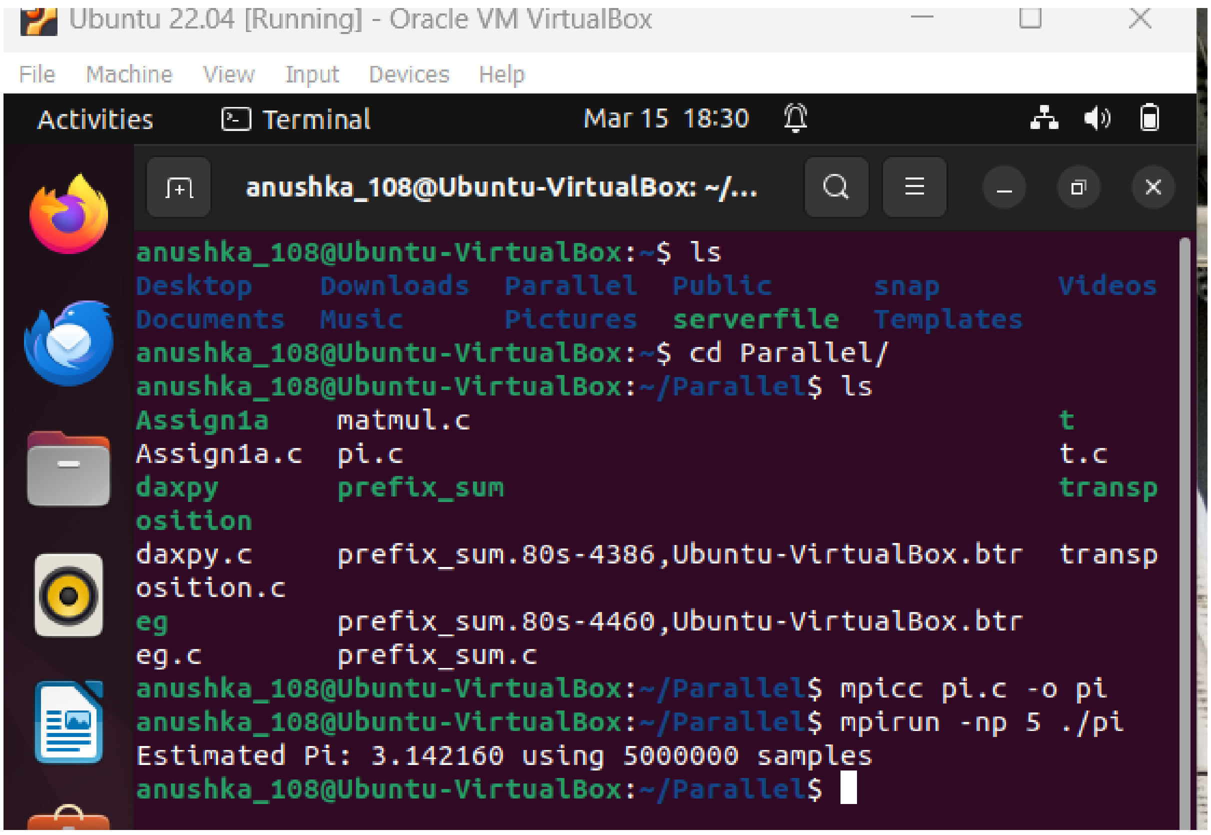
