demo-output

January 4, 2024

[1]:
```
import MODULE_CQS_Attention as cqs_att
import numpy as np
```

[2]:
```
# Set the value for whole sequence length (N), head dimension (d), and # worker devices (W)
N = 11
d = 5
W = 7
```

[3]:
```
# Initialize Q, K, V, and CQS_Attention
Q = np.random.rand(N,d)
K = np.random.rand(N,d)
V = np.random.rand(N,d)
cqs_attention = cqs_att.CQS_Attention(Q,K,V,W)
```

[4]:
```
################ WORKFLOW ################

# Run the workflow, including Scheduler computing the partition,
# Workers doing local computation, and Titler putting local results together.
# Output attention is stored as "cqs_attention.O"
# For a clean look, set "display = False". It mutes the workflow details.
# We suggest to set it to True when N < 30, modify as you wish.

if N < 30:
    display = True
else:
    display = False
cqs_attention.workflow(display = display)
```

```
===== Scheduler =====
N = 11, W = 7, d = 5

Interest Set
[0, 1, 5]

TG-Tk map
{0: [0], 1: [1], 2: [2], 3: [3, 4], 4: [5, 6], 5: [7, 8], 6: [9, 10]}
```

CQS
[[0, 1, 5], [1, 2, 6], [2, 3, 0], [3, 4, 1], [4, 5, 2], [5, 6, 3], [6, 0, 4]]

undistilled pair list
[[(0, 1), (0, 5), (1, 5)], [(1, 2), (1, 6), (2, 6)], [(2, 3), (0, 2), (0, 3)],
[(3, 4), (1, 3), (1, 4)], [(4, 5), (2, 4), (2, 5)], [(5, 6), (3, 5), (3, 6)],
[(0, 6), (4, 6), (0, 4)]]

distilled pair list
[[(0, 1), (0, 5), (1, 5)], [(1, 2), (1, 6), (2, 6)], [(2, 3), (0, 2), (0, 3)],
[(3, 4), (1, 3), (1, 4)], [(4, 5), (2, 4), (2, 5)], [(5, 6), (3, 5), (3, 6)],
[(0, 6), (4, 6), (0, 4)]]

distilled CQS
[[0, 1, 5], [1, 2, 6], [0, 2, 3], [1, 3, 4], [2, 4, 5], [3, 5, 6], [0, 4, 6]]

MtrlL
[[0, 1, 7, 8], [1, 2, 9, 10], [0, 2, 3, 4], [1, 3, 4, 5, 6], [2, 5, 6, 7, 8],
[3, 4, 7, 8, 9, 10], [0, 5, 6, 9, 10]]

Task lists (ONLY for validation purpose)
[[(0, 0), (0, 1), (1, 0), (0, 7), (7, 0), (0, 8), (8, 0), (1, 7), (7, 1), (1,
8), (8, 1)], [(1, 1), (1, 2), (2, 1), (1, 9), (9, 1), (1, 10), (10, 1), (2, 9),
(9, 2), (2, 10), (10, 2)], [(2, 2), (2, 3), (3, 2), (2, 4), (4, 2), (0, 2), (2,
0), (0, 3), (3, 0), (0, 4), (4, 0)], [(3, 3), (3, 4), (4, 3), (4, 4), (3, 5),
(5, 3), (3, 6), (6, 3), (4, 5), (5, 4), (4, 6), (6, 4), (1, 3), (3, 1), (1, 4),
(4, 1), (1, 5), (5, 1), (1, 6), (6, 1)], [(5, 5), (5, 6), (6, 5), (6, 6), (5,
7), (7, 5), (5, 8), (8, 5), (6, 7), (7, 6), (6, 8), (8, 6), (2, 5), (5, 2), (2,
6), (6, 2), (2, 7), (7, 2), (2, 8), (8, 2)], [(7, 7), (7, 8), (8, 7), (8, 8),
(7, 9), (9, 7), (7, 10), (10, 7), (8, 9), (9, 8), (8, 10), (10, 8), (3, 7), (7,
3), (3, 8), (8, 3), (4, 7), (7, 4), (4, 8), (8, 4), (3, 9), (9, 3), (3, 10),
(10, 3), (4, 9), (9, 4), (4, 10), (10, 4)], [(9, 9), (9, 10), (10, 9), (10, 10),
(0, 9), (9, 0), (0, 10), (10, 0), (5, 9), (9, 5), (5, 10), (10, 5), (6, 9), (9,
6), (6, 10), (10, 6), (0, 5), (5, 0), (0, 6), (6, 0)]]

