TestPermute

March 19, 2024

[ ]:
```
import torch
torch.cuda.is_available()
!export LC_ALL="en_US.UTF-8"
!export LD_LIBRARY_PATH="/usr/lib64-nvidia"
!export LIBRARY_PATH="/usr/local/cuda/lib64/stubs"
!ldconfig /usr/lib64-nvidia
```

```
/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_0.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_5.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbb.so.12 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc_proxy.so.2 is not a symbolic
link

/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc.so.2 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind.so.3 is not a symbolic link
```

[ ]:
```
import torch

def fn(x, y):
    return torch.permute(x, (0, 2, 1, 3)) + y
fnc = torch.compile(fn)

bsz = 4
num_head = 32
seq_len = 2048
head_dim = 128
x = torch.randn([bsz, num_head, seq_len, head_dim]).cuda()
y = torch.randn([bsz, seq_len, num_head, head_dim]).cuda()
z = fnc(x, y)
print(z[0,0,0,0])
```

```
tensor(2.7243, device='cuda:0')
```

```
[ ]: !cat "/tmp/torchinductor_root/af/
      ↪cafv3o7yfvokrdc43dlfnnsbarspfkfa4mspui5idtwypok2n2a7.py"
```

```
from ctypes import c_void_p, c_long
import torch
import math
import random
import os
import tempfile
from math import inf, nan
from torch._inductor.hooks import run_intermediate_hooks
from torch._inductor.utils import maybe_profile
from torch._inductor.codegen.memory_planning import _align as align

from torch import device, empty, empty_strided
from torch._inductor.codecache import AsyncCompile
from torch._inductor.select_algorithm import extern_kernels

aten = torch.ops.aten
inductor_ops = torch.ops.inductor
assert_size_stride = torch._C._dynamo.guards.assert_size_stride
alloc_from_pool = torch.ops.inductor._alloc_from_pool
reinterpret_tensor = torch.ops.inductor._reinterpret_tensor
async_compile = AsyncCompile()


# kernel path: /tmp/torchinductor_root/t2/ct2f2733o5civzjcus4ql5aofxvaxgixan4psa
k3os3b36644txj.py
# Source Nodes: [add], Original ATen: [aten.add]
# add => add
triton_poi_fused_add_0 = async_compile.triton('triton_', '''
import triton
import triton.language as tl
from torch._inductor.ir import ReductionHint
from torch._inductor.ir import TileHint
from torch._inductor.triton_heuristics import AutotuneHint, pointwise
from torch._inductor.utils import instance_descriptor
from torch._inductor import triton_helpers

@pointwise(
    size_hints=[33554432],
    filename=__file__,
    triton_meta={'signature': {0: '*fp32', 1: '*fp32', 2: '*fp32', 3: 'i32'},
'device': 0, 'device_type': 'cuda', 'constants': {}, 'configs':
[instance_descriptor(divisible_by_16=(0, 1, 2, 3), equal_to_1=(),
ids_of_folded_args=(), divisible_by_8=(3,))]},
```

```
    inductor_meta={'autotune_hints': set(), 'kernel_name':
'triton_poi_fused_add_0', 'mutated_arg_names': []},
    min_elem_per_thread=0
)
@triton.jit
def triton_(in_ptr0, in_ptr1, out_ptr0, xnumel, XBLOCK : tl.constexpr):
    xnumel = 33554432
    xoffset = tl.program_id(0) * XBLOCK
    xindex = xoffset + tl.arange(0, XBLOCK)[:]
    xmask = xindex < xnumel
    x4 = xindex
    x0 = xindex % 128
    x1 = (xindex // 128) % 2048
    x2 = (xindex // 262144) % 32
    x3 = (xindex // 8388608)
    tmp0 = tl.load(in_ptr0 + (x4), None)
    tmp1 = tl.load(in_ptr1 + (x0 + (128*x2) + (4096*x1) + (8388608*x3)), None)
    tmp2 = tmp0 + tmp1
    tl.store(out_ptr0 + (x4), tmp2, None)
''')

import triton
import triton.language as tl
from torch._inductor.triton_heuristics import grid, start_graph, end_graph
from torch._C import _cuda_getCurrentRawStream as get_cuda_stream


async_compile.wait(globals())
del async_compile