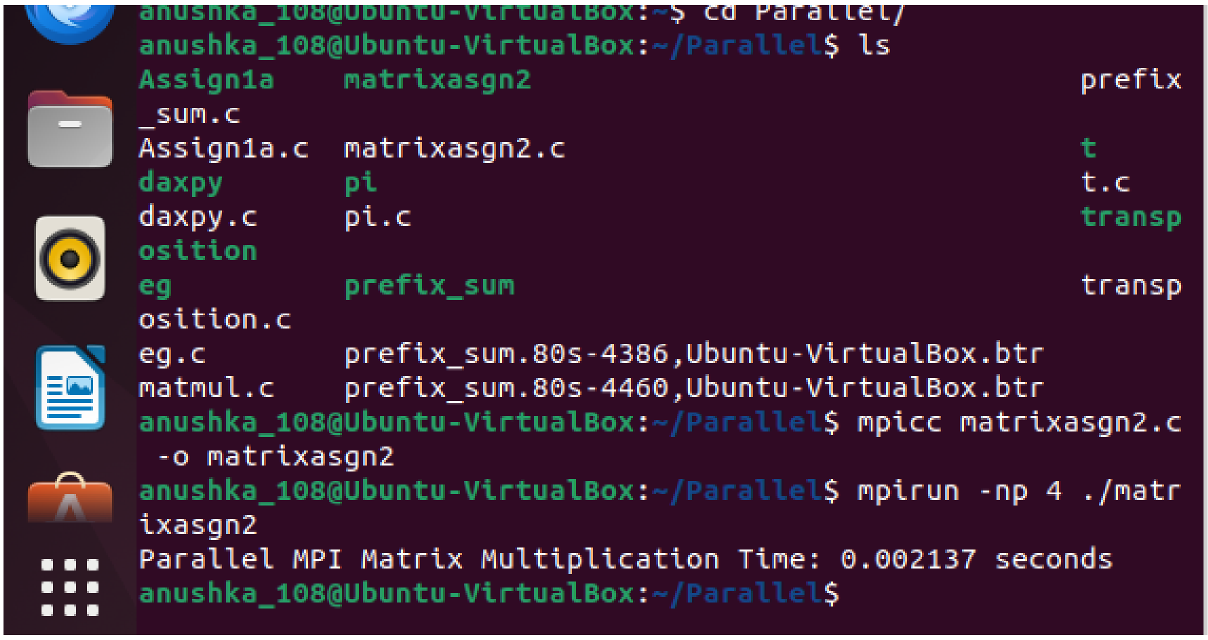
Assignment 2: MPI

