

GitHub: ToyAIX/triton-runner

 : [triton-runner.org](https://triton-runner.org)

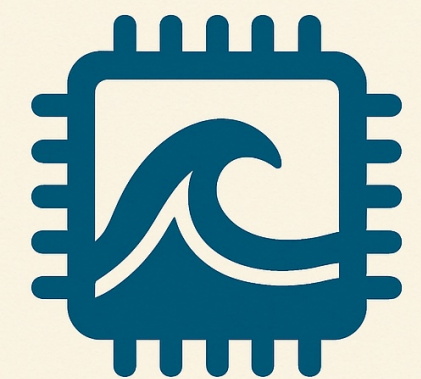
# Triton Runner

Multi-Level Triton Runner(Dump) 

BobHuang

Triton Compiler Engineer

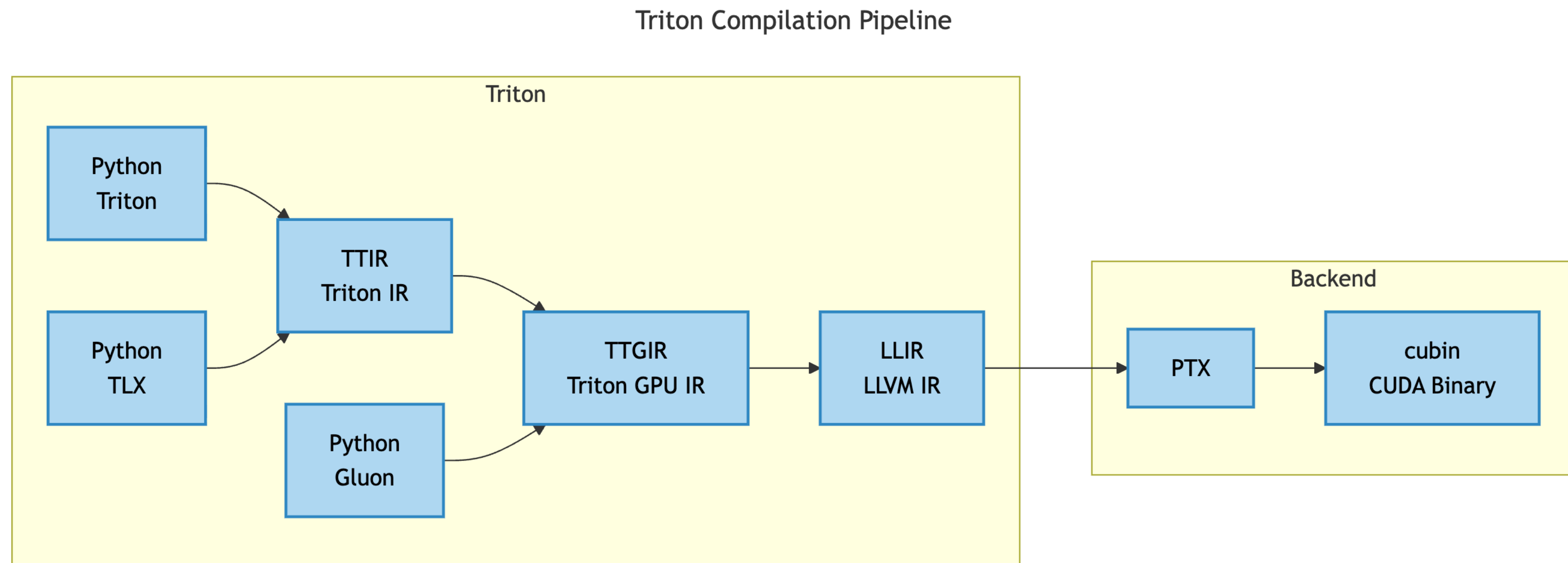
Oct 29, 2025



**Triton Runner**

# Introduction

- Multi-level execution engine for Triton
- Multi-level dump across Python/TTIR/TTGIR





# Installation

- Install from pip

```
pip install triton-runner
```

- Install from source

```
git clone https://github.com/toyaix/triton-runner
```

```
cd triton-runner
```

```
pip install -e .
```

# How To Use

## Python Runner

- using @triton\_runner.jit instead of @triton.jit

```
root ~/triton-runner# python examples/python_runner/matmul.py
[triton-runner] TRITON_ALWAYS_COMPILE defaults to ON.
For production, It is recommended to disable it by running: export TRITON_ALWAYS_COMPILE=0
[triton-runner] Triton cache with always_compile saved at /root/.triton/cache/4C73IUPJYEJEVPBKD73MOCFVNQIVL5U0MEMNJJJF6NCOYFQ0LM2A
✅ Triton and Torch match
```

```
import triton_runner

# @triton.jit
@triton_runner.jit
def matmul_kernel(
    a_ptr, b_ptr, c_ptr,
    M, N, K,
    stride_am, stride_ak,
    stride_bk, stride_bn,
    stride_cm, stride_cn,
    BLOCK_SIZE_M: tl.constexpr, BLOCK_SIZE_N: tl.constexpr
):
```

