Buscador Distribuido

Eric Luis López Tornas Marco Antonio Ochil Trujillo

July 18, 2024

Abstract

Para la implementación del sistema distribuido nos basamos en ideas recibidas a lo largo del curso, donde algunas recibieron modificaciones de mayor significación, pero pretendemos establecer las bases de los contenidos que utilizamos. Para la arquitectura pensamos en *Micro-servicios* al principio, donde los cambios se debieron principalmente a la problemática a resolver, para su creación utilizamos el anillo de *Chord* y los *Hilos* como medio concurrente, así como una combinación de *RPC* y *Objetos remotos* para la comunicación. Con respecto a coordinación utilizamos el algoritmo *Bully* para la elección de líder, pero no trabajamos con algoritmos de sincronización de tiempo. Como ya se mencionó anteriormente se utilizó *DHT*, además de *Broadcast* para autodescubrimiento entre los nodos servidores, así como en la comunicación entre los nodos clientes y el líder de la red. En el caso de la replicación trabajamos con la *Escritura Remota* como principio realizando replicas en el antecesor y sucesor de cada nodo de nuestro *DHT*.

Contents

Chapter 1

Problemática

Buscador de documentos distribuídos

Los buscadores son programas o aplicaciones que colectan informacion que, segun sea el caso, puede ser de manera local, en la red o en internet. Una vez colectada esta informacion es procesada y almacenada inteligentemente dentro de una base de datos a la cual puede acceder el buscador. A partir de esta base de datos entonces, desde el buscador y utilizando unos terminos de busqueda, se pueden recuperar los documentos que contengan informacion relevante y relacionada con lo que se busca. Existen buscadores para acceder a informacion en Internet o de manera local.

Especificaciones

El proyecto que se propone tiene como proposito implementar un buscador de archivos distribuidos en varias computadoras localizando los documentos a partir de un patron de busqueda y permitiendo el acceso a los documentos identificados. A diferencia de un buscador tradicional, en el buscador que se proyecta, cualquier computadora que se incorpore al sistema tendra la capacidad de proveer documentos que puedan ser localizados y accedidos desde otras maquinas del sistema. De igual manera desde cualquier maquina se podran realizar las busquedas. El sistema debera estar preparado para que la salida de una maquina del sistema, en cualquier momento (por ejemplo mientras se esta accediendo a ella) no cause fallas o inconsistencias en el sistema.

Recomendaciones

El sistema que se proponga debe poseer las sisguientes cararecterísticas:

• Arquitectura Distribuida.

• En cada computadora corre un cliente del sistema desde el cual se pueden buscar ficheros presentes en el sistema ası como permite que se pueda acceder, a partir de busquedas, a los ficheros que desde la computadora se comparten.

• Para la busqueda como mınimo es imprescindible tener en cuenta los nombres de los ficheros y sus tipos.

• Es necesario tener en cuenta que un fichero localizado mediante una busqueda puede no estar disponible para su acceso o pueden ocurrir errores en el proceso de transferencia.

• En el sistema pueden existir ficheros duplicados con diferente nombre siendo los mismos ficheros. En este caso el sistema debe proveer una estrategia de seleccion que logre la mayor eficiencia posible en la transferencia del fichero buscado.

• El proceso de busqueda debe utilizar mecanismos lo mas eficiente posibles con el objetivo de reducir el tiempo de respuesta a los usuarios.

Chapter 2

Arquitectura

Inicialmente se planteo desarrollar esta arquitectura, dado las posibilidades de separar problemas entre distintos nodos que realizaran tareas de busqueda, operadores específicos en la busqueda, actualización de embeddings para el cálculo de consultas, así como el intercambio CRUD con el cliente. Esta idea fue modificándose debido a la restricción del problema donde cada nodo debía ser capaz de interactuar por si solo con todas las operaciones que necesitaba el cliente, lo que vio afectada la separación de tareas que se pensaba desarrollar. Otra problemática encontrada fue la complejidad que añadía aumentar demasiado la cantidad de clases en nuestro programa, situación que fue consultada con los profesores y comprendida para disminuir la dificultad de la implementación.

En la estructura actual cada nodo realiza operaciones CRUD sobre los documentos que contiene, además de actualizaciones en su embedding particular para la realización de consultas, así como todo lo referente a comunicación interna para la selección de líder y chequeos de existencia de nodos predecesores. Todos los nodos actuales pertenecen a una misma clase donde el único que tiene funcionalidades diferentes es el escogido como líder, el cual se encarga de la comunicación con los clientes.

Chapter 3

Comunicación

La implementación proporcionada de Chord nos ayudó a manejar las referencias a sucesores y predecesores en cada momento y tener un cómodo panorama de comunicación entre los objetos contenidos en cada nodo, combinando envios referenciales con informaciones en forma de texto.

Chapter 4

Coordinación

Se realizó una implementación del algoritmo Bully donde cada nodo al llegar a la red cuenta con un objeto que solicita elecciones y que almacena el ip del ganador, este ganador es el único que responde a los clientes, los cuales utilizan Broadcast para comunicarse con la red. Como el cliente se comunica por Broadcast, todos los nodos escuchan su llamado pero solo el líder es quien contesta a sus peticiones por lo que la caída del lider no afecta la comunicación del cliente, ya que cada poco tiempo se hacen reelecciones para volver a escoger alguien que responda a los clientes.

No estamos utilizando algoritmos de sincronización de tiempo, comprendemos que esto trae deficiencias principalmente en la edición de archivos en nuestro problema, pero no contabamos con el tiempo suficiente para completar dicha implementación.

Chapter 5

Nombrado y Localización

Estamos utilizando tanto DHT (Anillo de Chord) como Broadcast. El anillo es utilizado para casi todo lo referente a intercambio de información entre servidores, así como para mantener la consistencia de la red con respecto a orden de nodos y manipulación de documentos, en nuestro anillo no culminamos la implementación de la tabla de fingers dado que nos presentó diversas dificultades en su actualización ante la caída de nodos de la red. En el caso del Broadcast se utiliza principalmente en funciones de descubrimiento como en la llegada de un nuevo servidor a la red, así como en la interacción del cliente con el nodo líder de la red; además de estas situaciones se necesita en la selección de líder para mantener la elección abierta a la participación de todos los nodos y en la resolución de consultas, en las cuales el líder envia la información a buscar a todos los nodos de la red y luego se encarga de combinar las respuestas acumuladas.

Chapter 6

Consistencia y Replicación

Nos basamos en el criterio de Escritura Remota para mantener la información en el sucesor y predecesor de cada nodo, utilizando distintas tablas para cada información. El cambio que realizamos es la actualización de la información en el nodo propietario de la información proseguida de la actualización en sus replicas. Las replicaciones provocadas por operaciones con los clientes fueron modeladas principalmente por el líder que está encargado de encontrar al propietario de la información y enviar otros datos pertinentes. Otras operaciones provocadas por la caída o inclusión de documentos fueron realizadas en la implementación de la función *notify* pues la encontramos como función necesaria para resolver cada uno de los casos tratados.

Chapter 7

Exposición

Contamos con un visual en la biblioteca Streamlit de Python para mostrar con mayor claridad el funcionamiento de nuestro proyecto. Esperamos que sea de su agrado.