SISTEMAS DISTRIBUIDOS

Práctica 3

Curso 2018-2019

Autores:
Ignacio Palacios Gracia 739359
Rubén Rodríguez Esteban 737215

1. Introducción

En esta práctica se va a trabajar de la siguiente forma, dado el intervalo [1, 1.000.000], se va a proceder a encontrar los pares de números amigos contenidos en él. Para ello, se utilizará una arquitectura de tipo Master-Worker. Sin embargo, la ejecución de los workers está sujeta a la presencia de fallos de tipo crash, omission y timing. Por ello, se ha utilizado el Algoritmo 1 proporcionado por los profesores de la asignatura para evitar fallos a la hora de pérdida de nodos.

2. Solución planteada

Para llevar a cabo la implementación de este chat se ha utilizado el algoritmo se ha empleado la arquitectura master-worker como solución. En el sistema interviene un proceso cliente que pide el cálculo de los números amigos en el intervalo [1..1000000]. Dicha petición es enviada a un proceso master, encargado de recibirla y de transmitirla a un proceso proxy.

El proceso proxy es el encargado de recibir las peticiones del máster y de enviarselas a los distintos workers. Concretamente existen tres workers que trabajan de acuerdo a tres protocolos distintos:

- El tipo de worker 1 calcula los divisores del número proporcionado, y devuelve al proxy una lista con todos ellos.
- El tipo de worker 2 calcula los divisores del número proporcionado, los suma, y devuelve esa suma al proxy.
- El tipo de worker 3 suma toda una lista de divisores proporcionada por el proxy, y le devuelve a éste la suma de la lista.

Mientras los workers trabajan o envían, el proxy se quedará esperando a su respuesta. Al llegar la respuesta del worker, la envía al máster.

La tolerancia a fallos se ha gestionado de manera que cuando el proxy no recibe la respuesta a la petición enviada antes de que expire el timeout un determinado número de veces, significa que el worker al que le ha enviado la petición ha caído por completo o temporalmente. Ante esta situación el proxy enviará al máster un mensaje de error (timeout).

En este contexto pueden ocurrir dos posibles sucesos, por un lado que el worker al que le mandó la petición haya caído por completo, es decir, que ha experimentado un fallo irreversible, o que el nodo se despierte porque se había quedado dormido temporalmente, recoja la petición que tenía en espera y la atienda, procediendo posteriormente a mandarsela al proxy.

Si el worker llega a mandar el mensaje pasado un tiempo, el proxy detectará que este worker estaba dormido (se comprueba mediante índices pasados entre el máster y el proxy, que se compara con el índice que se le pasó al worker cuando recibió la petición), mandará un mensaje al máster indicando que el worker ya se ha despertado, para poder volver a enviarle mensajes.

El proceso master es el encargado de recibir la petición del cliente para calcular las parejas de números de amigos del intervalo, y de enviar al proxy el número o lista de divisores, junto con el pid del worker que recibirá la petición. En el máster se encuentran las listas de los workers de cada tipo a conectar, una lista para cada uno de los tipos. Si el proxy manda un mensaje de que el worker ha superado el tiempo de timeout, el master borrará ese worker de la lista correspondiente, pero si recibe del proxy la confirmación de que el worker ya ha despertado, la vuelve a añadir a la lista.