# How To Use

## TTIR Runner

- using @triton\_runner.jit instead of @triton.jit
- provide TTIR path

```
ttir_dir=triton_runner.get_file_dir(__file__)
```



```
root ~/triton-runner# python examples/ttir_runner/matmul/mat
[triton-runner] TRITON_ALWAYS_COMPILE defaults to ON.
For production, It is recommended to disable it by running:
YS_COMPILE=0
[triton-runner] Triton cache with always_compile saved at /r
JYZ2BRNQHK3FSQHHDBTJHXG4Y3GSC3B7TM7BET3B3MBJYV7S4HKQ
✅ Triton and Torch match
```



```
matmul_kernel[grid](
    a, b, c,
    M, N, K,
    a.stride(0), a.stride(1),
    b.stride(0), b.stride(1),
    c.stride(0), c.stride(1),
    BLOCK_SIZE_M=16,
    BLOCK_SIZE_N=16,
    ttir_dir=triton_runner.get_file_dir(__file__),
)
return c
```





# How To Use

## TTGIR Runner

- using @triton\_runner.jit instead of @triton.jit
- provide TTGIR path
- perhaps need metadata json

 main ▾ [triton-runner](#) / [examples](#) / [runner](#) / [tlx](#) / [v3.4.0](#) / [ttgir](#) / [sm90](#) / 

 **sBobHuang** Fix tlx support (#51) 

Name	Last commit message
 ..	
 _attn_fwd_ws_pipelined_pingpong.json	Fix tlx support (#51)
 _attn_fwd_ws_pipelined_pingpong.ttgir	Fix tlx support (#51)
 hopper-fa-ws-pipelined-pingpong.py	Fix tlx support (#51)

# How To Use

## LLIR/PTX/cubin Runner

- using @triton\_runner.jit instead of @triton.jit
- provide LLIR/PTX/cubin path
- need metadata json




# How To Use

## More Runner

- Gluon(Tile-Based GPU Programming with Low-Level Control) Runner
- TLX(Minimally Invasive Paths to Performance Portability) Runner

main


[triton-runner](#) / [examples](#) / [runner](#) / [v3.4.0](#) / [gluon](#) /

 **sBobHuang** Release v0.2.6 (#40)

Name	Last commit message
..	
01-intro.py	Release v0.2.6 (#40)
02-layouts.py	Release v0.2.6 (#40)

main

[triton-runner](#) / [examples](#) / [runner](#) / [tlx](#) / [v3.4.0](#) /

 **sBobHuang** Fix tlx support (#51)

Name	Last commit message
..	
cubin/sm90	Fix tlx support (#51)
llir/sm90	Fix tlx support (#51)
ptx/sm90	Fix tlx support (#51)
python	Fix tlx support (#51)
ttgir/sm90	Fix tlx support (#51)
ttir/sm90	Fix tlx support (#51)



# Hopper Examples

**sm90 (H100, H200, H20, etc.)**

```
python examples/runner/v3.4.0/python/matmul-with-tma-v4.py
```

```
python examples/runner/v3.4.0/ttir/matmul-with-tma/matmul-with-tma-v4.py
```

```
python examples/runner/v3.4.0/ttgir/sm90/matmul-with-tma-v4.py
```

```
python examples/runner/v3.4.0/llir/sm90/matmul-with-tma-v4.py
```

```
python examples/runner/v3.4.0/ptx/sm90/matmul-with-tma-v4.py
```

```
python examples/runner/v3.4.0/cubin/sm90/matmul-with-tma-v4.py
```

```
python examples/runner/v3.4.0/gluon/01-intro.py
```

```
python examples/runner/v3.4.0/gluon/02-layouts.py
```

# More Architectures Examples

- sm90: Hopper (H100, H200, H20, etc.)
- sm80: Ampere (A100, A30)
- sm120: Blackwell (RTX PRO 6000, RTX 5090, etc.)
- sm86: Ampere (A10, RTX 3090, etc.)
- sm75: Turing (T4, RTX 2080, etc.)

