Actividad 5

Materia: Cómputo de Alto Rendimiento

Programa: Maestría en Ciencia de Datos e Información, INFOTEC

Docente: Dra. Magali Arellano Vázquez

Alumno: Rodrigo Guarneros Gutiérrez

Introducción

En este documento se encuentra la captura de pantalla de la ejecución de cada código y los resultados de las métricas de desempeño de cada problema específico. **Para cada uno de ellos se elaboró el código secuencial y el paralelo para efectos de comparar su desempeño y resultado.**

1. Cálculo de π.
2. Mensajes encadenados.
3. Producto punto de dos vectores.
4. Producto punto de una matriz NXN y un vector $NX1$.
5. Aplicación del paradigma map-reduce.

Los códigos están disponibles en anexo, se trata de los códigos originales y funcionales disponibles para su ejecución y comprobación.

1. Cálculo secuencial y paralelo de la aproximación de π a partir de la suma de Riemann.

A continuación se presentan las líneas de comandos a ejecutar y los resultados (los códigos están disponibles y adjuntos a este documento).

```
rodri@Computer1Rod MINGW64 ~/OneDrive/Imágenes/Documentos/INFOTEC/Segundo Trimestre/Cómputo de alto rendimiento/HW5
(lenguajes MPI y MapReduce)/parallel_computing (main)
$ python pi_noparalelo.py
Introduzca el valor de n para la suma de Rimann: 100
Instroduzca el valor de precision: 4
El valor aproximado de PI es: 3.141600986923125, con un error de 0.000008333333332
Tiempo total de ejecucion: 0.00100111961364746094 segundos

rodri@Computer1Rod MINGW64 ~/OneDrive/Imágenes/Documentos/INFOTEC/Segundo Trimestre/Cómputo de alto rendimiento/HW5
(lenguajes MPI y MapReduce)/parallel_computing (main)
$ mpiexec -n 4 python pi_paralelo.py
Indica el valor de n para la suma de Riemann: 100
Indica por favor el factor de precisisn: 4

Proceso 2 de 4 el Rank (n·mero de orden) es: 2, Size (cantidad de procesos) es: 4, Suma local es: 78.29244650957668
Proceso 3 de 4 el Rank (n·mero de orden) es: 3, Size (cantidad de procesos) es: 4, Suma local es: 77.78741525634219
Proceso 1 de 4 el Rank (n·mero de orden) es: 1, Size (cantidad de procesos) es: 4, Suma local es: 78.79260283629755
Proceso 0 de 4 el Rank (n·mero de orden) es: 0, Size (cantidad de procesos) es: 4, Suma local es: 79.28763409009609
La aproximacisn del valor de PI es: 3.141600986923125, con un error de 0.000008333333332
Tiempo total de ejecucisn en la parte paralela: 0.00200319290161132812 segundos
```

En resumen, los resultados son los siguientes en términos de desempeño:

	Métrica	Código Secuencial	Código Paralelo
0	Error	83.33X10^-7	83.33X10^-7
1	Tiempo_ejecución_segundos	10x10^-4	20x10^-4

In [6]:
```
# Dependencias
import pandas as pd
```

In [11]:
```
resumen_pi = {'Métrica':['Error', 'Tiempo_ejecución_segundos'], 'Código Secuencial'
              'Código Paralelo':['83.33X10^-7', '20x10^-4']}
resumen_pi = pd.DataFrame(resumen_pi)
print(resumen_pi.to_markdown())
```

```
|    | Métrica                   | Código Secuencial   | Código Paralelo   |
|---:|:--------------------------|:--------------------|:------------------|
|  0 | Error                     | 83.33X10^-7         | 83.33X10^-7       |
|  1 | Tiempo_ejecución_segundos | 10x10^-4            | 20x10^-4          |
```