Ban lists (global index)
[[(1, 1), (7, 7), (7, 8), (8, 7), (8, 8)], [(9, 9), (9, 10), (10, 9), (10, 10),
(2, 2)], [(3, 3), (3, 4), (4, 3), (4, 4), (0, 0)], [(5, 5), (5, 6), (6, 5), (6,
6), (1, 1)], [(7, 7), (7, 8), (8, 7), (8, 8), (2, 2)], [(9, 9), (9, 10), (10,
9), (10, 10), (3, 3), (3, 4), (4, 3), (4, 4)], [(5, 5), (5, 6), (6, 5), (6, 6),
(0, 0)]]

Ban lists (reindexed)
[[(1, 1), (2, 2), (2, 3), (3, 2), (3, 3)], [(2, 2), (2, 3), (3, 2), (3, 3), (1,
1)], [(2, 2), (2, 3), (3, 2), (3, 3), (0, 0)], [(3, 3), (3, 4), (4, 3), (4, 4),
(0, 0)], [(3, 3), (3, 4), (4, 3), (4, 4), (0, 0)], [(4, 4), (4, 5), (5, 4), (5,
5), (0, 0), (0, 1), (1, 0), (1, 1)], [(1, 1), (1, 2), (2, 1), (2, 2), (0, 0)]]

```
===== Workers =====
Worker 0: mTki = 4, d = 5
Pi: (4, 4)
[[6.24964367 5.1074789  2.88217038 2.62784611]
 [8.18990266 0.         5.50351193 4.13911604]
 [5.36170445 3.94461848 0.         0.        ]
 [2.92905058 2.19581023 0.         0.        ]]

Si.T: (4,)
[16.86713906 17.83253063  9.30632293  5.12486081]

Oi: (4, 5)
[[ 9.52717358  7.27742826 11.96338326  5.29616513  8.1918431 ]
 [ 8.10229432  8.12592018 13.7555127   8.32388644 10.92211307]
 [ 5.03646366  2.52819532  6.21806313  1.1782918   2.98963499]
 [ 2.7875527   1.40183708  3.42040322  0.64469275  1.64459821]]

Worker 1: mTki = 4, d = 5
Pi: (4, 4)
[[7.01997619 4.15256231 4.58972712 5.09688711]
 [2.09241169 0.         1.95304413 1.97707811]
 [2.17796368 2.7585929  0.         0.        ]
 [2.96680346 2.56439678 0.         0.        ]]

Si.T: (4,)
[20.85915273  6.02253393  4.93655658  5.53120024]

Oi: (4, 5)
[[ 8.53679746 10.27191397  9.21312198  5.76397505  6.53008317]
 [ 2.75557357  3.09870379  2.89958659  2.24336165  2.18392143]
 [ 2.00699222  2.19628578  1.92450038  0.29986328  1.01921922]
 [ 2.69908024  2.51866193  2.33107878  0.30184929  1.20978756]]

Worker 2: mTki = 4, d = 5
Pi: (4, 4)
[[0.         2.46791552 5.19994236 3.82604429]
 [2.66906956 1.72735613 1.92799114 1.56317242]
 [2.76133256 1.36861205 0.         0.        ]
 [4.62593019 1.91594046 0.         0.        ]]

Si.T: (4,)
[11.49390216  7.88758925  4.12994461  6.54187065]

Oi: (4, 5)
[[4.84146815 5.34633852 3.44421095 6.07255852 4.93823241]
 [2.69818032 2.62043589 3.45406967 2.97085928 2.96459979]
 [0.83682556 0.81613904 2.36668698 0.67931577 1.17888841]
 [1.39089662 1.21784401 3.873078   1.10435664 1.91854293]]
```

```
Worker 3: mTki = 5, d = 5
Pi: (5, 5)
[[ 0.          4.49669434  5.6367157  14.37770069  8.58791292]
 [ 2.99521167  2.2385523   2.26968932  3.16152805  2.75671761]
 [ 3.51897503  2.36876173  2.06954872  5.38024123  4.67208596]
 [ 5.1154839   5.1204938   4.61282928  0.          0.        ]
 [ 3.571005    3.91617068  2.35054788  0.          0.        ]]

Si.T: (5,)
[33.09902365 13.42169895 18.00961267 14.84880698  9.83772356]

Oi: (5, 5)
[[25.55041629  7.58285525 18.96507575  9.29397552 26.05770863]
 [10.34751841  4.5151009   6.84608538  3.72145948  8.54769982]
 [14.31971288  5.40701254  9.87943461  4.12134032 12.46494977]
 [ 9.85940855  7.19329281  5.82335894  6.52131953  6.33580066]
 [ 6.3498755   4.90758446  4.14566194  4.09704928  4.1761072 ]]

Worker 4: mTki = 5, d = 5
Pi: (5, 5)
[[0.         2.8648122  2.9387977  1.67429674 1.81837516]
 [4.46746454 9.58110051 7.80228421 3.31023316 4.32247465]
 [2.6181131  5.15189098 5.9961519  1.9988217  2.87413094]
 [2.66741365 5.0215082  5.20327861 0.         0.        ]
 [2.7830648  3.55450526 2.97526526 0.         0.        ]]

Si.T: (5,)
[ 9.2962818  29.48355706 18.63910862 12.89220046  9.31283532]

Oi: (5, 5)