1. Estimate the value of Pi using the Monte Carlo method and demonstrate basic MPI functions.


Ubuntu 22.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Mar 15 18:30
anushka_108@Ubuntu-VirtualBox: ~/...

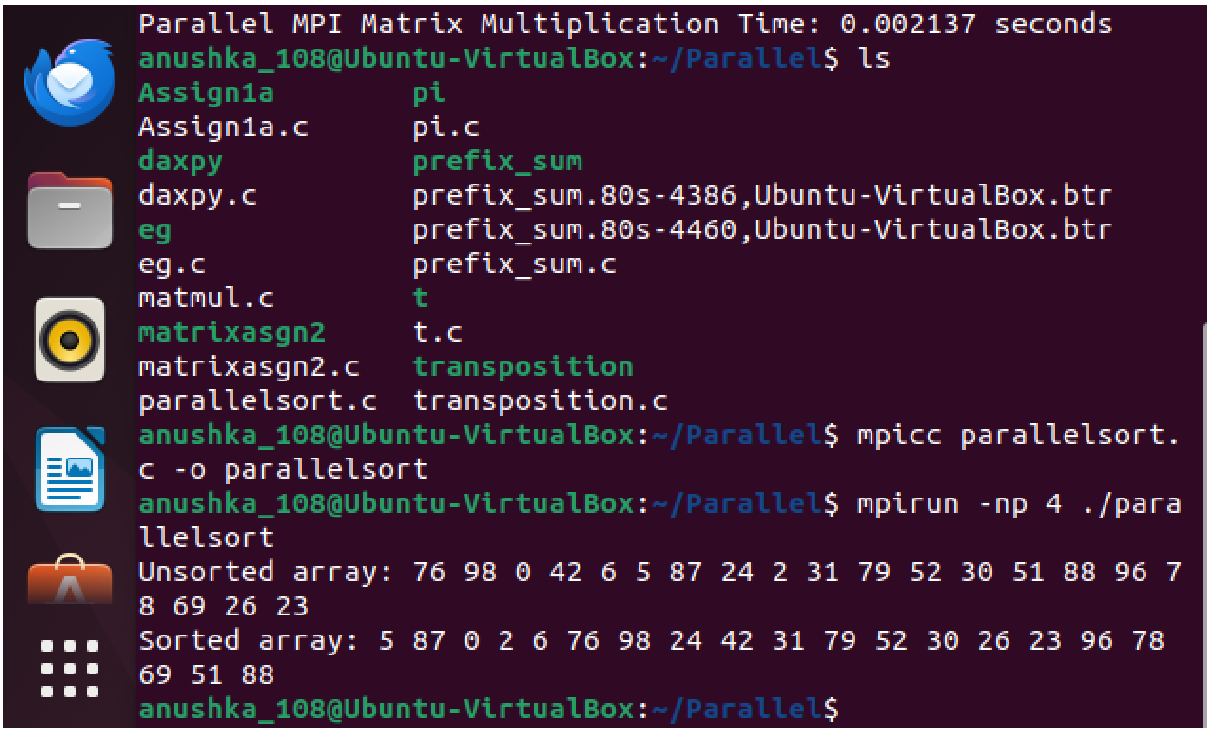
```
anushka_108@Ubuntu-VirtualBox:~$ ls
Desktop     Downloads   Parallel   Public      snap       Videos
Documents   Music       Pictures   serverfile  Templates
anushka_108@Ubuntu-VirtualBox:~$ cd Parallel/
anushka_108@Ubuntu-VirtualBox:~/Parallel$ ls
Assign1a     matmul.c                                t
Assign1a.c   pi.c                                    t.c
daxpy        prefix_sum                              transp
osition
daxpy.c      prefix_sum.80s-4386,Ubuntu-VirtualBox.btr  transp
osition.c
eg           prefix_sum.80s-4460,Ubuntu-VirtualBox.btr
eg.c         prefix_sum.c
anushka_108@Ubuntu-VirtualBox:~/Parallel$ mpicc pi.c -o pi
anushka_108@Ubuntu-VirtualBox:~/Parallel$ mpirun -np 5 ./pi
Estimated Pi: 3.142160 using 5000000 samples
anushka_108@Ubuntu-VirtualBox:~/Parallel$ 
```

2. Matrix Multiplication using MPI. Consider 70X70 matrix compute using serial sequential order and compare the time. For computing the time use double start_time, run_time; run_time = omp_get_wtime() - start_time; Time in seconds



