Scrapper Distribuido 💻

Autores ✒️

Nombre(s) y Apellidos	Grupo	Correo	GitHub
Reinaldo Barrera Travieso	C-411	r.barrera@estudiantes.matcom.uh.cu	@Reinaldo14
Ariel Plasencia Díaz	C-412	a.plasencia@estudiantes.matcom.uh.cu	@ArielXL

Empezando 🔧

Implementación

El proyecto está implementado en python 3.6.8. Para una mejor y mayor comprensión del código fuente propuesto entendemos que hay que tener profundos conocimientos acerca de python como lenguaje de programación. Nos apoyamos fundamentalmente en las librerías threading, socket, BeautifulSoup y urllib para su implementación.

Para la instalación de las dependencias ejecutamos el siguiente comando:

```
pip install -r requirements.txt
```

Ejecución

Para ejecutar nuestro proyecto de manera más sencilla proveemos un makefile con varias opciones. A continuación mostramos la ayuda.

```
cd src/
make help
info                        Display project description
version                     Show the project version
server                      Run a server with default parameters
client                      Run a client with default parameters
install                     Install the project dependencies
clean                       Remove temporary files
help                        Show this help
```

Para personalizar los parámetros de entrada consulte la documentación del fichero a ejecutar escribiendo las siguientes líneas:

```
cd src/
python server.py --help
python client.py --help
```

Sobre el Scrapper Distribuido 🕸

Chord

Chord es un protocolo de búsqueda distribuida que se puede utilizar para compartir archivos peer to peer (p2p). Chord distribuye objetos a través de una red dinámica de nodos e implementa un protocolo para encontrar estos objetos una vez que se han colocado en la red. La ubicación de los datos se implementa en la parte superior de Chord asociando una clave con cada elemento de datos y almacenando el par clave, elemento de datos en el nodo al que se asigna la clave. Cada nodo de esta red es un servidor capaz de buscar claves para aplicaciones clientes, pero también participa como almacén de claves. Además, se adapta de manera eficiente a medida que los nodos se unen y abandonan el sistema, y puede responder a consultas incluso si el sistema cambia continuamente. Por lo tanto, Chord es un sistema descentralizado en el que ningún nodo en particular es necesariamente un cuello de botella de rendimiento o un único punto de fallas.

Llaves

Cada clave insertada en la tabla de hash distribuida (DHT por sus siglas en inglés) tiene un hash para que quepa en el espacio de claves admitido por la implementación particular de Chord. El espacio de claves, en esta implementación, reside entre 0 y 2**m-1, donde m = 100 (indicado por MAX_BITS en el código).

Anillos de Nodos

Así como cada clave que se inserta en el DHT tiene un valor hash, cada nodo del sistema también tiene un valor hash en el espacio de claves del DHT. Para obtener este valor de hash, simplemente usamos el hash de la combinación ip:puerto, usando el mismo algoritmo de hash que usamos para las claves de hash insertadas en el DHT. Chord ordena el nodo de forma circular, en la que el sucesor de cada nodo es el nodo con el siguiente hash más alto. El nodo con el hash más grande, sin embargo, tiene el nodo con el
hash más pequeño como su sucesor. Es fácil imaginar los nodos colocados en un anillo, donde el sucesor de cada nodo es el nodo que le sigue cuando sigue una rotación en el sentido de las agujas del reloj.

Eficiencia y Escalabilidad

Chord está diseñado para ser altamente escalable, es decir, para que los cambios en las dimensiones de la red no afecten significativamente a su rendimiento. En particular, si n es el número de nodos de la red, su costo es proporcional a log(n). Es escalable porque solo depende del número de bits del que se compone un identificador. Si queremos más nodos, simplemente asignamos identificadores más largos. Es eficiente, porque hace búsquedas en un orden log(n), ya que en cada salto, se puede reducir a la mitad el número de saltos que quedan por hacer.

