TAREA 1: SISTEMAS DISTRIBUIDOS
Sistema de Caché
Profesor: Nicolás Hidalgo
Ayudantes: Cristián Villavicencio, Joaquín Fernández, Felipe Ulloa y Nicolás Núñez

LEA EL DOCUMENTO COMPLETO ANTES DE EMPEZAR A DESARROLLAR LA TAREA

Objetivo

El objetivo de la presente actividad radica en familiarizar al estudiantes con tecnologías de comunicación y funcionamiento de los sistemas de cache. En la actividad los/las estudiantes deberán implementar un sistema de búsqueda distribuido que utilice una API pública como fuente de datos y aplique un sistema de caché personalizado para mejorar el rendimiento de las consultas. Para esta actividad es crucial el análisis, el aplicar mejoras, y evaluar la propuesta. El análisis por parte de los grupos debe estar enfocado tanto en la teoría como en la práctica.

Enunciado

Usted es un trabajador de la empresa **SOA SPA**, una compañía con larga trayectoria que ofrece servicios informáticos a sus clientes y de logística en el área de transportes (**buses**). La empresa utiliza servicios programados en *C*, *pascal*, *Cobol* e incluso *assembly*. Como parte del equipo de mejoras tecnológicas, se le ha encomendado la tarea de evaluar las ventajas que puede ofrecer un sistema de caché y cómo puede mejorar la eficiencia y velocidad de los servicios de la empresa.

Para ello, deberá investigar sobre las diferentes opciones de sistemas de caché que existen en el mercado y presentar las más adecuadas (según su análisis) para la empresa, teniendo en cuenta sus necesidades y recursos. Deberá explicar cómo funciona el sistema de caché elegido y cuáles son sus principales ventajas que justifican su diseño.

Además, deberá utilizar una API pública de su elección para demostrar los beneficios de implementar un sistema de caché en los servicios informáticos de la empresa. Para ello, deberá diseñar una demo que muestre cómo el sistema de caché puede mejorar la velocidad de respuesta de los servicios informáticos que ofrece **SOA SPA**, y presentarla al equipo de dirección de la empresa (Directriz **JRGG**).

Como parte de su presentación, deberá explicar cómo se realizaría la implementación del sistema de caché en los servicios informáticos de la empresa y qué impacto tendría en el rendimiento y la eficiencia de los mismos. Asimismo, deberá considerar los posibles *riesgos y desventajas* de implementar un sistema de caché, y cómo se podrían mitigar estos riesgos.

Observaciones

- La tarea se debe de realizar en parejas. Sólo en casos especiales se aceptarán grupos de 3 integrantes considerando una extensión del problema para hacer equivalente la dificultad con el resto de grupos (leer detenidamente las instrucciones).
- ***¡¡¡Importante!!!*** No escoger API's con políticas de bloqueo ya que entorpecerán la actividad y no se logrará el objetivo principal de la tarea, por ello se sugiere que las prueben con antelación **(Recuerde que al momento de que se tome una API con restricciones, que las considere puesto que no podrá cumplir con la actividad, esto último influirá en su análisis y calificación)**.

Instrucciones

1. Selección de una API pública: Cada grupo deberá seleccionar una API pública de su elección que permita realizar búsquedas de cualquier índole. Algunos ejemplos de API populares pueden ser The Movie Database API, Google Books API, etc (Se debe de inscribir la URL en el siguiente archivo de google drive y cabe mencionar que la ***API es única por grupo***).
   El enlace del documento para la inscripción de la API es el siguiente:
   `https://docs.google.com/spreadsheets/d/1AUo2qFZM1EmK1ZhAUyGl2CJFDxd5HvqjRF1gQm1CjnY/edit?usp=sharing`

2. Implementación del sistema de búsqueda: Los/las estudiantes deberán implementar un sistema de búsqueda distribuido que utilice la API seleccionada como fuente de datos. Este sistema debe ser capaz de manejar consultas de múltiples usuarios simultáneamente acorde a algún supuesto propuesto por los grupos
(Un ejemplo: Se puede utilizar una API de Star Wars y consultar por algunos campos específicos dentro de la API tal es el caso de personajes, planetas, etc; dado lo anterior se puede estimar un supuesto de una gran cantidad de consultas a esta API en donde iría cada vez más lenta, por lo que el cache sería de gran utilidad y que mejor que usarlo de manera distribuida, el contexto habría de variar según la API y la problemática).
A continuación se tienen ejemplos tanto en Python, GO y Javascript.

