PRÁCTICA 01. INFORME

Alumnos:

- Younes Kabiri <younes.kabiri@estudiants.urv.cat>
- Safia Guellil <safia.guellil@urv.cat>

20-Jun-2021

Younes Kabiri
Safia Guellil

NOTA: Hemos tenido problemas para conectar con el git i el Visual Code.

Tabla de contenido

Younes Kabiri
Safia Guellil

Conceptos Generales

Comunicación Directa

La comunicación directa se realiza entre el **Cliente** y el **Master**. Como bien dice su nombre no necesita de un intermediario, sino que ambas entidades se comunican directamente. Para implementarla hacemos uso de **GRPC**, ya que encripta los datos y es más veloz que el XML-RPC.

Implementación sistema GRPC

Primeramente hemos tenido que crear el archivo cluster.proto, el cual define los elementos de la comunicación.

Aquí distinguimos 4 tipos de modelos RPC:

Unary RPC: En este modelo de comunicación el cliente envía una petición y el servidor le responde con una respuesta.
Hacemos uso de este modelo en las funciones:

- ***Create Workers***: el cliente envía el número de workers que quiere crear, el servidor le contesta con un mensaje de estado (ex:“Workers creados”)
- **Delete Workers**:el cliente envía el número de workers que quiere eliminar, el servidor le contesta con un mensaje de cuantos workers ha eliminado.

Server streaming RCP: En este modelo el cliente envía una petición al servidor, pero recibe varias respuestas.
Hacemos uso de este modelo en la función:

- ***List Workers***: el cliente envía una petición de listar los workers, el servidor le contesta con varios mensajes de respuesta (stream), en el cual cada uno corresponde a la información del worker.

Cliente streaming RPC: En este modelo el cliente envía varias peticiones al servidor y este último le contesta con una respuesta.
Hacemos uso de este modelo en la función:

- ***Counting Words***: El cliente envía al servidor varios ficheros, el servidor contará el número de palabras de cada fichero, unirá la suma y le retornará un único valor total (una respuesta).

Bidirectional streaming RPC: En este modelo el cliente envía varias peticiones al servidor y a su vez el servidor le contesta con varias respuestas.
Hacemos uso de este modelo en la función:

- ***Word Counts***: El cliente envía al servidor varios ficheros, este último los lee y le responde con tuplas (word, num). La palabra y la frecuencia de veces que aparece en los ficheros.

Younes Kabiri
Safia Guellil

Comunicación Indirecta

La comunicación indirecta se realiza entre el Master y los Workers. El Master no ordena directamente a los workers las tareas que deben realizar sino lo hace mediante un intermediario, en nuestro caso será las colas de REDIS.

Implementación sistema REDIS

REDIS estará formada por 3 estructuras:

1. **Cola Tasks**: Esta cola será en la cual el Master colocará las tareas a medida que le lleguen peticiones del cliente. Por otro lado, los Workers adquirirán las tareas (publicadas por el master) de esta cola.

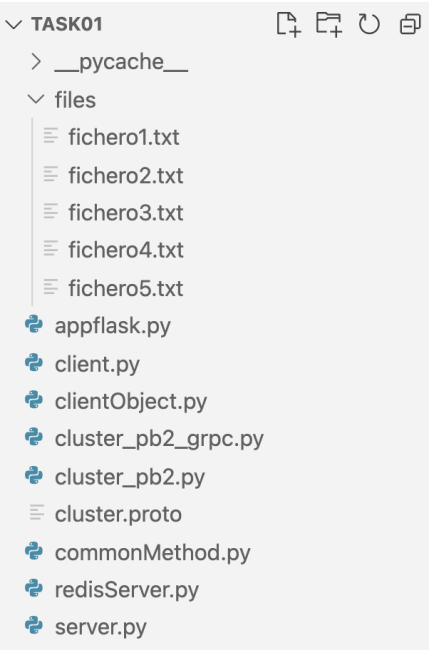
2. **Cola Intermediate**: En esta cola almacenaremos los resultados (NO FINALES) generados por los Workers, para que el worker que haga la tarea de “UNIFICAR”, las pueda adquirir y unificar. En esta cola los Workers suben sus resultados y el worker “unificador” los adquiere para generar el resultado final.

3. **Hash**: En esta estructura almacenaremos el resultado final del JOB. El worker unificador se encargará de subir en esta cola el resultado, por otro lado, el master adquirirá de esta estructura el resultado y se lo comunicará al cliente.

