Payman Behnam-U1078370

Parallel Programming of Many-Core

Assignment1

| Size | nelem1 | nelem2 | nelem4 | nelem8 | nelem16 | nelem32 | nelem64 | nelem128 | nelem256 | nelem512 |
|---|---|---|---|---|---|---|---|---|---|---|
| Sz10 | 0.05 | 0.013 | 0.011 | 0.011 | 0.013 | 0.018 | 0.025 | 0.042 | 0.075 | 0.14 |
| Sz100 | 0.01 | 0.01 | 0.01 | 0.011 | 0.013 | 0.018 | 0.025 | 0.044 | 0.075 | 0.141 |
| Sz1000 | 0.025 | 0.01 | 0.01 | 0.011 | 0.014 | 0.018 | 0.028 | 0.047 | 0.085 | 0.162 |
| Sz10000 | 0.009 | 0.009 | 0.01 | 0.011 | 0.013 | 0.017 | 0.027 | 0.045 | 0.084 | 0.161 |
| Sz100000 | 0.019 | 0.011 | 0.016 | 0.02 | 0.024 | 0.032 | 0.053 | 0.088 | 0.148 | 0.28 |
| Sz1000000 | 0.035 | 0.048 | 0.062 | 0.085 | 0.122 | 0.195 | 0.347 | 0.665 | 1.217 | 2.366 |
| Sz10000000 | 0.257 | 0.37 | 0.487 | 0.709 | 1.099 | 1.824 | 3.223 | 6 | 11.657 | 22.899 |
| Sz100000000 | 2.456 | 3.357 | 4.511 | 6.317 | 9.99 | 16.817 | 29.657 | 55.279 | 106.53 | 208.81 |
| Sz1000000000 | 25.276 | 33.444 | 45.131 | 63.06 | 96.046 | 165.87 | 297.33 | 554.34 | 1065.9 | 2090.8 |

*Figure 1-Timing Result for Cyclic distribution*

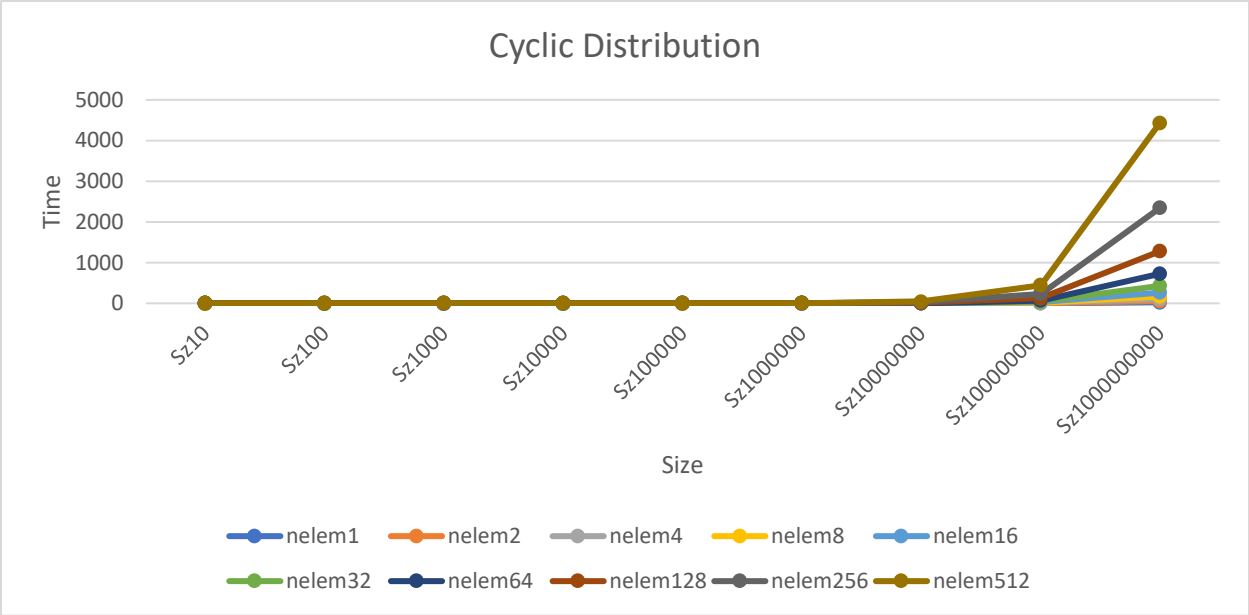
| Size | nelem1 | nelem2 | nelem4 | nelem8 | nelem16 | nelem32 | nelem64 | nelem128 | nelem256 | nelem512 |
|---|---|---|---|---|---|---|---|---|---|---|
| Sz10 | 0.039 | 0.015 | 0.013 | 0.014 | 0.015 | 0.018 | 0.025 | 0.038 | 0.065 | 0.118 |
| Sz100 | 0.012 | 0.012 | 0.012 | 0.013 | 0.016 | 0.021 | 0.029 | 0.044 | 0.065 | 0.119 |
| Sz1000 | 0.013 | 0.012 | 0.013 | 0.014 | 0.017 | 0.023 | 0.03 | 0.049 | 0.085 | 0.158 |
| Sz10000 | 0.013 | 0.012 | 0.015 | 0.025 | 0.033 | 0.044 | 0.044 | 0.054 | 0.098 | 0.173 |
| Sz100000 | 0.014 | 0.013 | 0.016 | 0.028 | 0.052 | 0.129 | 0.245 | 0.373 | 0.397 | 0.444 |
| Sz1000000 | 0.039 | 0.049 | 0.075 | 0.183 | 0.259 | 0.348 | 0.373 | 0.631 | 1.242 | 2.431 |
| Sz10000000 | 0.26 | 0.359 | 0.585 | 1.419 | 2.834 | 3.945 | 6.217 | 8.257 | 14.147 | 19.82 |
| Sz100000000 | 2.482 | 3.446 | 5.2 | 13.964 | 28.141 | 38.872 | 63.185 | 86.865 | 165.508 | 214.959 |
| Sz1000000000 | 24.503 | 33.706 | 51.299 | 139.166 | 281.587 | 389.955 | 632.052 | 868.577 | 1683.033 | 2171.948 |