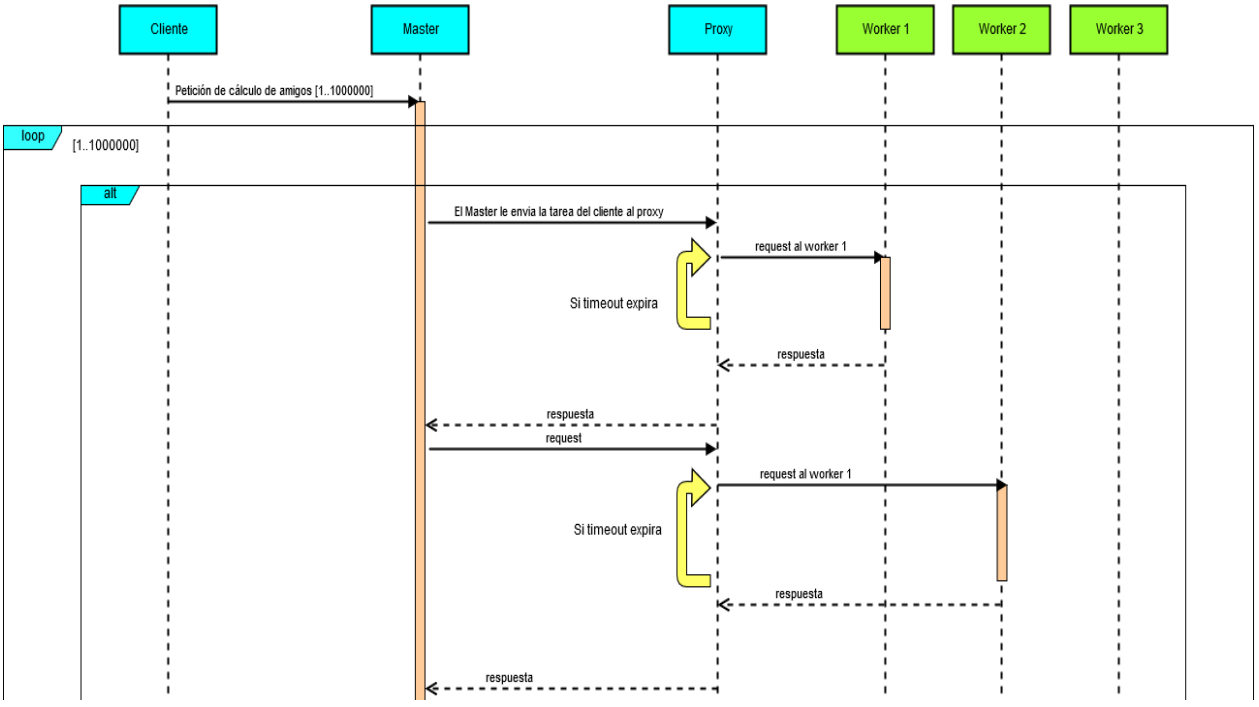
El master también contiene dos lista importantes más. Una de ellas almacena todas las sumas calculadas, colocadas de tal forma que el número x tiene su suma en la