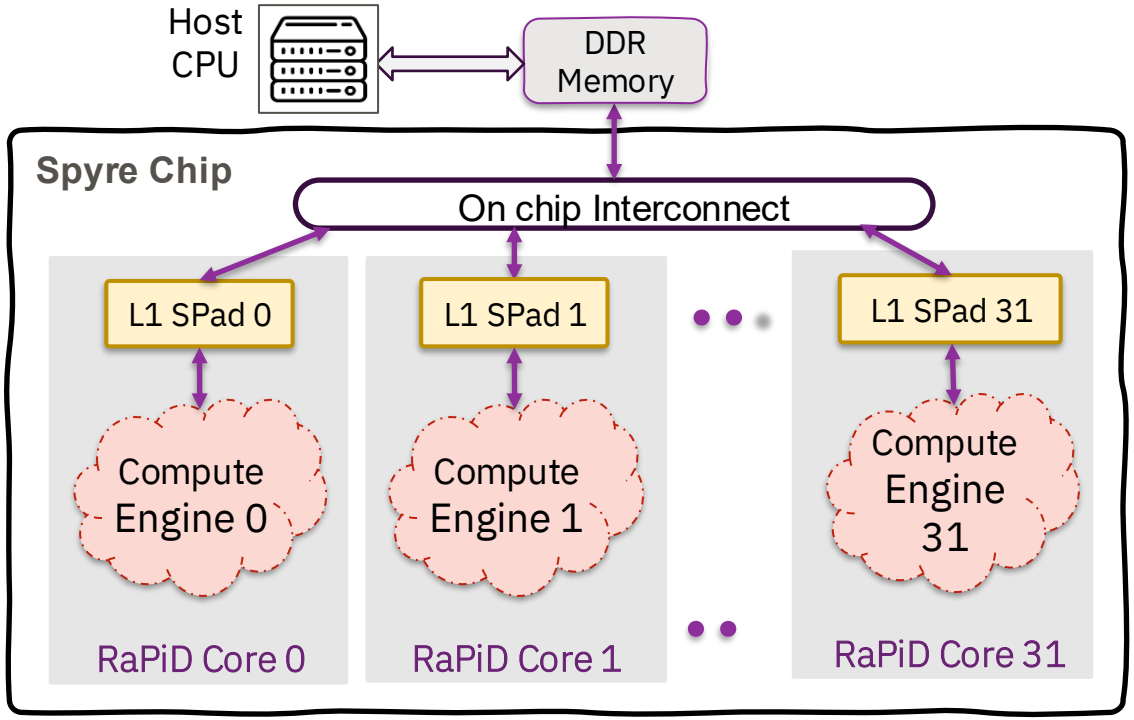
- A large team effort

https://github.com/torch-spyre/torch-spyre

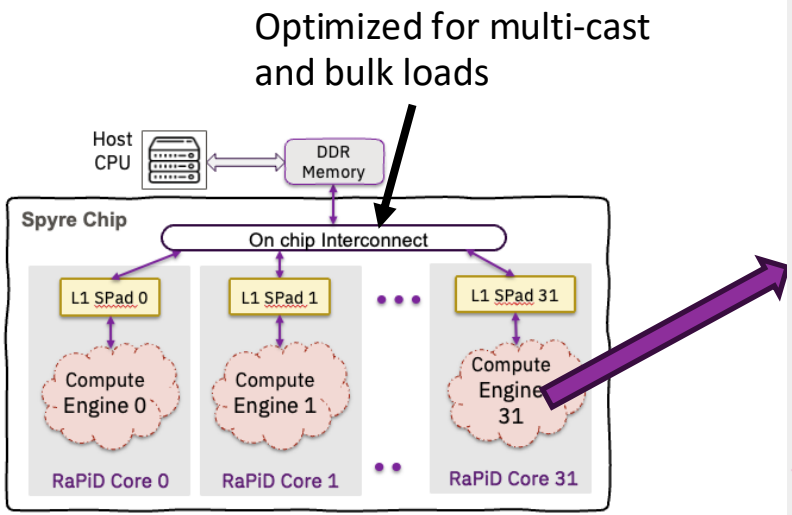


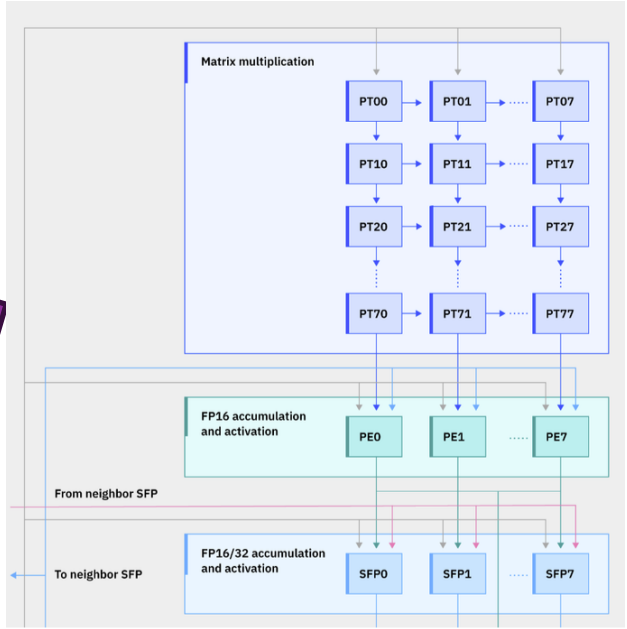
IBM Spyre Accelerator in a Nutshell

- System on a chip
- 32 compute cores
- Compiler-managed scratchpad (L1 SPad)
- Optimized for AI models
  a. 8, 16, 32 bit floats
  b. specialized compute units
  c. on chip interconnect



IBM Spyre Accelerator in a Nutshell





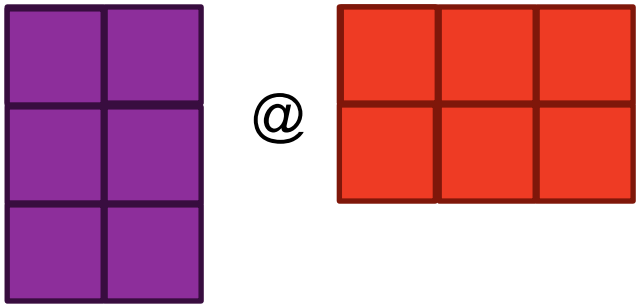
