



CONFERENCE

—NORTH AMERICA 2026—

Device-Aware Tensor Layouts

Avery Blanchard, Matt Arnold, Olivier Tardieu
IBM Research

Outline

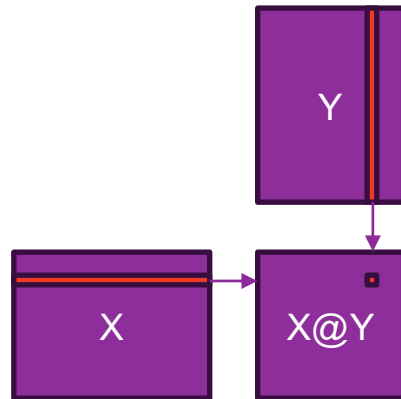
- Tensor Layouts Matter
 - Optimal Layouts Depend on Accelerators
- Inductor Layouts (`torch.compile`)
 - Limited Options, GPU Bias
- Device-Aware Layouts in Inductor
 - Example: Tiled Layouts
- Take Away

Tensor Layouts Matter

- Alignment
- Coalescing
- Extra transpositions
- Hardware tile size
- Memory channels and banks
- Cache hierarchies

Valid/optimal choices
depend on the device

Matrix multiplication



GPUs prefer row-major @ column-major
Spyre prefers tiled row-major @ tiled row-major

Inductor-Inferred Layouts are Limited

- `tensor.strided`
 - `size` = elements per dimension
 - `stride` = elements to next dimension
 - `storage_offset` = 1st element offset
- `torch.compile (inductor)`
 - decide to materialize tensors
 - decide tensor strides
 - + account for op constraints and some preferences
 - limited to expressible layouts
 - biased toward GPUs



0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	47
48	49	50	51	52	53	54	55

`size` = [4, 6]

`stride` = [8, 1]

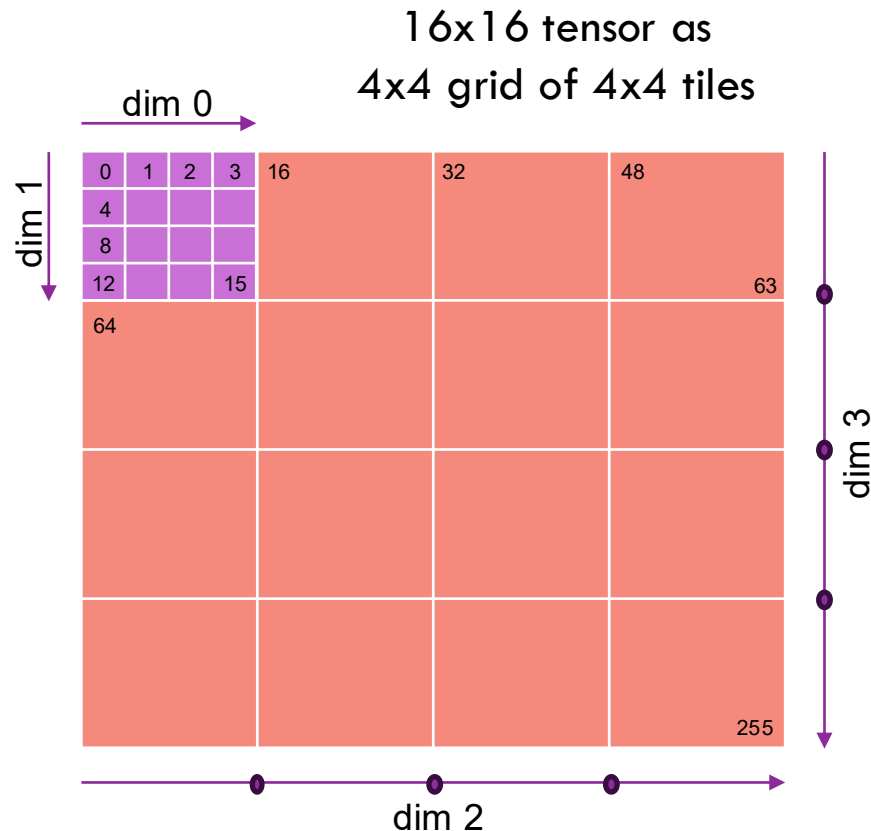
`storage_offset` = 16

Device-Aware Layouts

Richer layouts have benefits

Example: Tiled Layouts

- Arrange tensor elements as grid of tiles
- “Flatten in place”
- Tiles are contiguous
- Transposed tensors also have contiguous tiles



