Take an integer array of 10*4, 105, 10*6 elements.

try accessing the array elements using np 1,2,4 and tell me the run total run_time for this accessing elements.

Experimental Setup

- Language: C
- Parallel Model: MPI (Distributed Memory Model)
- Compiler: mpicc
- Execution Command:

mpirun -np <p> ./program

Where <p> = 1, 2, 4

Work Division Strategy

Each process calculates:

count = N / nprocs

start = rank * count

stop = start + count

Each process accesses elements from start to stop-1.

C Code with clock() function used to calculate run_time

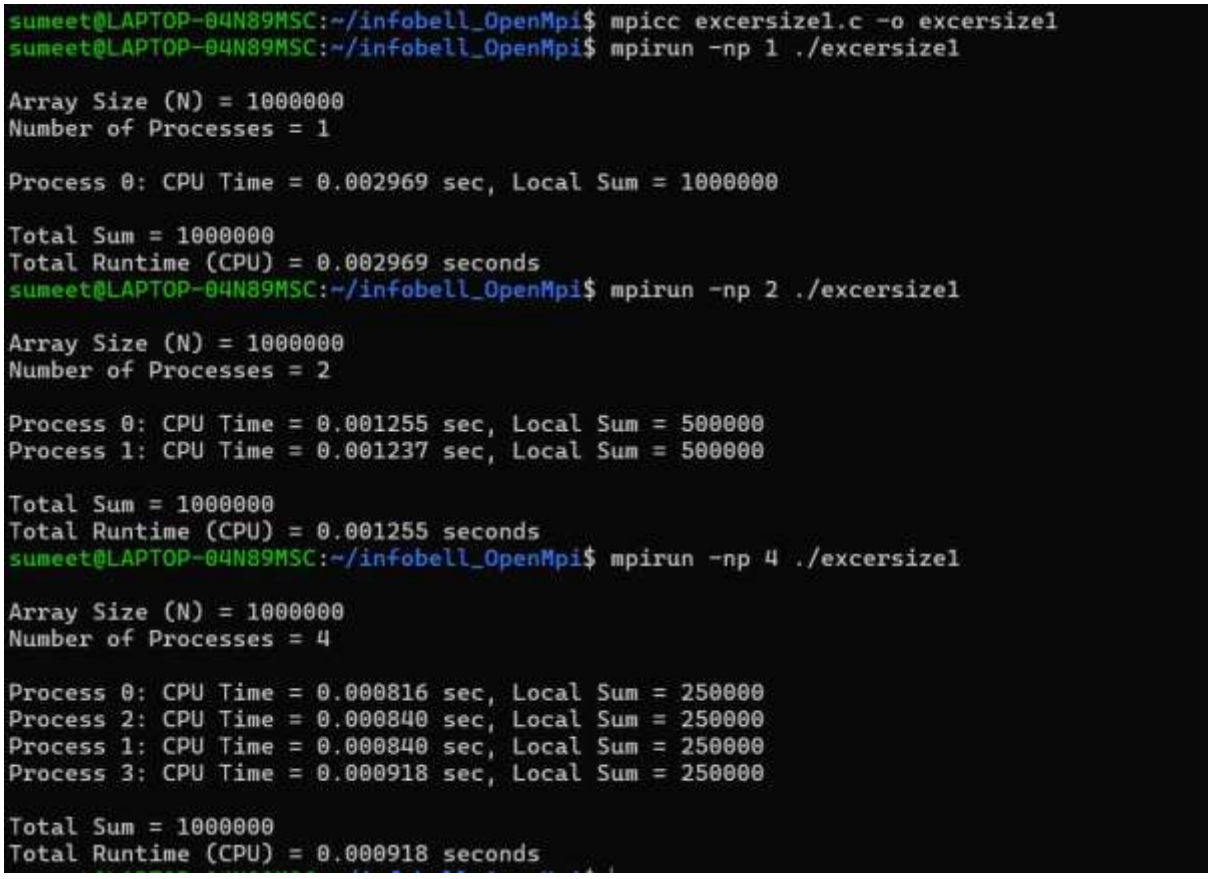
```
include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define N 10000 // this changed accordingly

int main(int argc, char **argv)
{
   int rank, nprocs;
   int *arr;
   long long loc_sum = 0, t_sum = 0;
   clock_t start_clock, end_clock;

   MPI_Init(&argc, &argv);

   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   MPI_Comm_size(MPI_COMM_WORLD, &nprocs);

   if(rank == 0)
   {
      printf("\nArray Size (N) = %d\n", N);
      printf("Number of Processes = %d\n\n", nprocs);
   }

   arr = (int*) malloc(N * sizeof(int));

   for(int i = 0; i < N; i++)
      arr[i] = 1;
```

```
int count = N / nprocs;
int start = rank * count;
int stop  = start + count;

start_clock = clock();

for(int i = start; i < stop; i++)
   loc_sum += arr[i];

end_clock = clock();

double local_time =
   (double)(end_clock - start_clock) / CLOCKS_PER_SEC;

printf("Process %d: CPU Time = %f sec, Local Sum = %lld\n",
      rank, local_time, loc_sum);

double total_time;

MPI_Reduce(&loc_sum, &t_sum, 1,
         MPI_LONG_LONG, MPI_SUM, 0, MPI_COMM_WORLD);

MPI_Reduce(&local_time, &total_time, 1,
         MPI_DOUBLE, MPI_MAX, 0, MPI_COMM_WORLD);

if(rank == 0)
{
   printf("\nTotal Sum = %lld\n", t_sum);
   printf("Total Runtime (CPU) = %f seconds\n", total_time);
```

```
    }

    free(arr);
    MPI_Finalize();
    return 0;
}
```

