Los lenguajes más modernos (Java, C#, etc) poseen un mecanismo que permite "liberar" la memoria de un objeto de manera automática cuando este ya no es necesario, por medio de un servicio denominado recolector de basura

¿Cuál es el objetivo de una llamada al método clean?

- Disminuir en uno el contador de referencias
- Eliminar el tiempo de arrendamiento asignado para una referencia remota que tiene un cliente
- Eliminar el identificador del cliente de tipo VMID de la lista de referencia de cada objeto remoto indicado en el parámetro ids.

¿Cuál es el objetivo de una llamada al método dirty?

- Asignar un periodo de arrendamiento para una referencia remota que tiene un cliente.
- Agregar el VMID del cliente a la lista de referencias para cada objeto remoto indicado en el parámetro ids.
- Remover un periodo de arrendamiento para una referencia remota que tiene un cliente.

¿Cuándo se realiza una llamada a clean?

- Cuando no existen más referencias a la referencia remota en el cliente.

¿Cómo se identifica un cliente?

- Mediante un objeto de la clase VMID

¿Cómo se identifica un objeto remoto?

- Mediante un objeto de la clase ObjID y un objeto de la clase VMID.

¿Por qué para cada cliente que tiene una referencia remota se asigna un tiempo de arrendamiento?

- Para prevenir que el contador de referencias, en el JVM del servidor, tenga un número que no corresponde con la cantidad de clientes que tienen una referencia al objeto remoto.

¿Qué valores retorna una llamada al método dirty?

- Un identificador del cliente y un periodo de arrendamiento que puede ser más pequeño que el que solicita el cliente.
- Un identificador de la máquina virtual del cliente de tipo VMID y el periodo de arrendamiento.

¿De quién es responsabilidad renovar los arrendamientos?

- Un cliente identificado por un VMID.

¿Si la petición que realiza la JVM del cliente asociada a un método clean o dirty se pierde en la red, que mecanismos se utilizan en la interface DGC para tratar estas llamadas perdidas?

- El parametro sequenceNum, el arrendamiento de la referencia y la excepcioón RemoteException.

¿Qué sucede cuando el contador de referencias llega a 0?

- Eso indica que no hay ningún cliente que tenga una referencia al objeto remoto y por lo tanto se elimina el objeto remoto.

LECTURA

¿Qué valores retorna una llamada al método dirty?

- Un identificador del cliente y un período de arrendamiento que puede ser más pequeño que el que solicita el cliente
- Un identificador de la maquina virtual del cliente de tipo VMID y el periodo de arrendamiento

¿Cuál es el objetivo de una llamada al método clean?

- Disminuir en uno el contador de referencias
- Eliminar el tiempo de arrendamiento asignado para una referencia remota que tiene un cliente
- Eliminar el identificador del cliente de tipo VMID de la lista de referencia de cada objeto remoto indicado en el parámetro ids

¿De quién es la responsabilidad de renovar los arrendamientos?

- Un cliente identificado por un VMID

¿Por qué para cada cliente que tiene una referencia remota se asigna un tiempo de arrendamiento?

- Para prevenir que el contador de referencias, en la JVM del servidor, tenga un numero que no corresponde con la cantidad de clientes que tienen una referencia al objeto remoto

¿Cuál es el objetivo de una llamada al método dirty?

- Agrega el VMID del cliente a la lista de referencias para cada objeto remoto indicado en el parametro 'ids'
- Asignar un periodo de arrendamiento para una referencia remota que tiene un cliente
- Renovar un periodo de arrendamiento para una referencia remota que tiene un cliente

¿Cómo se identifica un cliente?

- Mediante un objeto de la clase VMID

¿Cómo se identifica un objeto remoto?

- Mediante un objeto de la clase ObjID y un objeto de la clase VMID.

¿Qué sucede cuando el contador de referencias llega a 0?

- Eso indica que no hay ningún cliente que tenga una referencia al objeto remoto y por lo tanto se elimina el objeto remoto.

¿Cuándo se realiza una llamada a clean?

- Cuando no existen más referencias a la referencia remota en el cliente

¿Si la petición que realiza la JVM del cliente asociada a un método clean o dirty se pierde en la red, que mecanismos se utilizan en la interface DGC para tratar esas llamadas perdidas?

