



# UNIVERSIDAD DE GRANADA

Calculadora - RPC Apache Thrift

# Índice

---

<b>Índice</b>	<b>2</b>
<b>Introducción</b>	<b>3</b>
<b>Funcionalidad</b>	<b>3</b>
Calculadora simple	3
Calculadora compleja	3
<b>Esquema de comunicación</b>	<b>4</b>
<b>Instrucciones para la ejecución</b>	<b>5</b>
Calculadora simple	5
Python	5
Java	6
Calculadora compleja	6
Python	6
Java	7
<b>Ejemplos de ejecución</b>	<b>8</b>

## Introducción

Se ha implementado una calculadora distribuida. Para ello se han utilizado llamadas a procedimientos remotos, RPC, es decir, el programa cliente es capaz de ejecutar funciones que se encuentran en otra máquina, que hace de servidor. Es capaz de efectuar la comunicación gracias a protocolos de red como TCP o UDP.

## Funcionalidad

Se han implementado 4 programas clientes y 4 servidores. Los clientes hacen llamadas a procedimientos que se encuentran implementados en los servidores, quienes les enviarán la respuesta a sus solicitudes.

Clientes:

- Operaciones básicas (python y java)
- Operaciones con vectores (python y java)

Servidores:

- Operaciones básicas (python y java)
- Operaciones con vectores (python y java)

## Calculadora simple

Permite realizar las siguientes operaciones con números en coma flotante.

- Suma
- Resta
- División
- Multiplicación

## Calculadora compleja

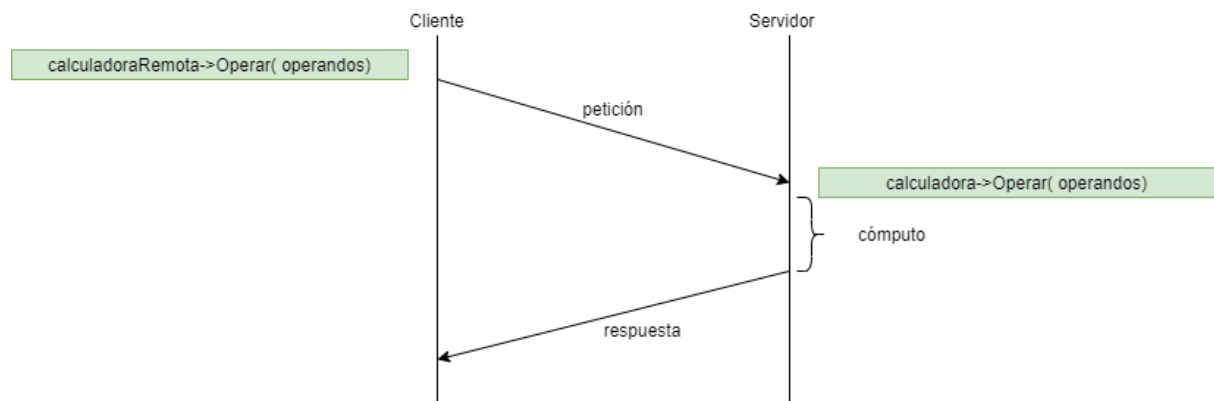
Permite realizar las siguientes operaciones con vectores:

- Suma
- Resta
- Multiplicación

## Esquema de comunicación

El cliente, ya sea el simple que opera con floats o el complejo que opera con vectores, pide al servidor el resultado de una función. Nótese que ambos programas pueden ser ejecutados en máquinas distintas con espacios de direcciones totalmente diferentes. Además, la tecnología de apache thrift es agnóstica del lenguaje de programación. Así pues, podremos ejecutar el cliente en java y hacer que se comunique con el servidor en python.

Estas operaciones han sido definidas en una interfaz. Gracias a la misma, ambos programas saben cómo deben de comunicarse (qué puede pedirle el cliente al servidor. cómo debe pedirlo (argumentos) y qué debería de devolverle el servidor (respuesta) ).



