PROGRAMACIÓN DISTRIBUIDA Y TIEMPO REAL

TRABAJO PRÁCTICO N°4

Juan Cruz Cassera Botta 17072/7
Lucio Bianchi Pradas 19341/8

1) **En una plataforma JADE de dos contenedores (Main-Container y Container-1, los nombres usualmente se asignan automáticamente por JADE) programe**

   a) Un agente con nombre de clase AgenteA para que cree un agente de clase AgenteB y lo migre al otro contenedor.

   Para resolver el ejercicio, el AgenteA se crea en el contenedor automáticamente creado por JADE llamado Container-1 a través de la terminal con el script ./jade.ps1 agente (una vez compilado los .java e inicializado el gui).

   Para crear al AgenteB, el AgenteA debe primero obtener un AgentContainer a través del método getContainerController, el cual devuelve el contenedor donde el AgenteA se encuentra actualmente, para lugar ejecutar el método createNewAgent el cual crea al AgenteB en ese mismo container donde está AgenteA.

   Una vez creado el AgenteB, se puede realizar la migración, obteniendo el Location a través del id del Main-Container, y se migra al AgenteB utilizando el método move(destino).

   Links de las clases utilizadas:
   https://jade.tilab.com/doc/api/jade/wrapper/AgentContainer.html
   https://jade.tilab.com/doc/api/jade/core/Agent.html

   b) El AgenteB debe mostrar que se ha completado su migración.

   Para mostrar un mensaje una vez completada la migración. Se sobreescribió el método afterMove() de la clase Agent en la clase del AgenteB. Este método ejecuta acciones luego de completar la migración del agente.

2) **Agregue en los agentes de la implementación anterior:**

a) En el AgenteA genere un mensaje de error si la migración del AgenteB falla.

Se soluciona agregando un simple try - catch a la hora de realizar la migración con el método move(destino).

```
try {
    // Crear AgenteB en el contenedor secundario.
    AgentController agenteB = container.createNewAgent("AgenteB", AgenteB.class.getName(), null);
    agenteB.start();
} catch (Exception e) {
    System.err.println("Error creando al agente B: " + e.getMessage());
}
```

b) En el AgenteB retorne al contenedor de donde fue migrado después de una espera de 2 segundos.

Para este ejercicio agregamos la variable de instancia **destino** de tipo Location en el AgenteB, a la cual se le asigna el valor de here() durante el setup(). El here() nos indica en qué container estaba el AgenteB originalmente, así en el afterMove() podemos migrarlo nuevamente hacia ese container con el método doMove().

3) **Programa una aplicación JADE para que recopile periódicamente el estado de un conjunto de computadoras de manera tal que desde un agente (similar al AgenteA programado anteriormente) se envíe a cada uno de los demás contenedores otro agente (similar al AgenteB programado anteriormente) que tome la información y retorne al lugar de origen. La información a recopilar debería ser:**
a) La carga de procesamiento de cada una de ellas.
b) La cantidad de memoria total disponible.
c) Nombre de la computadora.

Además, se debe registrar el tiempo total de la recopilación de la información de todo el sistema. Comente la relación entre este posible estado del sistema distribuido y el estado que se obtendría implementando el algoritmo de instantánea.

Para resolver el ejercicio, creamos un AgenteA que hace lo siguiente:

1. Crea una lista con los contenedores que debe recorrer el AgenteB. Además, agrega al final de esa lista el contenedor origen (en el que se encuentra el AgenteA), al cual debe retornar el AgenteB cuando termina su recorrido.
2. Crea al AgenteB y le pasa como argumento la lista anterior mencionada.
3. Comienza la ejecución del AgenteB.

Por otro lado, el AgenteB una vez creado inicializa el contador que se utiliza para obtener el tiempo que se tarda en recopilar la información, y comienza a recorrer uno por uno los contenedores que le envió el AgenteA. Mientras el contenedor que esté procesando sea distinto al de origen, se recopila la información solicitada (CPU, RAM y el nombre de la PC) y se mueve el siguiente contenedor.

