Tabla de Hash Distribuida

Autor

Nombre y Apellidos	Correo	GitHub
Ariel Plasencia Díaz	arielplasencia00@gmail.com	@ArielXL

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 $2^m - 1$ inclusive, donde $m = 10$ (indicado por MAX_BITS en el código). Entonces, el espacio de claves está entre 0 y 1023.

Anillo 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 coste 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.

Ejecución

Dado que se trata de un sistema descentralizado, no hay scripts de servidor y cliente separados. En su lugar, cada secuencia de comandos actúa como servidor y como cliente, lo que permite conexiones p2p a otros nodos. Cualquier nodo puede unirse a la red pero inicialmente debe conocer la ip y puerto de otro nodo que ya es parte de la red Chord.

Para ejecutar este proyecto escriba las siguientes líneas en una terminal abierta desde esta misma dirección:

```
cd src/
python chord.py 127.0.0.1 7000              # primer nodo
python chord.py 127.0.0.1 8000              # segundo nodo
python chord.py 127.0.0.1 9000              # tercer nodo
```

Tener en cuenta que un nodo puede tener la misma dirección ip, pero no debe coincidir los puertos. Cuando comiencen los nodos, se le mostrarán varias opciones:

1. *Entrar a la red*: Puede conectar un nodo a otro de cualquier forma. Una vez que elija unirse a la red, puede usar la combinación de ip y puerto de cualquier otro nodo para unirse a la red.
2. *Dejar la red*: Los nodos pueden salir de la red informando al algoritmo y, por tanto, a otros nodos de su salida. Chord también admite la desconexión o falla no informada de los nodos, sin embargo, se prefiere la salida informada.
3. *Imprimir la finger table*: Imprime a través de la terminal la finger table de este nodo.
4. *Imprimir id, predecesor y sucesor*: Imprime a través de la terminal tanto el id del nodo actual como su predecesor y sucesor.

Referencia

Para más información puede consultar el documento *Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications*, el cual, nos sirvió de gran ayuda para una mejor y mayor comprensión e implementación del protocolo.