Trabajo práctico especial

Objetivo

Construir una simulación distribuida basada en agentes, con capacidad de **escalar horizontalmente de forma dinámica** en la cantidad de computadoras utilizadas.

Enunciado

Junto al enunciado, el alumno recibirá la implementación no distribuída de una simulación basada en agentes.

La simulación esta estructurada a través de un motor simple que permite registrar agentes en la simulación. Cada agente no es más que un programa que es ejecutado en su propio thread, que contribuye de manera independiente al fenómeno estudiado. El ciclo de vida de cada agente es controlado por el motor. Los agentes son independientes entre sí, aunque pueden enviar eventos al resto de los agentes, y obtener objetos “de entorno”, que proveen información de contexto para todos los agentes.

Como caso de prueba, se presenta una simulación simple de cadenas de productores / consumidores, **basada en cuatro tipos de agentes**.

Se pide construir implementaciones del motor de simulación y del entorno de los agentes, de tal manera que pueda coordinarse una simulación con agentes distribuidos en varias computadoras.

Los requisitos de la nueva simulación distribuida son:

- Deben poder **agregarse** y **quitarse** agentes mientras la simulación esta corriendo. **La cantidad de agentes por nodo deberá mantenerse lo más pareja posible**. Es decir que la heurística utilizada para definir la afinidad de un agente a un nodo debe tender a que en estado permanente la diferencia de agentes entre todos los nodos no exceda la unidad. Los agentes deberan poder ser agregados y quitados desde cualquier nodo, sin importar donde esten corriendo.

- Deben poder **agregarse nodos mientras la simulación está funcionando**. La aparición de un nuevo nodo implicará la redistribución de agentes existentes para **mantener balanceada la cantidad de agentes que se ejecutan por nodo**.

- Deben poder **quitarse nodos mientras la simulación esta funcionando**. Los agentes que se encontraban corriendo en dicho nodo deberán ser migrados a otros nodos manteniendo el principio de **ecualizar la cantidad de agentes por nodo**. Si el nodo que se quita es el único presente, se finalizará la simulación.

- Deberá poder **consultarse información de la simulaciónen su totalidad**, desde cualquier nodo del sistema.

Como requisito adicional, **NO se podrán modificar las interfaces provistas** para la simulación, así como tampoco la implementación de los agentes provistos.

Diseño del sistema

Dentro del código fuente entregado por la cátedra, se destacan las siguientes interfaces:

`ar.edu.itba.pod.agent.runner.Simulation`

Representa el punto de partida para gestionar una simulación. Contiene métodos que permiten registrar y quitar agentes de la simulación, comenzarla o finalizarla.

La implementación local de dicha interfaz puede encontrarse en el paquete `ar.edu.itba.pod.multithread`.

`ar.edu.itba.pod.multithread.EventDispatcher`

Interfaz destinada a intercambiar eventos entre agentes. Mediante ella se pueden registrar listeners, bloquearse hasta que un evento llegue, desregistrarse y publicar eventos para hacerlos accesibles al resto.

La implementación local se encuentra en el mismo paquete y se llama MultiThreadEventDispatcher.

`ar.edu.itba.event.RemoteEventDispatcher`

Interfaz remota para distribución de eventos entre nodos.

`ar.edu.itba.balance.api.*`

Interfaces que encapsulan la funcionalidad requerida para el load balancing de agentes. En particular existen dos interfaces remotas para la comunicación entre nodos:

1. `AgentsBalancer:` Interfaz para elección de coordinador y comunicación con el mismo
2. `AgentsTransfer:` Interfaz para la transferencia de agentes entre nodos.

`ar.edu.itba.node.api.*`

`ClusterAdministration` interfaz remota que encapsula la funcionalidad requerida para la comunicación del nodo dentro del cluster.

Por otro lado, ***StatisticReports*** es una interfaz remota destinada a conocer las estadísticas de los agentes corriendo en cada nodo.

Cada una de estas interfaces se encuentra debidamente documentada en cuanto a su propósito, las acciones realizadas por cada método, los parámetros de entrada, los parámetros de salida y las validaciones a realizar.

El alumno deberá respetar estrictamente cada interfaz debido a que parte de la evaluación se realizará ejecutando casos de prueba contra las mismas.

Todas las implementaciones del alumno deben ubicarse en el siguiente paquete:

ar.edu.itba.pod.legajoXXX

Donde XXX es el número de legajo del alumno. Cualquier clase, interfaz o archivo fuera del paquete será ignorado.

La implementación del alumno **debe instanciar el RMI Registry de manera embebida**. Es decir, no debe hacer falta levantar el mismo desde afuera de la aplicación.

Toda librería utilizada para la implementación del trabajo deberá ser autorizada por la cátedra.

Modalidad y Calificación

El trabajo será desarrollado individualmente. Cualquier **plagio** detectado será penalizado con una calificación de 0 - no recuperable – reservándose la cátedra el derecho de tomar medidas administrativas adicionales.

La fecha límite de entrega será **21/11/2010 a las 19:00 horas**. Quienen no entreguen en dicha fecha dispondrán de una fecha tardía, que vence el **28/11/2010 a las 19:00 horas**

La entrega será calificada con una nota entre 0 y 10. **La nota mínima para considerar aprobado el trabajo práctico será de 7 puntos**. En el caso de las entregas en primera fecha que esten aprobadas, la nota definitiva es la obtenida en el trabajo. En el caso de las **entregas no aprobadas en primera fecha pero si aprobadas en segunda fecha, la nota definitiva es de 4 puntos** independientemente del resultado obtenido. En el caso de **entregas aprobadas directamente en segunda fecha, la nota definitiva es de 5 puntos** independientemente del resultado. Recordar que siempre se considera un trabajo práctico como aprobado si su nota es igual o superior a los 7 puntos.

Entregables

Se deberá entregar los siguientes ítems:

- Código fuente completo del trabajo práctico. El mismo debe ser enviado por mail a1 mail de la cátedra.

- Manual para crear el jar ejecutable. El mismo puede ser utilizando **Maven**. En ambos casos, la ejecución **no debe contener error** (classpath seteados a mano dependientes de la ubicación, etc). Toda

la ejecución de prueba se va a realizar por línea de comando. **No asumir la existencia de algun IDE** (Eclipse o similares).

Distribución de la nota / criterios de calificación

A la hora de calificar el trabajo práctico se tendrán en cuestión los siguientes aspectos:

- ¿La simulación funciona correctamente en un solo nodo? (1 puntos)
- ¿La simulación funciona correctamente en varios nodos? (2 puntos)
- ¿Se pueden agregar nodos cuando la simulación corre? (1 punto)
- ¿Se pueden quitar nodos cuando la simulación corre? (2 puntos)
- ¿La simulación es capaz de escalar quasi-linealmente? (2 puntos)
- ¿Pueden los nodos interactuar con los nodos de referencia implementados por la cátedra? (2 puntos)

La cátedra se reserva el derecho de otorgar puntos adicionales, así como también de otorgar puntuación parcial de acuerdo a su entendimiento de como se han cumplido los requisitos de la simulación en cada caso.