[[ 7.11027111  3.45365144  6.41887453  3.19397136  8.23372119]
 [19.52312508  9.69479145 18.70762545  8.00395583 22.89395402]
 [12.66301372  6.29377588 11.2713693   5.17535351 14.68719127]
 [ 9.10469725  2.70992321  6.99354004  1.39944474  9.83457613]
 [ 5.77794824  2.07578501  4.9573573   0.97683914  6.41724437]]

Worker 5: mTki = 6, d = 5
Pi: (6, 6)
[[0.         0.         2.64672912 1.42601376 2.05391949 1.5103148 ]
 [0.         0.         2.70397915 2.17260339 2.68231131 2.72021408]
 [4.44417146 3.12758998 2.25165615 2.79351192 4.89424572 2.40065458]
 [2.3472219  2.30486981 1.65738359 2.70471808 2.65830177 2.67903657]
 [2.32220075 2.368862   1.70930259 2.63327026 0.         0.        ]
 [2.37146626 3.15913568 2.23316095 2.6267218  0.         0.        ]]

Si.T: (6,)
[ 7.63697717 10.27910792 19.9118298  14.35153172  9.0336356  10.39048468]
```

```
Oi: (6, 5)
[[ 3.3644963   5.01346313  4.80988131  4.91557063  4.86676176]
 [ 4.12359684  6.47076904  6.23982151  6.41501691  6.25638072]
 [ 8.61008634 10.78267466  9.71688553 13.71934024 10.9096236 ]
 [ 6.18190682  7.91605588  7.16329045  9.38449096  8.07340498]
 [ 4.99780448  5.16702857  4.83635114  6.39926866  5.90684595]
 [ 5.98392136  5.96376867  5.34168396  7.29241679  6.77084297]]

Worker 6: mTki = 5, d = 5
Pi: (5, 5)
[[0.         5.800527   5.71098208 5.64542484 2.22507319]
 [6.98053316 0.         0.         5.10643104 4.27391314]
 [5.69581792 0.         0.         4.40218761 2.72241023]
 [2.7714028  3.49119378 2.88011116 2.5086496  2.60597135]
 [3.51211656 4.33659452 2.60665688 3.48788264 2.68378043]]

Si.T: (5,)
[19.38200711 16.36087733 12.82041576 14.2573287  16.62703103]

Oi: (5, 5)
[[12.13010485  5.55983839 10.75897995  7.10036063 13.76280806]
 [ 4.22121344  5.46939343  9.21990955  7.00162188  6.2700404 ]
 [ 3.37541727  4.08759605  7.33022889  5.82284927  4.8842035 ]
 [ 7.52771869  3.87859807  8.43662263  4.09460103  8.91616188]
 [ 8.42752723  4.36691077 10.19627137  5.24568165 10.08078131]]

===== Tiler =====
Putting local Oi and Si together:
O
[[26.49874658 18.18360517 26.16657415 18.46908428 26.89288357]
 [42.18950807 25.98068941 41.93371043 23.38183701 43.50990488]
 [12.564025    9.17279112 12.77253079  8.4081923  13.38224241]
 [14.54884028 10.34470306 14.02265366  9.31634587 14.59334999]
 [19.83420634 13.09562558 19.99233412 11.64071387 20.63987341]
 [33.60374707 22.3574777  33.75089395 21.52689723 35.49979509]
 [22.38830649 15.28895639 22.74726013 15.09525206 23.74750197]
 [22.75124725 16.02079319 22.9284887  16.29707678 23.73383473]
 [14.74740776 11.39367796 15.54105097 11.00602285 16.13524756]
 [14.53251539 11.24191243 15.19747415 10.79373297 15.84222706]
 [17.11052883 12.84934137 17.86903411 12.83994773 18.06141184]]

S.T
[47.74304833 71.79070701 23.20640498 25.18862073 34.83059125 60.69324137
 41.29724795 42.11035319 28.78922785 28.22752089 32.54871595]

Final Attention O = O/S
[[0.55502838 0.38086393 0.54807087 0.38684342 0.56328376]
```

```
[0.58767367 0.36189488 0.58411056 0.32569448 0.60606598]
[0.54140333 0.3952698  0.55038817 0.36232205 0.57666159]
[0.57759575 0.41068954 0.5567059  0.36986328 0.57936281]
[0.56944788 0.37598057 0.57398779 0.33420948 0.5925789 ]
[0.55366539 0.36836849 0.55608982 0.3546836  0.58490524]
[0.54212587 0.37021732 0.55081782 0.36552683 0.57503837]
[0.54027681 0.38044785 0.54448578 0.38700879 0.56361044]
[0.51225437 0.39576185 0.53982174 0.38229656 0.56046128]
[0.51483499 0.39826071 0.5383921  0.38238331 0.56123338]
[0.52568983 0.3947726  0.54899352 0.394484   0.55490397]]
```

