



Cómputo Paralelo

y en la Nube

Proyecto

Clausura

Bolsa de Palabras
con **MPI** para el
Análisis de texto

Escenario ejemplo con 6 Libros para ejecución con 6 Procesos

- `shakespeare_the_merchant_of_venice`
- `shakespeare_romeo_juliet`
- `shakespeare_hamlet`
- `shakespeare_the_merchant_of_venice`
- `shakespeare_romeo_juliet`
- `shakespeare_hamlet`

	...	yourn	youth	youthful	youthfulness	zeal	zealous	zenith	zip	zone	zounds
0	...	0	8	1	0	1	0	0	0	0	0
1	...	0	6	3	0	0	0	0	1	0	2
2	...	0	14	0	0	0	0	0	2	1	0
3	...	0	1	0	0	1	0	0	1	0	0
4	...	0	9	6	0	0	2	1	0	0	0
5	...	1	10	3	1	0	2	0	1	0	0



Requerimientos



Entrada

- Número de procesos a utilizar
- Listado de nombres de los archivos donde se encuentran las palabras a contar (los archivos se pueden encontrar en el mismo lugar que el ejecutable)

Salida

- Archivo con matriz de Bolsa de Palabras en formato csv.

Requerimientos



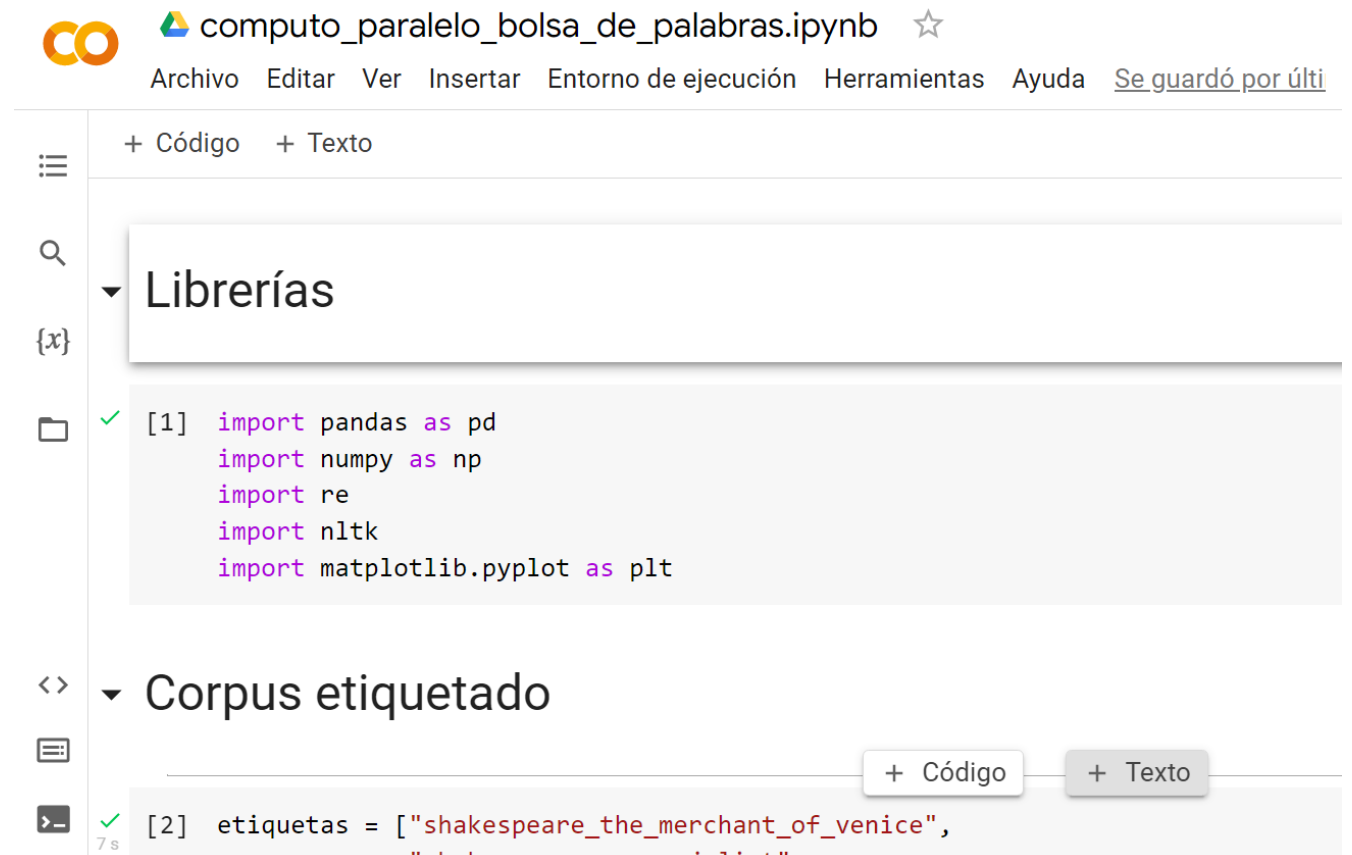
- Implementar versión paralela con MPI y Versión Serial.
- Calcular e imprimir speed-up, i.e., comparar el tiempo de ejecución de la version paralela con la versión serial.

Libreta de Google Colab

- https://colab.research.google.com/drive/1v4VvVDYpdeZjtnDcUzKeSVjDk5W6_YNf?usp=sharing



- Los libros (archivos txt de ejemplo) se encuentran en la carpeta de OneDrive del curso.



Criterios de evaluación



- Se deberá obtener un speed up mayor a 1.2x.
- Peso total del proyecto: 2.0 puntos de su calificación final.
- Ejecución del proyecto con todos los requerimientos indicados en su descripción (1.9 puntos)
- Código documentado correctamente (0.1 puntos)
- Fecha de entrega:
 - Martes **2 de Diciembre de 2025** en clase.
Equipo de 1 ó 2 Personas
- **NOTA 1:** Si el código paralelo no es más eficiente que el código en automático aplica un descuento de **1 punto**.
- **NOTA 1:** 30% menos por cada día natural de retraso.
- **NOTA 2:** Si se entrega después de la hora de clase, en automático aplica un día menos.

```
#include <map>
#include <string>
#include <iostream>
using namespace std;
```

```
int main (int argc, char *argv[]) {
    map<string, int> my_dictionary;
    my_dictionary.insert({"computo", 100});
    my_dictionary.insert({"paralelo", 200});
    my_dictionary.insert({"curso", 300});
    //my_dictionary.at("paralelo") = 3;
    my_dictionary["paralelo"] = 4;
    cout << my_dictionary.at("paralelo") << "\n";
    cout << my_dictionary.at("computo") << "\n";
    cout << my_dictionary["paralelo"] << "\n";
    cout << my_dictionary.count("inexistente") << "\n";
    cout << my_dictionary.count("paralelo") << "\n";
    my_dictionary.erase("curso");
    for (auto iterator = my_dictionary.begin(); iterator != my_dictionary.end(); ++iterator) {
        cout << iterator->first << " " << iterator->second << "\n";
    }
    return 0;
}
```

Hola Mundo de Diccionarios