2. Recepción y envío de mensajes encadenados.

A continuación se presentan las líneas de comandos a ejecutar y los resultados (los códigos están disponibles y adjuntos a este documento).

```
rodri@Computer1Rod MINGW64 ~/OneDrive/Imágenes/Documentos/INFOTEC/Segundo Trimestre/Cómputo de alto rendimiento/HW5
(lenguajes MPI y MapReduce)/parallel_computing (main)
$ python mensajes_noparalelo2.py
Introduzca el numero de procesos (entre 2 y el número de nucleos): 4
Hola, soy el proceso 0
Son el proceso 1 y recibi el mensaje del proceso 0: Hola, soy el proceso 0
Yo soy el proceso 2 y recibi el mensaje del proceso 1: Hola, soy el proceso 0
Yo soy el proceso 3 y recibi el mensaje del proceso 2: Hola, soy el proceso 0
El tiempo total de ejecución es de: 0.0010020732879638672 segundos

rodri@Computer1Rod MINGW64 ~/OneDrive/Imágenes/Documentos/INFOTEC/Segundo Trimestre/Cómputo de alto rendimiento/HW5
(lenguajes MPI y MapReduce)/parallel_computing (main)
$ mpiexec -n 4 python mensajes_paralelo2.py
Introudce por favor el n·mero de procesos (entre 2 y el n·mero de n·cleos): 4

Soy el proceso 1 y he recibido el mensaje de 0 que dice: Hola soy el proceso 0
Hola soy el proceso 0
El tiempo total de este algoritmo fue de: 0.0020112991333007812 segundos
Soy el proceso 2 y he recibido el mensaje de 1 que dice: Hola soy el proceso 0
Soy el proceso 3 y he recibido el mensaje de 2 que dice: Hola soy el proceso 0
```

En resumen, el desempeño de ambos códigos es el siguiente: | | Métrica | Código Secuencial | Código Paralelo | |---:|:-------------------------|:-------------------|:-----------------| | 0 | Tiempo_ejecución_segundos | 10x10^-4 | 20x10^-4 |

In [16]:
```
resumen_mensajes = {'Métrica': ['Tiempo_ejecución_segundos'], 'Código Secuencial':
             'Código Paralelo':['20x10^-4']}
resumen_mensajes = pd.DataFrame(resumen_mensajes)
print(resumen_mensajes.to_markdown())
```

```
|    | Métrica                   | Código Secuencial   | Código Paralelo   |
|---:|:--------------------------|:--------------------|:------------------|
|  0 | Tiempo_ejecución_segundos | 10x10^-4            | 20x10^-4          |
```

3. Producto punto de dos vectores (selecciona su dimensión)

A continuación se presentan las líneas de comandos a ejecutar y los resultados (los códigos están disponibles y adjuntos a este documento).

```
rodri@Computer1Rod MINGW64 ~/OneDrive/Imágenes/Documentos/INFOTEC/Segundo Trimestre/Cómputo de alto rendimiento/HW5
(lenguajes MPI y MapReduce)/parallel_computing (main)
$ python producto_punto_noparalelo.py
Introduzca el número de elementos para el vector A y B: 4
Introduzca los elementos del vector A:
Elemento 1: 1
Elemento 2: 2
Elemento 3: 3
Elemento 4: 4
Introduzca los elementos del vector B:
Elemento 1: 5
Elemento 2: 6
Elemento 3: 7
Elemento 4: 8
Producto Punto Total = 70.0
Tiempo de ejecucion: 0.002998590469360 segundos
```

```
rodri@Computer1Rod MINGW64 ~/OneDrive/Imágenes/Documentos/INFOTEC/Segundo Trimestre/Cómputo de alto rendimiento/HW5
(lenguajes MPI y MapReduce)/parallel_computing (main)
$ mpiexec -n 4 python producto_punto_paralelo.py
Introduzca el n·mero de elementos para el vector A y B: 4
Introduzca los elementos del vector A:
Elemento 1: 1
Elemento 2: 2
Elemento 3: 3
Elemento 4: 4
Introduzca los elementos del vector B:
Elemento 1: 5
Elemento 2: 6
Elemento 3: 7
Elemento 4: 8
Producto Punto Total = 70.0
Tiempo de ejecucion: 0.000000000000000 segundos
```

En resumen, el desempeño de ambos códigos es el siguiente: | | Métrica | Código Secuencial | Código Paralelo | |---:|:-------------------------|:--------------------|-----------------:| | 0 | Tiempo_ejecución_segundos | 30x10^-4 | 0 |