Fixed[Tiled]Layout

Inductor's FixedLayout

size = [16, 16]

stride = [16, 1]

device_size = [4, 4, 4, 4]

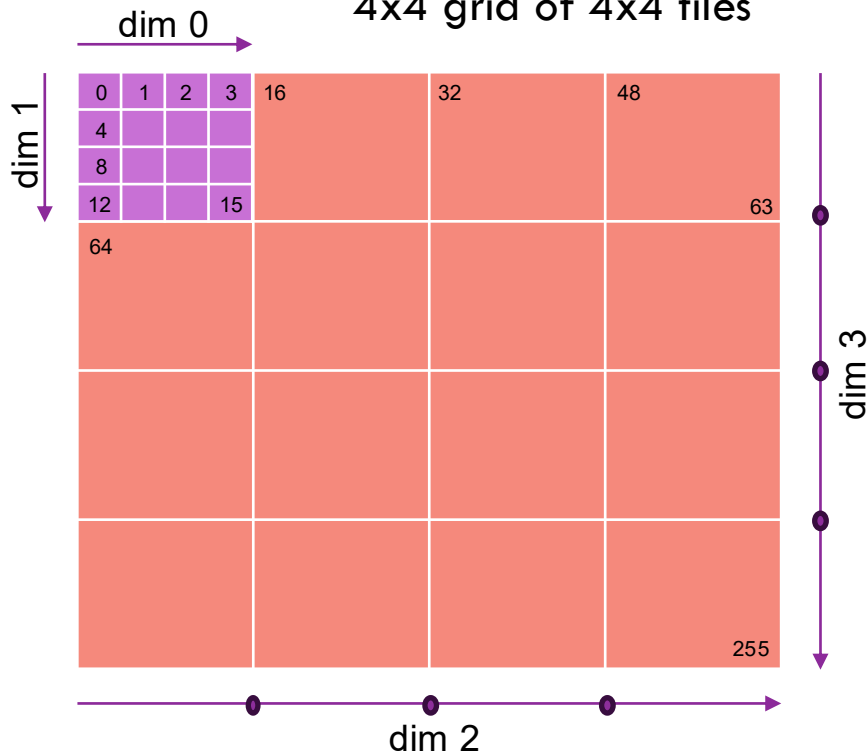
device_stride = [64, 16, 4, 1]

stride_map = [64, 4, 16, 1]

FixedTiledLayout

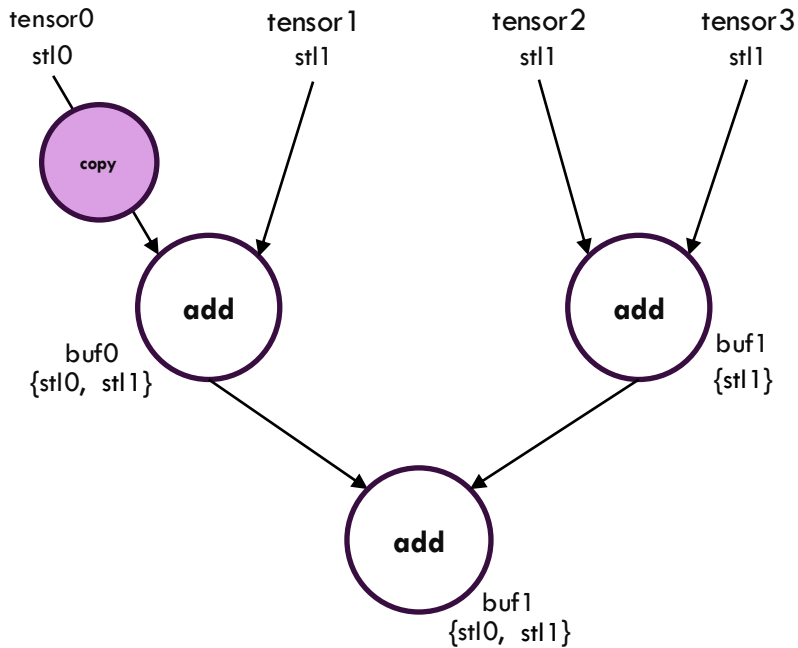
The FixedTiledLayout augments the FixedLayout instead of replacing it.