8 x 8 Systolic Array

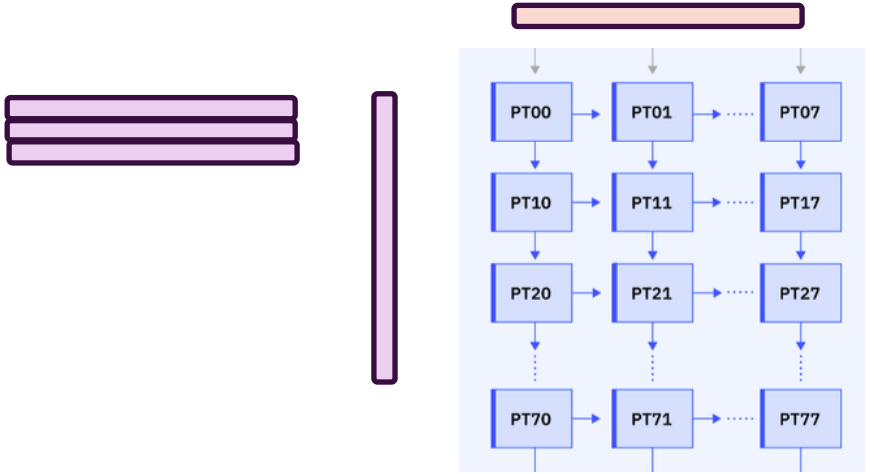
Every compute unit is a
16 byte SIMD engine

Achieving Efficient Matrix Multiply

- Compute Width: 128 bytes (16 x 8)
- **Utilize Systolic Array**

  ➔ **fp16: 64 x 64 tile**

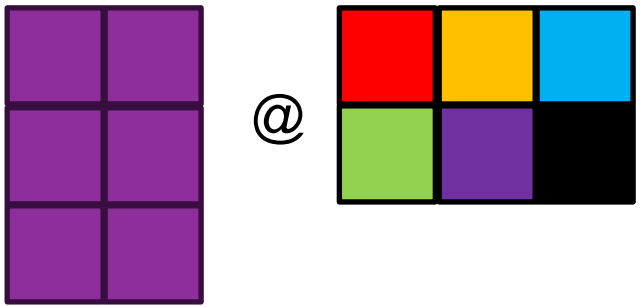


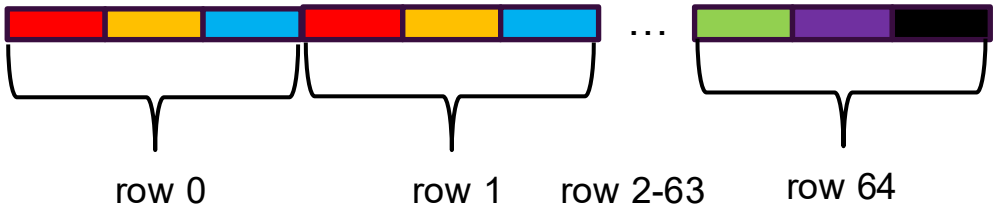


Achieving Efficient Matrix Multiply

- Compute Width: 128 bytes
- Utilize Systolic Array
  ➔ fp16: 64 x 64 tile
- **Exploit memory subsystem**
  ➔ **Bulk load of tiles**

