Arquitectura Software

Práctica 2

Descripción

La práctica consistirá en la construcción de una plataforma que simula un servidor de aplicaciones donde los clientes solicitan la realización de una tarea o cálculo.

Nos basaremos en una arquitectura cliente/servidor distribuido de manera aleatoria, con un balanceador de carga replicado que será el encargado de recibir las peticiones de los clientes y seleccionar un servidor disponible a quien enviarle dichas peticiones, servidores y clientes que realizarán las peticiones al servidor

Las ventajas de esta arquitectura se centran en la organización de funcionalidades como servicios independientes, proporcionados por componentes específicos y desplegados en máquinas diferentes

El balanceador a su vez proporciona métodos para poder activar y desactivar los nodos disponibles y administrar la disponibilidad de los servidores.

Haciendo referencia a los clientes, cabe destacar que nos centramos en un cliente ligero en donde los servidores implementan toda la lógica de servicios, siendo dichos clientes fáciles de mantener.

Los clientes también conocerán la lista de balanceadores y podrán consultar cuáles estań disponibles, por lo que si el balanceador cae, reenviará la petición a otro balanceador que se encuentre activo.

Manual de uso

Servidor

Primero se crearán tantos nodos como servidores deseemos tener.
Para arrancar un servidor se utiliza la función *servidor:start()*.

Balanceador

A continuación se crean los nodos para los balanceadores.
La función para iniciar un balanceador es *balanceador:start(NodosServidor)* donde *NodosServidor* es la lista de nodos en donde están corriendo los servidores. Por defecto el servidor entiende que todos los nodos servidor están activos.
Se podrán añadir nuevos nodos servidor al balanceador a través de la función *balanceador:addServer(NodoBal, NodoServ)* donde *NodoBal* será el nodo donde se está ejecutando el balanceador y *NodoServ* el nombre del nodo del nuevo servidor.

También se pueden activar y desactivar servidores con las funciones *balanceador:activeNode(NodeList)* e *balanceador:inactiveNode(NodeList)* donde *NodeList* sería la lista de servidores a activar o desactivar.

Cliente

Una vez creados los nodos para los clientes, la función para realizar las peticiones al servidor es *cliente:calcular(Balanceadores)* donde *Balanceadores* es la lista de balanceadores activos en el sistema.

Ejemplo de ejecución desde terminal

Parte del servidor

#erl -sname s

La s es el nombre que va a tener el servidor que al estar ejecutándose en una máquina quedará algo del estilo s@nombreMaquinaFisica.

#c(servidor).

Compilamos el módulo del servidor.

#servidor:start().

Arrancamos el servidor.

Parte del balanceador

Una vez tengamos un servidor arrancado

#erl -sname b

La b es el nombre que va a tener el servidor que al estar ejecutándose en una máquina quedará algo del estilo b@nombreMaquinaFisica.

#c(balanceador).

Compilamos el balanceador.

#balanceador:start([s@nombreMaquinaFísica]).

Iniciamos el balanceador con la lista de servidores.

A partir de este punto podemos empezar a lanzar peticiones desde el cliente

Parte del cliente

#erl -sname c

La c es el nombre que va a tener el servidor que al estar ejecutándose en una máquina quedará algo del estilo c@nombreMaquinaFisica.

#c(cliente).

Compilamos el cliente.

#cliente:calcular(b@nombreMaquinaFisica).

El cliente solicita un cálculo al servidor. En lugar de hacer cálculos como el objetivo de la práctica es la arquitectura, simulamos que tarda un tiempo aleatorio.

Test implementados

Se adjuntan pruebas automatizadas para testear el cliente y el balanceador.

Para ejecutar las pruebas del cliente:
Necesitamos un balanceador, un servidor y un cliente como mínimo para ejecutar este test. Con el servidor y balanceador operativos desde el cliente ejecutamos:

```
proper:quickcheck(cliente_props:prop_cliente(nombredelbalanceador@nombreMaquinaFisica).
```

Para ejecutar las pruebas del balanceador:
Necesitamos un balanceador, un servidor y un cliente como mínimo para ejecutar este test. Con el servidor y balanceador operativos desde el balanceador ejecutamos:

```
proper:quickcheck(balanceador_props:prop_start(nombrebalanceador@nombreMaquinaFisica).
```

