Programación Distribuida y Tiempo Real

Trabajo Práctico 4 - JADE

Integrantes

1

Se debe ejecutar el main container con el agente **walker**. Luego del inicio del agente, se disponen de 10 segundos para ejecutar otros cuatro containers, que serán por los que se moverá el mismo. Pasados estos 10 segundos, el agente obtiene los containers y los almacena en una cola, de la cual los va sacando de a uno para moverse hacia ellos, obteniendo los datos correspondientes y almacenándolos en una lista. Finalmente, cuando se acaban los containers de la cola, significa que llegó al main container, e imprime los resultados obtenidos.

Comando de compilación

```
javac -classpath ../jade.jar:. Walker.java
```

Comando de ejecución

```
java -cp ../jade.jar:. jade.Boot -gui -agents walker:Walker
```

> Comente la relación entre este posible estado del sistema distribuido y el estado que se obtendría implementando el algoritmo de instantánea.

Dado que no hay relación ni posibles inconsistencias entre los datos generados por cada computadora, los resultados obtenidos por el algoritmo de instantánea serían similares.

2

Se debe ejecutar el main container con el agente **adder**. Luego del inicio del agente, se disponen de 10 segundos para ejecutar otro container. Finalmente, el agente **adder** se mueve hacia el container 1, realiza la lectura del archivo `numeros` (cuyo formato debe ser de un número por línea), suma los números, vuelve al main container e imprime el resultado.

Comando de compilación

```
javac -classpath ../jade.jar:. Adder.java
```

Comando de ejecución

```
java -cp ../jade.jar:. jade.Boot -gui -agents adder:Adder
```

> Cómo se haría lo mismo con una aplicación cliente/servidor.

Se haría un `read` / `get` del archivo de números (esto podría ser de a bytes, de a líneas, o como resulte conveniente), y se efectuaría el procesamiento en el cliente.

> Qué pasaría si hubiera otros sitios con archivos que deben ser procesados de manera similar.

Se podría utilizar el esquema de recorrido del punto 1: el agente se movería por los containers, y en cada uno leería el archivo, efectuaría el procesamiento, y guardaría el resultado junto con container/host en donde se calculó. Una vez recorridos, se retornaría al origen, donde se dispondría de los datos obtenidos.

3a

Se utilizaron tres agentes:

- **reader**: a partir de un nombre de archivo, genera una copia de ese archivo del container *servidor* en el container *cliente*.
- **writer**: a partir de un nombre de archivo, genera una copia de ese archivo del container *cliente* en el container *servidor*.
- **main**: se encarga de recibir las peticiones externas, que indican operación a realizar y nombre de archivo, y delegarlo al agente correspondiente.

Se debe ejecutar el main container con los agente **main**, **reader** y **writer**. Luego del inicio del agente, se disponen de 10 segundos para ejecutar otro container (el servidor). Una vez pasados los 10 segundos, el agente **main** informará que está a la espera de operación. Desde la GUI se debe enviar un mensaje al agente **main** con la sintaxis `[read | write] [filename]`, que lo delegará al agente correspondiente. En caso de `read`, el agente **reader** genera un nuevo archivo `client_[filename]` y se mueve entre los containers, copiando los datos de a 1024 bytes. En caso de `write`, el agente **writer** genera un nuevo archivo `server_[filename]` y se mueve entre los containers, copiando los datos de a 1024 bytes.

Comando de compilación

```
javac -classpath ../jade.jar:. MainAgent.java ReaderAgent.java WriterAgent.java
```

Comando de ejecución

```
java -cp ../jade.jar:. jade.Boot -gui -agents "main:MainAgent;reader:ReaderAgent;writer:WriterAgent"
```

3b

Similar al **3a**, con la diferencia de que el mensaje que se debe enviar al agente **main** sólo debe contener el nombre del archivo a transferir. Aquí agregamos una sincronización entre el **writer** y el **main**: luego de delegar, el segundo se queda esperando a que el primero termine la escritura en el servidor; luego de recibir el mensaje de finalización por parte del **writer**, el **main** envía un mensaje al **reader** para que obtenga el archivo del servidor.