Unit Test for Fusion

测试数据指路

测试项目

Element-wise

1. Pointwise Chain

```
def pointwise_chain(x):
    y = torch.cos(x)
    x = x * y
    y = torch.max(x, y)
    x = x + y
    return x
```

Reduce

1. Softmax
2. Double Softmax

```
def double_softmax(x):
    return torch.softmax(torch.softmax(x, 1), 1)
```

3. Softmax Backward
4. LayerNorm
5. Norm Chain

```
def norm_chain(x):
    y = torch.sum(x, 1, keepdim=True)
    x = x * y
    y = torch.sum(x, 1, keepdim=True)
    x = x * y
    y = torch.sum(x, 1, keepdim=True)
    x = x * y
```

```
    return x
```

Horizontal

1. Horizontal Reduction Pointwise

```
def horizontal_reduction_pointwise(a):
    b = torch.sum(a, dim=1)
    c = torch.cos(a)
    return b, c
```

2. Horizontal Reduction Reduction

```
def horizontal_reduction_reduction(a):
    b = torch.sum(a, dim=1)
    c = torch.amax(a, dim=1)
    return b, c
```

Matmul

1. matmul+add
2. matmul+relu
3. matmul+add+relu
4. matmul+matmul

Convolution

1. conv+add
2. conv+relu
3. conv+bn

Mix

1. attention
    a. FlashAttention2
    b. Memory efficient attention, xformers.ops.fmha.memory_efficient_attention,reference: https://github.com/hpcaitech/ColossalAI/blob/main/colossalai/kernel/cuda_native/mha/mem_eff_attn.py

2. Fused normalization（常见conv+bn融合，已经包括在conv类下，没有发现其他的融合场景）
3. bias_dropout_add

```python
def bias_dropout_add(inp, bias, residual):
    # type: (Tensor, Tensor, Tensor, float, bool) -> Tensor
    out = torch.nn.functional.dropout(inp + bias, p=0.5, training=False)
    out = residual + out
    return out
```

4. bias_gelu

```python
def bias_gelu(inp, bias):
    x = inp + bias
    return  x * 0.5 * (1.0 + torch.tanh(0.79788456 * x * (1 + 0.044715 * x * x)))
```

测试指标

1. global memory 读写量（对于 torch compile 可测，但是非 compile 的原函数执行不好测）
2. buffer size (memory footprint);
3. compile time ;
4. number of kernels
5. run time;
6. numeric accuracy;
7. flops（fvcore工具可辅助，但是计算方式需要验证）
8. instruction number
9. memory peak

测试环境

RTX3090

1. 硬件：RTX3090单卡，内存24GB，位于120.92.72.3的cuda:2
2. 软件：Python 3.10.9，Cuda 12.2，PyTorch 2.1.0.dev20230814+cu121，Triton 2.1.0+e6216047b8，Pytest 7.4.0，Benchmark 4.0.0.
3. 参数：数据精度torch.float32，attention为float16

A100

1. 因为在a100的2307镜像上控制版本会把环境搞炸，所以直接使用镜像环境，与3090有一定差异
2. 硬件：A100单卡，内存40GB，位于九鼎平台
3. 软件：python 3.10.6，Cuda 12.1，Pytorch 2.1.0a0+b5021ba，Triton 2.1.0
4. 参数：数据精度torch.float32，attention为float16

测试数据：eager & inductor延时比较

第一版：pytest-benchmark

> ❗ 以下测试结果为pytest-benchmark方法，内含了算子运行前后torch.cuda.synchronize与torch.cuda.empty_cache的开销，参考价值有限，需要进一步测定

方法修正

1. Pytest benchmark不会自动删除计算结果的缓存，导致大批量形状先后测试时，靠后的批次会出现oom，可以通过在包装测试函数时不返回结果来避免这种情况（但是函数定义时一定需要返回，否则编译后的函数可能省略执行过程，导致输出异常）（后来的事实证明这种办法不总是奏效）
2. 再次完善测定方式后基本没有出现异常点，double softmax中的128*16384处的性能在eager和triton各自的耗时图中都属于正常的渐进
3. 在多个算子上都出现了相对性能劣化的分界线，有的也伴随着绝对性能相比更大形状上反而更差的情况，推测与gpu cache容量有关，需要更换环境补充实验证明（rtx3090 l1 cache 128KB, a100 l1 cache 192KB）（在a100上的少量实验暂时没有观察到这个现象，且面临更难办的问题）

结果汇总

1. Softmax

Softmax Median Relative Overhead

dim0\dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	5.03E+00	3.82E+00	3.84E+00	3.62E+00	1.43E+01	3.70E+00	3.32E+00	3.70E+00	3.05E+00	1.27E+00
256	3.77E+00	3.82E+00	3.80E+00	1.27E+01	3.59E+00	2.19E+00	2.79E+00	1.86E+00	1.71E+00	1.02E+00
512	3.77E+00	3.80E+00	1.41E+01	3.46E+00	2.48E+00	2.11E+00	1.84E+00	1.35E+00	1.25E+00	1.13E+00
1024	3.84E+00	1.41E+01	3.77E+00	2.42E+00	2.05E+00	1.95E+00	1.38E+00	1.13E+00	1.07E+00	1.05E+00
2048	1.44E+01	3.79E+00	2.56E+00	2.16E+00	1.07E+00	1.80E+00	1.22E+00	1.10E+00	1.07E+00	1.02E+00
4096	3.67E+00	2.67E+00	1.94E+00	1.05E+00	1.05E+00	1.75E+00	1.15E+00	1.10E+00	1.07E+00	1.02E+00
8192	2.61E+00	1.95E+00	1.06E+00	1.04E+00	9.99E-01	1.71E+00	1.08E+00	1.05E+00	1.04E+00	1.02E+00
16384	1.92E+00	1.05E+00	1.05E+00	9.98E-01	1.00E+00	1.65E+00	1.05E+00	1.03E+00	1.02E+00	1.01E+00
32768	1.04E+00	1.03E+00	1.01E+00	1.00E+00	1.25E+00	1.18E+00	1.02E+00	1.00E+00	9.74E-01	1.00E+00
65536	9.98E-01	1.01E+00	9.99E-01	1.01E+00	1.09E+00	9.87E-01	1.04E+00	1.00E+00	1.00E+00	#VALUE!

2. Layernorm

Layernorm: Median Relative Overhead										
dim0\dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	6.06E+00	4.16E+00	4.15E+00	4.14E+00	2.30E+00	4.33E+00	4.03E+00	1.16E+00	1.10E+00	1.06E+00
256	4.42E+00	4.16E+00	4.10E+00	1.76E+00	4.33E+00	4.35E+00	2.91E+00	1.11E+00	1.07E+00	1.03E+00
512	4.24E+00	4.15E+00	1.76E+00	4.10E+00	4.38E+00	3.00E+00	1.53E+00	1.06E+00	1.04E+00	1.02E+00
1024	4.21E+00	1.76E+00	4.10E+00	3.61E+00	2.07E+00	1.52E+00	1.27E+00	1.04E+00	1.02E+00	1.01E+00
2048	1.76E+00	4.13E+00	3.60E+00	2.40E+00	9.92E-01	1.18E+00	9.98E-01	1.02E+00	1.01E+00	1.00E+00
4096	3.70E+00	3.31E+00	2.62E+00	1.04E+00	9.87E-01	1.03E+00	1.01E+00	1.01E+00	1.01E+00	1.00E+00
8192	2.63E+00	2.41E+00	1.05E+00	1.00E+00	9.99E-01	9.28E-01	1.01E+00	1.01E+00	1.00E+00	1.00E+00
16384	1.72E+00	1.03E+00	1.01E+00	1.00E+00	9.99E-01	9.02E-01	1.00E+00	1.00E+00	1.00E+00	1.00E+00
32768	9.65E-01	9.98E-01	1.01E+00	9.99E-01	9.23E-01	1.08E+00	9.43E-01	1.00E+00	2.62E+00	1.00E+00
65536	9.27E-01	9.97E-01	1.00E+00	9.79E-01	9.77E-01	9.19E-01	1.01E+00	9.57E-01	1.00E+00	OOM

3. Horizontal Reduction Pointwise (sum & cos)

Horizontal Reduction Pointwise Median Relative Overhead										
dim0\dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	3.36E+00	2.48E+00	2.47E+00	2.47E+00	2.28E+00	3.31E-01	3.34E-01	4.12E-01	5.13E-01	4.44E-01
256	3.55E+00	2.46E+00	2.45E+00	1.34E+00	5.56E-01	3.37E-01	3.53E-01	4.00E-01	5.21E-01	5.36E-01
512	2.42E+00	2.46E+00	1.32E+00	3.77E-01	5.57E-01	5.18E-01	3.81E-01	4.37E-01	5.07E-01	5.96E-01
1024	2.41E+00	1.34E+00	3.73E-01	3.75E-01	5.27E-01	5.45E-01	4.39E-01	5.06E-01	5.87E-01	6.49E-01
2048	1.32E+00	3.80E-01	3.76E-01	4.12E-01	8.70E-01	5.81E-01	5.11E-01	5.81E-01	6.39E-01	6.71E-01
4096	3.71E-01	3.76E-01	4.04E-01	7.79E-01	9.20E-01	6.05E-01	5.79E-01	6.33E-01	6.67E-01	6.95E-01
8192	3.71E-01	3.97E-01	7.76E-01	8.56E-01	8.36E-01	6.36E-01	6.21E-01	6.66E-01	6.79E-01	7.04E-01
16384	3.89E-01	7.79E-01	8.57E-01	8.25E-01	6.18E-01	6.43E-01	6.68E-01	6.80E-01	6.86E-01	7.06E-01
32768	7.80E-01	8.59E-01	8.26E-01	6.14E-01	7.46E-01	8.97E-01	6.24E-01	4.67E-01	3.59E-01	1.00E+00
65536	8.61E-01	8.30E-01	6.14E-01	7.47E-01	5.37E-01	6.97E-01	4.68E-01	3.74E-01	9.97E-01	#VALUE!

4. Horizontal Reduction Reduction (sum & amax)

Horizontal Reduction Reduction Median Relative Overhead										
dim0\dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	2.07E+00	2.31E+00	2.31E+00	2.30E+00	2.31E+00	3.67E-01	3.66E-01	4.25E-01	5.09E-01	3.90E-01
256	2.25E+00	2.31E+00	2.29E+00	2.30E+00	3.64E-01	3.66E-01	3.55E-01	3.85E-01	4.24E-01	3.95E-01
512	2.26E+00	2.31E+00	2.30E+00	3.67E-01	3.66E-01	3.39E-01	3.60E-01	3.86E-01	4.16E-01	4.50E-01
1024	2.27E+00	2.29E+00	3.66E-01	3.65E-01	3.47E-01	3.56E-01	3.82E-01	4.16E-01	4.45E-01	4.68E-01
2048	2.27E+00	3.62E-01	3.66E-01	3.47E-01	3.57E-01	3.79E-01	4.15E-01	4.46E-01	4.69E-01	4.83E-01
4096	3.58E-01	3.66E-01	3.46E-01	3.51E-01	3.75E-01	4.03E-01	4.56E-01	4.82E-01	4.98E-01	4.92E-01
8192	3.69E-01	3.46E-01	3.57E-01	3.79E-01	4.11E-01	4.38E-01	4.70E-01	4.85E-01	4.94E-01	4.96E-01
16384	4.24E-01	3.59E-01	3.84E-01	4.15E-01	4.44E-01	4.66E-01	4.83E-01	4.91E-01	4.96E-01	4.98E-01
32768	5.06E-01	3.85E-01	4.17E-01	4.46E-01	4.69E-01	4.82E-01	4.92E-01	4.96E-01	4.98E-01	4.98E-01
65536	6.38E-01	4.19E-01	4.48E-01	4.69E-01	4.83E-01	4.91E-01	4.96E-01	4.98E-01	4.98E-01	4.97E-01

5. Double Softmax (softmax + softmax)

Double Softmax Median Relative Overhead										
dim0\dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	2.40E+00	2.63E+00	2.61E+00	2.59E+00	2.39E+00	2.67E+00	2.54E+00	4.55E+00	3.60E+00	1.10E+00
256	2.57E+00	2.61E+00	2.60E+00	2.59E+00	1.44E+00	1.79E+00	2.51E+00	1.86E+00	1.82E+00	7.72E-01
512	2.58E+00	2.61E+00	2.60E+00	2.35E+00	9.63E-01	1.84E+00	1.50E+00	1.17E+00	1.14E+00	8.50E-01
1024	2.56E+00	2.60E+00	2.57E+00	1.88E+00	7.39E-01	1.51E+00	9.83E-01	8.46E-01	8.25E-01	8.14E-01
2048	2.58E+00	2.58E+00	1.90E+00	1.37E+00	6.49E-01	1.43E+00	9.15E-01	8.48E-01	8.40E-01	7.76E-01
4096	2.57E+00	1.90E+00	1.14E+00	9.84E-01	5.20E-01	1.37E+00	8.76E-01	8.59E-01	8.51E-01	7.61E-01
8192	1.72E+00	1.12E+00	8.53E-01	9.14E-01	4.34E-01	1.30E+00	8.31E-01	8.02E-01	7.99E-01	7.55E-01
16384	1.04E+00	8.84E-01	7.03E-01	8.48E-01	4.00E-01	1.26E+00	8.00E-01	7.78E-01	7.75E-01	7.54E-01
32768	8.01E-01	7.73E-01	6.32E-01	8.10E-01	3.78E-01	1.25E+00	7.90E-01	7.69E-01	7.65E-01	#VALUE!
65536	6.60E-01	7.20E-01	6.04E-01	7.95E-01	7.43E-01	1.22E+00	7.81E-01	7.61E-01	#VALUE!	#VALUE!