OUTPUTS:

For N = 100000



For N = 10000

```
sumeet@LAPTOP-04N89MSC:~/infobell_OpenMpi$ mpicc excersize1.c -o excersize1
sumeet@LAPTOP-04N89MSC:~/infobell_OpenMpi$ mpirun -np 1 ./excersize1

Array Size (N) = 100000
Number of Processes = 1

Process 0: CPU Time = 0.000260 sec, Local Sum = 100000

Total Sum = 100000
Total Runtime (CPU) = 0.000260 seconds
sumeet@LAPTOP-04N89MSC:~/infobell_OpenMpi$ mpirun -np 2 ./excersize1

Array Size (N) = 100000
Number of Processes = 2

Process 0: CPU Time = 0.000124 sec, Local Sum = 50000
Process 1: CPU Time = 0.000148 sec, Local Sum = 50000

Total Sum = 100000
Total Runtime (CPU) = 0.000148 seconds
sumeet@LAPTOP-04N89MSC:~/infobell_OpenMpi$ mpirun -np 4 ./excersize1

Array Size (N) = 100000
Number of Processes = 4

Process 0: CPU Time = 0.000075 sec, Local Sum = 25000
Process 1: CPU Time = 0.000115 sec, Local Sum = 25000
Process 3: CPU Time = 0.000064 sec, Local Sum = 25000
Process 2: CPU Time = 0.000116 sec, Local Sum = 25000

Total Sum = 100000
Total Runtime (CPU) = 0.000116 seconds
```

For N = 10000

```
sumeet@LAPTOP-04N89MSC:~/infobell_OpenMpi$ mpicc excersize1.c -o excersize1
sumeet@LAPTOP-04N89MSC:~/infobell_OpenMpi$ mpirun -np 1 ./excersize1

Array Size (N) = 10000
Number of Processes = 1

Process 0: CPU Time = 0.000034 sec, Local Sum = 10000

Total Sum = 10000
Total Runtime (CPU) = 0.000034 seconds
sumeet@LAPTOP-04N89MSC:~/infobell_OpenMpi$ mpirun -np 2 ./excersize1

Array Size (N) = 10000
Number of Processes = 2

Process 1: CPU Time = 0.000015 sec, Local Sum = 5000
Process 0: CPU Time = 0.000015 sec, Local Sum = 5000

Total Sum = 10000
Total Runtime (CPU) = 0.000015 seconds
sumeet@LAPTOP-04N89MSC:~/infobell_OpenMpi$ mpirun -np 4 ./excersize1

Array Size (N) = 10000
Number of Processes = 4

Process 0: CPU Time = 0.000011 sec, Local Sum = 2500

Total Sum = 10000
Total Runtime (CPU) = 0.000015 seconds
Process 3: CPU Time = 0.000015 sec, Local Sum = 2500
Process 1: CPU Time = 0.000014 sec, Local Sum = 2500
Process 2: CPU Time = 0.000014 sec, Local Sum = 2500
```

