

Sistemas Distribuidos

La tendencia actual a la globalización exige de los sistemas y aplicaciones informáticos prestaciones que van más allá de lo alcanzable por cualquier ordenador aislado, por muy potente que sea. Ello hace que las aplicaciones distribuidas se vayan convirtiendo en el modelo generalizado.

Desde las más extendidas arquitecturas cliente-servidor hasta los sistemas en cluster que ofrecen la imagen de una única máquina, los sistemas distribuidos presentan una serie de ventajas frente a los centralizados, en cuanto a su potencial rendimiento, fiabilidad, escalabilidad y efectividad de coste. Sin embargo, su diseño y programación conllevan también dificultades específicas.

En este artículo revisamos las características generales de estos sistemas, objetivo de la actividad del grupo de investigación de Sistemas Distribuidos del ITI, y algunas de las opciones posibles a la hora de llevarlos a la práctica.

Introducción

La creciente necesidad de comunicación a todos los niveles entre particulares y organizaciones ha extendido el uso de los sistemas de comunicación entre ordenadores, desde las redes de área local y las redes de área extensa hasta Internet. En la actualidad el uso de ordenadores aparece generalmente asociado al de algún sistema de comunicación, y es posible disponer de grupos de ordenadores que presten conjuntamente un mismo servicio. En el futuro encontraremos cada vez más redes heterogéneas de ordenadores, con aplicaciones que compartan recursos geográficamente distribuidos, flujo de información y coordinación entre sus actividades. Como prueba de esta tendencia encontramos ejemplos actuales en los sistemas de reservas aéreas on-line o las aplicaciones de soporte a trabajo colaborativo, como video-conferencias.

El concepto de sistema distribuido, aparecido hace ya varias décadas, se puede definir como un conjunto de ordenadores interconectados mediante una red que colaboran para la realización de alguna tarea conjunta. Tales sistemas pueden estar compuestos de pequeños ordenadores de bajo coste que combinen sus capacidades. A diferencia de otros sistemas de ordenadores interconectados, los ordenadores (nodos) que forman un sistema distribuido mantienen un estado compartido. Esto requiere una coordinación entre ellos para mantener ese estado consistentemente.

Las principales ventajas que ofrecen los sistemas distribuidos son:

- Tolerancia a fallos. Una consecuencia de tal estado compartido es precisamente la capacidad de los sistemas distribuidos para tolerar fallos en alguno de sus componentes. En un sistema centralizado el estado se mantiene en un único ordenador, el servidor, de modo que el fallo de este (o cualquier reparación, sustitución, etc. a que deba someterse) hace que el sistema deje de estar disponible. Por el contrario, un sistema distribuido cuenta con diversas unidades de cómputo y gestión de recursos, y por tanto con una menor probabilidad de fallo de todo el sistema, si bien es necesario un

esfuerzo especial de gestión para lograr que las aplicaciones no adviertan los fallos de los distintos componentes.

- Escalabilidad. Para aumentar la capacidad de servicio de un sistema centralizado no existe más alternativa que sustituir el servidor por una máquina más potente. En cambio un sistema distribuido puede aumentar su capacidad añadiendo nuevos nodos al sistema existente.

A cambio de las ventajas ofrecidas, sin embargo, los sistemas distribuidos son notablemente más complejos que los centralizados. En particular es preciso contemplar en su diseño una serie de cuestiones específicas:

- Fallos en cada componente. En un sistema distribuido no existe un servidor central cuyos fallos inutilicen el sistema. Sin embargo, cada uno de los ordenadores que lo forma puede fallar independientemente. El sistema debe diseñarse de modo que sus especificaciones sigan siendo respetadas aunque falle algún componente aislado.

- No fiabilidad de las comunicaciones. Igualmente el diseño debe tener en cuenta la imperfección del sistema de comunicaciones entre nodos, y la posibilidad de pérdidas, desorden y duplicación en los mensajes.

