# 75.74 - Distribuidos I - TP4

Facultad de Ingeniería
Universidad de Buenos Aires

| del Mazo, Federico | Mermet, Ignacio Javier | Ventura, Julián |
|---|---|---|
| 100029 | 98153 | 102391 |


Junio 2022

# Índice

# 1 Sobre la entrega

El código de la entrega se puede encontrar en GitHub. - Mencionar tecnologías y que vean el readme para ve como se ejecuta.

# 2 Análisis del trabajo a realizar

## 2.1 Puntos clave

# 3 Hipótesis, simplificaciones y asunciones

- El server de rabbit no va a estar replicado

# 4 Arquitectura

- Lógica - De Procesos - De desarrollo - Física

## 4.1 Componentes

## 4.2 Servidor de cara al cliente

- Primero el cliente debe mandar un request por TCP de intención de arrancar a mandar comentarios y posts - Deberia mandar con sessionid nulo, con payload nulo y con type begin
    - status code no-zero es un tipo de error - Esto puede devolver un error de "intente mas tarde"

```
{
        "status_code": 1,
        "payload": "intente mas tarde"
}
```

Figura 1: ...

- O bien puede devolver un id de sesión valido

```
{
        "status_code": 0,
        "payload": "sessionid"
}
```

Figura 2: ...

- Entonces para cada comentario y post deberá mandar un payload en JSON

```
{
        "sessionid": "...",
        "type": "<type>",
        "payload": "..."
}
```

Figura 3: Donde `<type>` puede ser comment, post, begin, result o ACK.

- El server enviará un ACK por cada mensaje recibido y lo tratará como corresponda

```
{
        "status_code": 0,
        "payload": "ACK"
}
```

Figura 4: ...

O un mensaje de error, si surgiera.

- En caso que se caiga la conexión TCP, el cliente deberá mantener su estado y saber cual fue el ultimo mensaje que fue correctamente enviado. Desde ahí intentará mandar al siguiente servidor en el orden de la lista de preferencia. - Un payload vacío señaliza un end of stream. De este modo, el cliente deberá mandar un EoS tanto para comentarios como para posts.

- Para pedir el resultado debera mandar un mensaje con su sessionid, type result y con payload nulo. Una vez recibido correctamente el resultado, debera mandar un mensaje con su sessionid, type ACK y un payload vacio.

Si el resultado no estuviera disponible o el sessionid no fuera valido, se enviarán codigos de error correspondientes. Consultar el anexo para ver los posibles errores.

El servidor está replicado, por lo que requiere un algoritmo de elección de lider. Cualquier instancia debería poder contestar un mensaje de un cliente, pero solo el lider seria capaz de llevar una sesión. Es decir, solo el lider puede recibir nuevos posts y comentarios, y es el unico que puede contestar pedidos de resultados o nuevos ids de sesion.

En caso que una replica recibiera un mensaje de un cliente, deberá contestar con un codigo deerror y como payload la direccion del lider.

### 4.2.1 Workers Stateless

En líneas generales, un worker stateless sigue el siguiente patrón:

```
1 obtener mensaje de queue de entrada
2 procesar mensaje
3 depositar resultado en queue de salida
4 enviar ACK a la queue de entrada
```

Figura 5: Pseudocódigo de un worker stateless.

Imaginemos el escenario donde tenemos varios workers compitiendo por una misma queue...

| | **Direct Exchange** | | | **Workers** |
|---|---|---|---|---|
| prefetch | **0** | **1** | **1000** | **1** |
| *producer* | 5.513 | 5.676 | 5.635 | **5.343** |
| *consumer 0* | **3.536** | 6.860 | 4.266 | 8.796 |
| *consumer 1* | **2.353** | 6.834 | 3.210 | 7.239 |
| *consumer 2* | **2.112** | 6.762 | 2.484 | 4.275 |

Tabla 1: Experimento para comparar las dos alternativas de implementación. En todas las ejecuciones se mandaron 50000 mensajes, sin prints ni sleeps. Las queues fueron inicializadas como durables.

En la siguiente tabla podemos ver los resultados del experimento

Proponemos la siguiente estructura:

La capa de middleware se encargaría de repartir los mensajes al tópico correspondiente. Del mismo modo, cuando un productor deba señalizar un EoS, puede mandar el centinela a todos los consumiores por el topico correspondiente.

Si el worker 2 se cae, quedarian esperando que se procesen.

Explicar las desventajas de hacer el round robin a mano.

#### 4.2.2 Workers Stateful

**Remoción de duplicados**:

#### 4.2.3 Revividor

#### 4.2.4 Storage API

# 5 Algoritmo de elección de líder

# 6 Middleware

# 7 Procesamiento de mensajes

- Mencionar que es at-least-once - Mostrar diagrama de secuencia

# 8 Cliente

- Para el protocolo ver 4.2. - Tiene una lista de preferencias de direcciones (host:port) de instancias de servidores de cara al cliente. Estas se pueden pasar como variables de entorno o archivo de configuración. - Una vez que haya pedido y obtenido exitosamente el resultado, deberá enviar el ACK y tratarlo como lo requiera.