In [18]:
```
resumen_pp = {'Métrica': ['Tiempo_ejecución_segundos'], 'Código Secuencial': ['30x1
              'Código Paralelo':['0']}
resumen_pp = pd.DataFrame(resumen_pp)
print(resumen_pp.to_markdown())
```

```
|    | Métrica                   | Código Secuencial   |   Código Paralelo |
|---:|:--------------------------|:--------------------|------------------:|
|  0 | Tiempo_ejecución_segundos | 30x10^-4            |                 0 |
```

4. Producto punto de una matriz mxn y un vector $nx1$ (selecciona su dimensión)

A continuación se presentan las líneas de comandos a ejecutar y los resultados (los códigos están disponibles y adjuntos a este documento).

```
rodri@Computer1Rod MINGW64 ~/OneDrive/Imágenes/Documentos/INFOTEC/Segundo Trimestre/Cómputo de alto rendimiento/HW5
(lenguajes MPI y MapReduce)/parallel_computing (main)
$ python matriz_producto_noparalelo.py
Introduce el tamano de la matriz (n): 4
Matriz A:
[[373 738  39 626]
 [445 922 558 485]
 [708 996 641 640]
 [580 101 823 374]]
Vector x:
[88 59 97 83]
Resultado:
[132107 187939 236365 167872]
Tiempo de ejecucion: 0.0000000000 segundos
```

```
rodri@Computer1Rod MINGW64 ~/OneDrive/Imágenes/Documentos/INFOTEC/Segundo Trimestre/Cómputo de alto rendimiento/HW5
(lenguajes MPI y MapReduce)/parallel_computing (main)
$ mpiexec -n 4 python matriz_producto_paralelo.py
Introduce el tama±o de la matriz (n): 4

C:\Users\rodri\OneDrive\Imßgenes\Documentos\INFOTEC\Segundo Trimestre\C≤mputo de alto rendimiento\HW5 (lenguajes MPI
 y MapReduce)\parallel_computing\matriz_producto_paralelo.py:48: RuntimeWarning: overflow encountered in scalar mult
iply
  local_result[i] += A[i + rank * local_n][j] * local_x[j]
C:\Users\rodri\OneDrive\Imßgenes\Documentos\INFOTEC\Segundo Trimestre\C≤mputo de alto rendimiento\HW5 (lenguajes MPI
 y MapReduce)\parallel_computing\matriz_producto_paralelo.py:48: RuntimeWarning: overflow encountered in scalar mult
iply
  local_result[i] += A[i + rank * local_n][j] * local_x[j]
Matriz A:
[[ 92 294 210 990]
 [639 270 390 111]
 [235 937 939 576]
 [712  33 773 928]]
Vector x:
[86 93 12 63]
Resultado del producto punto del vector x y matriz:
[      7912      59427 1100047406 1310159999]
Tiempo de ejecuci≤n: 0.0000000000 segundos
```

En resumen, el desempeño de ambos códigos es el siguiente: | | Métrica | Código Secuencial | Código Paralelo | |---:|:----------|:----------|----------:| | 0 | Tiempo_ejecución_segundos | 0 | 0 |

In [19]:
```
resumen_pp = {'Métrica': ['Tiempo_ejecución_segundos'], 'Código Secuencial': ['0'],
              'Código Paralelo':['0']}
resumen_pp = pd.DataFrame(resumen_pp)
print(resumen_pp.to_markdown())
```

```
|    | Métrica                   |   Código Secuencial |   Código Paralelo |
|---:|:--------------------------|--------------------:|------------------:|
|  0 | Tiempo_ejecución_segundos |                   0 |                 0 |
```

