**Lab Course: Distributed Data Analytics Exercise Sheet 1**

Exercise 0

| | |
|---|---|
| Processor | Intel core i5 7th Gen |
| Operating system | Windows 10 |
| Memory | 8GB |
| Storage | Local Disk C: (299 GB) |
| Graphics | NVidia GeForce 920 MX |
| Programming language | Python |

Exercise 1

Solution file: mpimatvect.py

a)

Steps:

1) The rank is identified from the mpiexec command
2) Inputsize/rank is the batch size allocated to each of the processes except the last one.
3) The last process takes the remaining data after allocation to each process before it
4) Vector addition is performed in each slave
5) Results are returned to master

Vector addition

| Core/input size | 10^7 | 10^12 | 10^15 |
|---|---|---|---|
| 1 | 0.05500719975680113 | - | - |
| 2 | 1.803858700208366 | - | - |
| 3 | 1.844593099784106 | - | - |
| 4 | 1.8467364995740354 | - | - |

Note: For inputs of size 10^12, 10^15, the RAM is not big enough to perform operations of this magnitude

```
C:\Users\admin\Anaconda3\Scripts>mpiexec -n 1 python mpimatvect.py
rank 0
Traceback (most recent call last):
  File "mpimatvect.py", line 12, in <module>
    matrix_a = np.random.random((2,n))
  File "mtrand.pyx", line 430, in numpy.random.mtrand.RandomState.random
  File "mtrand.pyx", line 421, in numpy.random.mtrand.RandomState.random_sample
  File "_common.pyx", line 256, in numpy.random._common.double_fill
MemoryError: Unable to allocate 14.6 TiB for an array with shape (2, 1000000000000) and data type f
loat64
```

```
C:\Users\admin\Anaconda3\Scripts>mpiexec -n 1 python mpimatvect.py
rank 0
[1.18311933 1.5235067  0.97891462 0.96779836 1.13252036]
Time elapsed 0.00026950007304549217

C:\Users\admin\Anaconda3\Scripts>mpiexec -n 1 python mpimatvect.py
rank 0
[1.69407575 0.93302029 1.31820689 ... 1.04990507 0.257726   1.72507803]
Time elapsed 0.05500719975680113
```



Exercise 1

Solution file – average_vector.py

b)

Steps:

1) The rank is identified from the mpiexec command
2) Inputsize/rank is the batch size allocated to each of the processes except the last one.
3) The last process takes the remaining data after allocation to each process before it
4) Average is performed in each slave
5) Results are returned to master

| Cores/Input size | 10^7 | 10^12 | 10^15 |
|---|---|---|---|
| 1 | 0.012141900137066841 | - | - |
| 2 | 1.4384162002243102 | - | - |
| 3 | 1.4540423997677863 | - | - |
| 4 | 1.4918508999980986 | - | - |

Note: For inputs of size 10^12, 10^15, the RAM is not big enough to perform operations of this magnitude

```
rank 0
1000000000000
Traceback (most recent call last):
  File "average_vector.py", line 11, in <module>
    matrix_a = np.random.random((1,n))
  File "mtrand.pyx", line 430, in numpy.random.mtrand.RandomState.random
  File "mtrand.pyx", line 421, in numpy.random.mtrand.RandomState.random_sample
  File "_common.pyx", line 256, in numpy.random._common.double_fill
MemoryError: Unable to allocate 7.28 TiB for an array with shape (1, 1000000000000) and data type f
loat64
```

```
C:\Users\admin\Anaconda3\Scripts>mpiexec -n 3 python average_vector.py
rank 1
10000000
matrix [[0.96702984 0.54723225 0.97268436 ... 0.0664702  0.11850422 0.77935834]]
result for rank 1   [1667208.27545623]
rank 2
10000000
matrix [[0.96702984 0.54723225 0.97268436 ... 0.0664702  0.11850422 0.77935834]]
result for rank 2   [1665801.67454719]
rank 0
10000000
matrix [[0.96702984 0.54723225 0.97268436 ... 0.0664702  0.11850422 0.77935834]]
result for rank 0 [1666430.26574922]
Averge of the vector [0.49994402]
Time 1.5075397002510726
```

