ASSIGNMENT 1

Group 1

- Adam Taoufiq
- Lorena Goldoni
- Gabriel Panini

Kernel_2mm

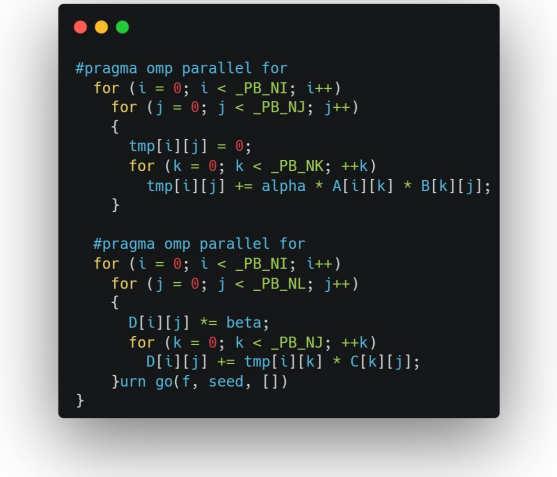
```c
static void kernel_2mm(int ni, int nj, int nk, int nl,
                       DATA_TYPE alpha,
                       DATA_TYPE beta,
                       DATA_TYPE POLYBENCH_2D(tmp, NI, NJ, ni, nj),
                       DATA_TYPE POLYBENCH_2D(A, NI, NK, ni, nk),
                       DATA_TYPE POLYBENCH_2D(B, NK, NJ, nk, nj),
                       DATA_TYPE POLYBENCH_2D(C, NL, NJ, nl, nj),
                       DATA_TYPE POLYBENCH_2D(D, NI, NL, ni, nl))
{
  int i, j, k;

  /* D := alpha*A*B*C + beta*D */
  {
    for (i = 0; i < _PB_NI; i++)
      for (j = 0; j < _PB_NJ; j++)
      {
        tmp[i][j] = 0;
        for (k = 0; k < _PB_NK; ++k)
          tmp[i][j] += alpha * A[i][k] * B[k][j];
      }
    for (i = 0; i < _PB_NI; i++)
      for (j = 0; j < _PB_NL; j++)
      {
        D[i][j] *= beta;
        for (k = 0; k < _PB_NJ; ++k)
          D[i][j] += tmp[i][k] * C[k][j];
      }
  }
}
```

Sum between the product of 3 2d matrixes and a scalar and a product between a 2d matrix and a scalar

D = ***alpha * A * B * C + beta * D***

Since the code is split into 2 phases and uses for loops, we have room for improvement using parallelization

Improvements



```
#pragma omp parallel for
  for (i = 0; i < _PB_NI; i++)
    for (j = 0; j < _PB_NJ; j++)
    {
      tmp[i][j] = 0;
      for (k = 0; k < _PB_NK; ++k)
        tmp[i][j] += alpha * A[i][k] * B[k][j];
    }

  #pragma omp parallel for
  for (i = 0; i < _PB_NI; i++)
    for (j = 0; j < _PB_NL; j++)
    {
      D[i][j] *= beta;
      for (k = 0; k < _PB_NJ; ++k)
        D[i][j] += tmp[i][k] * C[k][j];
    }urn go(f, seed, [])
}
```

2mm-omp-1

2 parallel zones for the 2 for loops

```
#pragma omp parallel
  {
    #pragma omp for
    for (i = 0; i < _PB_NI; i++)
      for (j = 0; j < _PB_NJ; j++)
      {
        tmp[i][j] = 0;
        for (k = 0; k < _PB_NK; ++k)
          tmp[i][j] += alpha * A[i][k] * B[k][j];
      }

    #pragma omp for
    for (i = 0; i < _PB_NI; i++)
      for (j = 0; j < _PB_NL; j++)
      {
        D[i][j] *= beta;
        for (k = 0; k < _PB_NJ; ++k)
          D[i][j] += tmp[i][k] * C[k][j];
      }

  }
```

2mm-omp-2

1 common parallel zone for the 2 for loops

Improvements

```
#pragma omp target parallel for map(to:i, k, j, _PB_NI, _PB_NJ, _PB_NK, alpha, A[0:_PB_NI][0:_PB_NK],
B[0:_PB_NK][0:_PB_NJ]) map(tofrom:tmp[0:_PB_NI][0:_PB_NJ])
    for (i = 0; i < _PB_NI; i++)
      for (j = 0; j < _PB_NJ; j++)
      {
        tmp[i][j] = 0;
        for (k = 0; k < _PB_NK; ++k)
          tmp[i][j] += alpha * A[i][k] * B[k][j];
      }
    #pragma omp target parallel for map(to:i, k, j, _PB_NI, _PB_NJ, _PB_NL, beta, tmp[0:_PB_NI]
[0:_PB_NJ], C[0:_PB_NJ][0:_PB_NL]) map(tofrom:D[0:_PB_NI][0:_PB_NL])
    for (i = 0; i < _PB_NI; i++)
      for (j = 0; j < _PB_NL; j++)
      {
        D[i][j] *= beta;
        for (k = 0; k < _PB_NJ; ++k)
          D[i][j] += tmp[i][k] * C[k][j];
      }
```

