TAREA 1: SISTEMAS DISTRIBUIDOS
Caché y RPC
Profesor: Nicolás Hidalgo
Ayudantes: Cristian Villavicencio, Joaquín Fernandez y Nicolás Núñez

LEA EL DOCUMENTO COMPLETO ANTES DE EMPEZAR A DESARROLLAR LA TAREA

Objetivo

El objetivo de esta tarea consiste en poner en práctica los conceptos de Caché y RPC. Para ello el alumno deberá hacer uso de Redis y gRPC que le permitan dar solución a la problemática planteada.

Conceptos previos

Web search engine: Es un sistema que utiliza como base un índice invertido que aloja los documentos de los diferentes sitios y palabras claves que lo agrupan. Los buscadores tradicionalmente poseen un *Front-end*, un sistema de índice y un apoyo al último a través de un sistema de caché. Su objetivo es entregar enlaces relacionados a ciertos documentos, palabras claves u otros mecanismos definidos por el motor de búsqueda.

Query logs: corresponden al registro de consultas o búsquedas realizadas en un *web search engine*. Contiene información que caracteriza distintas búsquedas tales como un usuario en concreto, hora, clicks, etc.

Problemática

Recientemente **Xoogle** ha caído en distintos problemas debido a que los sistemas de búsqueda trasgreden la privacidad de sus usuarios. Se ha puesto en evidencia la venta de la información que se recopila a través de las *Query Logs* generadas en un sistema de búsqueda. Debido a lo anterior, la universidad ha tomado como determinación crear un *web search engine* y lo han buscado a usted para desarrollarlo.

Como es de su conocimiento, los *Web search engine* son sistemas complejos que funcionan de manera distribuida para dar cabida a la alta carga que deben manejar. Por ello, es crucial contar con un sistema de cache que permita aliviar la carga del *backend* del buscador y con ello pueda lograr escalar en un futuro. Para apoyar al desarrollo del buscador, usted deberá implementar un sistema de caché e índice.

Resumen

El sistema que usted desarrolle deberá organizarse en las siguientes partes:

1. **Dataset**: se deberá elegir un dataset de algún *Query log*, que se encuentran en la siguiente lista:

   `http://www.cim.mcgill.ca/~dudek/206/Logs/AOL-user-ct-collection/`

2. **Bases de datos**: se deberá preparar una base de datos relacional a su elección y al menos tres instancias de *Redis*. A lo largo de la tarea usted necesitará una base de datos que le permita generar un índice con datos de distintas URL y datos importantes sobre estas búsquedas. Además, necesitará instanciar Redis para así minimizar las búsquedas de la base de datos. Lo anterior puede realizarse a través de contenedores Docker.

3. **Crawler**: del dataset elegido, se deberán desprender las distintas URL existentes y guardar la información que estas contengan en la base de datos, para así tener una indexación de distintas URLs. Esto se debe generar de manera automatizada con un *Script* para el dataset que usted elija. Se recomienda que usted genere los siguientes campos:

id	title	description	keywords	URL

4. **Backend para sistema de búsqueda**: se deberá crear un *backend* para el sistema de búsqueda, que permita buscar información de distintas URLs guardadas en la base de datos. Las reglas de búsqueda las propondrá usted, sin embargo, deberá justificarlas. Por otro lado, recuerde que este es sólamente el backend, es decir, la funcionalidad principal del sistema, sin embargo, este no se ejecutará a menos que un cliente se lo pida.

5. **Cliente para búsqueda**: se tendrá un cliente que permitirá realizar búsquedas para los distintos usuarios. Este cliente estará conectado con el *backend* o sistema de búsqueda generado en el punto 4, utilizando *gRPC*[1]. Por otro lado, también debe estar conectado al sistema de caché de *Redis*[2], que tratará de evitar que se generen búsquedas en el *backend* y es explicado con mayor profundidad en el siguiente punto.

6. **Sistema de caché**: El sistema de caché es un sistema que se deberá generar con distintas reglas, a partir del análisis del *Query log* utilizado. Este sistema deberá contar con una partición de IDs, distribuido entre tres instancias de *caché*. La configuración del caché utilizada deberá ser justificada por un análisis exhaustivo entre:

   - Políticas de remoción (LRU, LFU u otra a su elección).
   - Memoria utilizada.

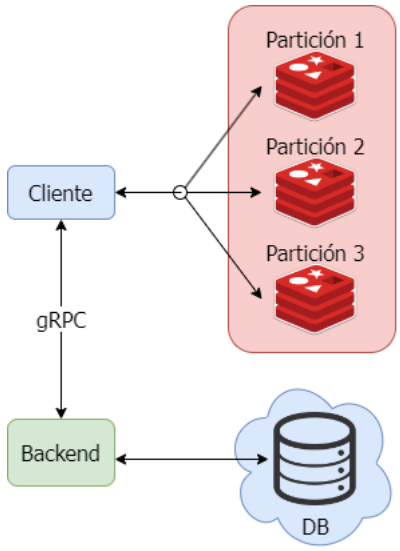
   Para el análisis solicitado, se deberán tomar en consideración las siguientes formas de comparación, siendo justificada en el análisis del informe:

   - Cantidad de aciertos del *caché* (veces que no se hizo una petición al *backend*).
   - Latencia en el tiempo de respuesta.