Exercise 2: Parallel Matrix vector multiplication using MPI:

Solution file – mpiexp2.py

Steps:

1) The rank is identified from the mpiexec command
2) Inputsize/rank is the batch size allocated to each of the processes except the last one.
3) The last process takes the remaining data after allocation to each process before it
4) Matrix vector is performed in each slave
5) Results are returned to master

```
C:\Users\admin\Anaconda3\Scripts>mpiexec -n 4 python mpiexp2.py
rank 1
result for rank 1   [[20.09526552]
 [24.24200587]
 [23.01842539]
 [27.43746367]
 [24.14116931]
 [24.76241191]
 [23.77776946]
 [25.11527172]
 [20.17607841]
 [24.30473943]
 [23.08857496]
 [25.75437113]
 [26.83043683]
 [20.4421288 ]
 [23.71749182]
 [22.88704724]
 [25.79793987]
 [23.67841687]
 [24.46944637]
 [24.86104392]
 [22.78831934]
 [24.56576931]
 [24.0089882 ]
 [21.92124151]
 [24.94284204]]
time 0.0014758999459445477
rank 2
result for rank 2   [[25.87430532]
 [22.05691752]
 [20.8303714 ]
 [23.47979888]
 [25.50354447]
```

| Core/inputsize | 10^2 | 10^3 | 10^4 |
|---|---|---|---|
| 1 | 0.0013326001353561878 | 0.011930100154131651 | 0.03754719998687506 |
| 2 | 0.003135299775749445 | 0.03129839990288019 | 1.193994800094515 |
| 3 | 0.0039114998653531075 | 0.04732789983972907 | 1.2676194002851844 |
| 4 | 0.015840400010347366 | 0.047501299995929 | 4.579885399900377 |

Exercise 3: Parallel Matrix operation using MPI:

Solution file – matrixmul.py

Steps:

1) The rank is identified from the mpiexec command
2) The matrix is broadcasted to all the processes
3) Inputsize/rank is the batch size allocated to each of the processes except the last one.
4) The last process takes the remaining data after allocation to each process before it
5) Matrix multiplication is performed in each slave
6) Result is given to the master process by allgather() method

```
Process 0 has been given this data: [[1.83045093 1.93212788 1.51311516 ... 1.0756
02534416]
 [1.33194721 1.89642837 1.05474197 ... 1.44569877 1.12979147 1.52953512]
 [1.27664395 1.79011491 1.82366065 ... 1.1717437  1.6234627  1.618894  ]
 ...
 [1.18351425 1.13784718 1.27693755 ... 1.31224477 1.51791726 1.07580222]
 [1.72611064 1.08378065 1.17724872 ... 1.87944075 1.11013658 1.20362394]
 [1.88091042 1.86874847 1.16528951 ... 1.3513995  1.0708804  1.09866774]]
Matrix multiplication: [[[70.99884088 70.91429066 73.22266159 ... 77.44144815 81.
   74.33852777]
  [73.56007732 71.92021902 73.65749405 ... 81.22422025 83.00301577
   75.97964383]
  [69.56673417 68.42940088 72.4513293  ... 75.79099071 78.47615037
   72.6539708 ]
  ...
  [70.24080852 69.03523395 72.15879805 ... 77.93307432 78.15465742
   71.42237834]
  [71.81684872 69.5068818  71.71476165 ... 78.14601128 78.07614546
   72.99762927]
  [73.02089048 73.03747282 74.01251403 ... 80.61253527 81.63457011
   75.79343431]]]
time 1.2127240002155304
```

| Core/input size | 10^2 | 10^3 | 10^4 |
|---|---|---|---|
| 1 | 1. 2127240002155304 | Does not run | Does not run |
| 2 | 0.7399005000479519 | Does not run | Does not run |
| 3 | 0.5918200998567045 | Does not run | Does not run |
| 4 | 0.5483559998683631 | Does not run | Does not run |