Una vez se finaliza el recorrido (se volvió al punto de origen), el AgenteB informa los datos recopilados de las distintas computadoras y el tiempo total de recopilación de la información.

Sobre la relación entre el estado del sistema distribuido y el estado que se obtendría usando el algoritmo de instantánea ...

El estado del sistema distribuido sería **inconsistente** porque, como

el AgenteB recorre las distintas PCs de forma secuencial y migrar de una PC a otra introduce overhead, se están recopilando los datos de uso de hardware de cada PC en tiempos físicos y lógicos distintos. Este tipo de estado no garantiza consistencia global, ya que las métricas observadas pueden no ser coherentes entre sí debido a cambios en la carga de procesamiento o la memoria disponible mientras el AgenteB realiza su trabajo.

Por otro lado, el algoritmo de instantánea de Chandy-Lamport está diseñado para capturar un estado global **consistente** de un sistema distribuido. lo que significa que todos los datos reflejarían un momento lógico único en el sistema, a pesar de no ser recopilados físicamente al mismo tiempo. Esto se logra porque el algoritmo coordina la captura del estado con marcas lógicas de tiempo (marcadores) que aseguran que los datos recogidos representan un estado estable del sistema.

4) **Comente cómo se recopilaría la misma información del ejercicio anterior con un modelo de procesamiento c/s.**

Para recopilar la misma información en un modelo de procesamiento c/s, los clientes serían los encargados de recopilar sus propios datos (carga de CPU, memoria disponible y nombre de la computadora) y de enviarlos al servidor central. Este envío se podría hacer, por ejemplo, a través del protocolo HTTP. El servidor central, sería el encargado de coordinar la recopilación de datos y almacenar la información obtenida.

Para que la información se pueda recopilar de forma periódica, el servidor central debería enviar una solicitud a todos sus clientes para avisarles que deben recopilar los datos, y que estos respondan con los datos solicitados.

5) **JADE vs c/s:**
a) Comente qué pasaría si se agregan o quitan computadoras a medida que avanza el tiempo en ambos sistemas.
b) Comente qué pasaría si se cambia el tipo de procesamiento a realizar en todas las computadoras en ambos sistemas.

a)

En un sistema JADE, si se agregan nuevas computadoras, los agentes pueden migrar y comunicarse con ellas sin necesidad de modificar el sistema base. Si se quitan computadoras, JADE también tiene la capacidad de manejar errores de migración o comunicación, enviando reportes o fallbacks (como implementamos en el AgenteA, ejercicio 2). Esta flexibilidad y tolerancia a fallos es propia de sistemas distribuidos basados en agentes, que suelen ser más robustos frente a cambios de infraestructura.

En un sistema c/s clásico, el servidor necesitaría estar configurado para aceptar conexiones desde las nuevas computadoras, y podría requerirse la configuración manual o actualización de la lista de clientes. Si se elimina una computadora cliente, el servidor podría sufrir errores o recibir respuestas incompletas, y el manejo de estos errores suele ser manual o basado en políticas de reconexión. Además, en el modelo c/s, la carga de conexiones y desconexiones dinámicas puede complicarse si el servidor tiene límites de conexiones o requiere reconfiguración**.**

b)

Con JADE, los agentes se pueden reprogramar o adaptar para realizar nuevas tareas mediante estrategias como migración de nuevos agentes especializados o ajustes en los comportamientos (behaviours) de los agentes existentes. Además, el sistema basado en agentes facilita la actualización de procesos, dado que los agentes pueden recibir instrucciones y adaptarse al procesamiento requerido sin necesidad de detener al sistema completo.

Cambiar el procesamiento en un modelo c/s implica modificar la lógica del servidor y, en algunos casos, también la de los clientes, lo cual puede requerir una reimplementación y redistribución de los programas a todas las computadoras. Esto puede ser complejo, ya que implica actualización y despliegue en cada cliente, o cambios en el servidor que requieren modificaciones en los clientes para asegurar compatibilidad.