

Sistema Distribuido tolerante a fallos

Sergio Herrero Barco 698521

Alex Oarga Hategan 718123

Noviembre de 2017

1. Introducción

El sistema diseñado consiste en un Sistema Distribuido tolerante a fallos, que permita encontrar todos los números amigos de un intervalo. Dos números a y b se dicen que son amigos si y solo si la suma de los divisores de " a " es igual a " b ", y la suma de los divisores de " b " es igual a " a ". Para ello se va a usar una arquitectura Master-Worker en la que se va a implementar un algoritmo de detección de fallos.

2. Arquitectura

Se han creado tres ficheros en los que esta implementado dicho sistema. Un fichero "amigos.exs" que contiene las operaciones relacionadas con los números, otro fichero "worker.exs" que contiene el módulo de los worker, y por ultimo un fichero "main.exs" que contiene el modulo del master.

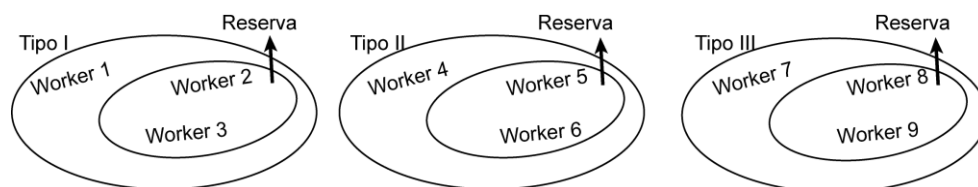
2.1. Arquitectura del Sistema

El sistema se compone de una serie de nodos que van a actuar siguiendo una Arquitectura Master-Worker. Dicha arquitectura consiste en un reparto de tareas, el master solicita operaciones al worker y este se las devuelve resueltas.

Existen 3 tipos de workers:

1. Los Worker de tipo 1 resuelven la tarea de encontrar los divisores de un número.
2. Los Worker de tipo 2 resuelven la tarea de sumar la lista de divisores de un número.
3. Los Worker de tipo 3 resuelven la tarea de sumar los números de una lista.

Cada tipo de worker **alberga 3 réplicas**, siendo una la líder, con la misma funcionalidad y así poder implementar un sistema de detección de fallos.

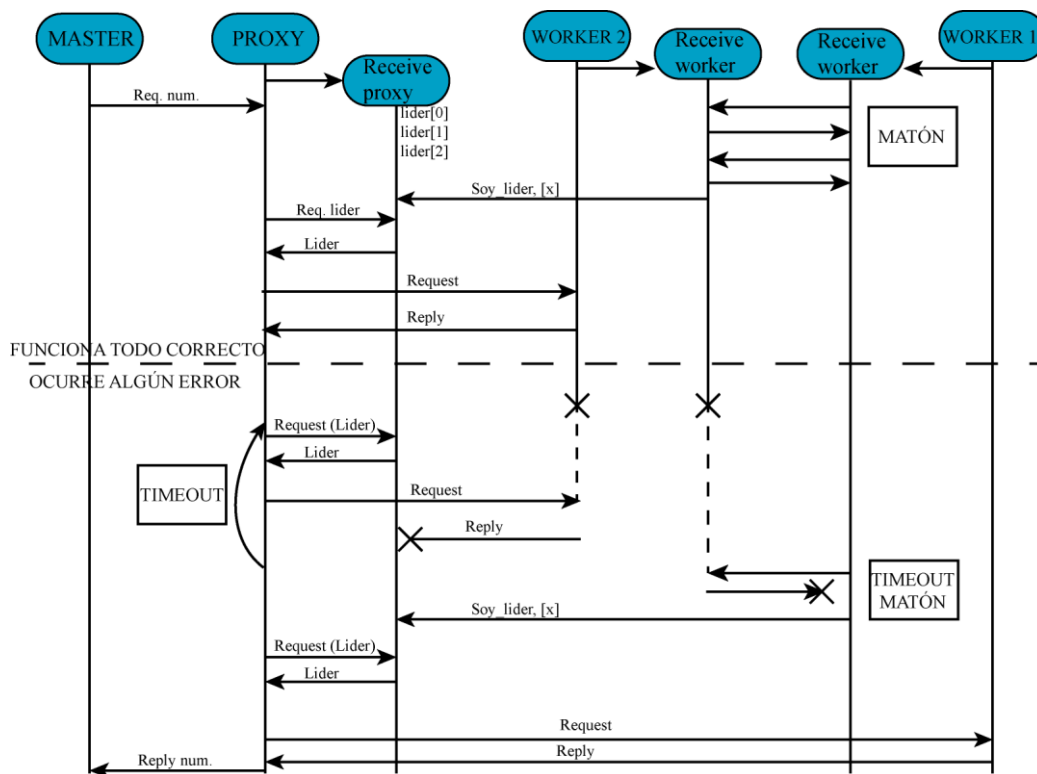


El **master** no se comunica directamente con los worker, sino que tiene implementado otra arquitectura Master-Worker que reparte la tarea entre dos procesos llamados **proxy**, y estos son los encargados de enviar las peticiones a los diferentes workers.

El **proxy** realizara peticiones a los worker y activara un temporizador esperando la respuesta. Si la respuesta no llega en un tiempo, vuelve a realizar la petición.

Los **worker** a la vez que responden a las peticiones que le llegan, tiene implementado el algoritmo del **matón**. Si en algún momento el worker lider de un tipo sufre un fallo, se activa una encuesta para elegir el nuevo líder.

2.2. Protocolo de comunicación



3. Tolerancia a fallos

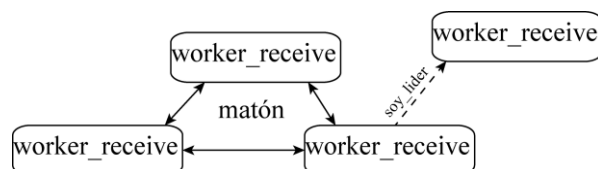
3.1. Master

El master tiene dos factores para tratar la tolerancia a fallos. El primero es la implementación del algoritmo 1 de tolerancia, que consiste en tener un servicio proxy que se comunica con los worker. Este le envía una petición, y si pasado un tiempo no contesta, vuelve a enviar la petición. Si tras 5 intentos no se recibe respuesta, el proxy envía un mensaje al master indicándole que el worker no funciona.

Además del algoritmo, el master envía primero la tarea al worker 2, y si este falla, le envía la petición al worker 1 y al worker 3. Siempre se le da preferencia al worker 2 ya que es el único que realiza la operación completa.

3.2. Worker

El worker implementa el algoritmo del matón. Dicho algoritmo consiste en la detección de que un nodo “líder” sufre una caída, y la elección de un nuevo líder que se ocupe de la funcionalidad. Tras la elección de un nuevo líder, este es el encargado de avisar al **proxy** de que es el nuevo líder y es el que va a realizar la funcionalidad.



4. Validación experimental

5. Conclusión