   Información sobre los puntos anteriormente mencionados pueden ser encontrados en el siguiente documento: `https://redis.io/docs/manual/`.

Diagrama

El sistema generado se puede resumir en el siguiente diagrama:



[1] `https://grpc.io/`
[2] `https://redis.io`

Ejemplo

Usted puede implementar el cliente de dos maneras, dependiendo de cuál sea la manera más cómoda para usted:

1. **Búsqueda a través de una terminal**: Se deberá tener un menú interactivo en el cliente:

```
{
> Inserte busqueda: mens
...
>  search results -> "mens"
    ----------------------------------
    "title": "mens - Traduccion al espanol     Linguee",
    "description": "men sustantivo, plural (singular: man)   . hombres pl m. menos frecuente:
        varones pl m ...",
    "URL": "https://www.linguee.es/ingles-espanol/traduccion/mens.html"
    ----------------------------------
    "name": "Mens - Wikipedia, la enciclopedia libre",
    "description": "Mens (mitologia), en la mitologia romana, Mens es la personificacion del
        pensamiento y la conciencia, tambien se conoce como Bona Mens, ...",
    "URL": "https://es.wikipedia.org/wiki/Mens",
    ----------------------------------
    ...
}
```

2. **Búsqueda a través de un CRUD**: El CURL para la búsqueda sería de la siguiente forma (utilizando una ruta previamente definida)

```
curl --location --request GET 'http://localhost:3000/search?q=Mens'
```

   Los resultados de la búsqueda tendrían que ser la siguiente lista de URLs, al no estar el resultado de la búsqueda "Mens" en cache, la búsqueda se realiza en la base de datos:

```
{
  "search_results": [
      {
        "title": "mens - Traduccion al espanol     Linguee",
        "description": "men sustantivo, plural (singular: man)   . hombres pl m. menos
            frecuente: varones pl m ...",
        "URL": "https://www.linguee.es/ingles-espanol/traduccion/mens.html"
      },{
        "name": "Mens - Wikipedia, la enciclopedia libre",
        "description": "Mens (mitologia), en la mitologia romana, Mens es la personificacion
            del pensamiento y la conciencia, tambien se conoce como Bona Mens, ...",
        "URL": "https://es.wikipedia.org/wiki/Mens",
        ...
  ]
}
```

Entrega

Para la entrega de esta tarea, usted deberá realizar:

- Un repositorio con todos los códigos utilizados.
- Un video con el funcionamiento del programa.
- Un informe donde explique, con sus palabras, brevemente el código desarrollado. Se deben tener las secciones: *Problema y solución*, *Explicación de módulos de código*, *Explicación del funcionamiento caché*, *Resultados y análisis en las configuraciones del caché* y *Conclusión*.

Aspectos formales de entrega

- **Fecha de entrega:** 16 de Septiembre 23:59 hrs.
- **Número de integrantes:** Grupos de 2 personas las cuales deben estar claramente identificadas en la tarea.
- **Querylog**: Deberá seleccionar un querylog de la lista entregada. Cualquier otro Querylog utilizado deberá ser conversado o no será considerado para la revisión.
- **Pauta de evaluación:** La pauta se puede encontrar en el siguiente link `https://docs.google.com/spreadsheets/d/1HHyyVhfbkPshcZknqw082UpHNtfe2Gc1WfFfqMM96qU/edit?usp=sharing`.
- **Lenguaje de Programación:** para la implementación debe escoger entre los siguientes lenguajes: **Python**, **GO** o **JavaScript**.
- **Formato de entrega:** Repositorio público (Github o Gitlab), video de funcionamiento e informe en formato PDF.
- **Tecnologías complementarias:** En caso de usar tecnologías complementarias, añadir una descripción en el informe.
- **Llamadas entre módulos:** vía gRPC, encontrada en el siguiente enlace: `https://grpc.io/`.
- **Contenedores de Redis y Postgres**: se recomienda utilizar las siguientes imágenes: `https://hub.docker.com/r/bitnami/redis/`, `https://hub.docker.com/r/bitnami/postgresql/`. En caso de utilizar otra imagen, deberá especificarlo en el README.
- Las copias de código serán penalizadas con nota mínima. Referente apropiadamente todo segmento de código que no sea de su autoría.
- No se debe implementar un front o interfaz, solo basta con implementar una API REST la cual contenga el método buscar o una interfaz interactiva en la terminal.
- Consultas: **nicolas.nunez2@mail.udp.cl** o **Naike#2258**