SISTEMAS DISTRIBUIDOS

Memoria práctica 3

Sergio García Esteban 755844
Irene Fumanal Lacoma 758325

Análisis de la carga de trabajo

En el caso peor, el cliente envía 8 tareas cada 2 segundos. En un escenario sin fallos y utilizando fibonacci_tr, 8 workers serían suficientes para que no se secuencialicen las tareas. Al diseñar una solución con réplica de tareas, el número de workers necesarios se multiplica por el número de réplicas que envíes.

Análisis de los fallos posibles

El código del worker suministrado, genera fallos aleatorios.

La probabilidad de que, al llegar una tarea, la operación que se ejecuta en ese worker cambie es del 29,1%. Pero el cambio no afectará hasta la siguiente tarea que reciba.

Operación	Probabilidad
System.halt	3,2%
Fib.fibonacci	5,8%
Fib.of	5,8%
Fib.fibonacci_tr	14,3%
Total	29,1%

Además, hay una probabilidad de 5,8% de que no envíe ninguna solución de la tarea actual.

¿Cómo afectan estos fallos a nuestro sistema distribuido?

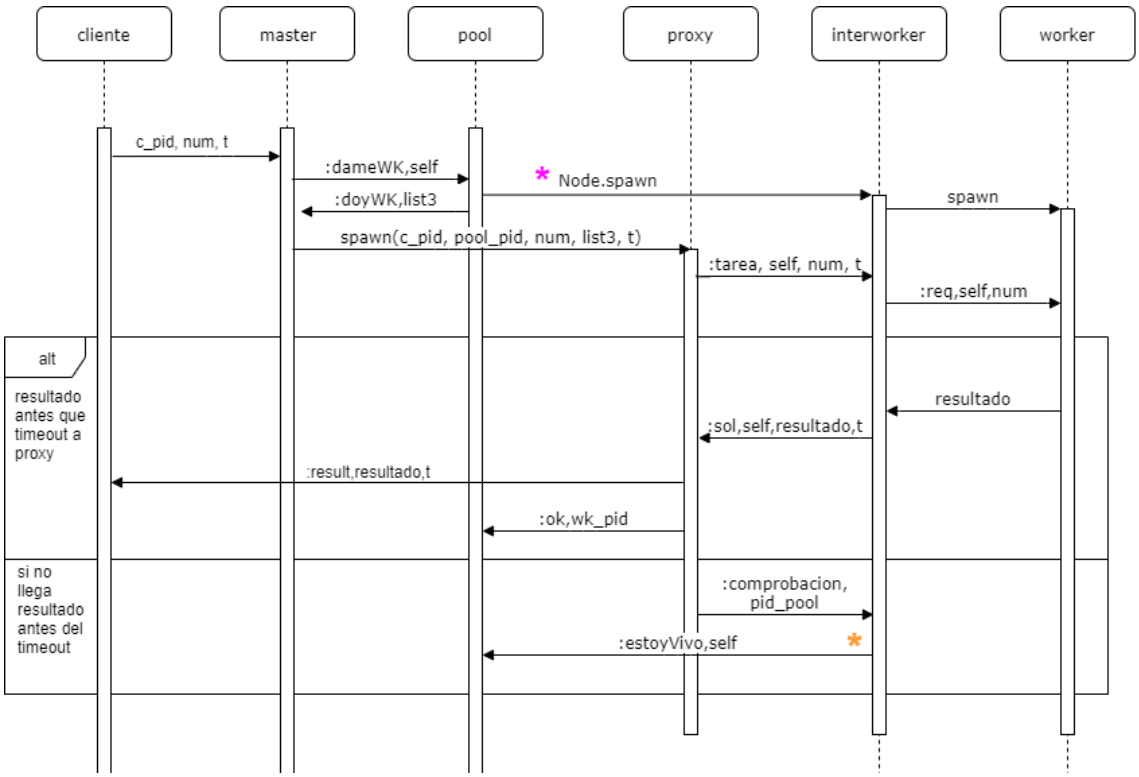
Si un worker cambia su operación a System.halt, en la siguiente tarea el worker muere.
Es un error tipo **crash**.

Si un worker cambia su operación a Fib.fibonacci, en la siguiente tarea calculará el resultado a enviar con esta operación, si el número es menor o igual a 36 dará un resultado cumpliendo el QoS, en otro caso tardará mucho más tiempo. Es un error tipo **timing**.

Si un worker cambia su operación a Fib.of, en la siguiente tarea calculará el resultado a enviar con esta operación, si el número es menor o igual a 100 dará un resultado, aunque en tipo float, cumpliendo el QoS. Es un error tipo **response**. Pero si el número es 1500 se produce una excepción aritmética y el worker muere. Es un error tipo **crash**.

Si un worker no envía ninguna solución, es un error tipo **omission**.