Ejemplos:

Python:

```python
import requests
import json
def get_planets():
    url = "https://swapi.dev/api/planets/1/"
    response = requests.get(url)
    data = response.json()
    jsonified_data = json.dumps(data)
    print(jsonified_data)

get_planets()
```

Golang:

```go
package main;

import ( "fmt" "net/http" "io/ioutil" )

var api_url = "https://swapi.dev/api/people/1/";

func main() {
    response, err := http.Get(api_url);

    if err != nil {
       fmt.Println("Error: ", err);
       return;
     }
     responseData, err := ioutil.ReadAll(response.Body);
     if err != nil {
       fmt.Println("Error: ", err);
       return;
     }
    fmt.Println(string(responseData));
}
```

Nodejs:

```javascript
const axios = require('axios');
const url = 'https://swapi.dev/api/people/1';
const getData = async url => {
    try {
        const response = await axios.get(url);
        console.log(response.data);
    } catch (error) {
        console.log(error);
    }
}
getData(url);
```

También se espera que los grupos realicen un script para automatizar las llamadas a la API. Ejemplo de ello es la utilización de un cronjob que se ejecute luego de una cantidad de milisegundos y se mantenga funcionando hasta identificar un bajo rendimiento (que los tiempos de espera aumenten, que se pierdan solicitudes por alguna métrica de pérdida de paquetes, queden muchas solicitudes en cola, etc). Para simplificar, se recomienda probar con un script que llame una cantidad de "n" veces a la API y "n" vendría a variar dado el momento que la API presente un bajo rendimiento.

3. Implementación del sistema de caché personalizado: Cada estudiante deberá diseñar e implementar un sistema de caché personalizado para el sistema de búsqueda distribuido. Se puede utilizar cualquier política de remoción para su caché: como LRU, LFU, etc. Recuerde que la selección debe estar justificada por su análisis.

4. Pruebas de rendimiento: Los/las estudiantes deberán realizar pruebas de rendimiento para evaluar el impacto del sistema de caché personalizado en el rendimiento de las consultas. Deberán realizar pruebas de tiempo en las consultas sin caché y con caché para calcular la mejora en el rendimiento aplicando métricas y gráficas que validen la investigación. (Un ejemplo, cambiando la política de remoción, cambiando los parámetros como la ram, CPU en cada una da las instancias de las particiones).
Para el caso de grupos con 3 integrantes, estos deben considerar la implementación de las interfaces de comunicación con REST y con gRPC con el objetivo de compararlos.

5. Evaluación de la escalabilidad: Los/las estudiantes deberán investigar sobre cómo el sistema de búsqueda con caché personalizado puede escalar de manera distribuida. Deberán proporcionar una evaluación detallada de cómo se puede agregar más capacidad de procesamiento para manejar una mayor cantidad de consultas. De igual forma realizar supuestos aplicados con respecto a restricciones de hardware, de que tan conveniente es usar el sistema propuesto dado:

   (a) TTL.

   (b) Técnica de particionamiento.

   (c) Política de remoción.

   (d) Tamaño del cache.

