HW4 보고서

2022-25448
박기범

1. 몬테 카를로

```
  int local_num_point = num_points / mpi_world_size;
  int start = mpi_rank * local_num_point;
  int end = start + local_num_point;
```

연산량을 포인트의 수로 나눕니다.
xs 에는 연산할 만큼의 포인트 갯수가 존재하고
각 포인트를 균등하게 분배해야하므로
이를 시작점과 끝점만 적당히 나누어

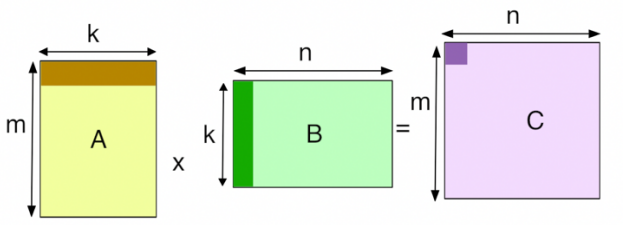
```
#pragma omp parallel for num_threads(threads_per_process) reduction(+:local_count)
  for (int i = start; i < end; i++) {
    double x = xs[i];
    double y = ys[i];

    if (x * x + y * y <= 1) {
      local_count++;
    }

  }
```

local count에 더해줍니다.

```
MPI_Reduce(&local_count, &global_count, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);
```

이를 합쳐주면 글로벌 카운트에 업데이트 됩니다.

2. matmul



1)병렬화 방식은 hw3와 거의 같습니다.
다만 A를 sub_A로 나누는데 이때 sub_A의 하나의 매트릭스가 full row를 갖게하여 sub_C가 독립적으로 일부분이 나올수 있게 쪼개었습니다.
그렇게 연산되 sub_C를

```
MPI_Gatherv(sub_C, (end_row - start_row) * N, MPI_FLOAT, C, sendcounts, displs,
MPI_FLOAT, 0, MPI_COMM_WORLD);
```

로 gather 해줍니다.

2)
위에서 말하였듯이 메모리 스트라이드를 없애기 위하여 문제를 쪼갤때에 각각의 서브매트릭스가 하나의 row를 가질수 있게 합니다.
네개의 프로세서가 같은 성능을 가지고있다는 전제하에 같은 크기만큼만 분배합니다.

3)

```
start_row = mpi_rank * rows_per_process + mpi_rank;
end_row = start_row + rows_per_process + 1;
```

매트릭스를 랭크를 반영하여 쪼개

```
float *sub_A = A + start_row * K;
float *sub_C = C + start_row * N;
```

포인터를 이용하여 a와 C를 쪼갭니다.

```
   MPI_Scatterv(A, sendcounts, displs, MPI_FLOAT, sub_A, (end_row - start_row) *
K, MPI_FLOAT, 0, MPI_COMM_WORLD);
   MPI_Bcast(B, K * N, MPI_FLOAT, 0, MPI_COMM_WORLD);
   matmul_multithread(sub_A, B, sub_C, end_row - start_row, N, K, num_threads);

   for (int i = 0; i < mpi_size; ++i) {
      if (i < remaining_rows) {
         sendcounts[i] = (rows_per_process + 1) * N;
         displs[i] = (i * rows_per_process + i) * N;
      } else {
         sendcounts[i] = rows_per_process * N;
         displs[i] = (i * rows_per_process + remaining_rows) * N;
      }
   }

   MPI_Gatherv(sub_C, (end_row - start_row) * N, MPI_FLOAT, C, sendcounts, displs,
MPI_FLOAT, 0, MPI_COMM_WORLD);
```

위에서 볼 수 있듯이 우선 end_row 와 start_row의 차를 이용하여 크기를 반영하여 scatter 합니다.

4)
Remainder가 생기도록 2의 지수승이 아닐 경우에는 지수를 처리하기 위한 리메인더가 복잡해지고 그런 문제를 넘어서 일을 균등하게 분배하는것이 어려워 질 수 있습니다. 더 나아가서 이를 반영한 적절한 워크로드 분배가 어려울수 있습니다.