*Figure 2-Timing Results for Adjacent Distribution*

1. What is the performance impact of increasing the problem size?

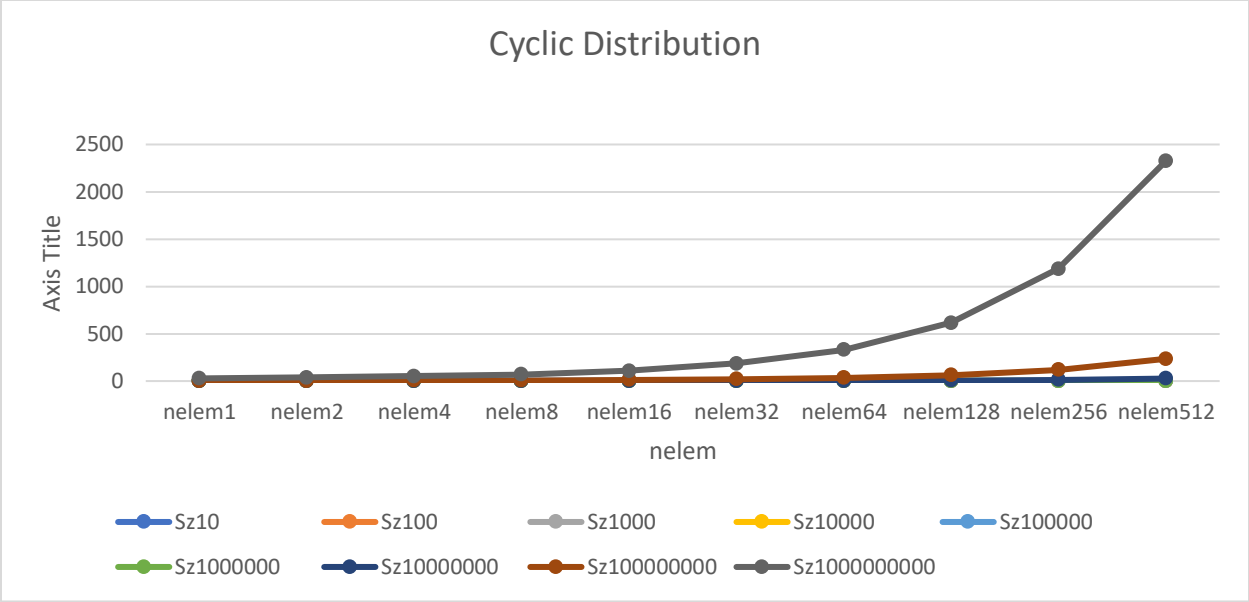
   By increasing the problem size, the execution time will be increased for both cyclic distribution and an adjacent distribution (Suppose the other parameters are fixed). When nelem is fixed, by increasing the problem size, the number of threads should be increased so each thread can do same amount of work. Accordingly we should expect a little bit increase (increase due to overhead, synchronization, condition checking etc ). For the large size, we can see huge difference (e.g. sz100000000-> sz100000000). The reason is that a lot of threads want to access the shared memory at the same time. So, congestion may happen and the timing will be increased. Although we are just considering the execution time, please pay attention that lerge vectores need to be coped form cpu to GPU through relatively slow PCIes bus and it overshadows the higher computational capability of GPU,