def call(args):
    arg0_1, arg1_1 = args
    args.clear()
    assert_size_stride(arg0_1, (4, 32, 2048, 128), (8388608, 262144, 128, 1))
    assert_size_stride(arg1_1, (4, 2048, 32, 128), (8388608, 4096, 128, 1))
    with torch.cuda._DeviceGuard(0):
        torch.cuda.set_device(0) # no-op to ensure context
        buf0 = empty_strided((4, 2048, 32, 128), (8388608, 128, 262144, 1),
device='cuda', dtype=torch.float32)
        # Source Nodes: [add], Original ATen: [aten.add]
        stream0 = get_cuda_stream(0)
        triton_poi_fused_add_0.run(arg0_1, arg1_1, buf0, 33554432,
grid=grid(33554432), stream=stream0)
        del arg0_1
        del arg1_1
        return (buf0, )
```

```
def benchmark_compiled_module(times=10, repeat=10):
    from torch._dynamo.testing import rand_strided
    from torch._inductor.utils import print_performance
    arg0_1 = rand_strided((4, 32, 2048, 128), (8388608, 262144, 128, 1),
device='cuda:0', dtype=torch.float32)
    arg1_1 = rand_strided((4, 2048, 32, 128), (8388608, 4096, 128, 1),
device='cuda:0', dtype=torch.float32)
    fn = lambda: call([arg0_1, arg1_1])
    return print_performance(fn, times=times, repeat=repeat)


if __name__ == "__main__":
    from torch._inductor.wrapper_benchmark import compiled_module_main
    compiled_module_main('None', benchmark_compiled_module)
```

[ ]:
```
import torch

def fn(x, y):
    return torch.permute(x, (0, 1, 3, 2)) + y
fnc = torch.compile(fn)

bsz = 4
num_head = 32
seq_len = 2048
head_dim = 128
x = torch.randn([bsz, num_head, seq_len, head_dim]).cuda()
y = torch.randn([bsz, num_head, head_dim, seq_len]).cuda()
z = fnc(x, y)
print(z[0,0,0,0])
```

```
tensor(1.2154, device='cuda:0')
```

[ ]:
```
!cat "/tmp/torchinductor_root/tf/
 ↪ctf2z67mjsizixjuil7z2h7ajirzmapgzicfyk5a63qad2ca7ke2.py"
```

```
from ctypes import c_void_p, c_long
import torch
import math
import random
import os
import tempfile
from math import inf, nan
from torch._inductor.hooks import run_intermediate_hooks
from torch._inductor.utils import maybe_profile
from torch._inductor.codegen.memory_planning import _align as align

from torch import device, empty, empty_strided
```

```
from torch._inductor.codecache import AsyncCompile
from torch._inductor.select_algorithm import extern_kernels

aten = torch.ops.aten
inductor_ops = torch.ops.inductor
assert_size_stride = torch._C._dynamo.guards.assert_size_stride
alloc_from_pool = torch.ops.inductor._alloc_from_pool
reinterpret_tensor = torch.ops.inductor._reinterpret_tensor
async_compile = AsyncCompile()


# kernel path: /tmp/torchinductor_root/u2/cu26ieqgivye67g7zgzwituhry3fpqnblmomg6
nnndq5ahl6uukx.py
# Source Nodes: [add], Original ATen: [aten.add]
# add => add
triton_poi_fused_add_0 = async_compile.triton('triton_', '''
import triton
import triton.language as tl
from torch._inductor.ir import ReductionHint
from torch._inductor.ir import TileHint
from torch._inductor.triton_heuristics import AutotuneHint, pointwise
from torch._inductor.utils import instance_descriptor
from torch._inductor import triton_helpers

@pointwise(
    size_hints=[262144, 128], tile_hint=TileHint.DEFAULT,
    filename=__file__,
    triton_meta={'signature': {0: '*fp32', 1: '*fp32', 2: '*fp32', 3: 'i32', 4:
'i32'}, 'device': 0, 'device_type': 'cuda', 'constants': {}, 'configs':
[instance_descriptor(divisible_by_16=(0, 1, 2, 3, 4), equal_to_1=(),
ids_of_folded_args=(), divisible_by_8=(3, 4))]},
    inductor_meta={'autotune_hints': set(), 'kernel_name':
'triton_poi_fused_add_0', 'mutated_arg_names': []},
    min_elem_per_thread=0
)
@triton.jit
