case-study-2-demo-output

January 4, 2024

[1]:

```
import sys
sys.path.append('../src')
import MODULE_CQS_Attention as cqs_att
import torch, math, gc
from time import time, sleep
from statistics import mean
```

[2]:

```
# Set W and N values
Ws = [4, 7, 8, 31]
Ns = [10000, 20000, 30000, 40000, 45000, 46000, 47000, 48000, 49000]
```

[3]:

```
# Determine the length of the longest subsequence a device receives

# W = 1 stores all N tokens locally
local_sequence_lengths = {1:Ns.copy()}
for W in Ws:
    longest_subsequence_length = []
    for N in Ns:
        scheduler = cqs_att.Scheduler(N,W)
        longest_subsequence_length.append(scheduler.longest_subsequence())
    local_sequence_lengths[W] = longest_subsequence_length

print('Longest subsequence a worker receives')
local_sequence_lengths
```

```
Longest subsequence a worker receives
```

[3]:

```
{1: [10000, 20000, 30000, 40000, 45000, 46000, 47000, 48000, 49000],
 4: [7500, 15000, 22500, 30000, 33750, 34500, 35250, 36000, 36750],
 7: [4287, 8572, 12858, 17144, 19287, 19715, 20144, 20572, 21000],
 8: [5000, 10000, 15000, 20000, 22500, 23000, 23500, 24000, 24500],
 31: [1937, 3873, 5808, 7743, 8711, 8904, 9099, 9292, 9486]}
```

[4]:

```
# This function is borrowed from Pytorch, available here: https://pytorch.org/
 ↪docs/stable/generated/torch.nn.functional.scaled_dot_product_attention.html
# We did not call scaled_dot_product_attention() from torch to avoid any␣
 ↪internal optimizations
```

```
# Therefore, the wall-clock time advantages are brought purely by␣
 ↪CQS_Attention, or fewer local tokens
def scaled_dot_product_attention(query, key, value, attn_mask=None, dropout_p=0.
 ↪0, is_causal=False, scale=None) -> torch.Tensor:
    # Efficient implementation equivalent to the following:
    L, S = query.size(-2), key.size(-2)
    scale_factor = 1 / math.sqrt(query.size(-1)) if scale is None else scale
    attn_bias = torch.zeros(L, S, dtype=query.dtype, device = 'cuda')
    attn_weight = query @ key.transpose(-2, -1) * scale_factor
    attn_weight += attn_bias
    attn_weight = torch.softmax(attn_weight, dim=-1)
    attn_weight = torch.dropout(attn_weight, dropout_p, train=True)  # applied␣
 ↪to balance the workload
    return attn_weight @ value

# To record the time of a single computation
def Attention_computation_timer(seq_len):
    t1 = time()
    res = scaled_dot_product_attention(torch.rand(1, 8, seq_len, 64,␣
 ↪dtype=torch.float16, device="cuda"),torch.rand(1, 8, seq_len, 64,␣
 ↪dtype=torch.float16, device="cuda"),torch.rand(1, 8, seq_len, 64,␣
 ↪dtype=torch.float16, device="cuda"))
    t_consumed = (time() - t1)*1000
    torch.cuda.empty_cache()
    gc.collect()
    return t_consumed

# Determine the average wall-clock time in each scenario
def average_wall_clock_time_in_each_scenario(local_sequence_lengths,␣
 ↪repeat_time, display = False):
    average_wall_clock_time_in_each_scenario = {}
    for k, v in local_sequence_lengths.items():
        if display:
            print(f'\nW = {k}')
        avg_wall_clock_times = []
        for N in v:
            if display:
                print(f'mTk = {N}')
            wall_clock_times = []
            for _ in range(repeat_time):
                sleep(2)
                torch.cuda.empty_cache()
                gc.collect()
                wall_clock_time = Attention_computation_timer(N)
                if display:
                    print(wall_clock_time)
                wall_clock_times.append(wall_clock_time)
            wall_clock_times.sort()
            med_val = wall_clock_times[len(wall_clock_times)//2]
            # remove outliers
            while wall_clock_times[-1] >= med_val * 1.5:
                wall_clock_times.pop()
            avg_wall_clock_times.append(mean(wall_clock_times))
        average_wall_clock_time_in_each_scenario[k] = avg_wall_clock_times.
 ↪copy()
    return average_wall_clock_time_in_each_scenario
```

