Distribución de recursos
¿Centralizada o distribuida?
Implementación en Golang con Grpc

Javier Moreno Gormaz 201673038-9
Bárbara Uribe Cataldo 201673074-5

December 2020

1 Qué se hizo y cómo se hizo

Realizamos un ejemplo de lo que podría ser la red de una bibilioteca con sus existencias digitales repartidas en chunks, distribuidas en 3 DataNodes. Estos se comunican con el cliente y el NameNode para hacer posible la carga y descarga de libros, además de la escritura del Log donde aparece en qué DataNode se encuentra cada parte de cada libro cargado con anterioridad.

Para poder conformar los DataNodes y el NameNode utilizamos protocol buffers, de esta forma generamos archivos que facilitan la comunicación entre nodos.

En los protocol buffers definimos los tipos de datos que utilizamos y las funciones que los relacionan, para DataNode.proto y NameNode.proto definimos los siguientes tipos de datos a utilizar:

```
message File{
    bytes log = 1;
}

message Book {
    string name = 1;
    int32 parts = 2;
    repeated Chunk chunks = 3;
}

message Chunk{
    string name = 1;
    bytes data = 2;
}

message Message {
```

```
    string text = 1;
}

message Proposal {
    string ip = 1;
    Chunk chunk = 2;
}
```

Luego, definimos las funciones de cada servicio:

```
service DataNode {

    rpc SendBookInfo(Book) returns (Message){}
    rpc DistributionType(Message) returns (Message){}

    rpc UploadBook(stream Chunk) returns (Message) {}
    rpc DownloadBook(Chunk) returns (Chunk) {}

    rpc DistributeChunks(Chunk) returns (Message){}
    rpc RescateChunks(Message) returns (Proposal){}
    rpc SendProposal(stream Proposal) returns (Message) {}

}

service NameNode {

    rpc GetAvaibleBooks(Message) returns (Message){}
    rpc GetLocations(Message) returns (stream Proposal){}

    rpc GetDistribution(Message) returns (Message){}
    rpc GetBookInfo(Book) returns (Message) {}

    rpc SendProposal(stream Proposal) returns (stream Proposal) {}
    rpc GetChunkDistribution(Message) returns (stream Proposal) {}
}
```

Para finalizar, estas funciones fueron implementadas tanto en el servidor de su respectivo servicio como en sus clientes. Para detalles sobre la funcionalidad del programa favor leer el Readme.md ubicado en todas las máquinas virtuales.

2 Resultados

- Realizamos la comparación entre cargar el mismo libro a través de ambas distribuciones, resultando en que, en la centralizada, la carga de este demoraba un poco más que en la distribuida. Con una diferencia de aproximadamente $2,5$ [seg].

- Cantidad de mensajes:

  Para cargar un libro de forma centralizada se realizan $5n + 11$ mensajes.

  Para cargar un libro de forma distribuida realizamos $9n+13$ mensajes.

  Donde n es la cantidad de chunks en las que el libro se divide.

3 Análisis

El que se pueda observar una diferencia de tiempo en la carga de uno de los libros puede hacer notar que la distribución centralizada no es tan escalable como la distribuida, en términos de cantidad de libros a cargar, ya que obviamente se demorará más.

El que se pueda observar una diferencia de esta magnitud en los mensajes realizados nos sugiere que, la distribución centralizada es más escalable que la distribuida en términos de quién maneja mejor una mayor cantidad de nodos donde almacenar los chunks.

4 Discusión

Que la distribución centralizada demore más tiempo en cargar un solo ejemplar tiene sentido ya que se debe esperar que el NameNode revise la propuesta entregada por los DataNodes, cree una nueva si es que esta no es satisfactoria, escriba el Log para poder recién realizar la distribución de datos en sus respectivos DataNodes.

Por otro lado, que se realicen una mayor cantidad de mensajes en la distribución distribuida tiene sentido, ya que esta no solo debe concentrarse en enviar mensajes al NameNode y al cliente, sino que también se deben enviar mensajes entre los DataNodes, a mayor cantidad de DataNodes existentes mayor será el flujo de mensajes.

5 Conclusión

Dependiendo de las necesidades del sistema son útiles ambas distribuciones, necesitando un tipo diferente de escalabilidad en cada caso. En este caso específico conviene utilizar una distribución centralizada ya que no manejamos grandes volúmenes de datos.