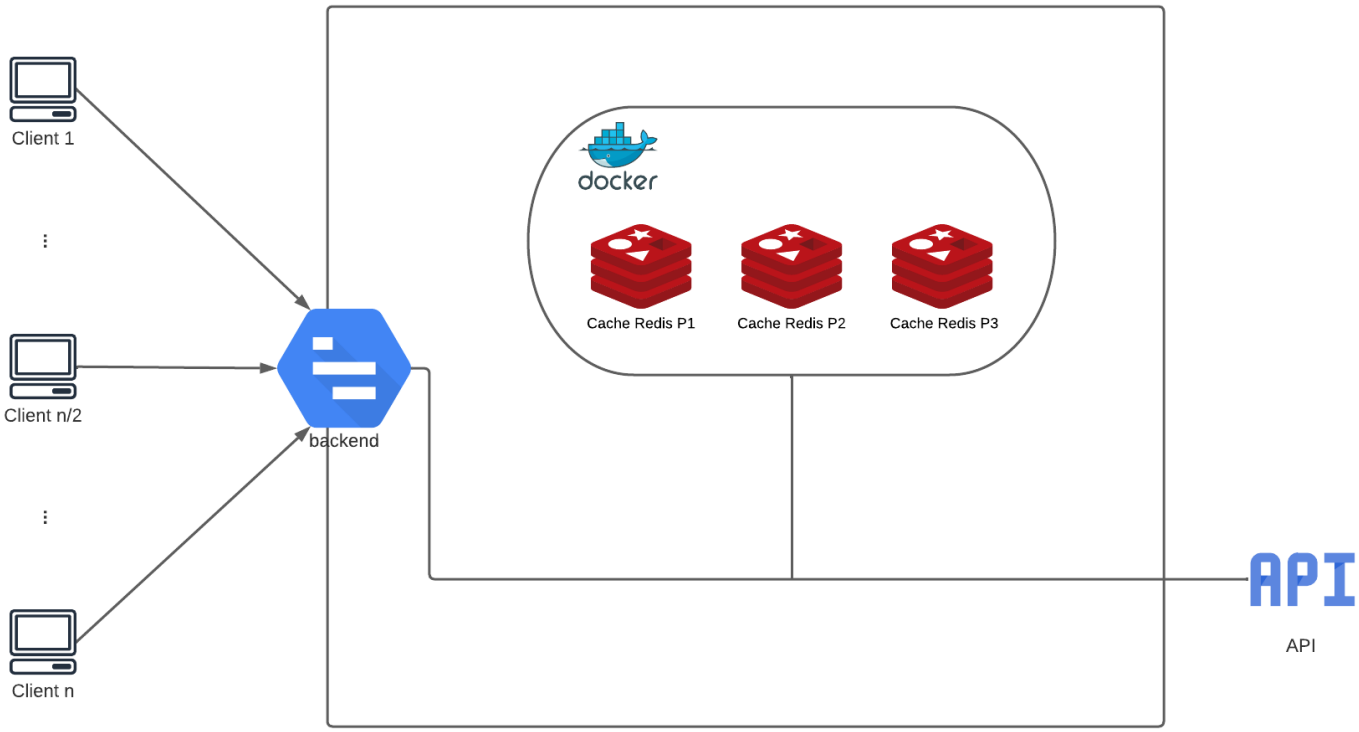
6. Para la implementación de todo el entorno se recomienda que utilicen contenedores *Docker* (muy recomendado sobre todo en la aplicación de las replicas de redis), considerar que el backend no tiene que estar necesariamente dentro de un contenedor. Además para el manejo del cache distribuido se debe usar índices, los índices pueden segmentar los datos en cada una de las replicas de cache redis, por lo que se debe de considerar el siguiente uso.

   - **Por rangos** (Ejemplo: con Id's [0 - 9] en la instancia 1, con Id's entre [10 - 19] en la instancia 2 y con Id's entre [20 - 29] en la instancia 3).

   - **Por Hash** (Ejemplo: Se asigna una función de hash consistente para alguna clave. Por ejemplo se transforma la palabra "Hola" a numérico y gracias a eso se asigna a una partición).

7. En caso de que los grupos sean conformados por 3 personas se deberá aplicar RPC, para el manejo de las request a la API de su elección, además de aplicar un análisis en función de que tan conveniente es usar RPC en contraste a REST; recordar que aquellos grupos conformados por 3 integrantes deben aplicar los procedimientos descritos anteriormente tanto con la API REST y como con RPC.

Diagrama



Entrega

Cada estudiante deberá hacer un vídeo que muestre la implementación del sistema de búsqueda con caché personalizado, las pruebas de rendimiento realizadas y la evaluación de la escalabilidad; La grabación debe ser explicativa y debe mostrar el proceso de implementación paso a paso. El vídeo debe ser acompañado de un informe escrito que incluya el código fuente (subido a github o gitlab), la documentación y los resultados de las pruebas de rendimiento.

Aspectos formales de entrega

- **Fecha de entrega:** 14 de abril 23:59 hrs.
- **Número de integrantes:** Grupos de 2 personas las cuales deben estar claramente identificadas en la tarea y en caso de ser 3 aplicar lo anterior tanto en REST y gRPC.
- **Pauta de evaluación:** La pauta se puede encontrar en el siguiente link `https://docs.google.com/spreadsheets/d/11b71NGI3EYu5g62G9UBMq8aVgyYhKQzpo3lc9tYUTKU/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/` (para equipos de 3 integrantes).

- **Contenedores de Redis bitnami y Redis oficial**: se recomienda utilizar las siguientes imágenes: `https://hub.docker.com/r/bitnami/redis/`, `https://hub.docker.com/_/redis`. 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 caso de ser 2 sino gRPC al ser 3, la cual contenga el método buscar o una interfaz interactiva en la terminal o también a través de un cliente HTTP a su elección (Postman, Thunder client, Insomnia, etc).
- Consultas: **nicolas.nunez2@mail.udp.cl** o **Naike#2258**, **joaquin.fernandez1@mail.udp.cl** o **Jøacø#9352**