6. Softmax Backward

Softmax Backward Median Relative Overhead										
dim0\dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	3.21E+00	2.40E+00	2.39E+00	2.38E+00	3.08E-01	2.30E+00	1.71E+00	1.37E+00	1.12E+00	5.25E-01
256	3.43E+00	2.38E+00	2.38E+00	3.49E-01	2.30E+00	1.56E+00	1.20E+00	9.57E-01	8.35E-01	4.60E-01
512	2.33E+00	2.39E+00	3.48E-01	1.96E+00	1.54E+00	1.21E+00	9.86E-01	8.43E-01	7.77E-01	4.00E-01
1024	2.33E+00	3.47E-01	1.97E+00	1.38E+00	1.10E+00	1.02E+00	8.72E-01	7.88E-01	7.53E-01	6.25E-01
2048	3.36E-01	1.97E+00	1.28E+00	9.95E-01	5.88E-01	8.54E-01	8.02E-01	7.62E-01	7.38E-01	5.16E-01
4096	1.93E+00	1.24E+00	9.78E-01	5.25E-01	5.32E-01	8.14E-01	7.63E-01	7.47E-01	7.35E-01	3.46E-01
8192	1.21E+00	9.73E-01	5.10E-01	5.29E-01	4.50E-01	7.97E-01	7.34E-01	7.24E-01	7.21E-01	#VALUE!
16384	9.77E-01	5.02E-01	5.17E-01	4.48E-01	4.00E-01	6.25E-01	5.31E-01	2.81E-01	2.40E-01	#VALUE!
32768	5.02E-01	5.03E-01	4.43E-01	5.00E-01	3.92E-01	5.15E-01	2.62E-01	2.40E-01	1.65E-01	#VALUE!
65536	5.00E-01	4.45E-01	5.00E-01	4.46E-01	6.03E-01	2.69E-01	2.41E-01	1.63E-01	#VALUE!	#VALUE!

7. Norm Chain

Norm Chain Median Relative Overhead										
dim0\dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	8.44E-01	9.14E-01	9.13E-01	9.10E-01	1.07E+00	1.61E-01	2.58E-01	5.94E-01	7.95E-01	1.30E-01
256	8.97E-01	9.04E-01	9.09E-01	1.04E+00	2.68E-01	1.74E-01	3.77E-01	4.96E-01	5.97E-01	1.28E-01
512	8.98E-01	9.36E-01	1.05E+00	2.70E-01	2.73E-01	2.42E-01	3.84E-01	4.54E-01	5.09E-01	2.08E-01
1024	8.91E-01	1.05E+00	2.71E-01	2.75E-01	8.44E-02	3.30E-01	4.27E-01	4.76E-01	5.13E-01	2.25E-01
2048	1.04E+00	2.68E-01	2.71E-01	8.16E-02	4.49E-01	3.93E-01	4.75E-01	5.14E-01	5.37E-01	2.24E-01
4096	2.63E-01	2.70E-01	7.32E-02	4.25E-01	4.70E-01	4.66E-01	5.25E-01	5.50E-01	5.65E-01	1.12E-01
8192	2.64E-01	7.15E-02	4.17E-01	4.45E-01	4.10E-01	5.13E-01	5.42E-01	5.58E-01	5.64E-01	9.05E-02
16384	7.03E-02	4.14E-01	4.40E-01	4.07E-01	3.36E-01	2.96E-01	3.31E-01	3.14E-01	2.59E-01	#VALUE!
32768	4.14E-01	4.34E-01	4.06E-01	3.36E-01	5.55E-01	3.70E-01	3.39E-01	2.69E-01	2.26E-01	#VALUE!
65536	4.34E-01	4.07E-01	3.36E-01	5.55E-01	3.67E-01	3.42E-01	2.72E-01	2.21E-01	#VALUE!	#VALUE!

8. Pointwise Chain

Norm Chain Median Relative Overhead										
dim0\dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	1.29E+00	1.43E+00	1.43E+00	1.45E+00	1.58E+00	1.36E+00	1.03E+00	5.44E-01	3.74E-01	4.29E-02
256	1.39E+00	1.42E+00	1.45E+00	1.18E+00	1.48E+00	1.06E+00	5.43E-01	3.76E-01	2.81E-01	4.08E-02
512	1.41E+00	1.44E+00	1.18E+00	1.42E+00	1.07E+00	5.38E-01	3.77E-01	2.84E-01	2.33E-01	4.35E-02
1024	1.41E+00	1.19E+00	1.43E+00	7.48E-01	1.05E-01	3.78E-01	2.85E-01	2.35E-01	2.09E-01	3.65E-02
2048	1.16E+00	1.41E+00	7.51E-01	8.85E-02	3.23E-01	2.86E-01	2.35E-01	2.09E-01	1.96E-01	3.28E-02
4096	1.39E+00	7.47E-01	8.82E-02	3.26E-01	3.32E-01	2.35E-01	2.10E-01	1.97E-01	1.88E-01	3.50E-02
8192	7.26E-01	8.81E-02	3.24E-01	3.23E-01	3.06E-01	2.10E-01	1.97E-01	1.90E-01	1.86E-01	#VALUE!
16384	8.69E-02	3.25E-01	3.21E-01	3.06E-01	2.50E-01	1.06E-01	9.24E-02	1.01E-01	4.52E-02	#VALUE!
32768	3.26E-01	3.22E-01	3.06E-01	2.50E-01	3.01E-01	1.79E-01	1.87E-01	1.44E-01	1.06E-01	#VALUE!
65536	3.23E-01	3.06E-01	2.50E-01	3.06E-01	1.68E-01	1.87E-01	1.49E-01	2.96E-01	#VALUE!	#VALUE!

9. MultiheadAttention（attention共使用4个维度，batch_size = 8, num_heads = 8）

emben_dim\nu	16	32	64	128	256	512	1024	2048	4096	8192
128	1.20E+00	1.25E+00	1.24E+00	1.26E+00	1.16E+00	8.49E-01	3.97E-01	9.67E-01	9.71E-01	9.86E-01
256	1.26E+00	1.25E+00	1.24E+00	1.14E+00	8.91E-01	1.27E+00	7.53E-01	9.97E-01	9.84E-01	9.92E-01
512	1.28E+00	1.25E+00	1.14E+00	8.54E-01	1.96E+00	1.20E+00	9.26E-01	9.98E-01	9.93E-01	1.00E+00
1024	1.19E+00	1.14E+00	8.58E-01	1.13E+00	1.08E+00	6.77E-01	6.77E-01	9.82E-01	1.00E+00	1.00E+00
2048	1.17E+00	1.08E+00	4.93E-01	1.06E+00	1.04E+00	7.88E-01	8.44E-01	9.97E-01	1.00E+00	9.95E-01
4096	1.09E+00	1.10E+00	1.07E+00	1.04E+00	7.96E-01	8.55E-01	8.91E-01	1.00E+00	1.00E+00	1.00E+00
8192	9.36E-01	9.12E-01	7.10E-01	9.26E-01	8.68E-01	7.08E-01	1.00E+00	1.00E+00	1.00E+00	1.00E+00
16384	8.99E-01	9.27E-01	8.26E-01	8.28E-01	8.04E-01	6.72E-01	1.10E+00	1.00E+00	1.00E+00	1.00E+00
32768	9.39E-01	8.45E-01	8.37E-01	8.24E-01	7.81E-01	8.89E-01	1.00E+00	1.01E+00	1.00E+00	9.99E-01
65536	8.84E-01	8.37E-01	7.94E-01	8.46E-01	6.56E-01	7.76E-01	1.02E+00	1.01E+00	9.99E-01	#VALUE!

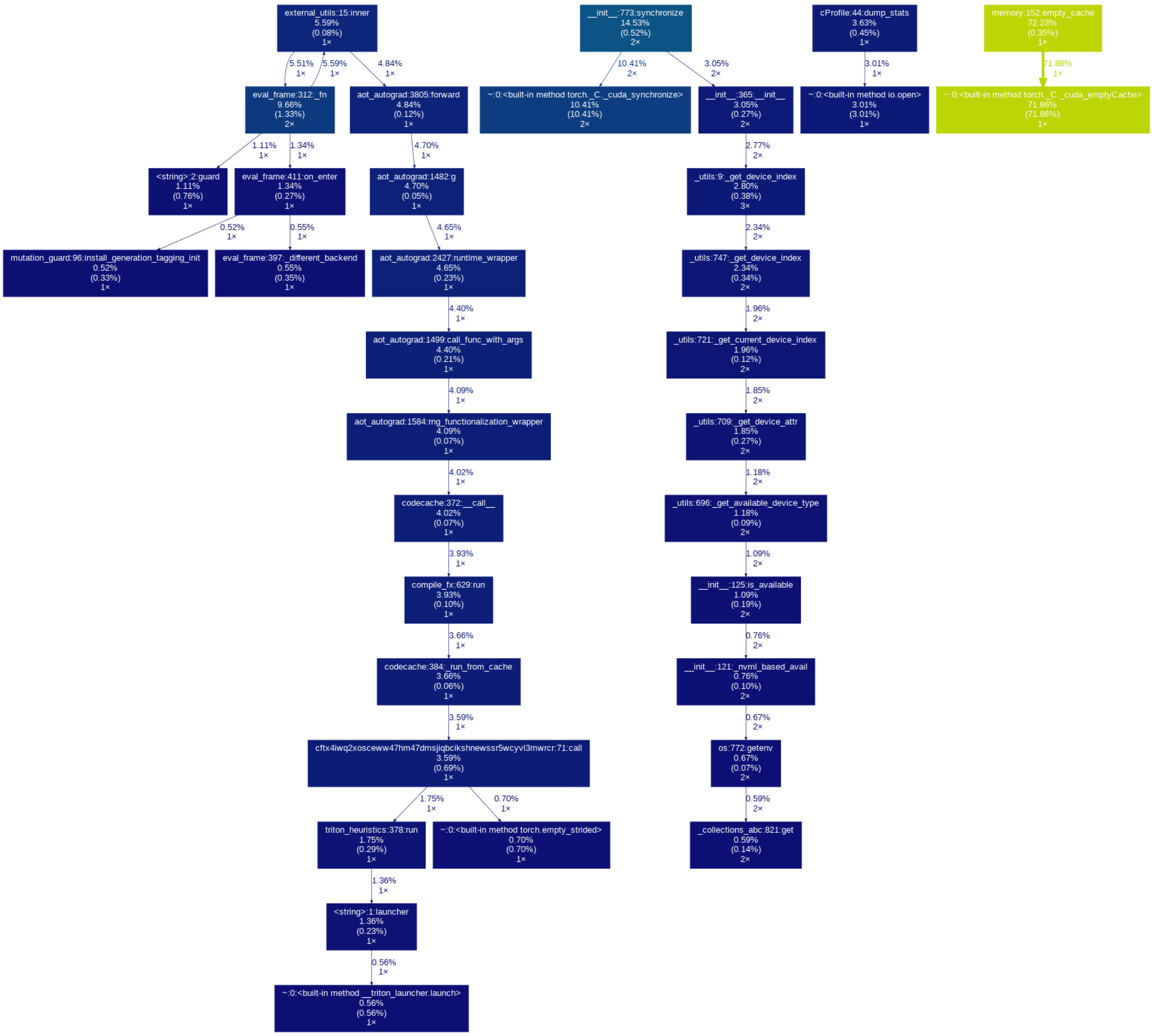
第二版：without empty_cache

> ❗ 去除了empty_cache时间，但是synchronize时间仍有影响，尤其是小形状张量，以下数据仅用于展示差异，不作为最终分析的依据
>
> 但去除synchronize包装的函数在直接送入benchmark时会使测试异常缓慢，原因不明
>
> 计时单位为s

FusionTests

方法修正

1. 经过profile监测，发现torch.cuda.empty_cache和torch.cuda.synchronize在运行时间中的占比非常高，远远超过了kernel本身的执行时间。如图，左1为函数执行，占比5.59%；左2为cuda.synchronize，占比14.53%；右二为profile输出状态，在benchmark中不出现，无需关心；右一为cuda.empty_cache，占比最高，达到了72.23%。



2. 仍然未解决的问题是，大批量形状依旧会造成最后批次更容易oom，empty_cache没有起到作用，暂未明确判断原因是来自torch还是benchmark。

第三版：triton.testing.do_bench

> ❗ 经抽样对比，triton.testing.do_bench测得时间与nsys输出的gpu kernel时间相近，目前认为是最准确的测定方式
>
> 计时单位为us

FusionTests-dobench

方法修正：do_bench利用GPU异步

- 普通的cuda event计时：gpu上空闲时间长，任务松散，cpu发射队列在gpu上并非连续执行，记录时间中含有大量空转

GPU	start.record		Kernel launch				end.record
CPU	start.record	Kernel launch	end.record				

- 加入cache.zero_的计时：gpu被zero占用后一直繁忙，任务排列紧密，cpu发射到gpu上后在zero的空隙中被连续执行，记录时间更加精确

GPU	zero	zero	zero	zero	start.record	Kernel launch	end.record
CPU	start.record	Kernel launch	end.record				

结果汇总

1. Softmax

Relative Performance										
dim0 \ dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	125.00%	103.13%	100.00%	100.00%	77.50%	66.67%	51.85%	30.09%	35.55%	34.80%
256	68.23%	62.50%	54.55%	47.06%	25.00%	17.40%	15.46%	22.34%	21.93%	22.25%
512	83.33%	62.50%	63.64%	58.82%	30.95%	25.24%	33.17%	40.30%	39.65%	39.40%
1024	71.43%	87.50%	75.00%	69.24%	44.83%	41.88%	60.52%	67.46%	67.39%	67.42%
2048	87.50%	90.00%	81.25%	57.50%	60.24%	56.42%	84.75%	89.62%	90.29%	90.51%
4096	75.00%	75.00%	72.73%	55.80%	65.16%	57.60%	90.68%	93.37%	93.31%	93.52%
8192	75.00%	82.03%	64.62%	54.36%	67.53%	57.75%	91.29%	92.69%	92.58%	92.84%
16384	88.10%	85.39%	64.29%	51.60%	65.72%	56.53%	89.48%	90.20%	89.46%	90.10%
32768	93.33%	86.17%	65.02%	52.51%	67.74%	57.67%	90.68%	91.47%	91.02%	#VALUE!
65536	14.46%	27.43%	51.40%	93.68%	118.20%	84.02%	103.29%	99.64%	#VALUE!	#VALUE!