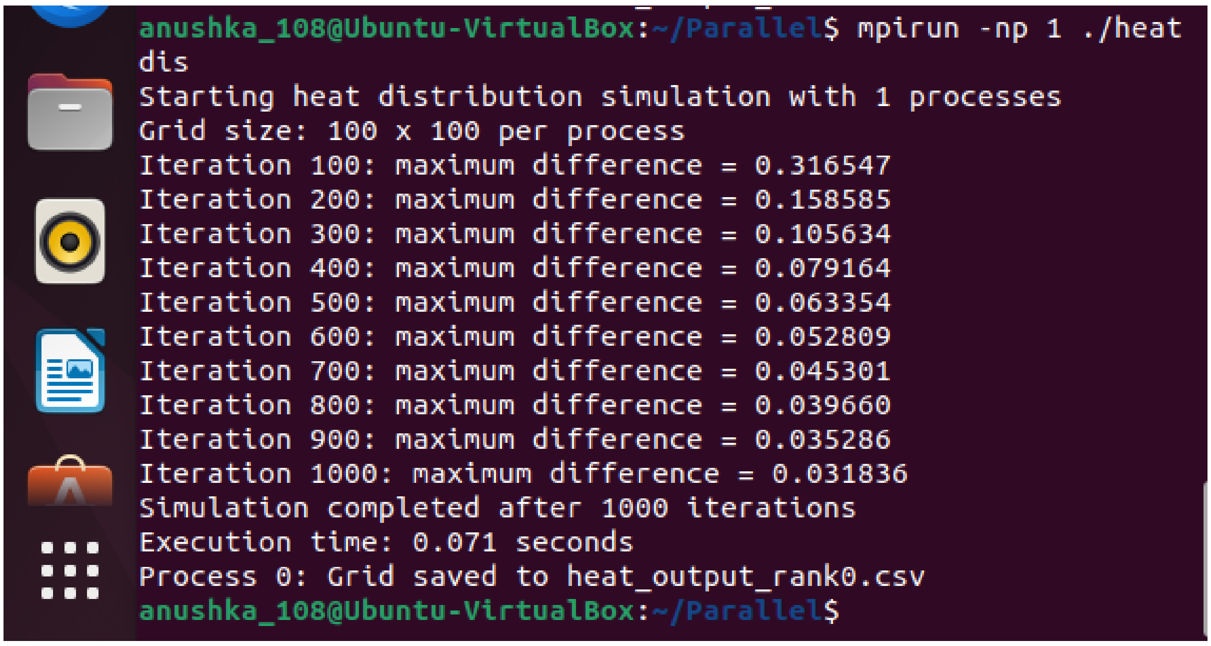
```
anushka_108@Ubuntu-VirtualBox:~$ cd Parallel/
anushka_108@Ubuntu-VirtualBox:~/Parallel$ ls
Assign1a    matrixasgn2                                  prefix
_sum.c
Assign1a.c  matrixasgn2.c                                t
daxpy       pi                                           t.c
daxpy.c     pi.c                                         transp
osition
eg          prefix_sum                                   transp
osition.c
eg.c        prefix_sum.80s-4386,Ubuntu-VirtualBox.btr
matmul.c    prefix_sum.80s-4460,Ubuntu-VirtualBox.btr
anushka_108@Ubuntu-VirtualBox:~/Parallel$ mpicc matrixasgn2.c
 -o matrixasgn2
anushka_108@Ubuntu-VirtualBox:~/Parallel$ mpirun -np 4 ./matr
ixasgn2
Parallel MPI Matrix Multiplication Time: 0.002137 seconds
anushka_108@Ubuntu-VirtualBox:~/Parallel$
```

