Implementación en Paralelo del Cálculo de Pi y Ordenamiento por MergeSort

Student:
Ayrton Fabio Coronado Huamán-@uni.pe
Luis Angel Flores Huamaní-@uni.pe
Herless Brayan Barrientos Porras-@uni.pe
Universidad Nacional de Ingeniería, Facultad de Ciencias,

Curso:
CC462A Sistemas Concurrentes y Distribuidos
Laboratorio 01

Abstract

Implementación de los códigos del Calculo de Pi y de Ordenamiento MergeSort, con la mejora en la eficiencia de los códigos al paralelizarlos
Keywords: Pi, MergeSort, Threads.

Contents

---

1 Introduction

En este artículo desarrollamos la implementación del cálculo de Pi mediante el algoritmo de Integrales de Rieman y Ordenamiento por el algoritmo MergeSort.Los cuales compararemos tanto en secuencial como en paralelo.

2 Theoretical Framework

2.1 Cálculo de Pi con Aproximación de una Integral

El método de la aproximación de la integral del Calculo de Pi, La aproximación de una integral mediante la suma de Riemann permite dividir el trabajo en unidades independientes en la que mayor numero de particiones mayor precisión.

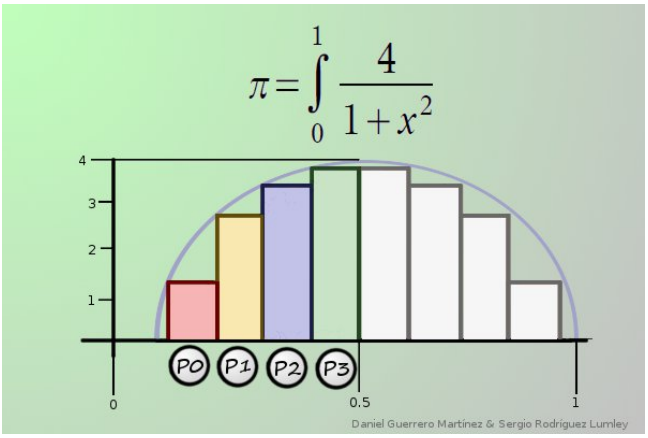
$$\pi = \int_0^1 \frac{4}{(1+x^2)} \cdot dx$$

A lo que estariamos calculando es:

$$\pi = \sum_{x=0}^1 \frac{4}{(1+x^2)} \cdot dx$$

El algoritmo que usaremos será el siguiente:

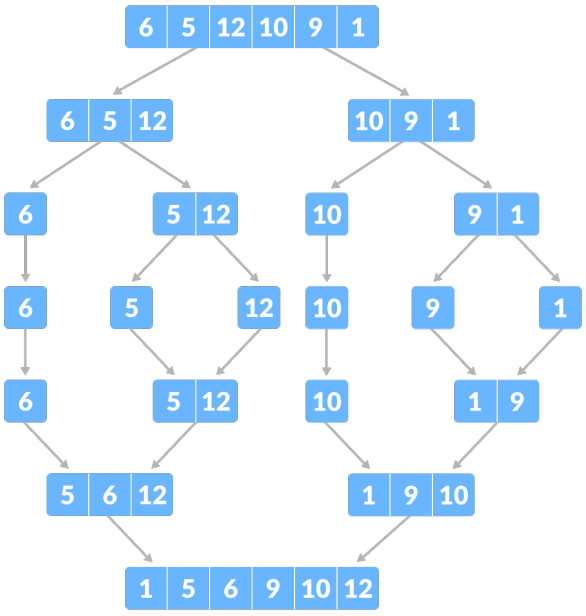
- Dividimos la longuitud de 0 a 1 en la cantidad de segmentos a tratar N.
- Calculamos cada partición de la sumatoria.
- Finalmente sumamos todas las particiones.



2.2 Ordenamiento MergeSort

El algoritmo de MergeSort de ordenamiento externo estable basado en la técnica divide y vencerás. Es de complejidad O(n log n).
El algoritmo será el siguiente:

- Si la longuitud de la lista es 0 o 1, entonces ya está ordenada.En otro caso:
- Dividir la lista desordenada en 2 sublistas de aproximadamente la mitad de tamaño.
- Ordenar cada sublista aplicando el ordenamiento por merge.
- Mezclar las 2 sublistas en una sola.