2. Layernorm

Relative Performance										
dim0 \ dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	112.50%	112.50%	112.50%	111.50%	91.67%	122.40%	127.78%	84.31%	84.95%	87.72%
256	112.50%	128.57%	102.43%	102.50%	80.93%	104.65%	105.71%	101.52%	98.39%	99.56%
512	98.26%	103.23%	113.89%	108.12%	84.83%	119.54%	104.76%	100.81%	100.42%	100.42%
1024	111.11%	122.22%	110.34%	112.50%	91.43%	116.07%	102.69%	100.84%	100.00%	99.47%
2048	108.33%	116.67%	101.12%	115.25%	100.00%	118.27%	101.69%	100.64%	100.53%	100.06%
4096	112.50%	117.65%	104.33%	121.74%	103.57%	118.32%	100.86%	100.32%	100.70%	100.35%
8192	116.67%	118.52%	106.52%	121.18%	108.41%	117.51%	100.86%	100.44%	100.51%	100.56%
16384	116.67%	119.23%	110.59%	131.90%	108.49%	119.36%	100.82%	100.38%	100.67%	100.52%
32768	118.18%	123.26%	113.50%	135.45%	109.05%	119.68%	100.73%	100.43%	100.71%	#VALUE!
65536	120.41%	126.83%	115.99%	134.82%	108.87%	117.58%	100.84%	100.45%	#VALUE!	#VALUE!

3. Horizontal Reduction Pointwise

Relative Performance										
dim0 \ dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	175.00%	175.00%	161.87%	166.67%	144.44%	146.15%	86.36%	85.21%	83.39%	82.43%
256	140.00%	119.79%	112.50%	83.33%	70.00%	64.86%	52.24%	50.40%	49.80%	48.80%
512	143.75%	114.29%	100.00%	92.31%	90.48%	97.37%	92.86%	90.44%	88.48%	87.17%
1024	133.33%	128.57%	120.00%	120.00%	134.62%	135.42%	134.07%	130.22%	130.22%	130.28%
2048	128.57%	133.33%	120.00%	136.00%	142.22%	137.08%	136.52%	132.10%	133.67%	131.56%
4096	133.33%	126.67%	138.79%	140.36%	137.36%	129.95%	123.02%	120.89%	120.66%	119.63%
8192	141.74%	150.00%	144.44%	143.02%	138.54%	133.52%	126.47%	124.68%	124.08%	123.05%
16384	150.00%	147.73%	145.24%	142.77%	139.14%	134.49%	127.72%	125.53%	125.60%	125.33%
32768	147.73%	148.78%	146.30%	143.38%	139.91%	134.06%	128.76%	126.34%	126.30%	#VALUE!
65536	24.80%	46.75%	87.43%	147.68%	147.23%	147.48%	146.85%	144.67%	#VALUE!	#VALUE!

4. Horizontal Reduction Reduction

Relative Performance										
dim0 \ dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	225.00%	200.00%	181.88%	197.92%	228.57%	232.81%	106.25%	106.90%	98.04%	92.55%
256	160.00%	134.37%	112.50%	109.09%	100.00%	92.86%	62.75%	56.38%	51.96%	49.15%
512	133.33%	150.00%	125.00%	118.18%	111.76%	123.33%	96.33%	90.72%	88.65%	86.30%
1024	150.00%	142.86%	150.00%	141.67%	160.00%	155.88%	150.00%	149.53%	150.98%	150.88%
2048	125.00%	133.33%	141.67%	182.35%	172.20%	181.63%	186.05%	186.06%	189.87%	193.80%
4096	118.18%	130.14%	168.42%	176.67%	189.80%	200.11%	192.14%	194.36%	197.18%	198.15%
8192	120.00%	142.86%	161.29%	176.00%	188.51%	194.51%	192.63%	192.86%	192.97%	192.16%
16384	134.78%	148.12%	167.31%	180.90%	191.93%	196.17%	197.01%	198.40%	199.07%	199.60%
32768	128.21%	155.36%	176.92%	186.42%	193.57%	195.11%	197.16%	199.07%	199.47%	200.13%
65536	24.24%	43.63%	82.18%	149.63%	198.66%	200.00%	199.58%	199.89%	200.10%	200.24%

5. Double Softmax

	Relative Performance									
dim0 \ dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	160.00%	116.67%	127.23%	120.00%	80.00%	63.75%	52.17%	22.19%	28.14%	27.90%
256	77.78%	60.94%	52.94%	44.83%	18.07%	14.13%	10.44%	15.75%	15.50%	15.44%
512	88.89%	66.67%	60.42%	51.72%	26.67%	17.86%	24.01%	29.14%	28.75%	28.61%
1024	88.89%	83.33%	72.22%	67.74%	38.76%	33.22%	48.44%	53.77%	53.69%	53.82%
2048	83.33%	75.78%	79.17%	53.31%	58.87%	54.92%	83.82%	89.20%	89.43%	89.47%
4096	68.42%	67.86%	97.78%	75.00%	58.11%	53.99%	85.38%	88.16%	88.34%	88.53%
8192	61.29%	95.65%	99.03%	77.78%	63.84%	56.38%	85.42%	87.52%	87.49%	87.61%
16384	83.02%	105.06%	108.78%	85.91%	69.10%	58.82%	94.11%	95.84%	95.69%	95.58%
32768	85.71%	109.52%	116.96%	91.12%	70.92%	61.49%	98.50%	100.73%	100.55%	#VALUE!
65536	13.73%	26.18%	49.44%	94.10%	225.66%	154.40%	141.80%	133.19%	#VALUE!	#VALUE!

6. Softmax Backward

	Relative Performance									
dim0 \ dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	150.00%	145.45%	140.00%	133.33%	132.29%	121.43%	106.45%	108.06%	113.11%	115.00%
256	120.00%	116.67%	88.89%	78.57%	62.96%	57.14%	63.81%	67.47%	67.48%	66.81%
512	116.67%	114.29%	111.11%	94.12%	96.97%	100.00%	106.35%	110.04%	110.97%	110.81%
1024	133.33%	136.33%	145.45%	129.17%	136.10%	125.74%	133.97%	136.46%	136.85%	137.48%
2048	137.50%	145.45%	140.91%	138.86%	133.33%	127.18%	134.52%	136.05%	135.75%	135.48%
4096	133.33%	147.62%	136.17%	126.04%	129.23%	122.52%	129.52%	129.30%	128.89%	128.63%
8192	163.16%	160.00%	134.57%	126.06%	131.09%	124.05%	131.42%	131.32%	130.75%	130.04%
16384	177.01%	148.78%	129.51%	122.83%	128.97%	122.72%	130.20%	130.64%	129.66%	128.52%
32768	182.09%	150.96%	131.05%	123.08%	131.27%	123.84%	133.86%	134.08%	132.86%	#VALUE!
65536	63.03%	119.69%	200.71%	178.38%	180.59%	140.69%	140.67%	140.41%	#VALUE!	#VALUE!

7. Norm Chain

	Relative Performance									
dim0 \ dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	450.00%	340.00%	380.00%	328.57%	323.26%	307.14%	188.00%	76.38%	73.62%	71.26%
256	283.33%	257.14%	216.22%	177.23%	109.09%	102.54%	60.61%	60.00%	57.96%	56.35%
512	300.00%	265.50%	220.00%	206.47%	150.00%	116.30%	109.33%	103.94%	100.70%	98.66%
1024	285.71%	275.00%	245.45%	288.24%	175.00%	166.09%	162.39%	156.89%	155.01%	153.48%
2048	253.12%	245.45%	311.39%	246.15%	197.87%	182.50%	181.54%	176.97%	174.68%	173.61%
4096	207.14%	266.67%	306.25%	202.11%	194.27%	186.70%	174.75%	172.06%	170.05%	169.48%
8192	252.84%	343.33%	265.75%	202.76%	188.53%	181.11%	175.38%	173.02%	171.47%	170.40%
16384	332.80%	366.04%	251.72%	186.54%	178.69%	175.62%	170.45%	169.32%	168.34%	168.07%
32768	354.55%	370.71%	246.85%	185.81%	179.84%	176.53%	173.33%	171.49%	170.46%	#VALUE!
65536	62.96%	116.96%	217.16%	389.39%	382.30%	286.07%	188.49%	181.40%	#VALUE!	#VALUE!

8. Pointwise Chain

	Relative Performance									
dim0 \ dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	250.00%	225.00%	250.00%	240.00%	233.33%	222.22%	385.71%	503.66%	516.28%	530.49%
256	225.00%	250.00%	240.00%	233.33%	219.44%	385.71%	503.66%	516.28%	530.49%	536.88%
512	250.00%	240.00%	233.33%	223.00%	385.71%	502.30%	516.64%	530.49%	536.88%	541.90%
1024	240.00%	233.33%	222.22%	385.71%	483.33%	518.31%	530.49%	536.88%	541.59%	543.36%
2048	233.33%	250.00%	385.71%	502.30%	518.60%	530.49%	536.88%	541.59%	543.36%	544.06%
4096	250.00%	385.71%	483.33%	518.60%	530.49%	536.88%	541.59%	543.36%	544.58%	544.85%
8192	385.71%	502.98%	518.60%	530.49%	536.88%	541.61%	543.52%	544.14%	544.69%	545.04%
16384	502.30%	518.60%	530.49%	536.88%	541.59%	543.52%	544.20%	544.69%	544.85%	545.92%
32768	516.28%	530.49%	536.88%	541.29%	543.68%	544.30%	545.29%	545.56%	546.29%	#VALUE!
65536	530.49%	536.88%	541.27%	543.68%	544.66%	545.03%	545.81%	545.58%	#VALUE!	#VALUE!

9. Attention

Relative Performance (headdim=64, nheads=32)							
triton / eager	128	256	512	1024	2048	4096	8192
	177.43%	140.00%	139.67%	155.73%	152.09%	125.73%	133.13%
	125.58%	132.98%	131.46%	132.71%	130.81%	119.38%	#VALUE!
	30.99%	37.48%	50.85%	82.63%	136.32%	132.66%	#VALUE!
	27.61%	34.66%	49.69%	81.41%	134.67%	#VALUE!	#VALUE!
	25.20%	33.43%	49.16%	81.04%	133.88%	#VALUE!	#VALUE!
	24.25%	33.02%	48.85%	80.70%	#VALUE!	#VALUE!	#VALUE!
flash2 / triton							
	131.51%	222.22%	302.50%	347.53%	529.39%	668.88%	769.77%
	307.14%	310.42%	398.51%	524.57%	672.45%	722.77%	743.73%
	1352.38%	1239.58%	1151.79%	918.31%	670.28%	659.94%	#VALUE!
	1569.44%	1460.49%	1305.58%	963.10%	686.40%	#VALUE!	#VALUE!
	1733.85%	1600.68%	1371.31%	975.61%	693.93%	#VALUE!	#VALUE!
	1853.20%	1663.96%	1394.86%	981.06%	#VALUE!	#VALUE!	#VALUE!
flash2 / eager							
	233.33%	311.11%	422.50%	541.23%	805.17%	841.00%	1024.77%
	385.71%	412.80%	523.88%	696.17%	879.63%	862.80%	#VALUE!
	419.05%	464.58%	585.71%	758.84%	913.72%	875.50%	#VALUE!
	433.33%	506.17%	648.73%	784.03%	924.35%	#VALUE!	#VALUE!
	436.92%	535.14%	674.19%	790.68%	929.04%	#VALUE!	#VALUE!
	449.36%	549.47%	681.35%	791.71%	#VALUE!	#VALUE!	#VALUE!