posición x-1 de la lista. La otra lista guarda todos los pares de números amigos encontrados.

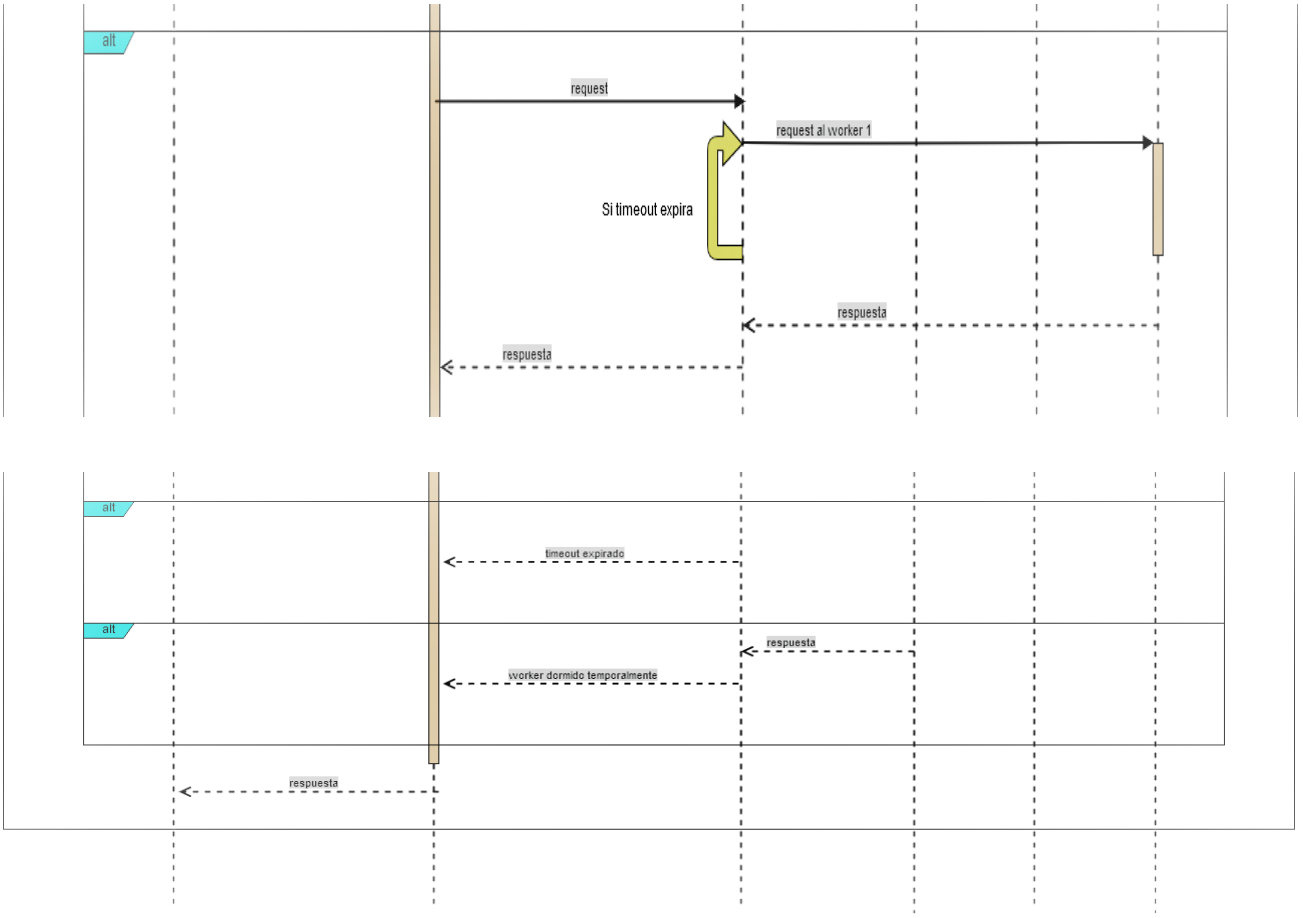
El envío de mensajes se maneja según dos tipos de protocolos de mensajes:

- El primer pide a los workers de tipos 1 y 3 que calculen la suma de divisores del número proporcionado, buscando luego si ese número tiene número amigo en las sumas ya calculadas. Tanto si tiene como si no, se introduce la suma en la posición correspondiente de la lista.
- El segundo pide al worker 2 que calcule la suma de los divisores del número proporcionado, y luego realiza el mismo procedimiento que la segunda parte del protocolo 1.

Si el proxy llega a enviar un mensaje de error de un worker por timeout, el master eliminará a ese worker de la lista hasta que se despierte, y además cambiará de protocolo. Por ejemplo, si está en el primer protocolo, y el worker 1 o 3 fallan, el master cambiará de protocolo, y dirá al proxy que mande mensajes al worker 2.

En el diagrama de secuencia mostrado a continuación se describe la sincronización y el paso de mensajes entre los diferentes procesos:





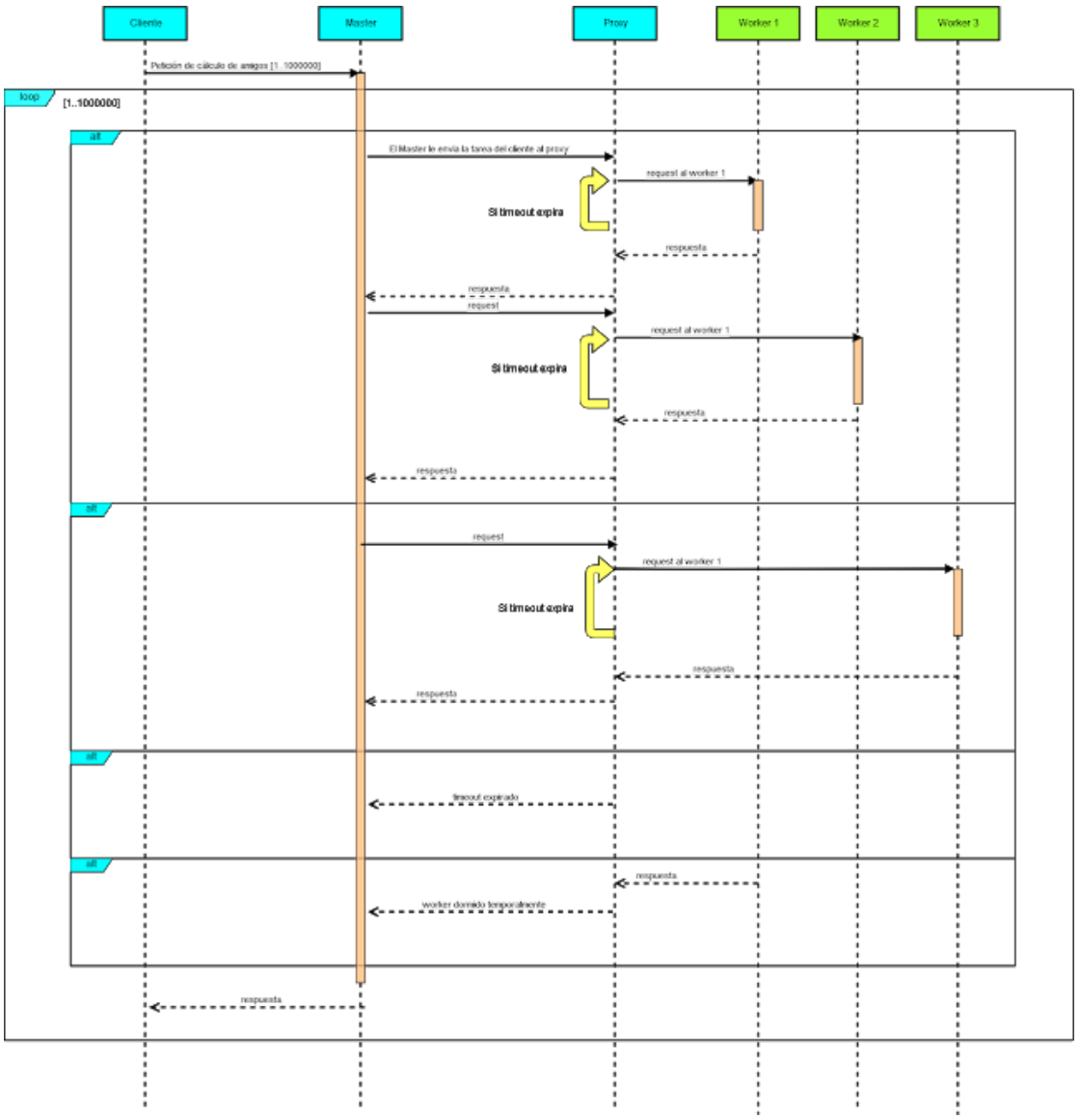
4. Validación

Para probar el correcto funcionamiento del sistema se ha probado el programa con números diferentes de nodos, llegando hasta poner 9 workers diferentes (3 de cada tipo) mas los tres principales (cliente, master y proxy). Primero se probó que funcionaba correctamente con workers sin fallos, y cada vez se fue probando a más (con retraso, con omisión, y al final, con crash). También se redujo el intervalo de [1..1000000] ya que el tiempo en realizar la tarea sería muy elevado.

5. Conclusión

A lo largo de esta práctica se ha aprendido a implementar el Algoritmo 1 proporcionado por los profesores de la asignatura. Este algoritmo se ha implementado mediante un cliente, un master, un proxy y un worker que interaccionan entre sí. En el master se guardan tanto la suma de todos los divisores de los números, como las parejas de números amigos encontradas. Además, existen dos tipos de interacción con los workers mediante el proxy, las cuales se van intercambiando entre sí cuando falle algún workers de ese protocolo. El proxy enviará tres tipos de mensajes al master, una confirmación con una lista o la suma resultado, un error de timeout del worker, y una confirmación de que un worker ha despertado y se le pueden enviar mensajes de nuevo. El master actuará de una manera o de otra según el mensaje que le llegue. Una vez acabe el intervalo, se enviará una lista con los números amigos del intervalo al cliente.

6. Anexo


Cliente
Master
Proxy
Worker 1
Worker 2
Worker 3
Petición de cálculo de amigos [1..1000000]
loop
[1..1000000]
alt
El Master le envía la tarea del cliente al proxy
request al worker 1
Si timeout expira
respuesta
respuesta
request
request al worker 1
Si timeout expira
respuesta
respuesta
alt
request
request al worker 1
Si timeout expira
respuesta
respuesta
alt
timeout expirado
alt
respuesta
worker dormido temporalmente
respuesta