3. Parallel Sorting using MPI (Odd-Even Sort)



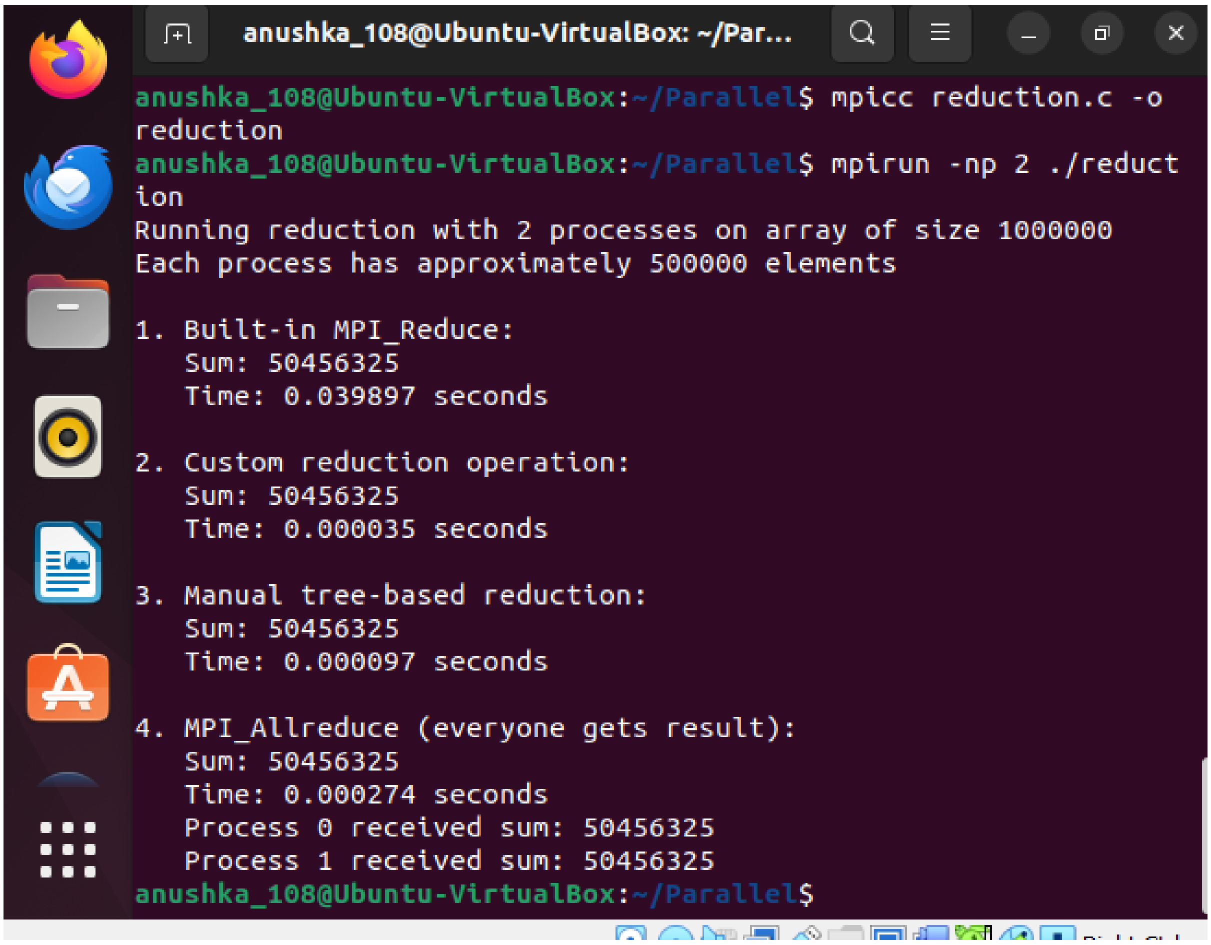
```
Parallel MPI Matrix Multiplication Time: 0.002137 seconds
anushka_108@Ubuntu-VirtualBox:~/Parallel$ ls
Assign1a        pi
Assign1a.c      pi.c
daxpy           prefix_sum
daxpy.c         prefix_sum.80s-4386,Ubuntu-VirtualBox.btr
eg              prefix_sum.80s-4460,Ubuntu-VirtualBox.btr
eg.c            prefix_sum.c
matmul.c        t
matrixasgn2     t.c
matrixasgn2.c   transposition
parallelsort.c  transposition.c
anushka_108@Ubuntu-VirtualBox:~/Parallel$ mpicc parallelsort.
c -o parallelsort
anushka_108@Ubuntu-VirtualBox:~/Parallel$ mpirun -np 4 ./para
llelsort
Unsorted array: 76 98 0 42 6 5 87 24 2 31 79 52 30 51 88 96 7
8 69 26 23
Sorted array: 5 87 0 2 6 76 98 24 42 31 79 52 30 26 23 96 78
69 51 88
anushka_108@Ubuntu-VirtualBox:~/Parallel$
```