# How To Use

## Python dump

- `triton_runner.language.dump()`

```
def dump(val: tl.tensor, offset: int = 0, dump_grid=None, _semantic=None):
```

Example:

```
import triton_runner.language as dl
```

```
dl.dump(output)
```

```
dl.dump(y, 0, (3))
```

```
dl.dump(y, 0, (3, 0))
```

```
dl.dump(y, 0, (3, 0, 0))
```

```
dl.dump(y, 0, 3)
```

```
dl.dump(y, BLOCK_SIZE, 1)
```

```
BLOCK_SIZE = 1024
dump_tensor = torch.empty((BLOCK_SIZE), dtype=x.dtype, device=x.device)

# NOTE:
# - Each torch.tensor object is implicitly converted into a pointer to its first
# - `triton.jit`'ed functions can be indexed with a launch grid to obtain a call
# - Don't forget to pass meta-parameters as keywords arguments.
add_kernel[grid](x, y, output, n_elements, BLOCK_SIZE=BLOCK_SIZE,
                 dump_tensor=dump_tensor,
)
```

# How To Use

## Python dump

- `triton_runner.language.dump()`

```
• (triton) ubuntu@VM-0-6-ubuntu:~/triton-runner$ python examples/dump/python/01-vec_add/dump_output.py
[Triton Runner] In dump mode, ssa=%36, op=arith.addf, loc=loc(#loc12), size=1024
[Triton Runner] Triton kernel cache hint and saved at /home/ubuntu/.triton/cache/6X7IIHFITW
I7EVD0ZIW4SY6YZTKJ2MVERL04KHXBMOQILOBMTBPQ
debug tensor([1.5409, 0.8883, 1.6324, ..., 1.4323, 1.2401, 0.7313], device='cuda:0')
The maximum difference between torch and dump is 0.0
✅ Triton and Torch match
```

# How To Use

**Python dump**

# How To Use

## Python dump

- `triton_runner.language.dump_boundary()`

```
def dump_boundary(val: tl.tensor, offset=0, _semantic=None):
```

Example:

```
import triton_runner.language as dl
```

```
dl.dump_boundary(transposed_block)
```

```
dl.dump(transposed_block, 0 ,  
        (tl.num_programs(axis=0) - 1,  
         tl.num_programs(axis=1) - 1,  
         tl.num_programs(axis=1) - 1))
```

# How To Use

## Python dump

- `triton_runner.language.dump_grids()`

`def dump_grids(val: tl.tensor, offset=0, _semantic=None):`

Example:

`import triton_runner.language as dl`

`dl.dump_grids(output)`

```
add_kernel[grid](x, y, output, n_elements, BLOCK_SIZE=BLOCK_SIZE,
                 dump_tensor=dump_tensor,
                 )
triton_runner.color_print.blue_print(f"debug {dump_tensor}")
dump_torch = x + y
max_diff = torch.max(torch.abs(dump_torch - dump_tensor[:n_elements]))
triton_runner.color_print.yellow_print(f"The maximum difference between torch and dump is {max_diff}")
# We return a handle to z but, since `torch.cuda.synchronize()` hasn't been called, the kernel is still
# running asynchronously at this point
```



# How To Use

## Python dump

- `triton_runner.language.dump_grids()`

`dl.dump_grids(block)`

```
# is same (triton.cdiv(rows, BLOCK_SIZE), triton.cdiv(cols, BLOCK_SIZE))
grid_dim = triton_runner.torch_utils.get_grid_dim([rows, cols], block_shape)
block_reshape = dump_tensor.reshape(*grid_dim, *block_shape)
block_permute = block_reshape.permute(0, 2, 1, 3)
reshape_tensor = block_permute.reshape(grid_dim[0] * BLOCK_SIZE, grid_dim[1] * BLOCK_SIZE)
dump_torch = input
max_diff = torch.max(torch.abs(dump_torch - reshape_tensor[:rows,:cols]))
triton_runner.color_print.yellow_print(f"The maximum difference between torch and dump is {max_diff}")
```

# How To Use

**TTIR dump**

# How To Use

## TTIR dump

```
BLOCK_SIZE = 1024
dump_tensor = torch.empty((BLOCK_SIZE), dtype=x.dtype, device=x.device)
# dump_value can be "%13"(x+y)
dump_value = "%13"

# NOTE:
# - Each torch.tensor object is implicitly converted into a pointer to its first element.
# - `triton.jit`'ed functions can be indexed with a launch grid to obtain a callable GPU kernel.
# - Don't forget to pass meta-parameters as keywords arguments.
add_kernel[grid](x, y, output, n_elements, BLOCK_SIZE=BLOCK_SIZE,
                 ttir_dir=triton_runner.get_file_dir(__file__),
                 dump_tensor=dump_tensor,
                 dump_value=dump_value,
)

triton_runner.color_print.blue_print(f"debug {dump_tensor}")
dump_torch = x + y
max_diff = torch.max(torch.abs(dump_torch[:BLOCK_SIZE] - dump_tensor))
triton_runner.color_print.yellow_print(f"The maximum difference between torch and dump is {max_diff}")
# We return a handle to z but, since `torch.cuda.synchronize()` hasn't been called, the kernel is still
# running asynchronously at this point.
return output
```