need 64 loads to get one tile ☹





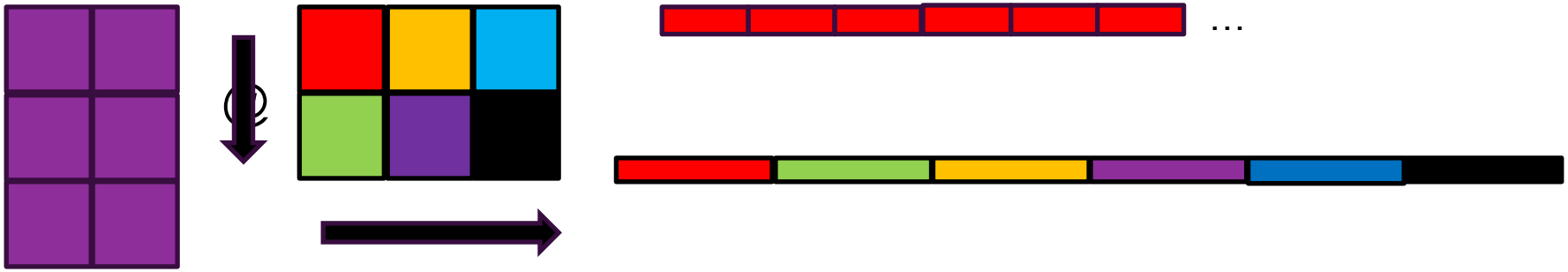
PyTorch default: Row Major Layout

Achieving Efficient Matrix Multiply

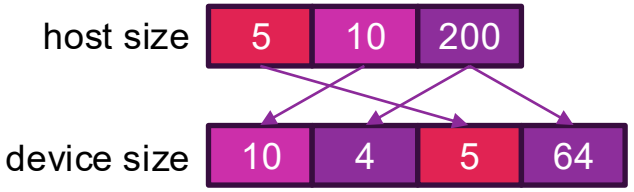
- Compute Width: 128 bytes
- Utilize Systolic array
  ➔ fp16: 64 x 64 tile
- **Exploit memory subsystem**
  ➔ **tile tensors to enable bulk load**

Our Solution

Tiled Tensor Layout



Spyre Tensor Layouts



tensor.to("spyre")

- host_size = [5, 10, 200]
- device_size = [10, 4, 5, 64]
-

host_stride = [2000, 200, 1]

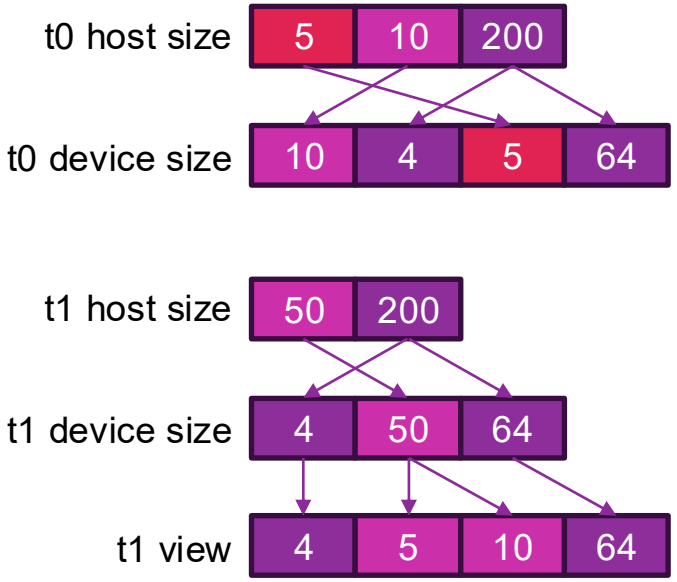
device_stride = [1280, 320, 64, 1]

stride_map = [200, 64, 2000, 1]

- host_offset = dot(host coordinates, host stride)
- device_offset = dot(device coordinates, device stride)
- **host_offset = dot(device coordinates, stride_map)**

The stride_map encodes the relationship between host and device coordinates

Compiling for Spyre



- t0 = torch.rand((5, 10, 200)).flatten(0, 1)
  t1 = torch.rand((50, 200))
  t0 + t1

- LLIR describes ops on host tensors
  - var_range = {p0: 50, p1: 200}
    index0 = 200*p0+p1

- **We need to convert these ops to ops on device tensors and simplify**
  - coordinates0 = [p0%10, p1//64, p0//10, p1%64]
    coordinates1 = [p1//64, p0, p1%64]

→ [q0, s0, q1, s1]
[s0, q1, q0, s1]

Take Away

Session Feedback ➔

- FX graph construction
- FX graph passes
  - Op decompositions
  - Padding
- Lowering
  - Custom lowerings
- Loop-level IR passes
  - Tiling
  - Core division
  - Scratchpad management
- Spyre code generation

Torch-Spyre adds device-specific passes and configurations to inductor.

By augmenting inductor instead of developing from scratch we replaced
in the order of 100K LOCs with 10K LOCs.

https://github.com/torch-spyre/torch-spyre