SISTEMAS DISTRIBUIDOS: TAREA I

Profesor: Nicolás Hidalgo
Ayudante: Yerko Ortiz

Objetivo:

El objetivo de esta tarea consiste en poner en práctica los conceptos de RPC y Cache vistos en clases. Para ello el alumno deberá hacer uso de un lenguaje de programación y tecnologías que le permitan dar solución a la problemática planteada.

Problemática:

Amazonia una pequeña tienda y start-up de e-commerce ha tenido un creciente aumento en usuarios en su plataforma, lo cual se traduce en un aumento en las ventas pero también en un aumento en la carga de los servidores. La arquitectura de su sistema se basa en el mítico cliente-servidor (C-S) de toda la vida, con un monolito recibiendo las *requests* (consultas) de los usuarios a toda hora. El explosivo crecimiento de la plataforma ha hecho ver a los ingenieros de Amazonia múltiples problemas de rendimiento y escalabilidad en el sistema. Ante esta situación la empresa ha decidido contactarlo a usted para resolver algunos de estos problemas. Las actividades que usted tendrá que realizar para dar solución a este problema consisten en implementar un sistema de inventarios que consta de los siguientes módulos: buscador, cache e inventario.

Descripción de los módulos del sistema

- **Buscador**: este módulo ha de tener un método buscar, dicho método recibe como entrada un string y como output ha de entregar una lista de productos cuyos nombres contengan el string enviado en el request del método buscar. Para realizar esta tarea el módulo buscador primero ha de verificar si el resultado para la búsqueda se encuentra en el cache, en caso que el resultado esté previamente almacenado en el cache, ha de retornarlo como response. En caso contrario, es decir que el resultado de la búsqueda del string no se encuentre en cache, el buscador ha de comunicarse vía RPC con el módulo de inventario y realizar la búsqueda en la lista de productos del inventario, el inventario responderá vía RPC al buscador la lista de productos cuyos nombres contienen el string de la búsqueda.
- **Cache**: El cache ha de ir guardando los resultados de las búsquedas, para el caso de remoción cuando la memoria del cache esté llena, este ha de usar una política del tipo LRU. Para almacenar los resultados de la búsqueda se recomienda guardarlos de forma llave valor, es decir el string de la búsqueda es la llave y el valor es el resultado.
- **Inventario**: Con el fin simplificar el problema y verificar que las llamadas RPC y el uso del cache funcionan de la forma solicitada, los productos del inventario se pueden representar mediante un archivo **JSON** o **XML**. En otras palabras el módulo de inventario en lugar de realizar consultas a una base de datos, realiza directamente la búsqueda del string en el archivo de inventario, como si este archivo fuese la tabla de productos en la base de datos. (De esta forma no es necesario implementar una base de datos, con un archivo que contenga los productos basta).

1 2

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

Resumen

El sistema consta de tres partes, el buscador que ha de ser una API REST con el método buscar(no es necesario nada mas, no se complique la vida), este método ha de consultar al cache si este contiene el resultado de la búsqueda en alguna key. El cache por su lado almacenará los resultados de las búsquedas para evitar consultar al inventario, el cache ha de usar una política de remoción LRU(cuando el cache alcanza su máxima capacidad de memoria, este remueve el elemento que lleva mas tiempo sin ser utilizado). El inventario por otro lado realiza la búsqueda en los nombres de los productos que contengan el string enviado en el request del método buscar, enviando el resultado al módulo buscar y este último es quien retorna el resultado como response. El módulo buscador e inventario deben comunicarse mediante RPC. En la figura 1 se muestra un diagrama que representa el sistema que debe implementar.

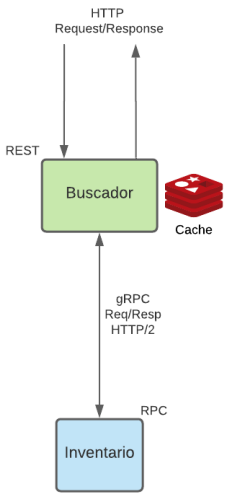

Figure 1: Ejemplo del sistema de inventario

Ejemplos

Para los ejemplos se considerará que el método **buscar** se encuentra en la siguiente ruta:

```
http://localhost:3000/inventory/search [GET]
```

De la misma forma se considerará la siguiente lista de productos:

```
{
  "products_list": [
      {
        "name": "cpu fah7",
        "price": 123,
        "stock": 12
      },{
        "name": "microcpu 211z",
        "price": 1,
        "stock": 1
      },{
        "name": "microscopy",
        "price": 54,
        "stock": 67
      }
  ]
}
```

[3] [4]

[3] https://redis.io
[4] https://grpc.io/

Nota: Para los ejemplos se supondrá que el cache se encuentra vacío inicialmente.

