DSD P2-2

PABLO BLANCO LOPEZ

Marzo 2024

1. Introducción

La práctica 2 de Desarrollo de Sistemas Distribuidos consta de hacer un sistema cliente-servidor de una calculadora con llamadas a procedimiento remoto (RPC), pero en al contrario que en la parte uno de esta misma práctica que se usaba RPC Sun, en esta se desarrolla con Apache Thrift. Se pide hacer que la calculadora haga operaciones básicas, y operaciones con grados como seno, coseno, tangente y conversión de grados a radianes, y como puntos extra que haga operaciones con cierta complejidad entre otras cosas, como montar el servidor y el cliente en diferentes idiomas, esta idea era la principal a desarrollar, pero hubo ciertos incovenientes que impedían que Thrift genere de forma correcta dado el IDL los demás archivos para java, que era dónde de primera mano se quería montar el servidor. Al ver estos problemas, se optó por introducir operaciones básicas entre enteros, operaciones de grados, operaciones de vectores y operaciones de matrices.

2. Funcionamiento

Una vez lanzado el servidor, si se ejecuta el cliente salta un menú donde pide introducir por consola un número del 1 al 4, siendo el siguiente menú:

1. Operación básica
2. Operación de grados
3. Operación vectorial
4. Operación matricial

Dentro del menú 1. Operación básica, pide introducir el primer valor numérico de la operación, después el operador, siendo estos suma, resta, multiplicación y división, + - * /, respectivamente, procediendo del segundo valor numérico.

Dentro del menú 2. Operación de grados, pide introducir la operación, a elegir entre seno (sen), coseno (cos), tangente (tan) y conversión de grados a

radianes (RAD), despúes de introducir la operación, pide los grados de dicha operación.

Dentro el menú 3. Operación vectorial, pide introducir el tamaño de los vectores, seguido del primer vector, la operación, siendo estas sumar, restar, producto escalar y producto vectorial, + - * x, en el caso del último solo se puede con vectores de tamaño tres. Después pide introducir el segundo vector de la operación.

Dentro del menú 4. Operación matricial, pedirá primero ingresar el tamaño de las matrices, siendo este tamaño el de las dos matrices, acto seguido pide introducir los valores de la primera matriz, el operador, siendo estos suma, resta y multiplicación entre matrices, + - * , respectivamente, y por último pide introducir la segunda matriz de la operación.

3. Explicación

Para la realización de la práctica, primero se tuvo que instalar en el equipo de trabajo la herramienta de Apache Thrift, acto seguido una vez instalada y verificada que funciona correctamente con python, esto se hace habiendo seleccionado dicho lenguaje en el instalador. Una vez funcionando Thrift, se debe crear un archivo .thrift, siendo este el IDL de nuestro proyecto.

```
struct resultBasic {
   1: i32      error,
   2: double   res
}

struct resultVector {
   1: i32         error,
   2: list<i32>   res
}

struct resultMatrix {
   1: i32              error,
   2: list<list<i32>>   res
}

service Calculadora {
   void ping(),
   resultBasic operacionBasica(1:i32 firstNum, 2:i32 secondNum, 3:string operation),
   resultBasic operacionGrados(1:double degree, 2:string operation),
   resultVector operacionVectorial(1:list<i32> firstVector, 2:list<i32> secondVector, 3:string operation),
   resultMatrix operacionMatricial(1:list<list<i32>> firstMatrix, 2:list<list<i32>> secondMatrix, 3:string operation)
}

```

Figura 1: calculadora.thrift

Una vez que tengamos el archivo .thrift que debe contener todos los servicios que dará nuestro servidor, al ser un IDL, thrift generará dichos archivos intermediarios entre nuestro servidor y cliente. Esto con thrift se hace con el siguiente comando.

```
$ thrift -gen py calculadora.thrift
```

Como se ha mencionado anteriormente, esto genera distintos archivos, con los tipos de estructuras que se pueden ver en el archivo calculadora.thrift, pero al ser python, que no tiene estructuras, genera para cada estructura una clase con dichos atributos. Esto se ha utilizado por si en el servidor ocurre algún tipo de error, se pone en el atributo error el número de dicho error, en el caso de que no hubiera sería un 0, y en la variable res de cada clase, el resultado de la operación correspondiente, a no ser que se haya notificado algún error, que el valor de dicho resultado será 0.

No se han experimentado muchos incovenientes, ya que en la anterior práctica se hizo en C, siendo este un idioma mucho más complicado que Python, sobre todo a la hora de alojar memoria, punteros y liberar dicha memoria. El único incoveniente, como se mencionó anteriormente era hacer el servidor en Java, tras horas de investigación, no se pudo hacer en Java por no encontrar el driver correspondiente para hacer que el código que se genera con Java funcione de una forma correcta.

4. Entrega

Para la entrega he comprimido mi carpeta de trabajo, que contiene principalmente los tres archivos con los que se ha trabajado, 'calculadora.thrift', 'servidor.py' y 'cliente.py'.