Relative Performance (headdim=128, nheads=16)							
triton / eager	128	256	512	1024	2048	4096	8192
	166.67%	132.26%	137.50%	147.55%	141.09%	122.57%	118.26%
	129.73%	127.78%	127.57%	127.56%	126.26%	117.08%	128.46%
	44.44%	48.89%	59.92%	88.35%	130.83%	127.03%	#VALUE!
	39.75%	45.37%	58.35%	86.77%	129.69%	125.99%	#VALUE!
	36.34%	42.78%	57.64%	86.27%	129.07%	#VALUE!	#VALUE!
	34.17%	41.61%	57.38%	85.58%	128.00%	#VALUE!	#VALUE!
flash2 / triton							
	136.36%	182.35%	186.05%	244.78%	351.93%	413.43%	481.35%
	284.62%	288.00%	293.65%	375.74%	437.21%	445.82%	419.90%
	810.00%	756.82%	743.43%	580.53%	428.76%	412.48%	#VALUE!
	960.61%	915.28%	858.93%	603.62%	436.71%	418.76%	#VALUE!
	1114.29%	1001.49%	901.52%	610.66%	441.35%	#VALUE!	#VALUE!
	1204.90%	1058.13%	910.97%	614.34%	445.76%	#VALUE!	#VALUE!
flash2 / eager							
	227.27%	241.18%	255.81%	361.17%	496.53%	506.76%	569.24%
	369.23%	368.00%	374.60%	479.29%	552.04%	521.97%	539.39%
	360.00%	370.03%	445.45%	512.87%	560.96%	523.97%	#VALUE!
	381.82%	415.28%	501.19%	523.76%	566.36%	527.59%	#VALUE!
	404.91%	428.46%	519.65%	526.84%	569.63%	#VALUE!	#VALUE!
	411.76%	440.24%	522.73%	525.75%	570.57%	#VALUE!	#VALUE!

第三版：a100

FusionTests-withA100

方法修正

- 在a100上遇到了可能是来自于torch的问题，compile的位置不恰当会导致部分函数报错，例如：
  - 第一段代码中，在func内部进行compile，会使benchmark中的函数被反复编译，延长测定时间，影响准确度

```python
x = torch.randn([dim0, dim1], device='cuda', dtype=torch.float32)
def func():
    fn = torch.nn.LayerNorm([dim1, ], device='cuda')
    fn = torch.compile(fn)
    res = fn(x)
    return res
result = triton.testing.do_bench(func, warmup=50, rep=1000,
return_mode="median")
```

- 第二段代码中，在func外层编译，但是由于torch的错误，可能引发dispatcher not implemented

```python
x = torch.randn([dim0, dim1], device='cuda', dtype=torch.float32)
def func():
    fn = torch.nn.LayerNorm([dim1, ], device='cuda')
    res = fn(x)
    return res
result = triton.testing.do_bench(torch.compile(func), warmup=50, rep=1000,
return_mode="median")
```

- 第三段代码是目前采用的形式，在测试结果和程序稳定性上都有保障

```python
x = torch.randn([dim0, dim1], device='cuda', dtype=torch.float32)
fn = torch.nn.LayerNorm([dim1, ], device='cuda')
fn = torch.compile(fn)
def func():
    res = fn(x)
    return res
result = triton.testing.do_bench(func, warmup=50, rep=1000,
return_mode="median")
```

- 代码形式已经在8个benchmark上更新，attention更新待push
- 该问题只在a100上出现，3090不受影响

结果汇总

1. Double Softmax

Relative Performance										
dim0 \ dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	136.59%	144.44%	157.97%	190.78%	159.24%	137.04%	129.83%	78.66%	108.65%	137.99%
256	128.00%	144.03%	157.54%	178.83%	140.65%	139.30%	133.79%	130.65%	145.83%	139.31%
512	119.44%	129.65%	141.55%	172.82%	139.66%	138.88%	128.15%	143.93%	141.20%	143.65%
1024	133.01%	140.38%	145.71%	151.36%	151.93%	153.27%	131.21%	123.29%	136.55%	135.96%
2048	129.74%	142.62%	135.71%	147.75%	170.14%	198.15%	164.35%	187.29%	136.53%	135.71%
4096	119.20%	136.67%	144.95%	147.77%	215.08%	201.46%	158.76%	192.60%	136.44%	135.82%
8192	112.85%	139.81%	143.97%	192.95%	100.32%	100.79%	100.51%	100.70%	100.00%	99.92%
16384	98.78%	100.40%	100.03%	99.95%	100.19%	100.22%	105.22%	100.78%	100.00%	100.04%
32768	100.46%	100.03%	99.92%	99.98%	100.25%	101.18%	106.21%	100.76%	100.02%	100.04%
65536	99.75%	99.88%	99.84%	100.02%	100.21%	101.35%	99.44%	100.71%	99.80%	#VALUE!

Relative A100-RTX3090										
dim0 \ dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	67.90%	59.42%	65.37%	68.69%	65.53%	71.43%	81.70%	132.83%	183.22%	131.39%
256	58.33%	60.62%	64.14%	71.36%	73.39%	95.41%	143.51%	242.66%	168.87%	161.57%
512	66.15%	62.29%	70.45%	71.22%	94.00%	129.45%	215.93%	178.18%	155.13%	154.34%
1024	62.29%	71.91%	74.15%	86.38%	130.83%	152.38%	217.48%	175.27%	159.38%	159.95%
2048	71.91%	74.33%	91.43%	132.93%	159.52%	145.35%	250.24%	169.21%	158.75%	160.40%
4096	77.90%	94.85%	153.21%	169.38%	154.02%	157.07%	272.14%	164.29%	158.25%	159.74%
8192	93.54%	160.36%	176.71%	145.83%	173.97%	165.98%	283.13%	162.51%	157.46%	158.66%
16384	157.85%	177.30%	146.03%	152.54%	191.64%	174.27%	275.80%	161.79%	157.19%	157.98%
32768	175.80%	146.53%	153.55%	156.81%	198.02%	174.71%	271.55%	161.51%	156.93%	#VALUE!
65536	146.90%	154.45%	158.24%	159.00%	201.91%	177.04%	284.38%	161.43%	#VALUE!	#VALUE!

128	57.97%	73.56%	81.16%	109.22%	130.43%	153.55%	203.31%	470.86%	707.36%	649.69%
256	96.00%	143.28%	190.88%	284.66%	571.18%	940.58%	1839.16%	2013.44%	1589.01%	1457.51%
512	88.89%	121.14%	165.04%	237.95%	492.31%	1006.74%	1152.67%	880.04%	761.97%	775.03%
1024	93.20%	121.14%	149.61%	193.00%	512.79%	703.09%	589.04%	401.91%	405.32%	404.07%
2048	111.95%	139.89%	156.73%	368.41%	461.07%	524.40%	490.66%	355.30%	242.36%	243.30%
4096	135.71%	191.04%	227.13%	333.71%	570.05%	586.07%	506.02%	358.90%	244.43%	245.06%
8192	172.22%	234.39%	256.90%	361.77%	273.40%	296.72%	333.12%	186.98%	179.99%	180.96%
16384	187.82%	169.44%	134.28%	177.48%	277.87%	296.94%	308.36%	170.12%	164.26%	165.35%
32768	206.04%	133.83%	131.19%	172.05%	279.94%	287.51%	292.79%	161.55%	156.09%	#VALUE!
65536	1067.27%	589.23%	319.55%	169.00%	89.66%	116.22%	199.43%	122.07%	#VALUE!	#VALUE!

2. Horizontal Reduction Pointwise

dim0 \ dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	155.33%	143.75%	151.57%	173.56%	185.56%	255.76%	121.05%	108.92%	123.39%	130.35%
256	135.71%	148.60%	152.86%	162.39%	163.78%	224.73%	117.04%	120.97%	133.50%	138.69%
512	142.32%	159.00%	142.46%	151.70%	154.47%	176.69%	133.42%	137.01%	141.79%	141.34%
1024	127.56%	141.25%	132.81%	147.81%	129.22%	151.79%	148.89%	142.49%	141.15%	140.77%
2048	134.27%	132.88%	142.02%	134.52%	131.73%	149.49%	146.66%	144.40%	142.55%	142.45%
4096	135.50%	134.44%	125.85%	136.78%	141.80%	149.65%	146.74%	144.61%	143.71%	143.20%
8192	119.30%	128.06%	138.37%	148.29%	99.96%	100.58%	99.99%	100.04%	100.07%	100.06%
16384	99.15%	99.86%	100.18%	99.96%	99.93%	99.92%	100.06%	100.07%	100.07%	100.02%
32768	100.14%	100.18%	100.04%	99.97%	100.02%	100.32%	100.06%	100.10%	99.99%	99.94%
65536	100.18%	99.98%	100.04%	99.99%	100.07%	100.16%	100.08%	99.99%	99.92%	#VALUE!

Relative A100-RTX3090

dim0 \ dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	59.10%	57.29%	59.54%	62.50%	59.94%	62.23%	97.91%	130.76%	139.13%	143.37%
256	58.95%	54.12%	63.44%	60.26%	62.31%	73.49%	133.65%	145.96%	149.59%	153.31%
512	55.16%	61.69%	62.20%	66.09%	78.25%	100.77%	142.76%	143.28%	147.28%	154.12%
1024	64.32%	67.29%	75.89%	90.00%	122.94%	129.03%	138.88%	148.79%	156.26%	159.83%
2048	66.82%	77.89%	95.21%	130.46%	140.95%	141.40%	148.81%	156.75%	160.44%	162.15%
4096	76.80%	100.50%	137.76%	150.57%	151.75%	152.38%	157.14%	160.74%	162.29%	163.21%
8192	103.76%	139.47%	145.66%	146.16%	149.04%	154.13%	161.30%	162.58%	163.30%	163.84%
16384	141.35%	145.86%	141.04%	150.69%	157.71%	160.72%	162.99%	163.52%	163.88%	164.05%
32768	144.65%	139.73%	151.17%	158.12%	161.29%	161.87%	163.92%	164.02%	164.07%	#VALUE!
65536	140.02%	150.90%	157.93%	161.50%	163.17%	163.02%	164.53%	164.51%	#VALUE!	#VALUE!

128	52.46%	47.06%	55.75%	65.08%	77.01%	108.90%	137.23%	167.13%	205.87%	226.71%
256	57.14%	67.13%	86.20%	117.43%	145.79%	254.62%	299.44%	350.34%	401.00%	435.77%
512	54.61%	85.82%	88.62%	108.62%	133.60%	182.86%	205.13%	217.06%	236.03%	249.87%
1024	61.54%	73.93%	83.99%	110.85%	118.01%	144.63%	154.24%	162.82%	169.37%	172.69%
2048	69.78%	77.63%	112.68%	129.03%	130.55%	154.20%	159.86%	171.34%	171.09%	175.57%
4096	78.05%	106.67%	124.92%	146.73%	156.64%	175.48%	187.45%	192.27%	193.29%	195.37%
8192	87.33%	119.07%	139.53%	151.54%	107.54%	116.10%	127.53%	130.45%	131.70%	133.23%
16384	93.43%	98.60%	97.29%	105.50%	113.26%	119.40%	127.70%	130.35%	130.57%	130.92%
32768	98.05%	94.08%	103.37%	110.24%	115.31%	121.13%	127.38%	129.95%	129.89%	#VALUE!
65536	565.65%	322.74%	180.72%	109.34%	110.91%	110.71%	112.13%	113.70%	#VALUE!	#VALUE!

3. Horizontal Reduction Reduction

Relative Performance										
dim0 \ dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	176.40%	163.87%	205.79%	204.11%	214.86%	352.68%	151.75%	113.93%	125.39%	169.64%
256	156.94%	188.02%	173.29%	190.09%	192.74%	291.83%	143.78%	140.72%	171.96%	173.71%
512	151.54%	177.52%	164.87%	189.58%	169.86%	235.74%	147.32%	184.85%	182.54%	193.01%
1024	146.58%	144.89%	149.45%	156.31%	133.53%	178.04%	179.32%	175.81%	184.92%	191.72%
2048	164.98%	148.90%	158.84%	155.19%	128.19%	201.84%	177.91%	187.10%	193.60%	200.65%
4096	156.82%	163.08%	146.59%	144.22%	154.20%	180.84%	187.72%	194.59%	199.88%	202.68%
8192	139.00%	162.72%	145.59%	178.32%	100.00%	99.99%	100.02%	99.98%	100.00%	100.01%
16384	100.00%	100.00%	100.00%	100.00%	100.01%	100.02%	100.01%	100.00%	99.99%	100.01%
32768	100.00%	99.96%	100.03%	99.99%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
65536	99.96%	100.00%	100.03%	99.99%	100.00%	100.00%	100.00%	100.00%	100.01%	100.00%

Relative A100-RTX3090										
dim0 \ dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	65.31%	57.02%	58.43%	63.76%	64.40%	62.87%	83.56%	120.10%	125.10%	104.47%
256	58.05%	56.70%	56.92%	62.54%	62.21%	68.53%	124.73%	144.34%	121.02%	137.69%
512	57.66%	62.88%	61.42%	65.31%	71.45%	93.01%	129.38%	109.91%	123.98%	137.80%
1024	64.00%	68.38%	70.20%	81.32%	112.28%	113.67%	109.95%	125.37%	141.04%	150.19%
2048	65.31%	71.24%	86.49%	122.92%	128.25%	103.90%	125.89%	141.94%	151.18%	156.73%
4096	73.89%	85.53%	128.64%	147.74%	124.83%	132.96%	142.13%	151.79%	157.13%	160.20%
8192	85.97%	114.56%	134.68%	114.52%	127.13%	140.41%	152.80%	157.79%	160.39%	162.18%
16384	112.47%	129.25%	114.52%	129.48%	144.10%	151.57%	158.90%	160.91%	162.00%	162.65%
32768	117.04%	114.76%	129.81%	144.66%	153.77%	156.33%	162.14%	162.52%	162.51%	162.92%
65536	114.47%	129.51%	144.20%	153.81%	158.92%	159.76%	163.82%	163.02%	162.74%	163.05%