[5]:
```
################ CORRECTNESS CHECK ################

# Validate the computation correctness by comparing CQS_Attention result with␣
 ↪the normal computation result

# Compute O (named as O_0) directly from Q, K, V
from scipy.special import softmax
O_0 = softmax(Q @ K.T, axis = 1) @ V

# Compare O_0 with CQS_Attention result (cqs_attention.O)
# True for equal, False otherwise.
print('CQS_Attention results is correct:')
np.allclose(O_0,cqs_attention.O)
```

```
CQS_Attention results is correct:
```

[5]:
```
True
```

[6]:
```
# Alternatively, we implement the above validation process as␣
 ↪"validate_computation_correctness()"
print('CQS_Attention results is correct:')
cqs_attention.validate_computation_correctness()
```

```
CQS_Attention results is correct:
```

[6]:
```
True
```

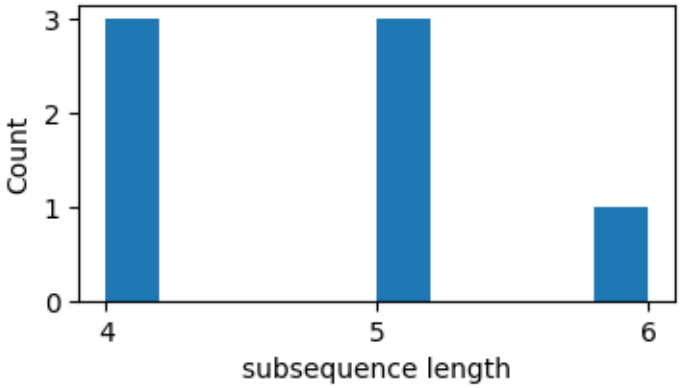
[7]:
```
################ MEMORY CONSUMPTION ################

cqs_attention.memory_consumption_summary()
```

```
FYI, approximation of subsequence length ratio is m / W: 0.42857142857142855.
Most actual ratios are lower, hence better.
Longest subsequence: 6, ratio to N: 0.5454545454545454
Shortest subsequence: 4, ratio to N: 0.36363636363636365
Average length: 4.714285714285714, ratio to N: 0.4285714285714286
```

```
Standard deviation: 0.7559289460184544

Subsequence length distribution
```



[10]:
```
################ VISUALIZE PARTITION OF P ################

# It is NOT recommended to visualize P partition when N is very large due to␣
 ↪long redering time.
# When W is large, visualization may not be as clear, because colors are␣
 ↪randomly generated and there may be very similar ones.
if N < 100:
    cqs_attention.visualize_P_partition()
```