def triton_(in_ptr0, in_ptr1, out_ptr0, ynumel, xnumel, YBLOCK : tl.constexpr,
XBLOCK : tl.constexpr):
    ynumel = 262144
    xnumel = 128
    yoffset = tl.program_id(1) * YBLOCK
    yindex = yoffset + tl.arange(0, YBLOCK)[None, :]
    ymask = yindex < ynumel
    xoffset = tl.program_id(0) * XBLOCK
    xindex = xoffset + tl.arange(0, XBLOCK)[:, None]
    xmask = xindex < xnumel
    x2 = xindex
    y3 = yindex
```

```
    y0 = yindex % 2048
    y1 = (yindex // 2048)
    tmp0 = tl.load(in_ptr0 + (x2 + (128*y3)), xmask,
eviction_policy='evict_last')
    tmp1 = tl.load(in_ptr1 + (y0 + (2048*x2) + (262144*y1)), xmask,
eviction_policy='evict_last')
    tmp2 = tmp0 + tmp1
    tl.store(out_ptr0 + (x2 + (128*y3)), tmp2, xmask)
''')

import triton
import triton.language as tl
from torch._inductor.triton_heuristics import grid, start_graph, end_graph
from torch._C import _cuda_getCurrentRawStream as get_cuda_stream


async_compile.wait(globals())
del async_compile

def call(args):
    arg0_1, arg1_1 = args
    args.clear()
    assert_size_stride(arg0_1, (4, 32, 2048, 128), (8388608, 262144, 128, 1))
    assert_size_stride(arg1_1, (4, 32, 128, 2048), (8388608, 262144, 2048, 1))
    with torch.cuda._DeviceGuard(0):
        torch.cuda.set_device(0) # no-op to ensure context
        buf0 = empty_strided((4, 32, 128, 2048), (8388608, 262144, 1, 128),
device='cuda', dtype=torch.float32)
        # Source Nodes: [add], Original ATen: [aten.add]
        stream0 = get_cuda_stream(0)
        triton_poi_fused_add_0.run(arg0_1, arg1_1, buf0, 262144, 128,
grid=grid(262144, 128), stream=stream0)
        del arg0_1
        del arg1_1
        return (buf0, )


def benchmark_compiled_module(times=10, repeat=10):
    from torch._dynamo.testing import rand_strided
    from torch._inductor.utils import print_performance
    arg0_1 = rand_strided((4, 32, 2048, 128), (8388608, 262144, 128, 1),
device='cuda:0', dtype=torch.float32)
    arg1_1 = rand_strided((4, 32, 128, 2048), (8388608, 262144, 2048, 1),
device='cuda:0', dtype=torch.float32)
    fn = lambda: call([arg0_1, arg1_1])
    return print_performance(fn, times=times, repeat=repeat)
```

```
if __name__ == "__main__":
    from torch._inductor.wrapper_benchmark import compiled_module_main
    compiled_module_main('None', benchmark_compiled_module)
```

[ ]:
```
import torch

def fn(x, y):
    return torch.permute(x, (1, 0, 2, 3)) + y
fnc = torch.compile(fn)

bsz = 4
num_head = 32
seq_len = 2048
head_dim = 128
x = torch.randn([bsz, num_head, seq_len, head_dim]).cuda()
y = torch.randn([num_head, bsz, seq_len, head_dim]).cuda()
z = fnc(x, y)
print(z[0,0,0,0])
```

```
tensor(-1.1328, device='cuda:0')
```

[ ]:
```
!cat "/tmp/torchinductor_root/we/
 ↪cwecycuvlsahhjs6yasgz3gjzln6ciegtgl7cevwkqk4o2sy27mq.py"
```

```
from ctypes import c_void_p, c_long
import torch
import math
import random
import os
import tempfile
from math import inf, nan
from torch._inductor.hooks import run_intermediate_hooks
from torch._inductor.utils import maybe_profile
from torch._inductor.codegen.memory_planning import _align as align

from torch import device, empty, empty_strided
from torch._inductor.codecache import AsyncCompile
from torch._inductor.select_algorithm import extern_kernels

aten = torch.ops.aten
inductor_ops = torch.ops.inductor
assert_size_stride = torch._C._dynamo.guards.assert_size_stride
alloc_from_pool = torch.ops.inductor._alloc_from_pool
reinterpret_tensor = torch.ops.inductor._reinterpret_tensor
async_compile = AsyncCompile()
```

```
# kernel path: /tmp/torchinductor_root/fk/cfkpixsvdhvwf2cq7kiofa7ueu4babihi3uzwh
gr7lc5xdtij3gd.py