128	51.20%	46.72%	66.12%	65.75%	60.54%	95.24%	119.35%	128.00%	160.00%	191.47%
256	56.94%	79.34%	87.67%	108.98%	119.91%	215.38%	285.81%	360.24%	400.56%	486.60%
512	65.53%	74.42%	81.01%	104.76%	108.58%	177.78%	197.86%	223.95%	255.28%	308.18%
1024	62.54%	69.35%	69.95%	89.72%	93.70%	129.83%	131.44%	147.40%	172.74%	190.85%
2048	86.20%	79.56%	96.97%	104.62%	95.47%	115.46%	120.38%	142.74%	154.15%	162.26%
4096	98.05%	107.18%	111.97%	120.60%	101.42%	120.16%	138.86%	151.96%	159.29%	163.87%
8192	99.59%	130.49%	121.57%	116.03%	67.44%	72.18%	79.34%	81.80%	83.11%	84.40%
16384	83.45%	87.26%	68.45%	71.58%	75.09%	77.28%	80.66%	81.10%	81.37%	81.49%
32768	91.29%	73.84%	73.39%	77.59%	79.44%	80.12%	82.23%	81.64%	81.47%	81.40%
65536	472.00%	296.83%	175.52%	102.79%	80.00%	79.89%	82.08%	81.56%	81.34%	81.43%

4. LayerNorm

Relative Performance										
dim0 \ dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	134.38%	104.96%	117.13%	143.98%	108.00%	136.12%	173.70%	131.80%	173.80%	173.93%
256	100.68%	119.22%	122.05%	125.39%	96.05%	136.40%	144.82%	132.55%	108.02%	105.41%
512	96.43%	118.25%	121.27%	119.21%	84.46%	119.34%	121.00%	96.40%	84.99%	83.70%
1024	100.62%	121.72%	123.34%	117.44%	76.15%	115.69%	133.32%	86.48%	75.72%	74.08%
2048	127.08%	137.75%	117.99%	120.94%	83.90%	124.42%	120.98%	91.38%	78.51%	80.34%
4096	149.75%	149.09%	114.98%	115.64%	81.70%	128.71%	119.70%	88.30%	78.16%	77.72%
8192	163.58%	150.00%	116.25%	110.76%	99.95%	99.64%	99.88%	100.00%	99.97%	100.16%
16384	97.91%	100.00%	100.00%	99.28%	100.00%	100.00%	99.95%	100.03%	99.99%	100.01%
32768	98.84%	99.93%	100.23%	99.85%	100.25%	100.06%	99.96%	100.03%	99.99%	99.98%
65536	100.02%	100.46%	100.38%	100.19%	100.00%	100.00%	100.00%	100.00%	100.01%	#VALUE!

Relative A100-RTX3090										
dim0 \ dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	83.72%	97.30%	85.97%	83.55%	81.48%	77.94%	78.46%	85.79%	83.40%	78.96%
256	97.30%	85.97%	91.90%	80.99%	90.04%	91.15%	115.96%	118.32%	105.56%	110.13%
512	94.95%	92.60%	85.86%	91.17%	107.06%	144.47%	153.38%	133.56%	139.31%	145.34%
1024	98.46%	99.72%	97.20%	108.27%	140.66%	162.12%	146.26%	151.48%	155.16%	158.20%
2048	100.73%	93.72%	96.45%	120.45%	156.04%	162.64%	151.15%	153.84%	156.16%	153.88%
4096	94.74%	98.01%	115.43%	146.89%	178.81%	164.97%	153.96%	156.11%	157.84%	160.13%
8192	103.34%	103.75%	130.45%	161.73%	203.40%	168.86%	159.30%	159.90%	160.55%	162.69%
16384	107.99%	109.60%	143.99%	192.66%	217.40%	169.68%	161.16%	160.89%	161.24%	163.39%
32768	110.43%	112.58%	153.45%	206.24%	224.13%	170.26%	161.98%	161.57%	161.73%	#VALUE!
65536	112.72%	116.45%	159.87%	209.66%	228.04%	171.24%	162.56%	162.06%	#VALUE!	#VALUE!

128	100.00%	90.78%	89.51%	107.89%	96.00%	86.68%	106.67%	134.10%	170.64%	156.57%
256	87.07%	79.72%	109.51%	99.07%	106.86%	118.81%	158.87%	154.50%	115.89%	116.60%
512	93.18%	106.08%	91.43%	100.53%	106.59%	144.23%	177.15%	127.71%	117.90%	121.14%
1024	89.16%	99.31%	108.65%	113.02%	117.15%	161.59%	189.88%	129.91%	117.49%	117.81%
2048	118.15%	110.66%	112.55%	126.40%	130.92%	171.11%	179.81%	139.69%	121.95%	123.55%
4096	126.11%	124.20%	127.22%	139.53%	141.05%	179.46%	182.71%	137.40%	122.51%	124.02%
8192	144.91%	131.31%	142.36%	147.83%	187.51%	143.18%	157.76%	159.21%	159.69%	162.05%
16384	90.63%	91.93%	130.21%	145.01%	200.38%	142.15%	159.76%	160.34%	160.14%	162.56%
32768	92.35%	91.28%	135.52%	152.04%	206.04%	142.35%	160.75%	160.92%	160.57%	#VALUE!
65536	93.63%	92.23%	138.36%	155.81%	209.45%	145.64%	161.21%	161.33%	#VALUE!	#VALUE!

5. Norm Chain

Relative Performance										
dim0 \ dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	388.52%	367.30%	425.40%	455.76%	507.14%	540.69%	296.27%	146.81%	194.21%	193.15%
256	394.67%	344.91%	363.88%	436.05%	423.53%	518.57%	302.41%	277.37%	215.69%	195.39%
512	342.91%	340.60%	351.70%	359.95%	410.42%	441.28%	312.88%	285.47%	204.31%	195.23%
1024	325.41%	364.29%	351.14%	363.08%	371.09%	428.34%	395.18%	294.13%	194.43%	184.84%
2048	353.87%	325.13%	345.22%	314.46%	394.95%	431.36%	295.88%	258.33%	188.81%	184.51%
4096	345.43%	352.86%	319.26%	355.77%	437.29%	434.09%	276.83%	240.47%	185.19%	182.62%
8192	362.14%	313.28%	343.74%	445.09%	100.03%	100.25%	100.04%	100.07%	100.01%	100.05%
16384	100.00%	100.17%	99.71%	100.04%	100.06%	100.05%	99.99%	100.06%	100.06%	100.04%
32768	100.08%	99.65%	100.01%	100.07%	100.05%	99.94%	99.97%	100.10%	100.06%	100.00%
65536	99.96%	100.11%	100.09%	100.05%	100.07%	100.03%	100.04%	100.13%	99.97%	#VALUE!

Relative A100-RTX3090										
dim0 \ dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	60.76%	56.31%	56.72%	60.03%	59.60%	62.38%	99.54%	151.56%	155.15%	143.69%
256	56.49%	58.60%	58.82%	61.93%	63.77%	89.05%	159.60%	173.72%	152.59%	152.25%
512	59.57%	59.90%	61.97%	67.27%	93.35%	124.15%	174.44%	145.13%	144.82%	151.27%
1024	64.06%	65.73%	69.90%	100.90%	142.03%	136.16%	140.58%	145.68%	152.68%	156.55%
2048	66.45%	71.05%	101.55%	151.93%	155.36%	141.99%	145.38%	153.07%	156.79%	158.76%
4096	74.42%	103.64%	158.95%	173.22%	152.15%	149.90%	152.89%	156.90%	158.82%	159.76%
8192	105.39%	164.39%	176.72%	145.45%	145.86%	150.44%	156.92%	158.92%	159.82%	160.17%
16384	157.50%	171.02%	141.73%	146.70%	152.46%	156.31%	158.96%	159.92%	160.26%	160.52%
32768	160.41%	142.44%	146.83%	153.45%	156.24%	157.86%	160.05%	160.43%	160.49%	#VALUE!
65536	139.61%	147.39%	153.24%	157.06%	158.16%	158.85%	160.49%	160.54%	#VALUE!	#VALUE!

128	52.46%	60.84%	63.49%	83.27%	93.51%	109.80%	156.86%	291.33%	409.26%	389.46%
256	78.69%	78.60%	99.00%	152.38%	247.57%	450.32%	796.38%	803.09%	567.86%	527.95%
512	68.09%	76.85%	99.07%	117.28%	255.43%	471.04%	499.22%	398.60%	293.82%	299.33%
1024	72.96%	87.07%	100.00%	127.10%	301.18%	351.15%	342.10%	273.11%	191.51%	188.53%
2048	92.90%	94.12%	112.59%	194.09%	310.10%	335.61%	236.95%	223.43%	169.47%	168.73%
4096	124.10%	137.14%	165.70%	304.91%	342.47%	348.52%	242.20%	219.27%	172.96%	172.15%
8192	150.95%	150.00%	228.57%	319.29%	77.39%	83.28%	89.51%	91.91%	93.21%	94.04%
16384	47.33%	46.80%	56.14%	78.67%	85.38%	89.05%	93.25%	94.50%	95.26%	95.54%
32768	45.28%	38.29%	59.49%	82.64%	86.92%	89.37%	92.31%	93.64%	94.20%	#VALUE!
65536	221.65%	126.15%	70.63%	40.35%	41.40%	55.55%	85.17%	88.61%	#VALUE!	#VALUE!

6. Pointwise Chain

Relative Performance										
dim0 \ dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	190.53%	217.09%	224.79%	196.28%	198.44%	196.56%	202.38%	229.92%	423.82%	495.80%
256	193.16%	219.42%	194.92%	223.71%	196.31%	202.38%	230.29%	433.43%	495.85%	516.80%
512	195.22%	199.31%	202.18%	196.31%	202.16%	239.80%	432.66%	490.93%	520.25%	538.21%
1024	197.94%	202.18%	196.83%	201.72%	239.80%	432.56%	495.90%	520.28%	538.14%	548.17%
2048	223.02%	196.83%	202.16%	239.80%	423.28%	490.72%	520.25%	540.03%	548.15%	553.52%
4096	196.83%	213.93%	231.36%	432.56%	490.93%	517.03%	538.23%	548.15%	553.49%	556.49%
8192	213.70%	231.36%	432.56%	495.96%	99.98%	100.02%	100.00%	100.01%	100.00%	99.92%
16384	100.35%	99.84%	100.02%	99.98%	100.01%	99.99%	100.01%	100.00%	99.93%	99.97%
32768	99.84%	100.00%	99.97%	100.03%	100.01%	100.02%	99.98%	99.94%	99.97%	#VALUE!
65536	99.96%	99.93%	99.93%	99.93%	99.93%	99.94%	99.95%	99.98%	#VALUE!	#VALUE!

Relative A100-RTX3090										
dim0 \ dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	63.62%	56.69%	58.82%	66.09%	70.33%	86.14%	184.42%	258.32%	157.83%	151.27%
256	56.69%	60.26%	66.78%	68.82%	84.95%	184.42%	258.32%	157.90%	151.26%	156.48%
512	60.26%	66.67%	69.03%	86.02%	184.62%	256.71%	158.29%	151.30%	156.41%	158.28%
1024	66.67%	69.03%	86.02%	184.62%	256.71%	158.84%	151.32%	156.40%	158.19%	159.12%
2048	69.03%	86.02%	184.62%	256.71%	158.90%	151.29%	156.41%	158.18%	159.12%	159.72%
4096	86.02%	184.42%	256.71%	158.93%	151.30%	156.41%	158.19%	159.14%	159.74%	160.02%
8192	184.62%	256.71%	158.93%	151.30%	156.40%	158.19%	159.17%	159.74%	159.96%	160.19%
16384	256.71%	158.97%	151.29%	156.41%	158.18%	159.17%	159.74%	159.96%	160.12%	160.42%
32768	158.25%	151.30%	156.42%	158.07%	159.21%	159.79%	160.14%	160.31%	160.51%	#VALUE!
65536	151.39%	156.48%	158.20%	159.33%	159.90%	160.10%	160.35%	160.26%	#VALUE!	#VALUE!

128	48.48%	54.70%	52.89%	54.05%	59.81%	76.19%	96.76%	117.92%	129.57%	141.38%
256	48.67%	52.89%	54.24%	65.98%	75.99%	96.76%	118.11%	132.56%	141.38%	150.63%
512	47.06%	55.36%	59.81%	75.73%	96.76%	122.55%	132.56%	140.02%	151.57%	157.21%
1024	54.98%	59.81%	76.19%	96.55%	127.36%	132.56%	141.46%	151.57%	157.18%	160.53%
2048	65.98%	67.72%	96.76%	122.55%	129.69%	139.95%	151.57%	157.72%	160.53%	162.50%
4096	67.72%	102.28%	122.88%	132.56%	140.02%	150.63%	157.21%	160.54%	162.35%	163.44%
8192	102.28%	118.08%	132.56%	141.46%	29.13%	29.21%	29.29%	29.36%	29.37%	29.37%
16384	51.28%	30.60%	28.52%	29.13%	29.21%	29.28%	29.36%	29.37%	29.37%	29.38%
32768	30.60%	28.52%	29.13%	29.21%	29.29%	29.36%	29.36%	29.37%	29.37%	#VALUE!
65536	28.52%	29.13%	29.21%	29.28%	29.34%	29.36%	29.36%	29.37%	#VALUE!	#VALUE!