3 Results

Pi Secuencial.

```
11) 1024000 experimentos :     649 ms  3.14159265358987271132243904112836
12) 2048000 experimentos :    1178 ms  3.14159265358981310667756851348783 6
13) 4096000 experimentos :    2304 ms  3.14159265358979820551637466583158 6
14) 8192000 experimentos :    4643 ms  3.14159265358979448022607620391752 4
15) 16384000 experimentos :   10056 ms  3.14159265358979354890350158843900 8
16) 32768000 experimentos :   21192 ms  3.14159265358979331607285793456937 9
17) 65536000 experimentos :   44948 ms  3.14159265358979325786519702110197 2
18) 131072000 experimentos :  90729 ms  3.14159265358979324331328179273512 0
19) 262144000 experimentos :  195004 ms  3.14159265358979323967530298564340 7
20) 524288000 experimentos :  405527 ms  3.14159265358979323876580828387047 9
```

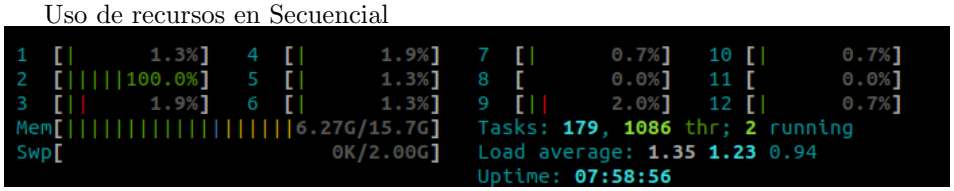
Pi Paralelo

```
<terminated> Pi [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Jun 16, 2020, 5
12) 2048000 experimentos :     349 ms  3.14159265358981310667756851348783 6
13) 4096000 experimentos :     665 ms  3.14159265358979820551637466583158 6
14) 8192000 experimentos :    1418 ms  3.14159265358979448022607620391752 4
15) 16384000 experimentos :    2935 ms  3.14159265358979354890350158843900 8
16) 32768000 experimentos :    6126 ms  3.14159265358979331607285793456937 9
17) 65536000 experimentos :   12538 ms  3.14159265358979325786519702110197 2
18) 131072000 experimentos :   24997 ms  3.14159265358979324331328179273512 0
19) 262144000 experimentos :   55595 ms  3.14159265358979323967530298564340 7
20) 524288000 experimentos :  111926 ms  3.14159265358979323876580828387047 9
```

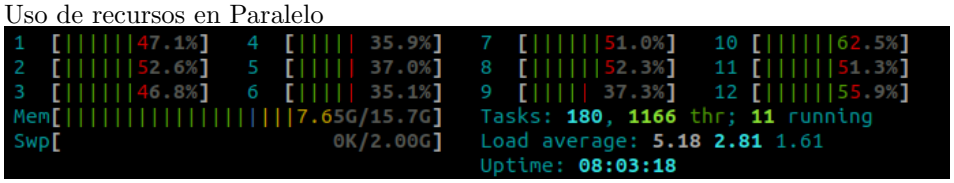
```
<terminated> MergeSort [Java Application] /usr/l
        Secuencial

1)       1500 elementos  :       1 ms
2)       3000 elementos  :       1 ms
3)       6000 elementos  :       2 ms
4)      12000 elementos  :       3 ms
5)      24000 elementos  :       5 ms
6)      48000 elementos  :       9 ms
7)      96000 elementos  :      14 ms
8)     192000 elementos  :      31 ms
9)     384000 elementos  :      59 ms
10)    768000 elementos  :     116 ms
11)   1536000 elementos  :     249 ms
12)   3072000 elementos  :     520 ms
13)   6144000 elementos  :    1087 ms
14)  12288000 elementos  :    2262 ms
15)  24576000 elementos  :    4497 ms
16)  49152000 elementos  :    9206 ms
        Paralelo

1)       1500 elementos  :       2 ms
2)       3000 elementos  :       1 ms
3)       6000 elementos  :       2 ms
4)      12000 elementos  :       2 ms
5)      24000 elementos  :       2 ms
6)      48000 elementos  :       3 ms
7)      96000 elementos  :       8 ms
8)     192000 elementos  :      11 ms
9)     384000 elementos  :      23 ms
10)    768000 elementos  :      41 ms
11)   1536000 elementos  :      82 ms
12)   3072000 elementos  :     158 ms
13)   6144000 elementos  :     343 ms
14)  12288000 elementos  :     664 ms
15)  24576000 elementos  :    1351 ms
16)  49152000 elementos  :    2887 ms
```

MergeSort

Uso de recursos en Secuencial



Uso de recursos en Paralelo



4 Conclusions

- Se puede observar la mejora en el tiempo de ejecucion tanto del Calculo de Pi como el de ordenamiento
- En los Programas en secuencial se puede observar el gran trabajo hecho por un procesador al 100%, a diferencia de los respectivos en paralelo trabajan todos a menor % .

5 Anexo Codigo

https://github.com/lfloresh/CC462Concurrentes

6 Anexo Documentación

- https://es.wikipedia.org/wiki/Ordenamiento_por_mezcla.
- https://lsi.ugr.es/jmantas/ppr/tutoriales/tutorial_mpi.php?tuto=03_pi

References

[1]