# How To Use

## TTIR dump

```
module {
  tt.func public @add_kernel(%arg0: !tt.ptr<f32> {tt.divisibility = 16 : i32} loc("/home/ubur
    %c1024_i32 = arith.constant 1024 : i32 loc(#loc1)
    %0 = tt.get_program_id x : i32 loc(#loc2)
    %1 = arith.muli %0, %c1024_i32 : i32 loc(#loc3)
    %2 = tt.make_range {end = 1024 : i32, start = 0 : i32} : tensor<1024xi32> loc(#loc4)
    %3 = tt.splat %1 : i32 -> tensor<1024xi32> loc(#loc5)
    %4 = arith.addi %3, %2 : tensor<1024xi32> loc(#loc5)
    %5 = tt.splat %arg3 : i32 -> tensor<1024xi32> loc(#loc6)
    %6 = arith.cmpi slt, %4, %5 : tensor<1024xi32> loc(#loc6)
    %7 = tt.splat %arg0 : !tt.ptr<f32> -> tensor<1024x!tt.ptr<f32>> loc(#loc7)
    %8 = tt.addptr %7, %4 : tensor<1024x!tt.ptr<f32>>, tensor<1024xi32> loc(#loc7)
    %9 = tt.load %8, %6 : tensor<1024x!tt.ptr<f32>> loc(#loc8)
    %10 = tt.splat %arg1 : !tt.ptr<f32> -> tensor<1024x!tt.ptr<f32>> loc(#loc9)
    %11 = tt.addptr %10, %4 : tensor<1024x!tt.ptr<f32>>, tensor<1024xi32> loc(#loc9)
    %12 = tt.load %11, %6 : tensor<1024x!tt.ptr<f32>> loc(#loc10)
    %13 = arith.addf %9, %12 : tensor<1024xf32> loc(#loc11)
    %14 = tt.splat %arg2 : !tt.ptr<f32> -> tensor<1024x!tt.ptr<f32>> loc(#loc12)
    %15 = tt.addptr %14, %4 : tensor<1024x!tt.ptr<f32>>, tensor<1024xi32> loc(#loc12)
    tt.store %15, %13, %6 : tensor<1024x!tt.ptr<f32>> loc(#loc13)
    tt.return loc(#loc14)
  } loc(#loc)
} loc(#loc)
```



# How To Use

## TTIR dump

```
(triton) ubuntu@VM-0-6-ubuntu:~/triton-runner$ python examples/dump/ttir/01-vector_add/dump_load.py
[Triton Runner] In dump mode, ssa=%9, op=tt.load, loc=loc(#loc8), size=1024
[Triton Runner] Triton kernel cache hint and saved at /home/ubuntu/.triton/cache/MNMF3666PRMM400TPQ2320U0FVJ5KRFYIU7NPDQDFNVG2KLFWV3A
debug tensor([0.9769, 0.6768, 0.9105, ..., 0.6336, 0.3570, 0.1050], device='cuda:0')
The maximum difference between torch and dump is 0.0
✅ Triton and Torch match
```

# How To Use

**TTGIR dump**

# How To Use

## TTGIR dump

```
BLOCK_SIZE_M, BLOCK_SIZE_K = 64, 32
dump_tensor = torch.empty((BLOCK_SIZE_M, BLOCK_SIZE_K), dtype=torch.float32, device=a.device)
# dump_value can be "%54"(acc in loop)
dump_value = "%54"
dump_grid = (0, 1)

matrix_multiplication_kernel[grid](
    a, b, c,
    M, N, K,
    a.stride(0), a.stride(1),
    b.stride(0), b.stride(1),
    c.stride(0), c.stride(1),
    BLOCK_SIZE_M=BLOCK_SIZE_M,
    BLOCK_SIZE_K=BLOCK_SIZE_K,
    ttgir_dir=triton_runner.get_file_dir(__file__),
    dump_tensor=dump_tensor,
    dump_value=dump_value,
    dump_grid=dump_grid,
)

triton_runner.color_print.blue_print(f"debug {dump_tensor}")
dump_torch = a @ b
start_K, start_M = dump_grid[0] * BLOCK_SIZE_K, dump_grid[1] * BLOCK_SIZE_M
dump_torch_slice = dump_torch[start_M:start_M+BLOCK_SIZE_M, start_K:start_K+BLOCK_SIZE_K]
max_diff = torch.max(torch.abs(dump_torch_slice - dump_tensor))
triton_runner.color_print.yellow_print(f"The maximum difference between torch and dump is {max_diff}")
```



# Thanks!

For more details, visit: [triton-runner.org](http://triton-runner.org)