There are is an exception when the size is 10 and nelem is less than 16. In this case, we can see an anomaly behavior in a way the execution time is decreased by increasing the size from 10 to 100 ; I think the reason is that there is some fixed overhead for all the sizes that will be appeared in the small sizes. Besides, for the very large size we can see quite difference in timing results (e.g. for performing the last row, last column the time is 2172).




2. What is the performance impact of increasing the work per thread?

   By increasing the number of elements work by one thread(nelem), the execution time will be increased for both cyclic distribution and an adjacent distribution (The other parameters are fixed). The reason is that each thread should do more tasks (rather than distributing tasks) and accordingly the performance is decreased.

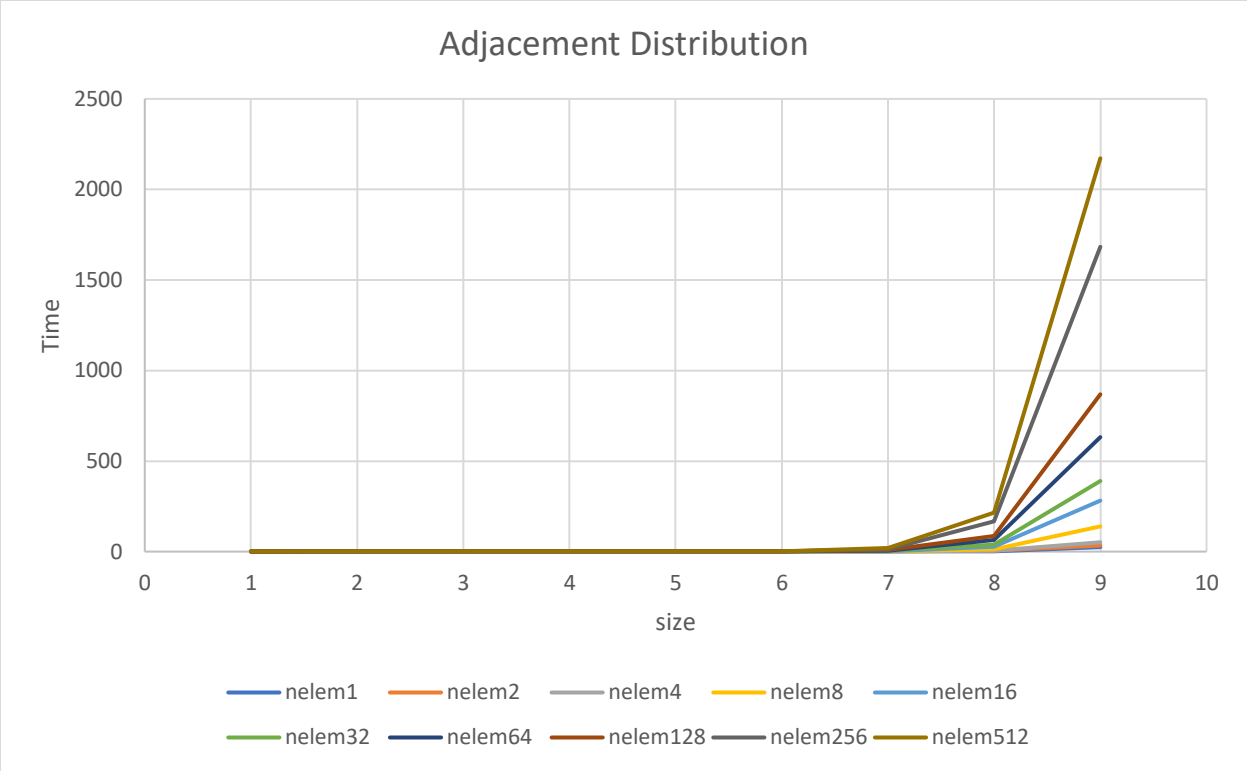


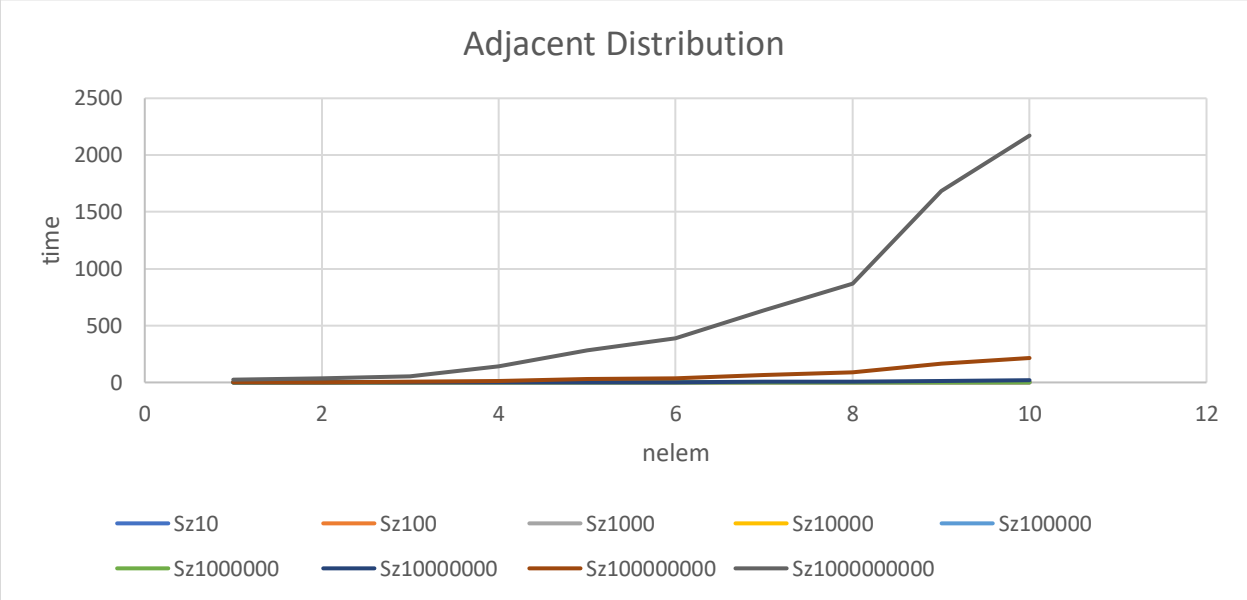

(Optional) 3. What is the performance impact of a cyclic distribution vs. each thread computing adjacent elements?

As the results show, the trend of "performance vs size" and "performance vs nelem" for both the cyclic and adjacent distribution are the same.

In general, the timing result for cyclic is better than adjacent. The reason is because of locality and the way the data stored in memory. In cyclic approach, by bringing some data, each thread can have access its own data instantly. This doesn't happen for adjacent distribution. Hence, the amount of parallelism in the cyclic distribution is more than adjacent distribution.

There are soma anomaly behavior that for some specific size and nelem, we can see that the timing of adjacent is better than cyclic.


Adjacement Distribution
Time
size
2500
2000
1500
1000
500
0
0 1 2 3 4 5 6 7 8 9 10
nelem1
nelem2
nelem4
nelem8
nelem16
nelem32
nelem64
nelem128
nelem256
nelem512


Adjacent Distribution
time
nelem
2500
2000
1500
1000
500
0
0 2 4 6 8 10 12
Sz10
Sz100
Sz1000
Sz10000
Sz100000
Sz1000000
Sz10000000
Sz100000000
Sz1000000000