Estructura Proyecto

En la imagen de la derecha podemos ver la estructura general del proyecto.
A continuación explicamos brevemente la función principal de cada archivo:

- **cluster.proto**: define los elementos de la comunicación GRPC.
- **client.py**: métodos del cliente para realizar peticiones al servidor y recibir respuestas.
- **server.py**: métodos del servidor para responder a las peticiones del cliente.
- **clientObject.py**: define la clase del Cliente.
- **commonMethod.py**: definen los métodos utilizados para
- **redisServer.py**: define todas las colas y estructuras de REDIS.
- **appflask.py:** define las rutas del servidor local.
- **files**: carpeta con 5 ficheros.



Younes Kabiri
Safia Guellil

Juego de Pruebas

Test: Crear Worker

#	Descripción	Resultado	¿Ok?
1	*Crear 1 worker.* *Resultado Esperado=* *“Workers created 1”*	Tiempo=0.0045 Resultado= *“Workers created 1”*	ok
2	*Crear 0 workers.* *Resultado Esperado=* *“Workers created 0”*	Tiempo=1.19 Resultado= *“Workers created 0”*	ok
3	*Crear 10 workers.* *Resultado Esperado=* *“Workers created 10”*	Tiempo=0.079 Resultado= *“Workers created 10”*	ok

Capturas: Crear Worker

Comand: python3 client.py worker create 1

```
workers: 1

message: "Workers created: 1"
```

Comand: python3 client.py worker delete 0

```
message: "Workers created: 0"
```

Comand: python3 client.py worker delete 20

```
workers: 10

message: "Workers created: 10"
```

Younes Kabiri
Safia Guellil

Test: Eliminar Worker

#	Descripción	Resultado	¿Ok?
1	*Eliminar 1 worker. (previamente creado)* *Resultado Esperado=* *"Workers deleted: 1"*	Tiempo=3.218 Resultado= *"Workers deleted: 1"*	ok
2	*Eliminar 0 workers.* *Resultado Esperado=* *"Workers deleted: 0"*	Tiempo=1.38 Resultado= *"Workers deleted: 0"*	ok
3	*Eliminar más workers de los creados.* *(eliminar 5, creados 2)* *Resultado Esperado=* *"Workers deleted: 2"*	Tiempo=7.03 Resultado= *"Workers deleted: 2"*	ok

Capturas: Eliminar Worker

Comand: python3 client.py worker delete 1

```
workers: 1

message: "Workers deleted: 1"
```

Comand: python3 client.py worker delete 0

```
message: "Workers deleted: 0"
```

Comand: python3 client.py worker delete 5

```
workers: 5

message: "Workers deleted: 2"
```

Test:Listar Workers

#	Descripción	Resultado	¿Ok?
1	*Listar Workers* *(2 previamente creados)* *Resultado Esperado=* *2 Workers*	Tiempo=0.00021 Resultado= <Process name='Process-1' pid=10134 parent=10124 started> <Process name='Process-2' pid=10135 parent=10124 started>	ok
2	*Listar Workers (sin crear ningún worker)* *Resultado Esperado=* *0 Workers*	Tiempo=2.14 Resultado= 0 *Workers*	ok
3	*Listar 5 Workers. (creados)* *Resultado Esperado=* *5 Workers*	Tiempo=0.0006 Resultado= <Process name='Process-3' pid=10150 parent=10124 started> <Process name='Process-4' pid=10151 parent=10124 started> <Process name='Process-5' pid=10152 parent=10124 started> <Process name='Process-6' pid=10153 parent=10124 started> <Process name='Process-7' pid=10154 parent=10124 started>	ok

Capturas: Listar Worker

Comand: python3 client.py worker list

```
<Process name='Process-12' pid=10555 parent=10451 started>
<Process name='Process-13' pid=10556 parent=10451 started>
```

Comand: python3 client.py worker list
(No aparece nada)

Comand: python3 client.py worker list

```
<Process name='Process-12' pid=10555 parent=10451 started>
<Process name='Process-13' pid=10556 parent=10451 started>
<Process name='Process-14' pid=10568 parent=10451 started>
<Process name='Process-15' pid=10569 parent=10451 started>
<Process name='Process-16' pid=10570 parent=10451 started>
```