Resistencia a fallas

Chord admite la desconexión o falla desinformada de los nodos al hacer ping continuamente a su nodo sucesor. Al detectar un nodo desconectado, el anillo de nodos se estabilizará automáticamente. Los archivos en la red también se replican en el nodo sucesor, por lo que en caso de que un nodo falle, otro nodo se encarga de él, este último nodo será redirigido a su sucesor.

Web Scrapping

Web scrapping es una técnica utilizada mediante programas de software para extraer información de sitios web. Usualmente, estos programas simulan la navegación de un humano en la World Wide Web ya sea utilizando el protocolo HTTP manualmente, o incrustando un navegador en una aplicación. El web scrapping está muy relacionado con la indexación de la web, la cual indexa la información de la web utilizando un robot y es una técnica universal adoptada por la mayoría de los motores de búsqueda. Sin embargo, el web scrapping se enfoca más en la transformación de datos sin estructura en la web (como el formato HTML) en datos estructurados que pueden ser almacenados y analizados en una base de datos central, en una hoja de cálculo o en alguna otra fuente de almacenamiento. Alguno de los usos del web scrapping son la comparación de precios en tiendas, la monitorización de datos relacionados con el clima de cierta región y la detección de cambios y la integración de datos en sitios webs. En los últimos años el web scrapping se ha convertido en una técnica muy utilizada dentro del sector del posicionamiento web gracias a su capacidad de generar grandes cantidades de datos para crear contenidos de calidad.

En nuestro caso para esta funcionalidad utilizamos las librerías BeautifulSoup y urllib provistas por el lenguaje de programación python. Con ellas podemos extraer de manera sencilla el texto html de una determinada url, así como sus fotos y demás archivos relacionados. Cabe mencionar que el web scrapping se lleva a cabo por niveles de profundidad, variable especificada en la entrada del algoritmo.

Sistema Distribuido

El sistema distribuido está constituido por nodos los cuales tienen ciertas responsabilidades o roles. La existencia de un único rol para cada nodo no impide que dos nodos no puedan estar en un mismo hostname. El sistema está disponible siempre que halla algún nodo disponible (eso no implica que deba dar una respuesta rápida ante los pedidos). En caso de que algún nodo abandone la red, ya sea tanto informada como desinformadamente, no se pierden los datos, éstos son replicados por los restantes nodos existentes en la red. También es capaz de reconectarse y funcionar como un solo sistema cuando exista la posibilidad de comunicación entre los nodos de los subsistemas.

En nuestro proyecto cada nodo tiene las siguientes funcionalidades:

1. *Join network*: Se une a otro nodo según IP y PORT especificados.
2. *Leave network*: Deja la red de manera informada y replica los archivos que contiene el nodo.
3. *Print finger table*: Imprime la finger table del nodo.
4. *Print my predecessor and successor*: Imprime tanto el ID del nodo actual como su predecesor y sucesor.
5. *Print IP, PORT and LEVEL*: Imprime tanto el ID, IP, PORT y LEVEL (nivel de profundidad) del nodo actual.
6. *Print files on the network*: Imprime los archivos contenidos en el nodo.
7. *Make web scrapping*: Dada una url y un nivel de profundidad, se descargan todos los ficheros correspondientes a la url en la carpeta *src/downloads/urls/* con la profundidad especificada y se sube el html principal hacia la red.
8. *Upload file*: Se sube un archivo determinado y replicado a varios nodos en la red.
9. *Download file*: Se descarga el archivo especificado en la dirección *src/downloads/files/*.

Un ejemplo completo 📃

Con la idea de evidenciar el cumplimiento de los requisitos pedidos para nuestro proyectos mostramos varios ejemplos, en forma de videos, basados en un servidor y varios clientes, los cuales llevarán a cabo diferentes consultas al sistema distribuido de manera concurrente. Estos videostutoriales podemos encontrarlos en la ruta *doc/*.

License 📄

Referencias 🗃️

Para la implementación del proyecto nos apoyamos fundamentalmente en algunas consultas hechas en la red de redes y en el documento Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, este último nos sirvió de gran ayuda para una mejor y mayor comprensión e implementación del protocolo chord.