# Source Nodes: [add], Original ATen: [aten.add]
# add => add
triton_poi_fused_add_0 = async_compile.triton('triton_', '''
import triton
import triton.language as tl
from torch._inductor.ir import ReductionHint
from torch._inductor.ir import TileHint
from torch._inductor.triton_heuristics import AutotuneHint, pointwise
from torch._inductor.utils import instance_descriptor
from torch._inductor import triton_helpers

@pointwise(
    size_hints=[33554432],
    filename=__file__,
    triton_meta={'signature': {0: '*fp32', 1: '*fp32', 2: '*fp32', 3: 'i32'},
'device': 0, 'device_type': 'cuda', 'constants': {}, 'configs':
[instance_descriptor(divisible_by_16=(0, 1, 2, 3), equal_to_1=(),
ids_of_folded_args=(), divisible_by_8=(3,))]},
    inductor_meta={'autotune_hints': set(), 'kernel_name':
'triton_poi_fused_add_0', 'mutated_arg_names': []},
    min_elem_per_thread=0
)
@triton.jit
def triton_(in_ptr0, in_ptr1, out_ptr0, xnumel, XBLOCK : tl.constexpr):
    xnumel = 33554432
    xoffset = tl.program_id(0) * XBLOCK
    xindex = xoffset + tl.arange(0, XBLOCK)[:]
    xmask = xindex < xnumel
    x3 = xindex
    x0 = xindex % 262144
    x1 = (xindex // 262144) % 32
    x2 = (xindex // 8388608)
    tmp0 = tl.load(in_ptr0 + (x3), None)
    tmp1 = tl.load(in_ptr1 + (x0 + (262144*x2) + (1048576*x1)), None)
    tmp2 = tmp0 + tmp1
    tl.store(out_ptr0 + (x3), tmp2, None)
''')

import triton
import triton.language as tl
from torch._inductor.triton_heuristics import grid, start_graph, end_graph
from torch._C import _cuda_getCurrentRawStream as get_cuda_stream


async_compile.wait(globals())
del async_compile
```

```
def call(args):
    arg0_1, arg1_1 = args
    args.clear()
    assert_size_stride(arg0_1, (4, 32, 2048, 128), (8388608, 262144, 128, 1))
    assert_size_stride(arg1_1, (32, 4, 2048, 128), (1048576, 262144, 128, 1))
    with torch.cuda._DeviceGuard(0):
        torch.cuda.set_device(0) # no-op to ensure context
        buf0 = empty_strided((32, 4, 2048, 128), (262144, 8388608, 128, 1),
device='cuda', dtype=torch.float32)
        # Source Nodes: [add], Original ATen: [aten.add]
        stream0 = get_cuda_stream(0)
        triton_poi_fused_add_0.run(arg0_1, arg1_1, buf0, 33554432,
grid=grid(33554432), stream=stream0)
        del arg0_1
        del arg1_1
        return (buf0, )


def benchmark_compiled_module(times=10, repeat=10):
    from torch._dynamo.testing import rand_strided
    from torch._inductor.utils import print_performance
    arg0_1 = rand_strided((4, 32, 2048, 128), (8388608, 262144, 128, 1),
device='cuda:0', dtype=torch.float32)
    arg1_1 = rand_strided((32, 4, 2048, 128), (1048576, 262144, 128, 1),
device='cuda:0', dtype=torch.float32)
    fn = lambda: call([arg0_1, arg1_1])
    return print_performance(fn, times=times, repeat=repeat)


if __name__ == "__main__":
    from torch._inductor.wrapper_benchmark import compiled_module_main
    compiled_module_main('None', benchmark_compiled_module)
```

[ ]:
```
import torch

def fn(x, y):
    return torch.permute(x, (2, 1, 0, 3)) + y
fnc = torch.compile(fn)

bsz = 4
num_head = 32
seq_len = 2048
head_dim = 128
x = torch.randn([bsz, num_head, seq_len, head_dim]).cuda()
y = torch.randn([seq_len, num_head, bsz, head_dim]).cuda()
z = fnc(x, y)
```

```
print(z[0,0,0,0])
```

```
tensor(-0.8373, device='cuda:0')
```

```
[ ]: !cat "/tmp/torchinductor_root/7r/
      ↪c7r2yifzq34wmhenriswpr5vbqfug7vhq5xlhi4x5ygntusgq6wb.py"
```

```
from ctypes import c_void_p, c_long
import torch
import math
import random
import os
import tempfile
from math import inf, nan
from torch._inductor.hooks import run_intermediate_hooks