Younes Kabiri
Safia Guellil

Test: Counting Words

#	Descripción	Resultado	¿Ok?
1	*Contar palabras ficheros 1. [1 worker]* *Total Esperado =4*	Tiempo=0.012 Resultado=4	ok
2	*Contar palabras ficheros 1,2 y 3. [1 worker]* *Total Esperado =18*	Tiempo=0.0206 Resultado=18	ok
3	*Contar palabras ficheros 1,2 y 3. [3 worker]* *Total Esperado =18*	Tiempo=1.0186 Resultado=8	ok
4	*Contar palabras ficheros 1,2,3,4 y5 (Todos). [1 worker]* *Total Esperado =305*	Tiempo=0.03668 Resultado=305	ok
5	*Contar palabras ficheros 1,2,3,4 y5 (Todos). [6 worker]* *Total Esperado =305*	Tiempo=1.037 Resultado=305	ok
6	*Llamar a la función con una dirección de fichero incorrecta.* *Resultado Esperado=* *ERROR*	Tiempo= Resultado= ERROR	ok

Capturas: Counting Words

Comand: python3 client.py job run-countwords http://localhost:8000/fichero1.txt

```
--->Run countwords
totalWords: 4
```

Comand: python3 client.py job run-countwords http://localhost:8000/fichero1.txt http://localhost:8000/fichero2.txt http://localhost:8000/fichero3.txt

```
--->Run countwords
totalWords: 18
```

Younes Kabiri
Safia Guellil

Comand: python3 client.py job run-countwords http://localhost:8000/fichero1.txt http://localhost:8000/fichero2.txt http://localhost:8000/fichero3.txt

```
--->Run countwords
totalWords: 18
```

Comand: python3 client.py job run-countwords http://localhost:8000/fichero1.txt http://localhost:8000/fichero2.txt http://localhost:8000/fichero3.txt http://localhost:8000/fichero4.txt http://localhost:8000/fichero5.txt

```
--->Run countwords
totalWords: 305
```

Comand: python3 client.py job run-countwords http://localhost:8000/fichero1.txt http://localhost:8000/fichero2.txt http://localhost:8000/fichero3.txt http://localhost:8000/fichero4.txt http://localhost:8000/fichero5.txt

```
--->Run countwords
totalWords: 305
```

Comand: python3 client.py job run-countwords http://localhost:8000/fichero100.txt

```
--->Run countwords
Warnig...Empty address:  http://localhost:8000/fichero100.txt
```

Younes Kabiri
Safia Guellil

Test:Word count

#	Descripción	Resultado	¿Ok?
1	*Mostrar palabras (tuplas) fichero 1.[1 worker]* *Resultado Esperado=*	Tiempo=0.0187 Resultado=	ok
2	*Mostrar palabras (tuplas) fichero 1 y 2.[1 worker]* *Resultado Esperado=*	Tiempo=0.0280 Resultado=	ok
	Mostrar palabras (tuplas) fichero 1 y 2.[3 worker] *Resultado Esperado=*	Tiempo=1.033 Resultado=	ok
3	*Llamar a la función con una dirección de fichero incorrecta.* *Resultado Esperado=*	Tiempo=non Resultado=	ok

Capturas: Word count

Comand: python3 client.py job run-wordcount http://localhost:8000/fichero1.txt

```
--->Run wordcount
word: "foo"
num: 2

word: "bar"
num: 2
```

Comand: python3 client.py job run-wordcount http://localhost:8000/fichero1.txt http://localhost:8000/fichero2.txt

```
--->Run wordcount
word: "foo"
num: 2

word: "bar"
num: 2

word: "Created"
num: 1

word: "by"
num: 1

word: "Younes"
num: 1

word: "and"
num: 1

word: "Safia"
num: 1
```

Comand: python3 client.py job run-wordcount http://localhost:8000/fichero1.txt http://localhost:8000/fichero2.txt

```
--->Run wordcount
word: "foo"
num: 2

word: "bar"
num: 2

word: "Created"
num: 1

word: "by"
num: 1

word: "Younes"
num: 1

word: "and"
num: 1

word: "Safia"
num: 1
```

Comand: python3 client.py job run-wordcount http://localhost:8000/fichero1.txt http://localhost:8000/fichero200.txt

```
--->Run wordcount
Warnig...Empty address:  http://localhost:8000/fichero200.txt
```

Younes Kabiri
Safia Guellil

Younes Kabiri
Safia Guellil

Conclusiones

Esta práctica nos ha ayudado a entender las diferencias entre la comunicación indirecta y la comunicación directa. Hemos podido observar que podemos **escalar** (aumentar el número de Workers), grácias a la comunicación indirecta entre el máster y los workers.