C Code with MPI_Wtime() function used to calculate run_time

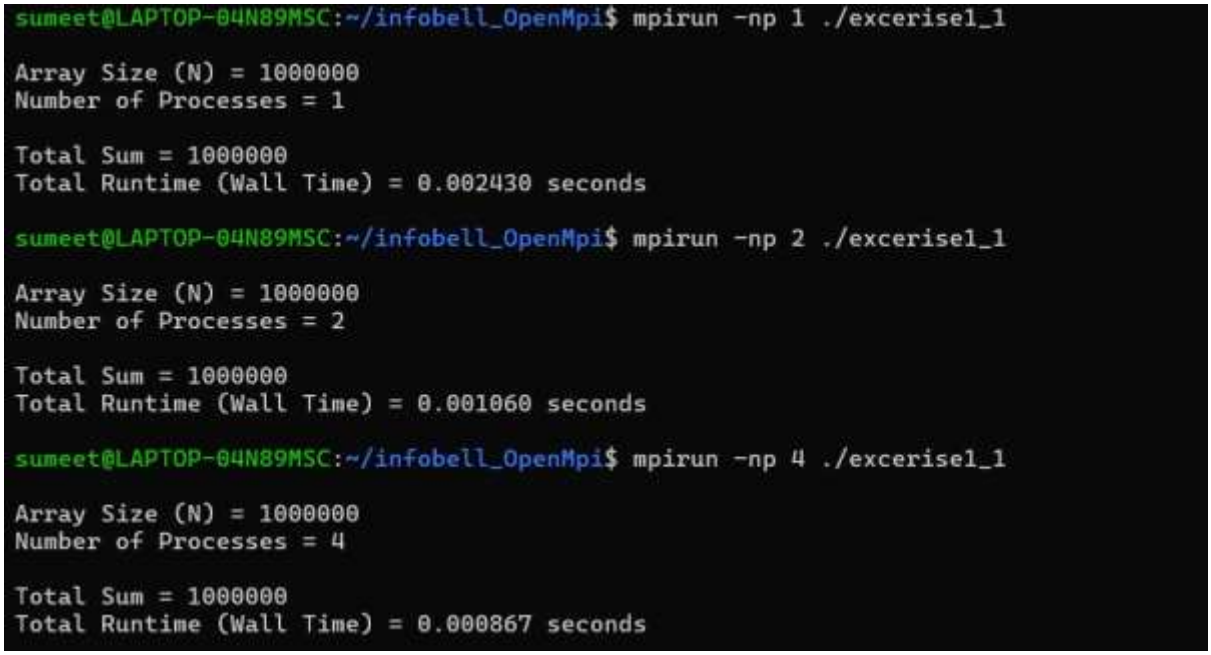
```
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>
#define N 1000000   // change to 10000, 100000, 1000000

int main(int argc, char **argv)
{
   int rank, nprocs;
   int *arr;
   long long local_sum = 0, total_sum = 0;

   MPI_Init(&argc, &argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   MPI_Comm_size(MPI_COMM_WORLD, &nprocs);

   if(rank == 0)
   {
      printf("\nArray Size (N) = %d\n", N);
      printf("Number of Processes = %d\n\n", nprocs);
   }

   arr = (int*) malloc(N * sizeof(int));

   for(int i = 0; i < N; i++)
      arr[i] = 1;
```

```
int count = N / nprocs;
int start = rank * count;
int stop  = start + count;

MPI_Barrier(MPI_COMM_WORLD);   // synchronize before timing

double start_time = MPI_Wtime();

for(int i = start; i < stop; i++)
   local_sum += arr[i];

double end_time = MPI_Wtime();

double local_time = end_time - start_time;
double total_time;

// Combine total sum
MPI_Reduce(&local_sum, &total_sum, 1,
        MPI_LONG_LONG, MPI_SUM, 0, MPI_COMM_WORLD);

// Combine runtime (take maximum)
MPI_Reduce(&local_time, &total_time, 1,
        MPI_DOUBLE, MPI_MAX, 0, MPI_COMM_WORLD);

if(rank == 0)
{
   printf("Total Sum = %lld\n", total_sum);
```

```
        printf("Total Runtime (Wall Time) = %f seconds\n\n", total_time);
    }

    free(arr);
    MPI_Finalize();
    return 0;
}
```

OUTPUTS:

For N = 1000000



For N = 100000

```
sumeet@LAPTOP-04N89MSC:~/infobell_OpenMpi$ mpirun -np 1 ./excerise1_1

Array Size (N) = 100000
Number of Processes = 1

Total Sum = 100000
Total Runtime (Wall Time) = 0.000265 seconds

sumeet@LAPTOP-04N89MSC:~/infobell_OpenMpi$ mpirun -np 2 ./excerise1_1

Array Size (N) = 100000
Number of Processes = 2

Total Sum = 100000
Total Runtime (Wall Time) = 0.000091 seconds

sumeet@LAPTOP-04N89MSC:~/infobell_OpenMpi$ mpirun -np 4 ./excerise1_1

Array Size (N) = 100000
Number of Processes = 4

Total Sum = 100000
Total Runtime (Wall Time) = 0.000081 seconds
```

For N = 10000

```
sumeet@LAPTOP-04N89MSC:~/infobell_OpenMpi$ mpirun -np 1 ./excerise1_1

Array Size (N) = 10000
Number of Processes = 1

Total Sum = 10000
Total Runtime (Wall Time) = 0.000018 seconds

sumeet@LAPTOP-04N89MSC:~/infobell_OpenMpi$ mpirun -np 2 ./excerise1_1

Array Size (N) = 10000
Number of Processes = 2

Total Sum = 10000
Total Runtime (Wall Time) = 0.000014 seconds

sumeet@LAPTOP-04N89MSC:~/infobell_OpenMpi$ mpirun -np 4 ./excerise1_1

Array Size (N) = 10000
Number of Processes = 4

Total Sum = 10000
Total Runtime (Wall Time) = 0.000006 seconds
```