16x16 tensor as
4x4 grid of 4x4 tiles



Validate and Optimize Layouts: torch-spyre Example

Override copy to/from
device to realize device
layout



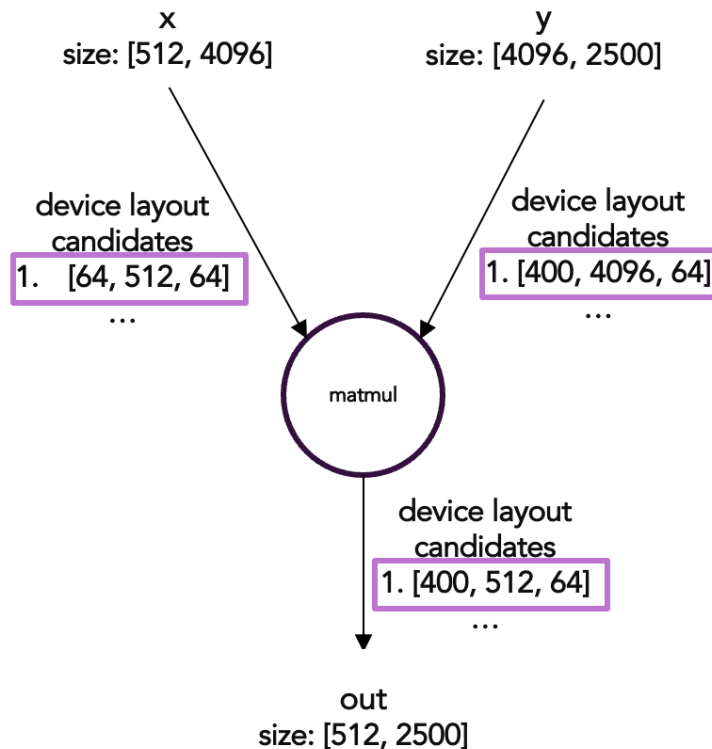
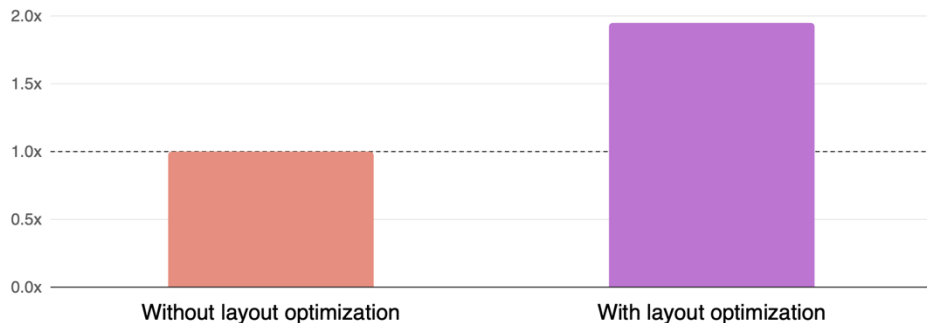
Select layouts, minimizing cost

Propagate layout candidates

Insert relay layout operations

Impact of Tiled Layouts on Spyre

- Tiling enables burst loads on these inputs to saturate off-chip memory bandwidth
- The tiled layouts provide **1.95x speedup** over the untiled layout for matrix multiply



Take Away

- In torch-spyre, we couple Inductor's layout with a device layout
- Other options explored:
 - Replacing tensor.strided
 - Deferring to Inductor backend (Triton)
- Should Inductor's layout be extended to enable more layout optimizations?
- See device layouts in action in [torch-spyre](#)
- Learn more about torch-spyre in our talk:
Integrating the IBM Spyre Accelerator
4:55 PM – LL21D (Lower Level)



Beyond Size and Stride: Unleash Performance with Device-Aware Tensor Layouts

IBM @ PyTorch2026
Presentation schedule,
publications, contacts, careers