Detección y corrección de los fallos producidos



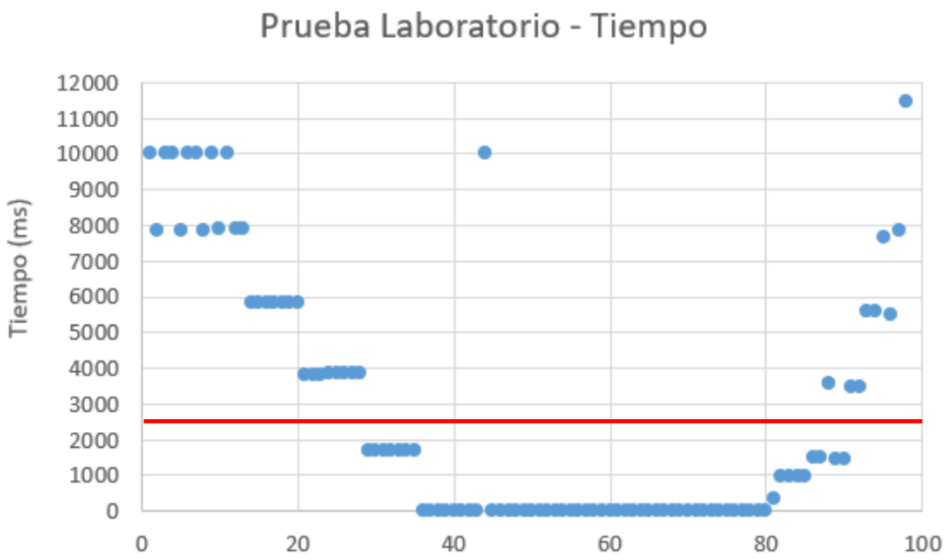
Al recibir una tarea en el servidor (máster), éste pide al Pool 3 workers que ejecutarán la tarea. El pool guarda 2 listas, una con nodos en los que no han sido lanzado ningún proceso y otra con los pids de los workers esperando a recibir una tarea (*).

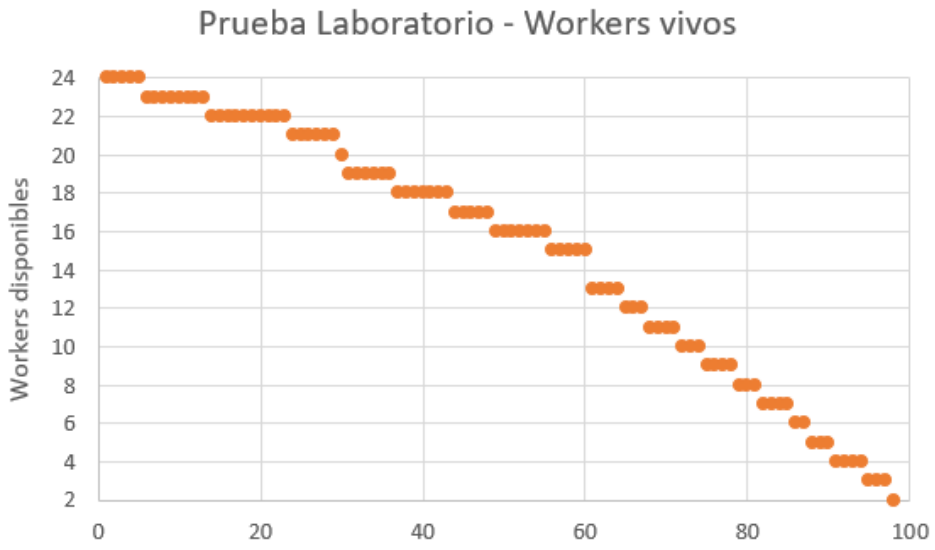
El número de workers por tarea configurado es 3, permite con bajo coste obtener una probabilidad de fallo igual a la probabilidad de que fallen los 3 workers.

La lista de los workers (list3) se envía a un proceso nuevo (proxy). El proxy se encarga de mandar las tareas a los workers. En cuanto llegue una solución la tratamos (truncando el resultado) y se la envía al cliente. Si las otras 2 soluciones llegan, se le comunicará al pool. Si vence el *timeout,* se enviará un mensaje al worker para comprobar si es error crash o error timing/omission (*). Si es un error del segundo tipo, enviará un mensaje al pool para indicarle que puede recibir más tareas.

El *timeout* configurado es de 2300 milisegundos, suficiente para detectar todas las soluciones que cumplen el QoS, basado en nuestras pruebas.

Pruebas en Laboratorio





En esta prueba se ha cumplido el QoS en el 65% de las tareas, hemos utilizado 24 workers.

Las primeras 25 tareas no cumplen el QoS porque nuestro sistema no lanza la ejecución de un worker hasta que sea necesario. El worker tiene un sleep de 10 segundos y por lo tanto, cada tarea cuyos workers sean nuevos, no cumple el QoS.

Las últimas 10 tareas tampoco cumplen el QoS, esto se debe a la cantidad de workers disponibles, que van disminuyendo con cada error tipo crash.

El final de la prueba se produce cuando la cantidad de workers es menor a 3.