Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Practica 3

Tolerancia a fallos en Servidores Sin Estado 30221 - Sistemas Distribuidos

Aarón Ibáñez Espés, 779088
Ángel Espinosa Gonzalo 775750

Índice

1. Resumen

Esta práctica tiene como objetivo el diseño y la implementación de un sistema distribuido con técnicas de tolerancia a fallos y RPC para la ejecución de la operación FindPrimes. En ella se parte sobre el código de la primera práctica en la que se implementó la arquitectura master-worker, la cual se modifica para implementar la comunicación usando RPC.
Finalmente, se ha diseñado una arquitectura master-worker elástica, de manera que el número de workers se modifique de forma automática en tiempo de ejecución.

2. Análisis y clasificación de los fallos que pueden aparecer

En la practica se encuentran 3 tipos de fallos de naturalezas diferentes:

- **Fallos por delay:** Este tipo de fallos ocurren cuando el worker no responde en un tiempo superior a 3 segundos, umbral del QoS.
- **Fallos por omisión:** De la misma forma que los fallos de delay, estos ocurren cuando el worker no contesta en menos de 3 segundos.
- **Fallos por crash:** Estos fallos suceden cuando ha sudedido un error en el worker y se had etenido su ejecucion, en estos casos se devuielve un error.

3. Estrategias de detección para cada uno de los fallos

Para la detección de los fallos se han seguido los siguientes métodos:

- **Fallos por delay u omisión:** Ambos fallos se han tratado de la misma manera, ya que en ambos el tiempo de respuesta del servidor es superior al umbral establecido para el QoS de 3 segundos. Para ello, en el momento en el que se realiza la petición al worker, se establece un timer de 3 segundos y si este salta antes de haber recibido respuesta, significa que se ha dado este fallo.
- **Fallos por crash:** El fallo por crash se detecta en la función encargada de procesar las peticiones del worker, en el caso de que se devuelva un error por parte del worker, se asume que este se ha caído.

4. Estrategias de corrección para cada uno de los fallos

Para la corrección de los fallos se han seguido los siguientes métodos:

- **Fallos por delay u omisión:** Cuando se detecta que el worker no responde en mas de 3 segundos, se reintroduce la petición a la cola de peticiones, para que así esta pueda ser procesada de nuevo y el resultado sea devuelto a su cliente.
- **Fallos por crash:** Cunado se detecta que uno de los workers se ha caído, se envía la ip por un canal síncrono a una rutina que esta escuchando dicho canal y que se encarga de levantarlo.

5. Modificaciones arquitecturales respecto de la practica 1

5.0.1. Comunicación RPC

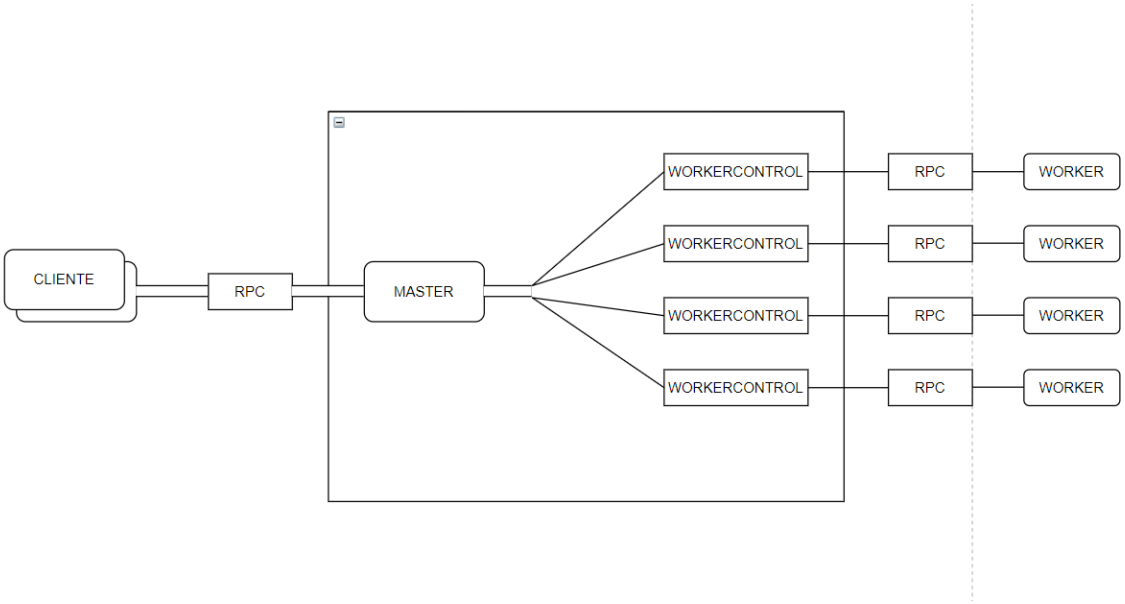

Figura 1: Diagrama de arquitectura máster-worker RPC

Como se ha mencionado anteriormente para el diseño de la arquitectura requerida se ha hecho uso de la arquitectura máster-worker planteada en la primera práctica.
A partir de la misma se ha utilizado el protocolo RPC para conectar al cliente con el máster y a su vez para conectar el máster con los workers.
El máster es el encargado de distribuir el trabajo entre los diferentes workers por medio de goroutines, las cuales están escuchando un canal donde se publican las peticiones.

5.0.2. Arquitectura RPC Elástica

Con respecto a las modificaciónes sugeridas en el enunciado, se ha decidido implementar una arquitectura elástica la cual es capaz de añadir o elimianr workers del sistema en funcion de la carga de trabajo.