7. Softmax Backward

Relative Performance										
dim0 \ dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	124.43%	139.74%	146.80%	142.03%	129.64%	131.84%	130.72%	112.38%	120.28%	136.61%
256	139.74%	136.80%	144.93%	135.80%	144.01%	139.89%	133.71%	122.90%	140.56%	141.34%
512	140.00%	139.01%	123.80%	135.71%	140.66%	142.10%	127.69%	141.75%	145.08%	145.62%
1024	128.95%	136.50%	140.00%	137.87%	149.88%	159.20%	149.56%	142.24%	142.91%	141.93%
2048	139.01%	137.41%	136.79%	146.75%	175.92%	169.96%	159.58%	148.72%	144.93%	143.51%
4096	140.10%	129.54%	142.00%	179.60%	187.71%	171.33%	161.72%	148.69%	145.07%	143.47%
8192	131.83%	140.94%	180.86%	191.30%	100.02%	99.97%	101.69%	100.06%	100.09%	100.00%
16384	99.51%	100.11%	100.16%	100.03%	100.08%	100.00%	101.67%	100.05%	99.97%	99.97%
32768	100.18%	100.14%	100.04%	99.99%	100.10%	99.97%	101.76%	99.99%	100.00%	#VALUE!
65536	100.00%	100.00%	99.99%	100.01%	99.97%	99.99%	100.02%	100.00%	#VALUE!	#VALUE!

Relative A100-RTX3090										
dim0 \ dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	58.90%	58.72%	61.04%	65.31%	81.41%	92.52%	132.00%	156.38%	155.44%	150.00%
256	58.72%	65.50%	64.00%	80.00%	92.36%	130.95%	166.33%	165.78%	153.44%	155.74%
512	64.00%	65.31%	73.23%	89.82%	133.33%	160.85%	160.06%	150.94%	153.65%	155.55%
1024	65.31%	78.43%	92.59%	134.24%	164.03%	150.97%	156.20%	154.70%	156.65%	158.56%
2048	78.40%	92.92%	136.83%	164.44%	152.38%	153.44%	162.70%	156.18%	158.30%	159.17%
4096	92.75%	136.26%	165.96%	143.25%	156.22%	162.63%	164.55%	157.58%	158.68%	159.16%
8192	136.08%	166.64%	142.43%	148.10%	166.28%	165.09%	164.30%	158.34%	158.95%	159.26%
16384	166.91%	143.27%	148.01%	154.95%	168.01%	165.23%	164.27%	158.71%	159.33%	159.53%
32768	143.27%	147.95%	154.90%	157.39%	169.68%	164.88%	165.04%	160.12%	159.56%	#VALUE!
65536	148.39%	154.90%	157.19%	158.38%	170.29%	165.59%	164.29%	159.17%	#VALUE!	#VALUE!

128	48.85%	56.41%	64.00%	69.57%	79.78%	100.45%	162.09%	162.62%	165.28%	178.19%
256	68.38%	76.80%	104.35%	138.27%	211.25%	320.57%	348.55%	302.02%	319.61%	329.48%
512	76.80%	79.43%	81.59%	129.52%	193.41%	228.57%	192.18%	194.44%	200.87%	204.42%
1024	63.16%	78.53%	89.11%	143.28%	180.65%	191.13%	174.38%	161.27%	163.60%	163.68%
2048	79.26%	87.78%	132.83%	173.79%	201.05%	205.06%	193.02%	170.72%	169.01%	168.61%
4096	97.46%	119.57%	173.07%	204.12%	226.91%	227.41%	205.46%	181.21%	178.61%	177.53%
8192	109.95%	146.79%	191.43%	224.73%	126.87%	133.04%	127.14%	120.64%	121.67%	122.47%
16384	93.84%	96.40%	114.46%	126.19%	130.37%	134.64%	128.27%	121.55%	122.84%	124.09%
32768	78.82%	98.14%	118.25%	127.86%	129.40%	133.10%	125.47%	119.42%	120.09%	#VALUE!
65536	235.41%	129.42%	78.31%	88.80%	94.27%	117.69%	116.81%	113.37%	#VALUE!	#VALUE!

8. Softmax

Relative Performance										
dim0 \ dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	93.43%	99.28%	130.12%	116.07%	130.07%	116.91%	123.40%	76.33%	99.02%	107.11%
256	102.81%	118.88%	111.04%	138.23%	112.59%	108.93%	111.16%	112.94%	108.47%	104.06%
512	91.00%	102.59%	111.76%	131.62%	116.84%	124.80%	106.22%	100.61%	105.48%	106.30%
1024	102.30%	103.09%	108.20%	118.22%	132.40%	127.03%	84.19%	93.50%	101.90%	101.67%
2048	112.75%	113.95%	107.40%	111.84%	130.84%	113.08%	98.22%	120.44%	101.80%	101.15%
4096	94.65%	103.12%	98.89%	102.07%	120.23%	109.28%	93.79%	122.71%	101.81%	101.58%
8192	102.57%	101.32%	102.37%	102.55%	99.35%	99.25%	100.11%	100.19%	99.90%	99.96%
16384	103.49%	99.80%	100.38%	100.92%	99.44%	101.22%	104.44%	100.06%	99.95%	100.09%
32768	102.59%	99.13%	100.27%	99.80%	100.85%	101.59%	107.90%	100.02%	99.96%	100.08%
65536	100.27%	100.18%	100.22%	99.96%	100.38%	100.96%	104.34%	99.96%	100.07%	#VALUE!

Relative A100-RTX3090										
dim0 \ dim1	128	256	512	1024	2048	4096	8192	16384	32768	65536
128	62.50%	48.18%	49.38%	63.28%	62.31%	66.12%	72.61%	120.49%	150.38%	125.65%
256	51.17%	54.05%	57.83%	63.21%	63.16%	82.15%	120.45%	194.63%	157.15%	162.10%
512	60.84%	57.76%	62.05%	69.26%	74.95%	104.00%	172.07%	165.04%	153.11%	153.61%
1024	51.45%	67.07%	70.42%	79.14%	103.35%	120.89%	207.16%	171.32%	158.38%	159.40%
2048	66.67%	75.00%	86.67%	108.24%	125.29%	140.84%	246.88%	167.03%	158.12%	160.13%
4096	74.04%	83.12%	123.08%	132.85%	149.84%	152.35%	268.78%	163.24%	157.82%	159.43%
8192	87.47%	119.51%	129.86%	140.18%	175.02%	164.59%	282.42%	161.82%	157.42%	158.53%
16384	117.66%	132.02%	141.79%	149.30%	189.46%	171.36%	276.60%	161.52%	157.22%	157.93%
32768	130.61%	142.89%	150.48%	156.07%	197.15%	174.50%	270.55%	161.39%	156.90%	#VALUE!
65536	141.95%	150.84%	156.43%	158.71%	201.81%	175.72%	265.57%	161.33%	#VALUE!	#VALUE!
128	46.72%	46.38%	64.26%	73.44%	104.58%	115.94%	172.80%	305.66%	418.92%	386.69%
256	77.11%	102.81%	117.73%	185.67%	284.44%	514.29%	865.83%	984.03%	777.17%	758.08%
512	66.44%	94.81%	108.98%	154.99%	282.95%	514.20%	551.07%	412.04%	407.30%	414.40%
1024	73.68%	79.01%	101.59%	135.11%	305.26%	366.70%	288.21%	237.44%	239.49%	240.36%
2048	85.91%	94.96%	114.56%	210.53%	272.13%	282.30%	286.10%	224.46%	178.30%	178.96%
4096	93.43%	114.29%	167.35%	243.00%	276.48%	289.04%	278.00%	214.52%	172.20%	173.17%
8192	119.63%	147.61%	205.74%	264.45%	257.47%	282.90%	309.72%	174.91%	169.87%	170.69%
16384	138.21%	154.31%	221.42%	292.00%	286.65%	306.83%	322.86%	179.16%	175.65%	175.43%
32768	143.57%	164.37%	232.05%	296.64%	293.50%	307.39%	321.92%	176.48%	172.31%	#VALUE!
65536	984.07%	550.87%	305.02%	169.35%	171.39%	211.13%	268.27%	161.84%	#VALUE!	#VALUE!

9. Attention

Relative Performance (headdim=64, nheads=32)							
triton / eager	128	256	512	1024	2048	4096	8192
1	173.92%	167.27%	187.18%	174.08%	192.01%	205.42%	190.86%
2	174.60%	176.97%	179.33%	177.13%	194.43%	205.17%	188.80%
4	172.00%	185.68%	173.94%	176.44%	192.29%	201.77%	#VALUE!
8	188.80%	180.69%	173.71%	175.47%	191.87%	202.01%	#VALUE!
16	182.37%	175.90%	175.52%	175.56%	191.42%	#VALUE!	#VALUE!
32	171.28%	176.27%	176.04%	175.90%	192.82%	#VALUE!	#VALUE!
flash2 / triton							
1	163.00%	202.97%	201.00%	437.54%	596.60%	754.11%	910.14%
2	247.24%	258.02%	392.60%	576.74%	731.76%	820.35%	942.15%
4	287.46%	305.33%	471.47%	692.11%	815.23%	854.90%	#VALUE!
8	251.37%	378.81%	576.67%	770.71%	833.04%	859.89%	#VALUE!
16	295.84%	449.86%	641.97%	799.54%	838.15%	#VALUE!	#VALUE!
32	357.47%	502.28%	686.32%	796.55%	834.73%	#VALUE!	#VALUE!
flash2 / eager							
1	283.48%	339.51%	376.23%	761.68%	1145.53%	1549.08%	1737.10%
2	431.69%	456.61%	704.04%	1021.59%	1422.75%	1683.14%	1778.73%
4	494.44%	566.95%	820.10%	1221.16%	1567.59%	1724.93%	#VALUE!
8	474.58%	684.50%	1001.76%	1352.36%	1598.34%	1737.08%	#VALUE!
16	539.54%	791.32%	1126.79%	1403.69%	1604.38%	#VALUE!	#VALUE!
32	612.27%	885.38%	1208.22%	1401.15%	1609.48%	#VALUE!	#VALUE!

Relative Performance (headdim=128, nheads=16)							
triton / eager	128	256	512	1024	2048	4096	8192
1	170.79%	158.39%	175.10%	169.66%	185.10%	199.71%	197.47%
2	173.62%	178.69%	184.23%	173.79%	186.96%	196.95%	183.56%
4	180.16%	180.46%	177.78%	172.82%	185.74%	194.84%	181.39%
8	181.84%	181.28%	171.93%	175.06%	190.04%	197.93%	#VALUE!
16	201.86%	173.85%	172.45%	174.70%	187.33%	197.24%	#VALUE!
32	178.16%	172.65%	173.87%	174.17%	188.49%	#VALUE!	#VALUE!
flash2 / triton							
1	138.36%	150.07%	169.33%	249.07%	336.51%	438.76%	516.23%
2	217.23%	230.67%	215.39%	354.70%	442.58%	518.94%	582.53%
4	232.54%	222.74%	307.41%	427.25%	523.11%	546.90%	601.03%
8	233.94%	265.75%	365.54%	489.80%	520.47%	530.98%	#VALUE!
16	215.11%	330.10%	437.37%	528.06%	535.98%	539.30%	#VALUE!
32	280.52%	361.30%	483.28%	540.43%	539.07%	#VALUE!	#VALUE!
flash2 / eager							
1	236.31%	237.70%	296.50%	422.58%	622.88%	876.25%	1019.40%
2	377.15%	412.19%	396.82%	616.44%	827.44%	1022.08%	1069.28%
4	418.96%	401.95%	546.49%	738.40%	971.61%	1065.57%	1090.17%
8	425.40%	481.74%	628.49%	857.44%	989.11%	1050.98%	#VALUE!
16	434.21%	573.88%	754.25%	922.50%	1004.06%	1063.68%	#VALUE!
32	499.78%	623.78%	840.26%	941.26%	1016.08%	#VALUE!	#VALUE!