4. Heat Distribution Simulation using MPI



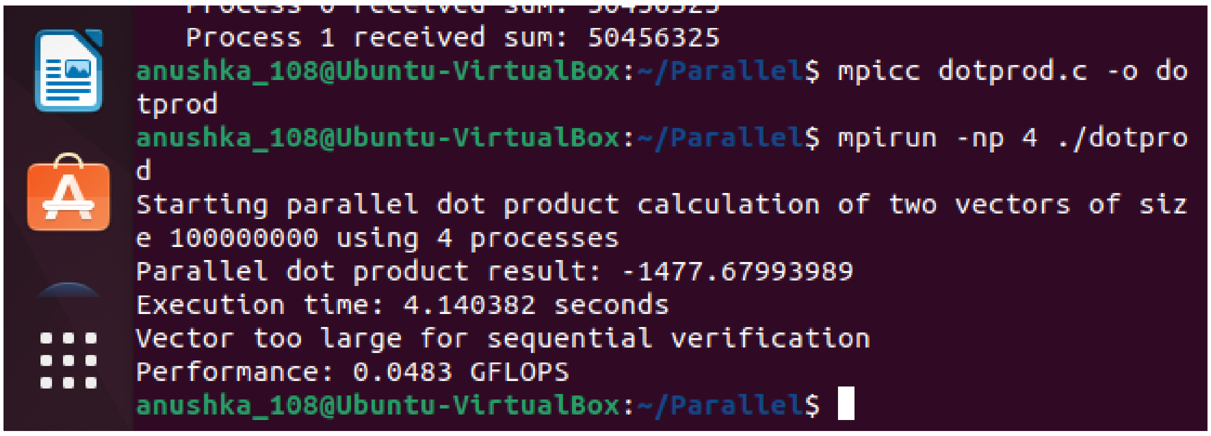
```
anushka_108@Ubuntu-VirtualBox:~/Parallel$ mpirun -np 1 ./heat
dis
Starting heat distribution simulation with 1 processes
Grid size: 100 x 100 per process
Iteration 100: maximum difference = 0.316547
Iteration 200: maximum difference = 0.158585
Iteration 300: maximum difference = 0.105634
Iteration 400: maximum difference = 0.079164
Iteration 500: maximum difference = 0.063354
Iteration 600: maximum difference = 0.052809
Iteration 700: maximum difference = 0.045301
Iteration 800: maximum difference = 0.039660
Iteration 900: maximum difference = 0.035286
Iteration 1000: maximum difference = 0.031836
Simulation completed after 1000 iterations
Execution time: 0.071 seconds
Process 0: Grid saved to heat_output_rank0.csv
anushka_108@Ubuntu-VirtualBox:~/Parallel$
```

5. Parallel Reduction using MPI



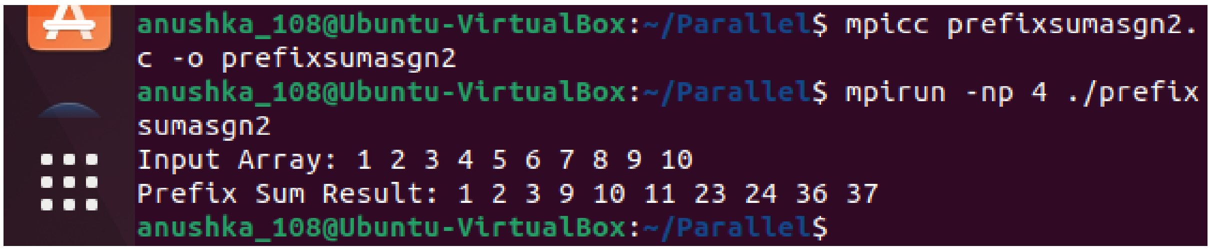
```
anushka_108@Ubuntu-VirtualBox:~/Parallel$ mpicc reduction.c -o
reduction
anushka_108@Ubuntu-VirtualBox:~/Parallel$ mpirun -np 2 ./reduct
ion
Running reduction with 2 processes on array of size 1000000
Each process has approximately 500000 elements

1. Built-in MPI_Reduce:
   Sum: 50456325
   Time: 0.039897 seconds

2. Custom reduction operation:
   Sum: 50456325
   Time: 0.000035 seconds

3. Manual tree-based reduction:
   Sum: 50456325
   Time: 0.000097 seconds

4. MPI_Allreduce (everyone gets result):
   Sum: 50456325
   Time: 0.000274 seconds
   Process 0 received sum: 50456325
   Process 1 received sum: 50456325
anushka_108@Ubuntu-VirtualBox:~/Parallel$
```