Ejemplo 1: búsqueda del string "cpu"

El CURL para la búsqueda sería de la siguiente forma(utilizando la ruta previamente definida)

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

Los resultados de la búsqueda tendrían que ser la siguiente lista de productos, al no estar el resultado de la búsqueda "cpu" en cache, la búsqueda se realiza en el archivo de inventario:

```
{
  "search_results": [
      {
        "name": "cpu fah7",
        "price": 123,
        "stock": 12
      },{
        "name": "microcpu 211z",
        "price": 1,
        "stock": 1
      }
  ]
}
```

Nota: Tras realizar la búsqueda inserta el resultado de la búsqueda en el cache, la próxima vez que se busque por el string "cpu" retornará este resultado.

Ejemplo 2: búsqueda del string "micro"

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

Nota: En cache actualmente solo está el resultado de la búsqueda de "cpu" pero no de micro, por lo que para este caso también consultará en inventario.

Los resultados de la búsqueda tendrían que ser la siguiente lista de productos:

```
{
  "search_results": [
      {
        "name": "microcpu 211z",
        "price": 1,
        "stock": 1
      },{
        "name": "microscopy",
        "price": 54,
        "stock": 67
      }
  ]
}
```

Nota: Después de realizar esta búsqueda también se guarda en cache el resultado, dado que el cache está configurado con política de remoción LRU si almacenar este resultado supera la memoria máxima establecida, el cache eliminará el elemento menos utilizado, dado que esta es la segunda búsqueda se asume que la memoria es lo suficientemente grande como para contener ambos resultados asociados a sus respectivas llaves.

Si posteriormente se vuelve a realizar la búsqueda de "micro" o "cpu", estos resultados estarán en cache por lo que no será necesario comunicarse con el modulo de inventario para realizar la búsqueda, de esta forma por cada **acierto al cache** estamos ahorrando tiempo de búsqueda y quitamos carga al módulo de inventario! [5] [6]

[5] https://redis.io

[6] https://grpc.io/

Requerimientos Entrega:

- El trabajo se desarrollará en grupos de **3 personas** las cuales deben estar claramente identificadas en la tarea.
- Para vuestra implementación puede utilizar alguno de los siguientes lenguajes de programación: python, go, c++, java o node (escoja el que más le acomode).
- El entregable (código) debe estar alojado en un repositorio github o gitlab, y se entrega se realiza haciendo llegar el link del repositorio al ayudante y profesor vía CANVAS. Se dispondrá una tarea en CANVAS para la carga de los enlaces.
- El entregable debe tener un archivo README en el repositorio. Dentro del README se deben especificar las instrucciones para la ejecución de su código. Si es necesario instalar alguna dependencia extra, es importante mencionarlo en este.
- Otras tecnologías complementarias podrán ser utilizadas a criterio de los estudiantes. En dicho caso, se solicita incluir una breve explicación en el README.
- Para la implementación del módulo de cache debe hacerse con REDIS. En el README debe mencionar la configuración de Redis utilizada y describirla de forma breve. REDIS se puede descargar en `https://redis.io`
- Las llamadas entre los otros módulos debe realizarse via RPC. No hay restricciones para frameworks o librerías de RPC, en ayudantía se mostrarán ejemplos de gRPC pero queda a su total criterio.
- Copias de código serán castigadas con la nota mínima. Referencie apropiadamente todo segmento de código que no sea de vuestra autoría.
- No es requisito implementar un front o interfaz gráfica, basta con implementar una API REST la cual contenga el método buscar.
- **Fecha de entrega**: La tarea debe entregarse como máximo el día 6 de Septiembre 2021 a las 23:59. No se aceptan tareas con atrasos.

Evaluación:

- README con instrucciones de ejecución y descripción de la configuración de REDIS (1 punto).
- Repositorio contiene el archivo de inventario que utilizaron para las pruebas (0.5 puntos).
- Método buscar funcionando acorde a los requerimientos establecidos (2 puntos).
- Método buscar hace uso del cache para evitar búsquedas en el módulo de inventario(1.5 puntos).
- El buscador se comunica vía RPC con el módulo de inventario (1 punto).

7 8

[7] `https://redis.io`
[8] `https://grpc.io/`