

Tarea-1-PI-Distribuidos

Pulido Bejarano Raymundo

4 de Octubre del 2020

1. Introducción

En esta Tarea se desarrollo un pequeño sistema distribuido usando una topologia de nodos de tipo estrella, contando de 4 nodos incluyendo al central, para con esta generar una aproximación a el número PI.

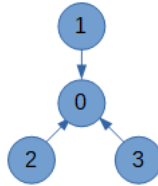


Figura 1: Topologia del Sistema

Para esto se tomo como base la Serie de Gregory-Leibniz que nos dice:

$$\pi = 4(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots)$$

Donde esta igualdad viene de un análisis matemático que tiene como base una serie de Taylor:

$$\frac{1}{1-y} = 1 + y + y^2 + \dots$$

Se le aplica una sustitución de $y = -x^2$, y con esto tenemos:

$$\frac{1}{1+x^2} = 1 - x^2 + x^4 - x^6 + \dots$$

Ahora recordando que:

$$\frac{d \tan^{-1} x}{dx} = x - \frac{x^5}{3} + \frac{x^5}{5} - \dots$$

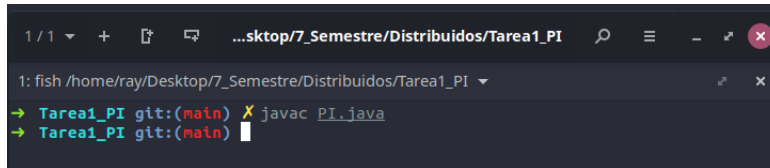
Sustituimos $x = 1$ y obtenemos :

$$\pi = 4(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots)$$

Por lo que esta serie se aproximara de gran manera a π entre mayor sea el numero de sumandos.

Con esto en mente, cada nodo de la topologia generara 10000000 de números a la iteración.

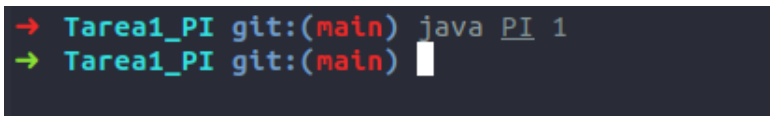
2. Capturas de Pantalla



```
1 / 1 + [ ] ...sktop/7_Semestre/Distribuidos/Tarea1_PI
1: fish /home/ray/Desktop/7_Semestre/Distribuidos/Tarea1_PI
→ Tarea1_PI git:(main) javac PI.java
→ Tarea1_PI git:(main)
```

Figura 2: Compilación

En la figura 2 se muestra la correcta compilación del archivo *PI.java*, en el cual está escrito el programa solución al problema planteado.



```
→ Tarea1_PI git:(main) java PI 1
→ Tarea1_PI git:(main)
```

Figura 3: Ejecución Nodo 1

En la figura 3, se muestra la correcta operación del programa usando de parámetro el *numero 1*, que nos dice que hará como el nodo 1 descrito en la topologia y a su vez de una forma abstracta de cliente al servidor, suministrándole el procesamiento de 10 millones de números parte en esta serie.

```
→ Tarea1_PI git:(main) java PI 2  
→ Tarea1_PI git:(main) □
```

Figura 4: Ejecución Nodo 2

En la figura 4, se muestra la correcta operación del programa usando de parámetro el *numero 2*, que nos dice que hará como el nodo 2 descrito en la topología y a su vez de una forma abstracta de cliente al servidor, suministrándole el procesamiento de 10 millones de números parte en esta serie.

```
→ Tarea1_PI git:(main) java PI 3  
→ Tarea1_PI git:(main) □
```

Figura 5: Ejecución Nodo 3

En la figura 5, se muestra la correcta operación del programa usando de parámetro el *numero 3*, que nos dice que hará como el nodo 3 descrito en la topología y a su vez de una forma abstracta de cliente al servidor, suministrándole el procesamiento de 10 millones de números parte en esta serie.

```
→ Tarea1_PI git:(main) java PI 0  
3.141592628592157  
→ Tarea1_PI git:(main) □
```

Figura 6: Ejecución Nodo 0

En la figura 5, se muestra la correcta operación del programa usando de parámetro el *numero 0* y de una forma funcional, hará de servidor central para gestionar las peticiones y operar los datos suministrados por los clientes.

3. Conclusión

Podemos apreciar las bondades de procesar un problema de forma distribuida ya que esto nos divide el costo de procesamiento de los datos en

el número de nodos existente, por lo que adaptar un proceso para poder realizarlo de forma distribuida, es sumamente rentable en tiempo de procesamiento.