## Instrucciones para la ejecución

### Calculadora simple

#### Python

El proyecto ha implementado utilizando el IDE de JetBrains PyCharm. Por tanto, puede ser abierto en el mismo y ejecutado. Le he pasado el intérprete virtual que he utilizado para crear la práctica. Si se desea hacer por línea de comandos:

```
>python3 server.py
```

```
jesus@LAPTOP-AK4H2F20:~/P3/calculadora/python$ python3 server.py  
iniciando servidor...
```

```
>python3 client.py
```

```
jesus@LAPTOP-AK4H2F20:~/P3/calculadora/python$ python3 client.py  
¡Bienvenido a la calculadora distribuida de Jesús!  
Atención: se usará como separador de decimales el punto ('.')  
Operando 1: 5  
Operador (+ - / * : x): /  
Operando 2: 0  
Hubo un error  
5.0 / 0.0 = None  
jesus@LAPTOP-AK4H2F20:~/P3/calculadora/python$ python3 client.py  
¡Bienvenido a la calculadora distribuida de Jesús!  
Atención: se usará como separador de decimales el punto ('.')  
Operando 1: 3  
Operador (+ - / * : x): *  
Operando 2: 7.5  
3.0 * 7.5 = 22.5
```

## Java

```
> java -jar server.jar
```

```
jesus@LAPTOP-AK4H2F20:~/P3/calculadora/java/out/artifacts/server_jar$ java -jar server.jar  
Starting the simple server...
```

```
> java -jar client.jar
```

```
jesus@LAPTOP-AK4H2F20:~/P3/calculadora/java/out/artifacts/client_jar$ java -jar client.jar  
¡Bienvenido a la calculadora distribuida de Jesús!  
Se utiliza la coma (',') como separador de decimales.  
Operando 1: 1  
Operador (+ - / : x *): +  
Operando 2: 1  
1.0 + 1.0 = 2.0
```

## Calculadora compleja

### Python

```
>python3 complex_server.py
```

```
jesus@LAPTOP-AK4H2F20:~/P3/calculadora/python$ python3 complex_server.py  
iniciando servidor complejo...
```

```
>python3 complex_client.py
```

```
jesus@LAPTOP-AK4H2F20:~/P3/calculadora/python$ python3 complex_client.py  
¡Bienvenido a la calculadora distribuida de Jesús!  
Atención: se usará como separador de decimales el punto ('.')  
Leyendo los vectores:  
Primer vector:  
Introduzca el tamaño del vector: 4  
Elemento 0 : 1  
Elemento 1 : 2  
Elemento 2 : 3  
Elemento 3 : 4  
Segundo vector:  
Introduzca el tamaño del vector: 4  
Elemento 0 : 5  
Elemento 1 : 6  
Elemento 2 : 0  
Elemento 3 : 7  
Introduzca la operación que desea realizar (+ - * x) : -  
Resultado: [-4.0, -4.0, 3.0, -3.0]
```

## Java

```
> java -jar complex_server.jar
```

```
^C^Vjesus@LAPTOP-AK4H2F20:~/P3/calculadora/java/out/artifacts/server_jar$ java -jar complex_server.jar
Starting the complex server...
```

```
> java -jar complex_client.jar
```

```
jesus@LAPTOP-AK4H2F20:~/P3/calculadora/java/out/artifacts/client_jar$ java -jar complex_client.jar
Bienvenido a la calculadora distribuida de Jesús!
Se utiliza la coma (',') como separador de decimales.
Introduzca el tamaño del primer vector: 3
Elemento 0: 1
Elemento 1: 2
Elemento 2: 3
Operador (+ - / : x *): +
Introduzca el tamaño del segundo vector: 3
Elemento 0: 1
Elemento 1: 2
Elemento 2: 3
[2.0, 4.0, 6.0]
```

## Ejemplos de ejecución

A partir de la interfaz, que se encuentra en el archivo con extensión .thrift, generamos el resto de archivos para poder realizar la calculadora. Una vez el cliente y el servidor hayan sido implementados, podremos ejecutar el programa.

```
..\Thrift\thrift-0.14.1.exe --gen py .\calculadora.thrift  
..\Thrift\thrift-0.14.1.exe --gen java .\calculadora.thrift
```