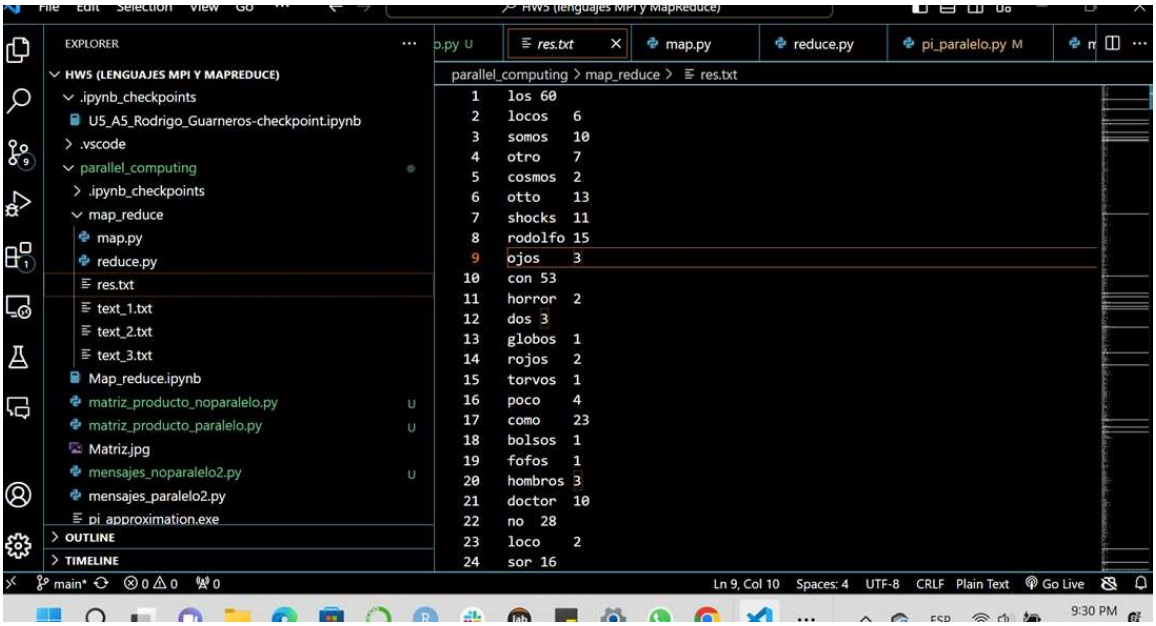
5. Implementación del código basado en el paradigma de map-reduce. Mapeando las palabras y reduciendo con base en su frecuencia absoluta

---

A continuación se presentan las líneas de comandos a ejecutar y los resultados (los códigos están disponibles y adjuntos a este documento).

```
rodri@Computer1Rod MINGW64 ~/OneDrive/Imágenes/Documentos/INFOTEC/Segundo Trimestre/Cómputo de alto rendimiento/HW5
(lenguajes MPI y MapReduce)/parallel_computing/map_reduce (main)
$ ls
map.py  reduce.py  text_1.txt  text_2.txt  text_3.txt

rodri@Computer1Rod MINGW64 ~/OneDrive/Imágenes/Documentos/INFOTEC/Segundo Trimestre/Cómputo de alto rendimiento/HW5
(lenguajes MPI y MapReduce)/parallel_computing/map_reduce (main)
$ ls *.txt | python map.py | python reduce.py > res.txt
```



Referencias

- Microsoft MPI Documentation (mpiexec). Consultado el viernes 6 de octubre. Disponible en:

https://learn.microsoft.com/en-us/powershell/high-performance-computing/mpiexec?view=hpc19-ps&source=recommendations

- Gropp, W., Gropp, W. D., Lusk, E., Skjellum, A., & Lusk, A. D. F. E. E. (1999). Using MPI: portable parallel programming with the message-passing interface (Vol. 1). MIT press. Disponible en: https://aulavirtual.infotec.mx/pluginfile.php/85009/mod_bootstrapelements/intro/usingmpi.
- Gropp, W., Hoefler, T., Thakur, R., & Lusk, E. (2014). Using advanced MPI: Modern features of the message-passing interface. MIT Press. Disponible en: https://aulavirtual.infotec.mx/pluginfile.php/85009/mod_bootstrapelements/intro/advanced
- Karniadakis, G., Karniadakis, G. E., & Kirby II, R. M. (2003). Parallel scientific computing in C++ and MPI: a seamless approach to parallel algorithms and their implementation (Vol. 2). Cambridge University Press. Disponible en: https://aulavirtual.infotec.mx/pluginfile.php/85009/mod_bootstrapelements/intro/sci.pdf