2mm-omp-3

*Use of offloading on gpu device with the use of the **target parallel** directive*

```
#pragma omp target teams num_teams(8) map(to:A[0:_PB_NI][0:_PB_NK], B[0:_PB_NK][0:_PB_NJ])
map(tofrom:tmp[0:_PB_NI][0:_PB_NJ])
    #pragma omp parallel for num_threads(16) collapse(2)
    for (i = 0; i < _PB_NI; i++)
      for (j = 0; j < _PB_NJ; j++)
      {
        tmp[i][j] = 0;
        for (k = 0; k < _PB_NK; ++k)
          tmp[i][j] += alpha * A[i][k] * B[k][j];
      }
    #pragma omp target teams num_teams(8) map(to:tmp[0:_PB_NI][0:_PB_NJ], C[0:_PB_NJ][0:_PB_NL])
map(tofrom:D[0:_PB_NI][0:_PB_NL])
    #pragma omp parallel for num_threads(16) collapse(2)
    for (i = 0; i < _PB_NI; i++)
      for (j = 0; j < _PB_NL; j++)
      {
        D[i][j] *= beta;
        for (k = 0; k < _PB_NJ; ++k)
          D[i][j] += tmp[i][k] * C[k][j];
      }
```

2mm-omp-4

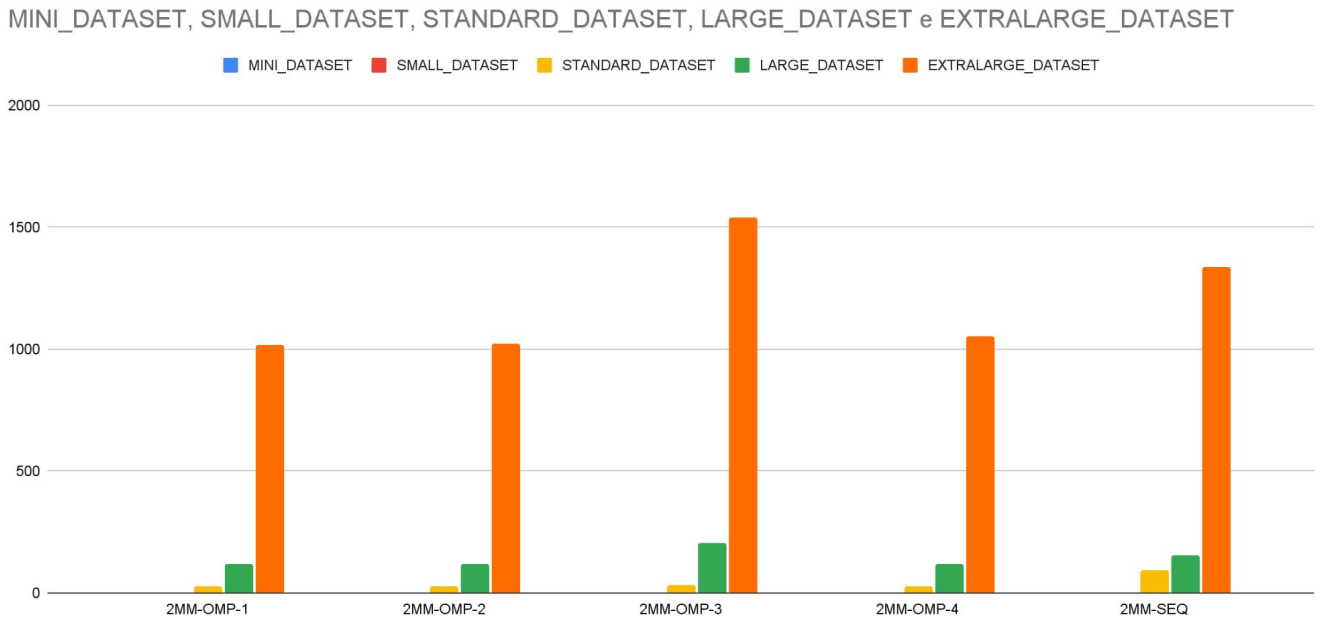
*Use of offloading on gpu device with the use of the **target teams** directive and **parallel for** directive*

Results

Exec. time	MINI_DATASET	SMALL_DATASET	STANDARD_DATASET	LARGE_DATASET	EXTRALARGE_DATASET
2MM-OMP-1	0,000102	0,01624	27,242825	119,804206	1018,048251
2MM-OMP-2	0,0001	0,016553	26,74888	117,848538	1023,896809
2MM-OMP-3	0,000982	0,039754	35,004477	204,705999	1538,485785
2MM-OMP-4	0,001491	0,017962	26,965387	120,903831	1050,288738
2MM-SEQ	0,00019	0,026052	95,855588	152,016992	1336,088332

Speedup	MINI_DATASET	SMALL_DATASET	STANDARD_DATASET	LARGE_DATASET	EXTRALARGE_DATASET
2MM-OMP-1	1,86	1,60	3,52	1,27	1,31
2MM-OMP-2	1,90	1,57	3,58	1,29	1,30
2MM-OMP-3	0,19	0,66	2,74	0,74	0,87
2MM-OMP-4	0,13	1,45	3,55	1,26	1,27

Execution time graph



Speedup graph