Para evaluar la carga de trabajo y si se requiere lanzar o eliminar workers, se monitoriza la longitud del canal donde se añaden las peticiones.
Si el numero de peticiones en el canal es 0 se elimina un worker, ya que se considera que el numero de peticiones restantes por atender se pueden atender garantizando el QoS con un worker menos, en caso de que el numero de peticiones en el canal sea mayor que 0 se entiende que el numero de workers activos no son suficientes para sartisfacer la demanda por lo que se levanta un worker.

La arquitectura final queda de la siguiente manera:

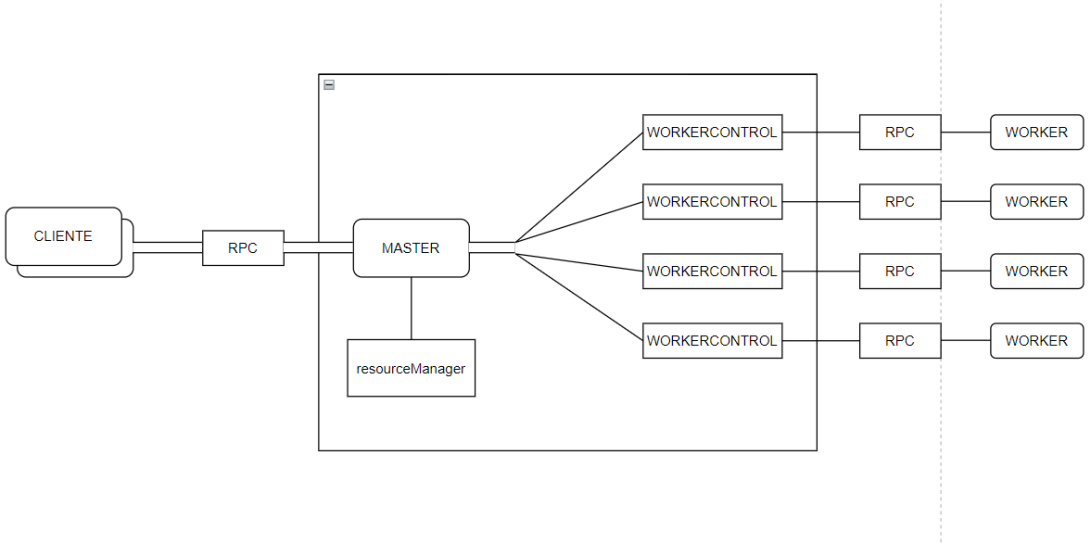

Figura 2: Diagrama de arquitectura máster-worker RPC elástica

6. Validación experimental

La gráfica que se muestra a continuación representa la carga de trabajo del servidor durante 6,30 minutos.

Para realizar la prueba se creado un script, el cual lanza un cliente cada 9 segundos hasta completar un total de 10 clientes conectados simultáneamente.

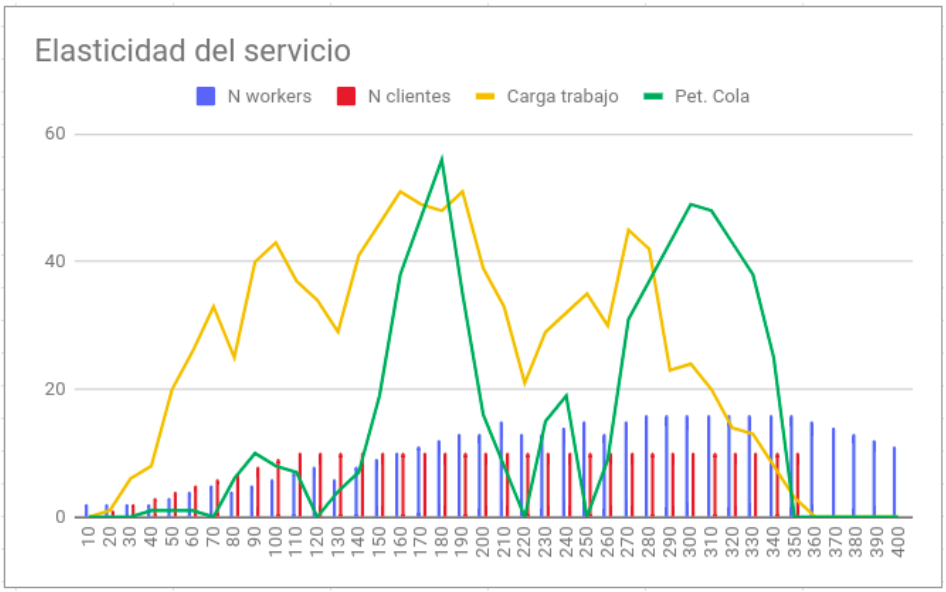

Figura 3: Estado del servidor

Como se puede observar en la gráfica, el numero de workers comienza a aumentar cuando se empiezan a acumular peticiones sin atender en el canal (línea verde). Cuando la rutina de monitorización detecta que el tamaño de la cola es superior a 1 añade un worker y cuando es igual a 0 elimina un worker.

Cabe destacar que en la gráfica aparecen momentos en los que el numero de workers disminuye pese a haber peticiones encoladas, esto es debido a que se ha dado un fallo de crash en uno de los workers y el sistema esta intentando levantarlo de nuevo.

7. Conclusiones

En esta practica se ha observado como gracias a la tolerancia a fallos y a la implementación de una arquitectura elástica se pueden aprovechar de forma mas eficiente los recursos del servidor, así como ajustar la potencia de procesado necesaria en función de la carga de trabajo en cada momento.