Diagrama de clases

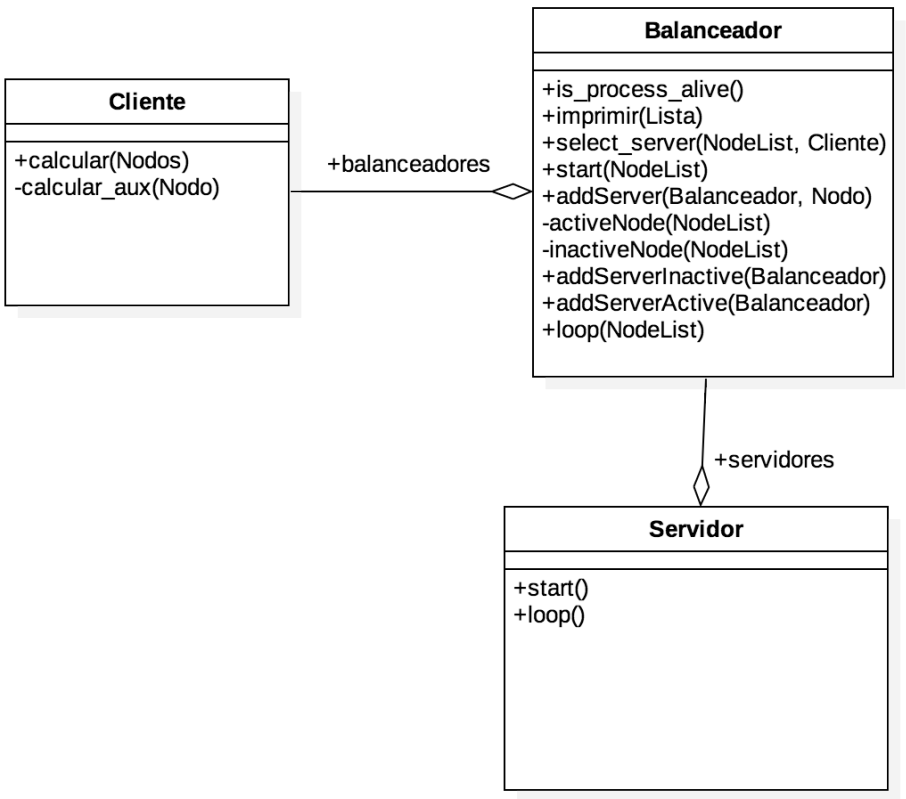


Diagrama de despliegue

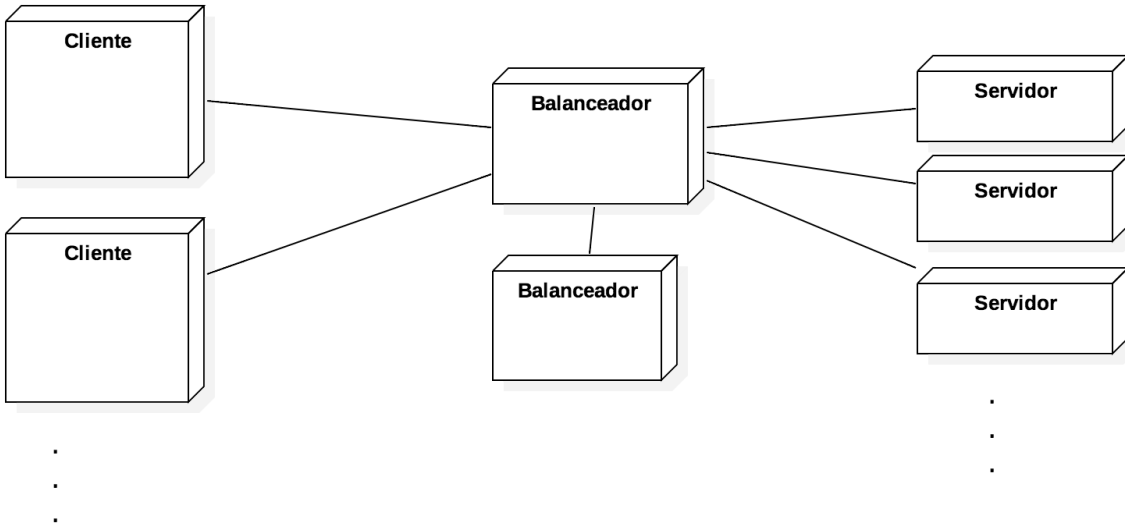
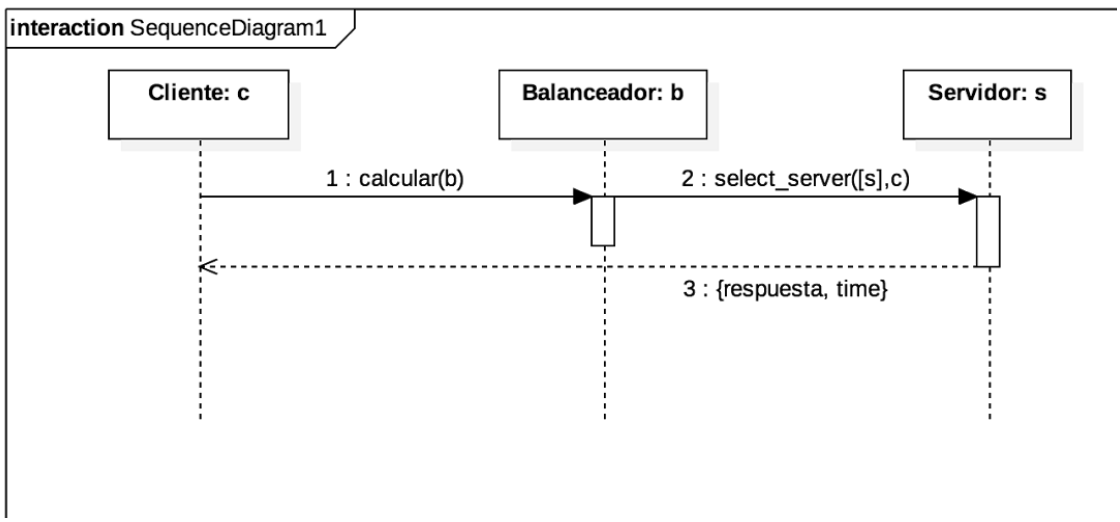
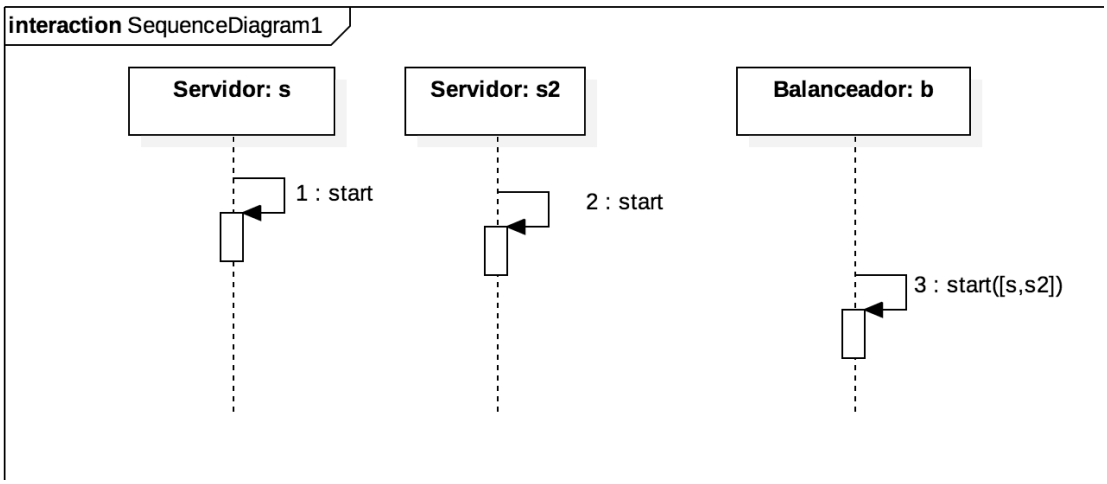


Diagrama de secuencia



En este diagrama queremos representar el funcionamiento del sistema desde que el cliente envía una petición hasta que el servidor la procesa.
El cliente llama a la función calcular() que le envía una petición al balanceador que a su vez llama a select_server() pasando como argumento la lista de servidores y el cliente. Una vez ahí comprueba los servidores activos y envía la petición al servidor. Una vez la procesa envía la respuesta al cliente.



En este diagrama queremos representar la puesta en marcha del sistema. En primer lugar se arrancan los servidores (da igual el orden) y posteriormente el balanceador al que se le pasa como argumento la lista de servidores.

Requisitos No-Funcionales

Debido a la naturaleza del sistema los requisitos no funcionales que consideramos relevantes son los siguientes:

Eficiencia: Nos interesa garantizar que la aplicación se ajusta a sus expectativas en cuanto a los tiempos de respuesta siendo capaz de prestar servicio adecuadamente de acuerdo al tipo y tamaño para el que ha sido concebido, para ello contamos con la posibilidad de incrementar el número de servidores disponibles en función de la demanda de los clientes.

Disponibilidad: Nos interesa que el servicio prestado por la calculadora se encuentre disponible cuando el usuario lo requiera, para ello contamos con un balanceador de carga replicado.

Seguridad: Suponemos que nuestro sistema es público, por lo que los datos no son confidenciales ni van cifrados.

Fiabilidad: Nos interesa mantener una recuperación frente a fallos al disponer del cambio automático del balanceador, y la elección por parte del balanceador de los servidores activos.

Escalabilidad: Al ser un sistema público el sistema puede llegar a crecer muy rápidamente para ello disponemos de funciones para poder dar de alta/baja en el balanceador servidores de cálculo según lo precise.