Para ejecutar la aplicación lo primero que debemos de hacer es levantar el servidor. Podemos elegir cualquiera de ellos. Veamos un ejemplo de ejecución con un servidor de python y cliente en java. Observamos que el servidor abrirá un socket que escuchará en el puerto 9090.

```
36  
37 ▶ if __name__ == "__main__":  
38     handler = CalculadoraHandler()  
39     processor = Calculator.Processor(handler)  
40     transport = TSocket.TServerSocket(host="127.0.0.1", port=9090)  
41  
42 Run: server (1) x  
43 D:\Escritorio\Universidad\DSO\Practicas\Repositorio\P3\calculadora\python\venv\Scripts\python.exe D:/  
44 iniciando servidor...  
45 |
```

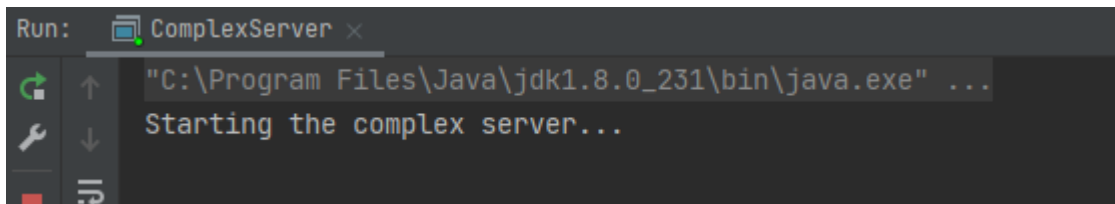
Por otro lado, debemos de ejecutar el cliente. En este caso, en Java.

```
14 try {  
15     // Set the client up  
16     TTransport transport = new TSocket( host: "localhost", port: 9090);  
17  
18     Calculator client = new Calculator(transport);  
19     client.open();  
20  
21     System.out.println("Bienvenido a la calculadora distribuida de Jesús!");  
22     System.out.println("Se utiliza la coma (',') como separador de decimales.");  
23  
24     System.out.print("Operando 1: ");  
25     double op1 = Double.parseDouble(System.console().readLine());  
26  
27     System.out.print("Operador (+ - / : x *): ");  
28     String op = System.console().readLine();  
29  
30     System.out.print("Operando 2: ");  
31     double op2 = Double.parseDouble(System.console().readLine());  
32  
33     double result = client.calculate(op1, op, op2);  
34     System.out.println(op1 + " " + op + " " + op2 + " = " + result);  
35  
36     client.close();  
37 } catch (Exception e) {  
38     e.printStackTrace();  
39 }  
40  
41 Process finished with exit code 0  
42 |
```



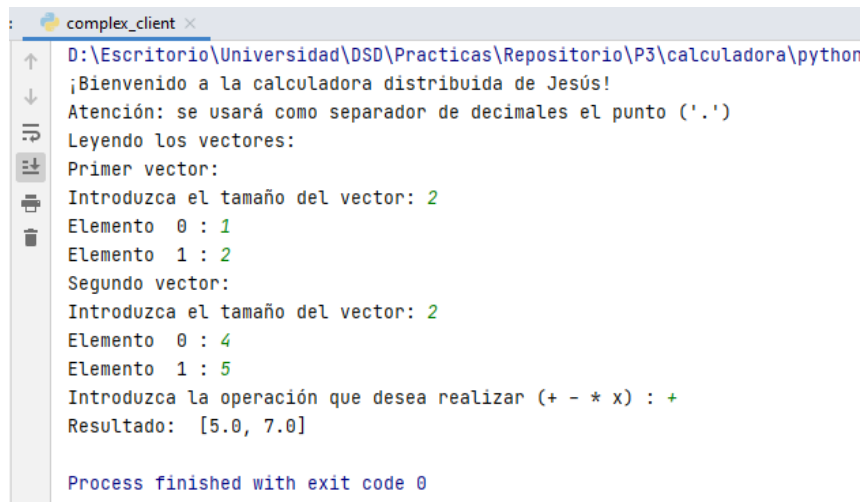
Podríamos hacerlo al revés. Utilizar el servidor en java y el cliente en python o ambos en el mismo lenguaje. Al fin y al cabo, esta tecnología es agnóstica al lenguaje porque el protocolo de transporte es común.