第三版：Matmul & Conv

1. matmul+add+relu
2. matmul+add
3. matmul+relu
4. matmul+matmul
5. conv+add
6. conv+bn
7. conv+relu

规律总结

1. 数据统计
   a. 在pytest-benchmark的第一版数据中，观察到算子耗时的平均值和中位数表现基本一致
2. 异常处理
   a. 65536*65536的大形状在多数算子上会出现oom
   b. 较复杂的算子在32768*65536或65536*32768的形状上也可能oom
   c. 有的compile可以使原本会oom的形状顺利执行（attention上最明显）
   d. 得益于A100更大的内存，oom的阈值也提高了，异常数量减少
3. 绝对性能
   a. 执行时间的整体趋势存在显著的阶梯形增长

4. 形状大小
    a. 对3090上的多数算子而言，torch.compile在维度尺寸相近的张量上表现更好（如2048*2048），维度相差较大的张量上表现较差（如128*32768）
    b. 但是compile的效果渐进与数据量并不是严格的相关
    c. Pointwize chain是例外，小形状上compile效果稍弱，大形状上效果较好，但是都优于eager
    d. 而attention控制了四维输入中的batch size与seq len变化，不便与其他算子比较
5. 算子类型
    a. Softmax和DoubleSoftmax算子的compile效果比较有限，在测试范围内的最佳性能只能略微优于eager，多数形状上只能勉强追平或更弱，推测是由于torch aten对关键算子做了精细的手动优化
    b. 相反，在非典型的算子融合场景中，aten没有专门手动处理，compile则体现出了优势，基本上都可以超过eager的性能
6. 实验平台
    a. 相比3090，a100在小规模张量上的表现更弱，大规模上更优
    b. 在A100上呈现了特殊的相对性能趋势，对于参与测试的8类融合算子均一致（attention的输入张量维度与其他8个算子不同，不归纳到此现象）
    c. compile在较小规模的张量上相比eager实现了明显的优化
    d. 但是在形状超过8192*2048后，相对性能鲜少超过100%，分界线十分清晰，在各个算子上位置相同
    e. 在softmax的测例中，该现象不如其他算子明显，但是8192*2048以上的形状相对性能仍然更差一些
    f. 形状kernel比较(matmul+add)
        i. 8192*1024-eager

Time (%)	Total Time (ns)	Instances	Avg (ns)	Med (ns)	Min (ns)	Max (ns)	StdDev (ns)	Name
51.2	788801463	4834	163177.8	163056.0	162048	167040	580.0	void at::native::vectorized_elementwise_kernel<(int)4, at::native::FillFunctor<int>, at::detail::Ar…
25.6	394626324	5081	77667.1	75265.0	74560	97569	6819.7	ampere_fp16_s16816gemm_fp16_128x128_ldg8_f2f_stages_32x5_nn
23.2	357431513	5081	70346.7	69728.0	69152	82176	1809.7	void at::native::unrolled_elementwise_kernel<at::native::CUDAFunctor_add<float>, at::detail::Array<…
0.0	64191	2	32095.5	32095.5	14048	50143	25523.0	void at::native::<unnamed>::distribution_elementwise_grid_stride_kernel<float, (int)4, void at::nat…
0.0	48992	1	48992.0	48992.0	48992	48992	0.0	void at::native::<unnamed>::distribution_elementwise_grid_stride_kernel<float, (int)4, void at::nat…

        ii. 8192*1024-compiled

Time (%)	Total Time (ns)	Instances	Avg (ns)	Med (ns)	Min (ns)	Max (ns)	StdDev (ns)	Name
43.8	464404227	2922	158933.7	152161.0	151008	194305	15303.0	void cutlass::Kernel<cutlass_80_tensorop_s1688gemm_128x128_16x5_nn_align4>(T1::Params)
42.9	454657091	2778	163663.5	163456.0	162496	172064	739.4	void at::native::vectorized_elementwise_kernel<(int)4, at::native::FillFunctor<int>, at::detail::Ar…
13.3	140858593	5844	24103.1	21376.0	6496	44512	17180.1	triton__0d1d2d
0.0	64864	2	32432.0	32432.0	14240	50624	25727.4	void at::native::<unnamed>::distribution_elementwise_grid_stride_kernel<float, (int)4, void at::nat…
0.0	49760	1	49760.0	49760.0	49760	49760	0.0	void at::native::<unnamed>::distribution_elementwise_grid_stride_kernel<float, (int)4, void at::nat…

        iii. 8192*2048-eager

Time (%)	Total Time (ns)	Instances	Avg (ns)	Med (ns)	Min (ns)	Max (ns)	StdDev (ns)	Name
34.8	448078486	2744	163293.9	163040.0	162048	170433	779.5	void at::native::vectorized_elementwise_kernel<(int)4, at::native::FillFunctor<int>, at::detail::Ar…
34.6	445865892	2887	154439.2	146688.0	145824	188832	15682.0	void cutlass::Kernel<cutlass_80_tensorop_f16_s16816gemm_relu_f16_128x128_32x4_nn_align8>(T1::Params)
30.6	394015791	2887	136479.3	134657.0	134080	151968	3724.5	void at::native::unrolled_elementwise_kernel<at::native::CUDAFunctor_add<float>, at::detail::Array<…
0.0	88736	1	88736.0	88736.0	88736	88736	0.0	void at::native::<unnamed>::distribution_elementwise_grid_stride_kernel<float, (int)4, void at::nat…
0.0	68768	2	34384.0	34384.0	18720	50048	22152.2	void at::native::<unnamed>::distribution_elementwise_grid_stride_kernel<float, (int)4, void at::nat…

        iv. 8192*2048-compiled

g. 这个位置的特殊性有可能不在size，而是在于8192*2048是测试程序中第65次调用torch.compile。由于存在环境变量DYNAMO_CACHE_SIZE_LIMIT（或者torch._dynamo.config.cache_size_limit）默认值为64，使得前64次编译被缓存到cache中，而之后不再重新编译和缓存。该现象只在a100上出现，3090具有相等的默认值，但是不会受到编译次数的影响。

重要问题

1. 在同设备上，compiled kernel的gpu执行时间优于eager kernel，但由于外层包装复杂，直接测得的端到端性能反而较差
2. 由于上文指出的a100分界线的存在，compile模式下的a100没有全面提升大形状张量的性能，反而有些劣于3090的compile

测试数据：eager & inductor全局访存比较

测试工具：nsight

1. 优点：cuda内置工具，能够最准确地获取gpu运行的各项信息，内容十分全面
2. 缺点：一次只能执行一个形状测例，不能批量分析，需要自行数据处理

指标：ncu metric: l1tex__t_sectors_pipe_lsu_mem_global_op_ld

1. 替代了旧版本的metric gld_transactions_per_request，记录kernel请求的global加载次数
2. 没有发现与性能具有明显相关性的特征，compile前后的global load数量或不变化、或上涨到3倍，此外没有其他现象
3. 但是另一个指标l1tex__t_sectors_pipe_lsu_mem_global_op_ld_lookup_hit记录了global load的命中数量，可能更影响性能
4. nsys测试准确度值得怀疑：相同的kernel在直接执行和送入do_bench执行，在nsys输出的时间不同，增加gpu时钟频率锁定后仍然如此

待测指标

1. l1tex__t_sectors_pipe_lsu_mem_global_op_st相关：global store次数及命中率
2. l1tex__t_sectors_pipe_lsu_mem_global_op_red相关：global reduction次数及命中率

Softmax分析

Naive_softmax

1. 通过cuda kernel的计时可以发现，naive的softmax几乎没有融合，共调用了2个elementwise和2个reduce

2. kernel名称显示不全，可以确认其中一个reduce为max，其他kernel未知
3. distribution_elementwise_grid_stride_kernel在各种softmax中均出现，推测是用于分布式并行的数据计算

compiled_naive_softmax

1. 编译形成了triton kernel，并调用了distributed kernel

torch_nn_softmax

1. 主要kernel为softmax_warp_forward，在aten/src/ATen/native/cuda/PersistentSoftmax.cuh中定义

compiler_torch_nn_softmax

1. 编译形成了triton kernel，并调用了distributed kernel
2. Cuda gpu执行时间与compiled_naive相差无几

torch_softmax

1. 是torch.nn.functional.softmax的别名，调用的kernel与torch.nn.Softmax相同
2. torch.nn.Softmax需要为backward做额外的准备，耗时更长一点
3. 只探讨forward的话，直接用torch.softmax更简洁高效

Common

1. distribution_elementwise_grid_stride_kernel在aten/src/ATen/native/cuda/DistributionTemplates.h中，用于处理分布式中kernel的步长、循环等计算
2. 除naive外，其他三种优化的gpu kernel调用时间相近
3. torch_nn的launch kernel花更多的时间，compiled的cudaStreamIsCapturing花更多时间
4. compile增加的cuda调用
    a. cudaMemcpyAsync
    b. cuModuleLoadData
    c. cuLaunchKernel(与cudaLaunchKernel有何不同？这里compile中节省了一次cudaLaunchKernel)
    d. cudaDeviceSynchronize
    e. cudaStreamSynchronize

Autotune

以attention_pytorch为例

```python
def attention_pytorch(q, k, v, dropout_p=0.0, causal=True):
    """
    Arguments:
        qkv: (batch_size, seqlen, 3, nheads, head_dim)
        dropout_p: float
    Output:
        output: (batch_size, seqlen, nheads, head_dim)
    """
    batch_size, seqlen, nheads, d = q.shape

    q = torch.permute(q, (0, 2, 1, 3)).reshape(-1, seqlen, d)
    k = torch.permute(k, (0, 2, 3, 1)).reshape(-1, d, seqlen)
    v = torch.permute(v, (0, 2, 1, 3)).reshape(-1, seqlen, d)

    softmax_scale = 1.0 / math.sqrt(d)

    # Preallocate attn_weights for `baddbmm`
    scores = torch.empty(batch_size * nheads, seqlen, seqlen, dtype=q.dtype, device=q.device)
    scores = torch.baddbmm(scores, q, k, beta=0, alpha=softmax_scale) # ((b h) t s)

    if causal:
        # "triu_tril_cuda_template" not implemented for 'BFloat16'
        # So we have to construct the mask in float
        causal_mask = torch.triu(torch.full((seqlen, seqlen), -10000.0, device=scores.device), 1)
        # TD [2022-09-30]: Adding is faster than masked_fill_ (idk why, just better kernel I guess)
        scores = scores + causal_mask.to(dtype=scores.dtype)
    attention = torch.softmax(scores, dim=-1)
    attention_drop = torch.nn.functional.dropout(attention, dropout_p)

    output = torch.bmm(attention_drop, v).reshape(batch_size, nheads, -1, d) # (b h t d)

    return output.to(dtype=q.dtype)
```

DEBUG信息

1. 通过设置TORCH_COMPILE_DEBUG=1，可以得到inductor编译流程
    a. Step 1: torchdynamo start tracing attention_pytorch
    b. Step 2: calling compiler function inductor
    c. Step 3: torchinductor compiling FORWARDS graph 0（在第3步会发生autotune，指定tuner并执行得到参数）
2. 由输出的output code文件可以得出
    a. Compiled attention的计算被切分到三个阶段，每个阶段的生成代码均调用一次triton_heuristic的调优函数（分别为pointwise, pointwise, persistent_reduction），以装饰器的形式放置于生成代码之前

```
@pointwise(
    size_hints=[4194304],
    filename=__file__,
    meta={
        'signature': {0: '*fp16', 1: '*fp16', 2: 'i32'},
        'device': 0,
        'constants': {},
        'mutated_arg_names': [],
        'autotune_hints': set(),
        'configs': [instance_descriptor(divisible_by_16=(0, 1, 2), equal_to_1=())]
    }
)
@triton.jit
def triton_(in_ptr0, out_ptr0, xnumel, XBLOCK : tl.constexpr)
```

    b. 三部分调优的结果分别为（按先后次序），其中

```
CachingAutotuner gets 2 configs
XBLOCK: 1024, num_warps: 4, num_stages: 1
XBLOCK: 512, num_warps: 8, num_stages: 1
CachingAutotuner gets 1 configs
XBLOCK: 32, YBLOCK: 32, num_warps: 4, num_stages: 1
CachingAutotuner gets 1 configs
num_warps: 4, num_stages: 1
```

        i. 第一阶段的pointwise调优产生两组参数
        ii. 第二阶段的pointwise调优增加了参数YBLOCK

    iii. 第三阶段的persistent_reduction调优固定了XBLOCK参数为1，RBLOCK为512，只调整num_warps和num_stages

   c. 调用benchmark比较第一阶段两组参数的性能，选择更优的一组存储在best_config文件中

```
Benchmark all input configs get:
XBLOCK: 1024, num_warps: 4, num_stages: 1: 0.008192, nreg 16, nspill 0, #shared-mem 256
XBLOCK: 512, num_warps: 8, num_stages: 1: 0.006144, nreg 16, nspill 0, #shared-mem 0
Save heuristic tuning result to /tmp/torchinductor_lizhixin/qb/cqbhkxcb5cfx457cvt73dibr5nbnjasevbn3l2gvm42733qw2i3r.best_config

Function                                    Runtimes (s)
----------------------------------------  --------------
CachingAutotuner.benchmark_all_configs           0.103
```

调优空间(triton_heuristics.py)

环境配置

1. max_autotune：打开后扩大参数空间，减缓调优速度，默认关闭
2. autotune_pointwise：为布局复杂的pointwise调优，默认打开，只有调试时关闭
3. TritonKernel参数reduction_hint：默认为ReductionHint.DEFAULT
4. torch.are_deterministic_algorithms_enabled：是否要求操作必须为确定性的（给定输入，输出固定）

公共参数

1. size_hints：轴值范围，self.numels数组的每个元素取next power of 2
2. Meta：包含signature, device, kernel_name等信息
   a. signature：kernel各个输入参数的类型
   b. Device & device_type：设备编号、后端类型
   c. constants：待调的tl.constexpr参数，dict形式，key值为其在参数列表中的序号
3. filename：参数存储位置，不必关心

选择规则

1. 不具有reduction计算的使用pointwise

2. 具有reduction计算的，要求最内层维度大于1。若满足以下全部条件，使用persistent_reduction，否则使用reduction
    a. 环境配置参数config.triton.persistent_reductions为真（没有发现设置该变量为真的位置？？？）
    b. 最后一维是静态形状
    c. 最后一维不超过阈值（INNER模式下阈值为1024，其他为64）

Pointwise

1. 独有参数：tile_hint，表示参数空间模式，只在size_hints为2维时有效
2. 处理size_hints长度为1，2，3的情况，分别选定不同的参数范围，使用triton_config包装后，组成的列表调用cached_autotune并返回
3. 参数选择依照的标准（nightly与stable版本在这里有区别）
    a. 一维空间
        i. 禁用：bs
        ii. 启用：bs, bs//2（示例中的第一组pointwise调优情况）
    b. 二维空间
        i. 禁用：32 & 32（示例中的第二组pointwise调优情况）
        ii. 启用：1, 16, 32, 64, 256, bs
    c. 三维空间
        i. 禁用：16 & 16 & 16
        ii. 启用：1, 8, 16, 64, bs
4. 禁用条件
    a. disable_pointwise_autotuning == False
    b. config.max_autotune == False
    c. config.max_autotune_pointwise == False
    d. tile_hint == TileHint.SQUARE