6. Parallel Dot Product using MPI



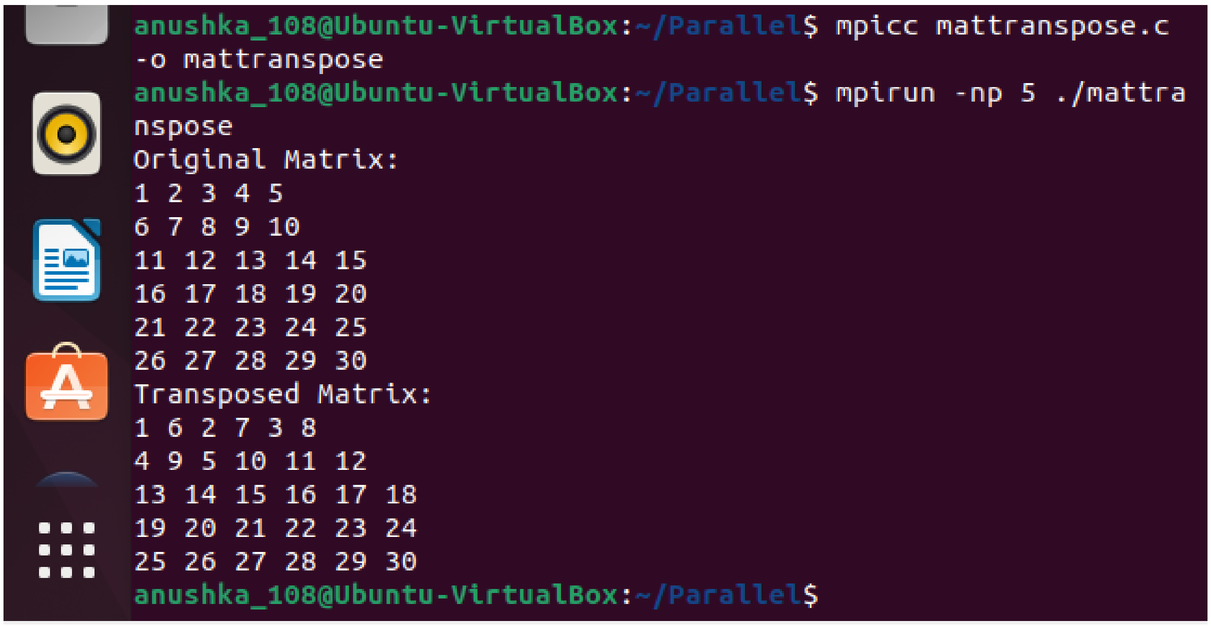
```
   Process 1 received sum: 50456325
anushka_108@Ubuntu-VirtualBox:~/Parallel$ mpicc dotprod.c -o do
tprod
anushka_108@Ubuntu-VirtualBox:~/Parallel$ mpirun -np 4 ./dotpro
d
Starting parallel dot product calculation of two vectors of siz
e 100000000 using 4 processes
Parallel dot product result: -1477.67993989
Execution time: 4.140382 seconds
Vector too large for sequential verification
Performance: 0.0483 GFLOPS
anushka_108@Ubuntu-VirtualBox:~/Parallel$
```

7. Parallel Prefix Sum (Scan) using MPI



```
anushka_108@Ubuntu-VirtualBox:~/Parallel$ mpicc prefixsumasgn2.
c -o prefixsumasgn2
anushka_108@Ubuntu-VirtualBox:~/Parallel$ mpirun -np 4 ./prefix
sumasgn2
Input Array: 1 2 3 4 5 6 7 8 9 10
Prefix Sum Result: 1 2 3 9 10 11 23 24 36 37
anushka_108@Ubuntu-VirtualBox:~/Parallel$
```

8. Parallel Matrix Transposition using MPI



```
anushka_108@Ubuntu-VirtualBox:~/Parallel$ mpicc mattranspose.c
-o mattranspose
anushka_108@Ubuntu-VirtualBox:~/Parallel$ mpirun -np 5 ./mattra
nspose
Original Matrix:
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25
26 27 28 29 30
Transposed Matrix:
1 6 2 7 3 8
4 9 5 10 11 12
13 14 15 16 17 18
19 20 21 22 23 24
25 26 27 28 29 30
anushka_108@Ubuntu-VirtualBox:~/Parallel$
```