| Esercitazione | 5 | 20/04/2020 | Array Smoothing |
|---|---|---|---|

**Sviluppare un programma MPI in C per il seguente problema.**

1. Sviluppare un programma MPI che esegue l'operazione di **Array Smoothing** di un array A di dimensione N, utilizzando P processi per un numero di iterazioni pari ad I. In ogni iterazione si deve calcolare un nuovo array B, dove l'i–esimo elemento di B è calcolato come la media di 2K elementi in A:

$$B_i = \frac{1}{2k-1} \sum_{j=-k}^{k} A_{i+k}$$

    - MASTER
        - genera un array A di dimensione N e lo inizializza con A[i]=i.
        - invia ai P–1 processi una porzione dell'array in modo equo.
        - riceve dagli slave i sub–array e scrive su standard output lo stato della computazione.
    - SLAVE
        - invia e riceve dai propri vicini in modo non bloccante gli elementi di bordo.
        - calcola lo smoothing dei soli elementi che non necessitano di elementi di competenza dei vicini.
        - controlla la ricezione degli elementi di bordo e calcola lo smoothing degli elementi rimanenti.
        - invia al MASTER il suo sub–array.

E' possibile utilizzare le funzioni di comunicazione collettiva dello standard MPI.

1. Ottimizzare utilizzando le funzioni PACK e UNPACK.
2. Ottimizzare utilizzando i tipi di dato derivato di MPI.
3. Valutare le prestazioni del programma in termini di scalabilità forte e debole, variando P, N e K, utilizzando un cluster Amazon AWS di t2.micro.
4. Descrivere il programma in un file Markdown:
    - Presentare la soluzione 10 righe.
    - Descrizione del codice e presentazione del codice.
    - Presentazione dei risultati per un certo numero di iterazioni I costante:
        - Al variare di K_test={K/2, K, 2K}:
        - Scalabilità debole variare P e mantenere costante la porzione di array per ogni processo.
        - Scalabilità forte variare P e mantenere costante la dimenezione N.

5. **TIPS**
    - I valori dei rank si riferiscono al valore dei processi nell'indicizzazione ottenuta dal communicator MPI_COMM_WORLD.
    - I valori di I, K, N e P devono essere scelti al fine di ottenere dei risultati che siano veritieri.
    - La valutazione delle prestazioni deve essere effettuato solo sulla versione più ottimizzata del programma.
    - Per ogni istanza t2.micro si deve eseguire un solo processo MPI.
    - Bisonga utilizzare un numero di istanze almeno pari a 16.
    - Esportare il file markdown in PDF per eventuali immagini, e cosegnare sia il file markdown che il PDF.
6. **Template file Markdown da consegnare**

````
| Array Smoothing | Nome e Cognome | Data di consegna |
| --- | --- | --- | --- |

# Descrizione della soluzione
10 righe

# Codice
<!--```c
#include <mpi.h>
#include <stdio.h>

int main(int argc, char** argv) {
    MPI_Init(NULL, NULL);
    int world_size;
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);
    int world_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
    char processor_name[MPI_MAX_PROCESSOR_NAME];
    int name_len;
    MPI_Get_processor_name(processor_name, &name_len);
    printf("Hello world from processor %s, rank %d out of %d processors\n",
           processor_name, world_rank, world_size);
    MPI_Finalize();
}
```-->
## Note sulla compilazione
## Note sull'implementazione

# Risulati
I={some value}
````
````

```
P={some value}

### K = {some value}
#### Scalabilità debole N={some value}
#### Scalabilità forte N={some value}

### K/2 = {some value}
#### Scalabilità debole N={some value}
#### Scalabilità forte N={some value}

### 2K = {some value}
#### Scalabilità debole N={some value}
#### Scalabilità forte N={some value}

## Descrizione dei risultati

# Conclusioni
```