Reduction

1. 独有参数：reduction_hint，默认为false，但不是布尔变量，可选项
    a. ReductionHint.INNER(0)
    b. ReductionHint.OUTER(1)
    c. ReductionHint.OUTER_TINY(2)

   d. ReductionHint.DEFAULT(3)
2. size_hints要求长度为2，使用triton_config_reduction包装
3. 多个候选参数，依据reduction_hint选择某一组或若干组
   a. INNER: contiguous_config, xblock小，rblock大
   b. OUTER: outer_config, xblock大，rblock小
   c. OUTER_TINY: tiny_config
   d. disable_pointwise_autotuning: 32&128
   e. DEFAULT: contiguous + outer + tiny + 64&64 + 8&512 + 64&4

Persistent Reduction

1. size_hints要求长度为2，使用triton_config_reduction包装
2. 默认：1&rnumel, 8&rnumel, 32&rnumel, 128&rnumel
3. 根据reduction_hint选择不同参数，默认全选
   a. INNER&disable_pointwise_autotuning: 1&rnumel
   b. OUTER: 128&rnumel
   c. OUTER_TINY:
   d. DEFAULT: all configs

参数生成

triton_config

1. 参数
   a. size_hints：节点的size_hints
   b. x, y, z：节点的XBLOCK, YBLOCK, ZBLOCK初始参数（y和z可以为空）
   c. num_stages：默认为1
2. 边界maxGridSize：最好能从设备信息获取
3. 条件：x, y, z不能超过size_hints相应维度上的值，且与该维度的MaxGridSize相乘不能超过size_hints，xyz三项乘积不超过初始值的乘积（blocksize总数固定）
4. 调整：在满足以上条件的情况下，每次翻倍，xyz先后处理
5. 其他：num_warps为blocksize除以256后在1～8范围内的2的幂，num_stages直接使用传入参数值

triton_config_reduction

1. 参数
    a. size_hints
    b. x, r：节点的XBLOCK, RBLOCK初始参数，均必需
    c. num_stages默认为2
2. 条件：x, r不能超过size_hints相应维度上的值，与triton_config同样的增长方式
3. 其他：num_warps为blocksize除以128后在2～8范围内的2的幂，num_stages直接使用传入参数值

调优器

cached_autotune

1. 本质是triton.autotune，增加了调试、异常处理、盘上缓存
2. 返回一个CachingAutotuner的实例

CachingAutotuner

1. 继承了triton的KernelInterface，是Triton autotuner的简化版
2. 增加了最佳参数的缓存，可以不依赖triton jit而预编译参数（使用torch.compile接口）
3. 运行tuner时，调用precompile函数中预编译和执行参数launcher，调用autotune_to_one_config选择最佳参数，返回结果

Layout探索：手写triton kernel

Pointwise bias dropout add

1. 令bias, residual采用与inp不同的layout，三个张量均为二维，数据类型为float32
    a. torch实现

    ```
    def bias_dp_add(inp, bias, residual):
        bias = torch.transpose(bias, 0, 1)
        out = torch.nn.functional.dropout(inp + bias, p=0.5, training=False)
        residual = torch.transpose(residual, 0, 1)
        out = residual + out
        return out
    ```

    b. triton实现使输出布局与bias, residual保持一致：XBLOCK, YBLOCK, num_warps, num_stages参与调优

```python
@triton.autotune(configs=configs, key=['xnumel', 'ynumel'])
@triton.jit
def bias_dp_add(
    inp_ptr, bias_ptr, resi_ptr, out_ptr,
    xnumel, ynumel,
    XBLOCK: tl.constexpr,
    YBLOCK: tl.constexpr):
    xindex = tl.program_id(0) * XBLOCK
    yindex = tl.program_id(1) * YBLOCK
    xoffset = tl.arange(0, XBLOCK)[:, None]
    yoffset = tl.arange(0, YBLOCK)[None, :]
    xmask = xindex + xoffset < xnumel
    ymask = yindex + yoffset < ynumel
    inp_ptrs = inp_ptr + (xindex + xoffset) * ynumel + yindex + yoffset
    bias_ptrs = bias_ptr + (yindex + yoffset) * xnumel + xindex + xoffset
    resi_ptrs = resi_ptr + (yindex + yoffset) * xnumel + xindex + xoffset
    out_ptrs = out_ptr + (yindex + yoffset) * xnumel + xindex + xoffset
    inp = tl.load(inp_ptrs, mask=xmask & ymask, eviction_policy='evict_last')
    bias = tl.load(bias_ptrs, mask=xmask & ymask, eviction_policy='evict_last')
    resi = tl.load(resi_ptrs, mask=xmask & ymask, eviction_policy='evict_last')

    tmp = inp + bias
    out = tmp + resi
    tl.store(out_ptrs, out, mask=xmask & ymask)
```

2. 正确性检查通过的条件：atol=1e-3, rtol=1e-2
3. 部分结果：大部分形状与inductor compiled性能持平，调优得到的BLOCK_SIZE比较单一（手写的自动调优范围还是比较大的，性能居然刚刚追上inductor，只在少数形状上互有胜负，看来inductor的调优空间确实精简得不错）

```
32  :  {'num_warps': 2, 'num_stages': 1, 'constants': {6: 32, 7: 16}, 'debug': None, 'arch': 86, 'device_type': 'cuda', 'shared': 2112, 'name': 'bias_dp_add_0d1d2d3d4d5d'}
64  :  {'num_warps': 2, 'num_stages': 1, 'constants': {6: 32, 7: 16}, 'debug': None, 'arch': 86, 'device_type': 'cuda', 'shared': 2112, 'name': 'bias_dp_add_0d1d2d3d4d5d'}
128  :  {'num_warps': 2, 'num_stages': 1, 'constants': {6: 32, 7: 16}, 'debug': None, 'arch': 86, 'device_type': 'cuda', 'shared': 2112, 'name': 'bias_dp_add_0d1d2d3d4d5d'}
256  :  {'num_warps': 2, 'num_stages': 1, 'constants': {6: 32, 7: 64}, 'debug': None, 'arch': 86, 'device_type': 'cuda', 'shared': 8448, 'name':
```

```
'bias_dp_add_0d1d2d3d4d5d'}
512   :  {'num_warps': 4, 'num_stages': 3, 'constants': {6: 64, 7: 64}, 'debug': None, 'arch': 86, 'device_type': 'cuda', 'shared': 16640, 'name': 'bias_dp_add_0d1d2d3d4d5d'}
1024   :  {'num_warps': 2, 'num_stages': 1, 'constants': {6: 32, 7: 64}, 'debug': None, 'arch': 86, 'device_type': 'cuda', 'shared': 8448, 'name': 'bias_dp_add_0d1d2d3d4d5d'}
2048   :  {'num_warps': 4, 'num_stages': 2, 'constants': {6: 32, 7: 64}, 'debug': None, 'arch': 86, 'device_type': 'cuda', 'shared': 8448, 'name': 'bias_dp_add_0d1d2d3d4d5d'}
4096   :  {'num_warps': 4, 'num_stages': 1, 'constants': {6: 32, 7: 64}, 'debug': None, 'arch': 86, 'device_type': 'cuda', 'shared': 8448, 'name': 'bias_dp_add_0d1d2d3d4d5d'}
8192   :  {'num_warps': 2, 'num_stages': 1, 'constants': {6: 32, 7: 64}, 'debug': None, 'arch': 86, 'device_type': 'cuda', 'shared': 8448, 'name': 'bias_dp_add_0d1d2d3d4d5d'}
16384   :  {'num_warps': 2, 'num_stages': 1, 'constants': {6: 32, 7: 64}, 'debug': None, 'arch': 86, 'device_type': 'cuda', 'shared': 8448, 'name': 'bias_dp_add_0d1d2d3d4d5d'}
32768   :  {'num_warps': 2, 'num_stages': 1, 'constants': {6: 32, 7: 64}, 'debug': None, 'arch': 86, 'device_type': 'cuda', 'shared': 8448, 'name': 'bias_dp_add_0d1d2d3d4d5d'}
65536   :  {'num_warps': 4, 'num_stages': 3, 'constants': {6: 32, 7: 64}, 'debug': None, 'arch': 86, 'device_type': 'cuda', 'shared': 8448, 'name': 'bias_dp_add_0d1d2d3d4d5d'}
bias-dp-add-perf: (GB/s)
      dim_0      Triton       Torch    Inductor
0      32.0  102.400003   64.000000  102.400003
1      64.0  204.800005  113.777774  204.800005
2     128.0  341.333321  204.800005  341.333321
3     256.0  458.293714  292.571425  504.123076
4     512.0  630.153853  356.173905  630.153853
5    1024.0  744.727267  409.600010  738.433810
6    2048.0  798.610823  442.810792  799.219525
7    4096.0  829.569645  428.339867  829.569645
8    8192.0  851.116917  384.375359  851.116917
9   16384.0  862.315828  378.820824  862.315828
10  32768.0  868.026489  268.452641  868.026489
11  65536.0  870.345562  111.089001  868.745655
```

4. 以形状256*128为例，inductor autotune的最佳配置是2*128，num_warps=4, num_stages=1

```
32  :  {'num_warps': 2, 'num_stages': 3, 'constants': {6: 2, 7: 16}, 'debug': None, 'arch': 86, 'device_type': 'cuda', 'shared': 0, 'name': 'bias_dp_add_0d1d2d3d4d5d'}
```

```
64  :  {'num_warps': 2, 'num_stages': 3, 'constants': {6: 2, 7: 16}, 'debug': None, 'arch': 86, 'device_type': 'cuda', 'shared': 0, 'name': 'bias_dp_add_0d1d2d3d4d5d'}
128  :  {'num_warps': 4, 'num_stages': 1, 'constants': {6: 2, 7: 32}, 'debug': None, 'arch': 86, 'device_type': 'cuda', 'shared': 0, 'name': 'bias_dp_add_0d1d2d3d4d5d'}
256  :  {'num_warps': 4, 'num_stages': 1, 'constants': {6: 32, 7: 16}, 'debug': None, 'arch': 86, 'device_type': 'cuda', 'shared': 2112, 'name': 'bias_dp_add_0d1d2d3d4d5d'}
512  :  {'num_warps': 2, 'num_stages': 1, 'constants': {6: 32, 7: 16}, 'debug': None, 'arch': 86, 'device_type': 'cuda', 'shared': 2112, 'name': 'bias_dp_add_0d1d2d3d4d5d'}
1024  :  {'num_warps': 4, 'num_stages': 1, 'constants': {6: 32, 7: 16}, 'debug': None, 'arch': 86, 'device_type': 'cuda', 'shared': 2112, 'name': 'bias_dp_add_0d1d2d3d4d5d'}
2048  :  {'num_warps': 2, 'num_stages': 1, 'constants': {6: 32, 7: 16}, 'debug': None, 'arch': 86, 'device_type': 'cuda', 'shared': 2112, 'name': 'bias_dp_add_0d1d2d3d4d5d'}
4096  :  {'num_warps': 4, 'num_stages': 2, 'constants': {6: 128, 7: 32}, 'debug': None, 'arch': 86, 'device_type': 'cuda', 'shared': 16512, 'name': 'bias_dp_add_0d1d2d3d4d5d'}
8192  :  {'num_warps': 2, 'num_stages': 1, 'constants': {6: 32, 7: 64}, 'debug': None, 'arch': 86, 'device_type': 'cuda', 'shared': 8448, 'name': 'bias_dp_add_0d1d2d3d4d5d'}
16384  :  {'num_warps': 2, 'num_stages': 1, 'constants': {6: 32, 7: 32}, 'debug': None, 'arch': 86, 'device_type': 'cuda', 'shared': 4224, 'name': 'bias_dp_add_0d1d2d3d4d5d'}
32768  :  {'num_warps': 2, 'num_stages': 2, 'constants': {6: 32, 7: 32}, 'debug': None, 'arch': 86, 'device_type': 'cuda', 'shared': 4224, 'name': 'bias_dp_add_0d1d2d3d4d5d'}
65536  :  {'num_warps': 2, 'num_stages': 2, 'constants': {6: 32, 7: 16}, 'debug': None, 'arch': 86, 'device_type': 'cuda', 'shared': 2112, 'name': 'bias_dp_add_0d1d2d3d4d5d'}
bias-dp-add-perf:
      dim_0      Triton       Torch    Inductor
0      32.0   19.230048    9.142857   16.000000
1      64.0   32.000000   18.285714   32.000000
2     128.0   64.000000   35.617391   64.000000
3     256.0  102.400003   64.000000  106.389615
4     512.0  204.800005  113.777774  204.800005
5    1024.0  341.333321  204.800005  341.333321
6    2048.0  455.111095  278.876596  455.111095
7    4096.0  630.153853  372.363633  630.153853
8    8192.0  713.316998  431.157914  728.683801
9   16384.0  799.219525  468.114273  799.219525
10  32768.0  829.569645  451.972420  829.569645
11  65536.0  846.820877  183.574224  851.116917
```

5. inductor每次调出的带宽也不都是一样好