from torch._inductor.utils import maybe_profile
from torch._inductor.codegen.memory_planning import _align as align

from torch import device, empty, empty_strided
from torch._inductor.codecache import AsyncCompile
from torch._inductor.select_algorithm import extern_kernels

aten = torch.ops.aten
inductor_ops = torch.ops.inductor
assert_size_stride = torch._C._dynamo.guards.assert_size_stride
alloc_from_pool = torch.ops.inductor._alloc_from_pool
reinterpret_tensor = torch.ops.inductor._reinterpret_tensor
async_compile = AsyncCompile()


# kernel path: /tmp/torchinductor_root/35/c3567yrrecffglcevq6lqmfcvht3nyv2trmfnu
kiy5cfypwgjeih.py
# Source Nodes: [add], Original ATen: [aten.add]
# add => add
triton_poi_fused_add_0 = async_compile.triton('triton_', '''
import triton
import triton.language as tl
from torch._inductor.ir import ReductionHint
from torch._inductor.ir import TileHint
from torch._inductor.triton_heuristics import AutotuneHint, pointwise
from torch._inductor.utils import instance_descriptor
from torch._inductor import triton_helpers

@pointwise(
    size_hints=[33554432],
    filename=__file__,
    triton_meta={'signature': {0: '*fp32', 1: '*fp32', 2: '*fp32', 3: 'i32'},
```

```
'device': 0, 'device_type': 'cuda', 'constants': {}, 'configs':
[instance_descriptor(divisible_by_16=(0, 1, 2, 3), equal_to_1=(),
ids_of_folded_args=(), divisible_by_8=(3,))]},
    inductor_meta={'autotune_hints': set(), 'kernel_name':
'triton_poi_fused_add_0', 'mutated_arg_names': []},
    min_elem_per_thread=0
)
@triton.jit
def triton_(in_ptr0, in_ptr1, out_ptr0, xnumel, XBLOCK : tl.constexpr):
    xnumel = 33554432
    xoffset = tl.program_id(0) * XBLOCK
    xindex = xoffset + tl.arange(0, XBLOCK)[:]
    xmask = xindex < xnumel
    x4 = xindex
    x0 = xindex % 128
    x1 = (xindex // 128) % 2048
    x2 = (xindex // 262144) % 32
    x3 = (xindex // 8388608)
    tmp0 = tl.load(in_ptr0 + (x4), None)
    tmp1 = tl.load(in_ptr1 + (x0 + (128*x3) + (512*x2) + (16384*x1)), None)
    tmp2 = tmp0 + tmp1
    tl.store(out_ptr0 + (x4), tmp2, None)
''')

import triton
import triton.language as tl
from torch._inductor.triton_heuristics import grid, start_graph, end_graph
from torch._C import _cuda_getCurrentRawStream as get_cuda_stream


async_compile.wait(globals())
del async_compile

def call(args):
    arg0_1, arg1_1 = args
    args.clear()
    assert_size_stride(arg0_1, (4, 32, 2048, 128), (8388608, 262144, 128, 1))
    assert_size_stride(arg1_1, (2048, 32, 4, 128), (16384, 512, 128, 1))
    with torch.cuda._DeviceGuard(0):
        torch.cuda.set_device(0) # no-op to ensure context
        buf0 = empty_strided((2048, 32, 4, 128), (128, 262144, 8388608, 1),
device='cuda', dtype=torch.float32)
        # Source Nodes: [add], Original ATen: [aten.add]
        stream0 = get_cuda_stream(0)
        triton_poi_fused_add_0.run(arg0_1, arg1_1, buf0, 33554432,
grid=grid(33554432), stream=stream0)
        del arg0_1
        del arg1_1
```

```
        return (buf0, )


def benchmark_compiled_module(times=10, repeat=10):
    from torch._dynamo.testing import rand_strided
    from torch._inductor.utils import print_performance
    arg0_1 = rand_strided((4, 32, 2048, 128), (8388608, 262144, 128, 1),
device='cuda:0', dtype=torch.float32)
    arg1_1 = rand_strided((2048, 32, 4, 128), (16384, 512, 128, 1),
device='cuda:0', dtype=torch.float32)
    fn = lambda: call([arg0_1, arg1_1])
    return print_performance(fn, times=times, repeat=repeat)


if __name__ == "__main__":
    from torch._inductor.wrapper_benchmark import compiled_module_main
    compiled_module_main('None', benchmark_compiled_module)
```