Veamos un ejemplo de la calculadora compleja, donde utilizamos vectores:



```
Run: ComplexServer x
"C:\Program Files\Java\jdk1.8.0_231\bin\java.exe" ...
Starting the complex server...
```

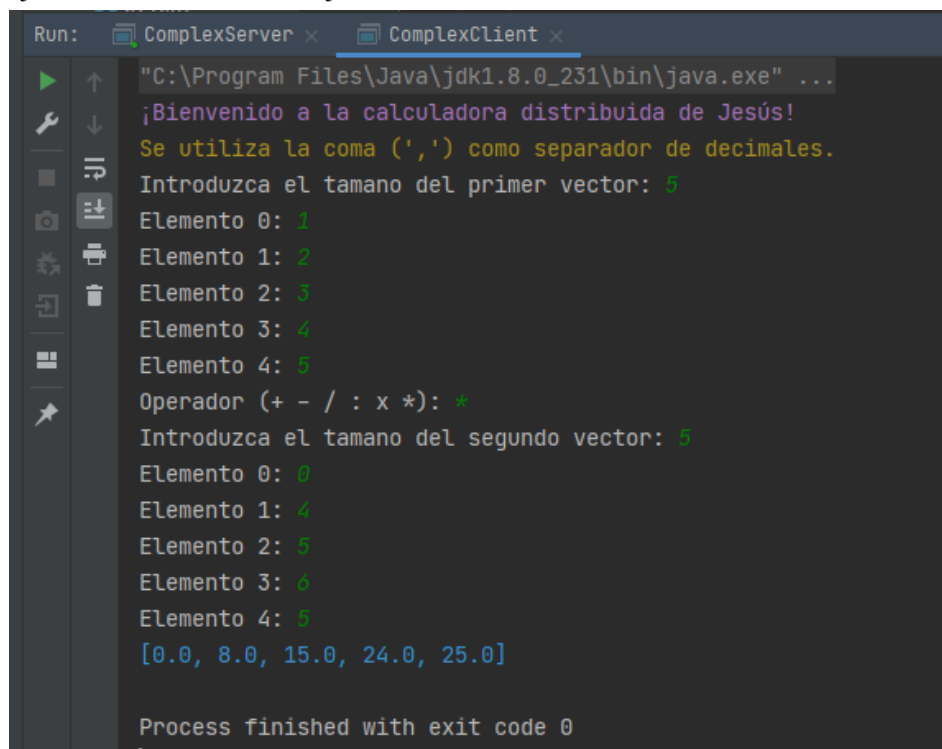
Ejecución con el cliente de python:



```
complex_client x
D:\Escritorio\Universidad\DSD\Practicas\Repositorio\P3\calculadora\python
¡Bienvenido a la calculadora distribuida de Jesús!
Atención: se usará como separador de decimales el punto ('.')
Leyendo los vectores:
Primer vector:
Introduzca el tamaño del vector: 2
Elemento 0 : 1
Elemento 1 : 2
Segundo vector:
Introduzca el tamaño del vector: 2
Elemento 0 : 4
Elemento 1 : 5
Introduzca la operación que desea realizar (+ - * x) : +
Resultado: [5.0, 7.0]

Process finished with exit code 0
```

Ejecución con el cliente en java:



```
Run: ComplexServer x ComplexClient x
"C:\Program Files\Java\jdk1.8.0_231\bin\java.exe" ...
¡Bienvenido a la calculadora distribuida de Jesús!
Se utiliza la coma (',') como separador de decimales.
Introduzca el tamaño del primer vector: 5
Elemento 0: 1
Elemento 1: 2
Elemento 2: 3
Elemento 3: 4
Elemento 4: 5
Operador (+ - / : x *): +
Introduzca el tamaño del segundo vector: 5
Elemento 0: 0
Elemento 1: 4
Elemento 2: 5
Elemento 3: 6
Elemento 4: 5
[0.0, 8.0, 15.0, 24.0, 25.0]

Process finished with exit code 0
```