- El parametro sequenceNum, el arrendamiento de la referencia y la excepción RemoteException

Tarea 2: Recolector de basura distribuido

El recolector de basura distribuido utilizado en java RMI controla las referencias a los objetos remotos. Cuando un objeto remoto no tiene ninguna referencia se plantea que no está siendo usado por lo tanto el recolector puede eliminarlo de la memoria. El paquete java.rmi.dgc contiene una interfaz y dos clases que soportan el recolector de basura distribuido en RMI. Las siguientes lecturas permiten conocer con mayor detalle estos elementos:

a) https://docs.oracle.com/javase/7/docs/api/java/rmi/dgc/DGC.html
b) http://docstore.mik.ua/orelly/java-ent/jenut/ch15_01.htm

A partir de las lecturas responder de forma clara y concisa las siguientes preguntas:

1) ¿Cuál es el objetivo de una llamada al método clean? ¿Cuándo el cliente programa una llamada a clean?

2) ¿Cuál es el objetivo de una llamada al método dirty? ¿De quien es responsabilidad de renovar los arrendamientos?

3) ¿Un objeto de tipo Lease, que valores almacena en sus atributos?

4) ¿Cómo se identifica un cliente?

5) ¿Cómo se identifica un objeto remoto?

Solución

1) El cliente llama al método clean cuando ya no se encuentran más referencias al objeto remoto del servidor. Cuando se llama al método clean se elimina el registro de la tabla de arrendamientos que correspondía a un id de la referencia, al mismo tiempo se disminuye en 1 el contador de referencias.

2) El método dirty se llama cuando se quiere pedir arrendamiento a un objeto remoto del servidor relacionado con un array de id's. Es la máquina virtual del Cliente quien se encarga de renovar los arrendamientos por medio de llamados sucesivos al método dirty.

3) Un **lease** o **arrendamiento** contiene un identificador único de VM (VMID) y una duración de arrendamiento. Un objeto **Lease** se utiliza para solicitar y conceder arrendamientos a referencias de objetos remotos.

Lease

```
public Lease(VMID id,
     long duration)
```

Constructs a lease with a specific VMID and lease duration. The vmid may be null.

Parameters:

`id` - VMID associated with this lease

`duration` - lease duration

4) Mediante un objeto de la clase VMID

5) Mediante un objeto de la clase ObjID y un objeto de la clase VMID.

```
Lease dirty(ObjID[] ids,
            long sequenceNum,
            Lease lease)
              throws RemoteException
```

The dirty call requests leases for the remote object references associated with the object identifiers contained in the array 'ids'. The 'lease' contains a client's unique VM identifier (VMID) and a requested lease period. For each remote object exported in the local VM, the garbage collector maintains a reference list-a list of clients that hold references to it. If the lease is granted, the garbage collector adds the client's VMID to the reference list for each remote object indicated in 'ids'. The 'sequenceNum' parameter is a sequence number that is used to detect and discard late calls to the garbage collector. The sequence number should always increase for each subsequent call to the garbage collector. Some clients are unable to generate a VMID, since a VMID is a universally unique identifier that contains a host address which some clients are unable to obtain due to security restrictions. In this case, a client can use a VMID of null, and the distributed garbage collector will assign a VMID for the client. The dirty call returns a Lease object that contains the VMID used and the lease period granted for the remote references (a server may decide to grant a smaller lease period than the client requests). A client must use the VMID the garbage collector uses in order to make corresponding clean calls when the client drops remote object references. A client VM need only make one initial dirty call for each remote reference referenced in the VM (even if it has multiple references to the same remote object). The client must also make a dirty call to renew leases on remote references before such leases expire. When the client no longer has any references to a specific remote object, it must schedule a clean call for the object ID associated with the reference.

Parameters:

`ids` - IDs of objects to mark as referenced by calling client

`sequenceNum` - sequence number

`lease` - requested lease

Returns:

granted lease

Throws:

`RemoteException` - if dirty call fails

```
void clean(ObjID[] ids,
           long sequenceNum,
           VMID vmid,
           boolean strong)
             throws RemoteException
```

The clean call removes the 'vmid' from the reference list of each remote object indicated in 'id's. The sequence number is used to detect late clean calls. If the argument 'strong' is true, then the clean call is a result of a failed dirty call, thus the sequence number for the client 'vmid' needs to be remembered.

Parameters:

`ids` - IDs of objects to mark as unreferenced by calling client

`sequenceNum` - sequence number

`vmid` - client VMID

`strong` - make 'strong' clean call

Throws:

`RemoteException` - if clean call fails