[5]:

```
repeat_time = 5
average_wall_clock_times =␣
 ↪average_wall_clock_time_in_each_scenario(local_sequence_lengths,␣
 ↪repeat_time, display = True)
average_wall_clock_times
```

```
W = 1
mTk = 10000
382.587194442749
3.394603729248047
3.4351348876953125
3.4613609313964844
3.4329891204833984
mTk = 20000
8.713245391845703
7.965803146362305
7.990121841430664
8.147001266479492
8.01396369934082
mTk = 30000
16.299962997436523
16.207218170166016
16.134262084960938
16.1135196685791
16.17908477783203
mTk = 40000
27.44913101196289
27.431011199951172
27.425765991210938
27.831554412841797
27.40025520324707
mTk = 45000
646.38352394104
824.3598937988281
826.7350196838379
826.5635967254639
```

```
819.6568489074707
mTk = 46000
845.4561233520508
866.6074275970459
867.631196975708
863.3990287780762
864.3929958343506
mTk = 47000
887.4256610870361
908.390998840332
908.710241317749
900.5296230316162
904.8421382904053
mTk = 48000
923.9037036895752
943.6993598937988
943.6748027801514
943.0835247039795
947.0946788787842
mTk = 49000
970.177173614502
985.8963489532471
988.3999824523926
988.5752201080322
987.5962734222412

W = 4
mTk = 7500
19.75393295288086
2.4864673614501953
2.454996109008789
2.4645328521728516
2.4993419647216797
mTk = 15000
5.496740341186523
5.156040191650391
5.128145217895508
5.157709121704102
5.134105682373047
mTk = 22500
9.888648986816406
9.746074676513672
9.687662124633789
9.732484817504883
9.767293930053711
mTk = 30000
16.097545623779297
16.112804412841797
```

```
16.150712966918945
16.093015670776367
17.731428146362305
mTk = 33750
20.267248153686523
20.064830780029297
21.403074264526367
19.997835159301758
20.050525665283203
mTk = 34500
20.99442481994629
20.876169204711914
21.068334579467773
20.79296112060547
20.835399627685547
mTk = 35250
21.912813186645508
21.625280380249023
21.85797691345215
21.69346809387207
21.663665771484375
mTk = 36000
22.618532180786133
22.56917953491211
22.541046142578125
22.847414016723633
22.487163543701172
mTk = 36750
511.120080947876
23.5750675201416
23.38576316833496
23.39005470275879
23.37360382080078

W = 7
mTk = 4287
1.5082359313964844
1.3935565948486328
1.4081001281738281
1.390218734741211
1.4271736145019531
mTk = 8572
2.481698989868164
2.8197765350341797
2.8629302978515625
2.8710365295410156
2.8543472290039062
mTk = 12858
```

```
4.955053329467773
6.749391555786133
4.188776016235352
4.157781600952148
4.159212112426758
mTk = 17144
6.304025650024414
6.289958953857422
6.262540817260742
6.218671798706055
6.236791610717773
mTk = 19287
85.2663516998291
7.470130920410156
9.140253067016602
7.562398910522461
7.526874542236328
mTk = 19715
7.897615432739258
7.765531539916992
7.752180099487305
7.807493209838867
7.797002792358398
mTk = 20144
8.115768432617188
8.055925369262695
8.093833923339844
8.046865463256836
7.957935333251953
mTk = 20572
8.480072021484375
8.396387100219727
8.3770751953125
8.413314819335938
8.416891098022461
mTk = 21000
8.733987808227539
8.621454238891602
8.938074111938477
10.587453842163086
8.771896362304688

W = 8
mTk = 5000
1.5997886657714844
1.5151500701904297
1.5554428100585938
1.5101432800292969
```

```
1.5401840209960938
mTk = 10000
2.818584442138672
3.4639835357666016
3.4592151641845703
3.4334659576416016
3.441333770751953
mTk = 15000
5.70225715637207
5.178213119506836
5.157947540283203
5.152225494384766
5.142688751220703
mTk = 20000
7.965087890625
7.979154586791992
8.009195327758789
11.105775833129883
7.99250602722168
mTk = 22500
9.792327880859375
9.734153747558594
9.654045104980469
9.692668914794922
9.608268737792969
mTk = 23000
10.155677795410156
10.058879852294922
10.072469711303711
10.074853897094727
10.199785232543945
mTk = 23500
10.756492614746094
10.46299934387207
10.511398315429688
10.524988174438477
10.568380355834961
mTk = 24000
10.988235473632812
10.902643203735352
10.871410369873047
11.012077331542969
10.864973068237305
mTk = 24500
11.40451431274414
11.271953582763672
11.240482330322266
11.346578598022461
```

```
11.317968368530273

W = 31
mTk = 1937
1.1074542999267578
5.014657974243164
1.1212825775146484
1.1324882507324219
1.168966293334961
mTk = 3873
1.4147758483886719
1.4050006866455078
1.3060569763183594
1.367330551147461
1.356363296508789
mTk = 5808
1.7044544219970703
1.6760826110839844
1.7180442810058594
1.7142295837402344
1.6717910766601562
mTk = 7743
2.2535324096679688
2.594470977783203
2.5148391723632812
2.5360584259033203
2.568960189819336
mTk = 8711
2.8870105743408203
2.8998851776123047
2.9518604278564453
2.9070377349853516
2.9098987579345703
mTk = 8904
3.0126571655273438
3.011465072631836
2.9706954956054688
3.0045509338378906
2.96783447265625
mTk = 9099
3.1316280364990234
3.0405521392822266
3.058195114135742
3.0956268310546875
3.0515193939208984
mTk = 9292
3.373861312866211
3.153562545776367
```

```
3.2930374145507812
4.288196563720703
3.1545162200927734
mTk = 9486
3.300189971923828
3.293752670288086
3.2417774200439453
3.220796585083008
3.2279491424560547
```

```
[5]: {1: [3.4310221672058105,
       8.166027069091797,
       16.186809539794922,
       27.507543563842773,
       788.7397766113281,
       861.4973545074463,
       901.9797325134277,
       940.2912139892578,
       984.128999710083],
      4: [2.476334571838379,
       5.214548110961914,
       9.764432907104492,
       16.437101364135742,
       20.35670280456543,
       20.9134578704834,
       21.750640869140625,
       22.612667083740234,
       23.431122303009033],
      7: [1.4254570007324219,
       2.7779579162597656,
       4.365205764770508,
       6.262397766113281,
       7.924914360046387,
       7.803964614868164,
       8.054065704345703,
       8.416748046875,
       9.130573272705078],
      8: [1.5441417694091797,
       3.3233165740966797,
       5.266666412353516,
       8.610343933105469,
       9.696292877197266,
       10.112333297729492,
       10.564851760864258,
       10.927867889404297,
       11.316299438476562],
      31: [1.1325478553771973,
1.3699054718017578,
1.696920394897461,
2.493572235107422,
2.9111385345458984,
2.993440628051758,
3.0755043029785156,
3.452634811401367,
3.2568931579589844]}
```