- Coste de las comunicaciones. En general los canales de comunicación que interconectan los distintos nodos de un sistema distribuido tienen mayor retardo y coste que los que conectarían distintos procesos dentro del mismo ordenador.

- Mantenimiento de la consistencia. Si la solución adoptada pasa por la replicación, en cada uno de los nodos, de la información necesaria para que la aplicación realice sus tareas, resulta imprescindible realizar una gestión adecuada de la consistencia de cada una de las vistas que los diferentes nodos tengan de esa información global. Esto obliga a que se incluya en el diseño algoritmos específicos que proporcionen, en cada caso, las garantías suficientes para el correcto funcionamiento de las aplicaciones.

Asimismo, hay otros factores que dificultan el desarrollo de los sistemas distribuidos. Por una parte, la programación en estos entornos es más compleja, y por tanto también la depuración y testeo

del software son más costosos. Por otro lado, un sistema distribuido cuenta con un mayor número de potenciales puntos de intrusión, lo cual, unido a la posible inseguridad de la red de comunicaciones, multiplica los riesgos para la seguridad del sistema y complica el diseño de un sistema distribuido seguro.

Alta Disponibilidad

En buena medida los sistemas distribuidos se diseñan para eliminar las limitaciones en el servicio proporcionado debido a fallos en el servidor. Una forma de cuantificar la tolerancia a fallos de un sistema es mediante el concepto de disponibilidad. Este parámetro representa el porcentaje de tiempo durante el cual el sistema se encuentra operativo y depende tanto de la probabilidad de que se den fallos como de la capacidad del sistema para recuperarse frente a estos. En función de su valor se puede establecer una clasificación del grado de disponibilidad de un sistema. Así se habla de alta disponibilidad cuando el tiempo en que el sistema no está operativo es menor del 0.001‰ (menos de 30 segundos al año).

La técnica fundamental para mejorar la disponibilidad de un servicio es la replicación de los componentes que lo proporcionan. Cada réplica de un mismo componente se ubicará en un nodo diferente del sistema, de modo que si un componente falla, una de sus réplicas puede atender el servicio correspondiente. Para lograr esta continuidad del servicio en caso de fallo, la replicación ha de apoyarse en una serie de servicios complementarios, como el monitor de pertenencia, que vigila el estado de los componentes del sistema para detectar y notificar sus fallos y recuperaciones, y los

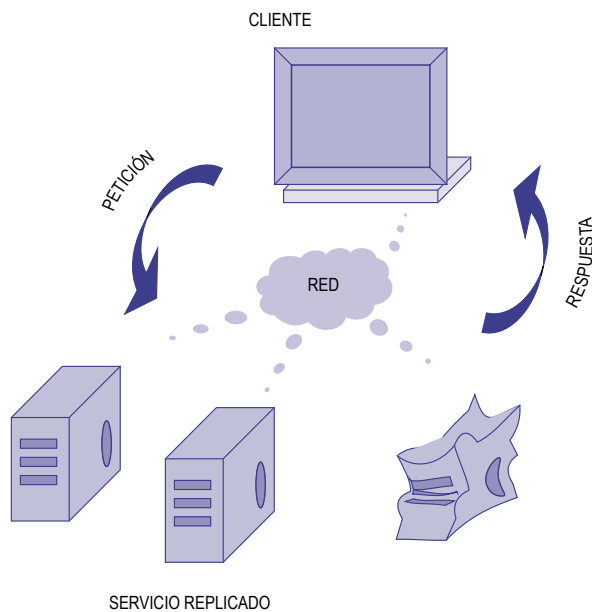


Figura 1: Replicación para disponibilidad.

mecanismos de checkpoints, para realizar actualizaciones de estado, lo cual generalmente requiere a su vez el uso de algún soporte transaccional.

Existen diversas alternativas a la hora de implementar la replicación. Así hay que decidir cuántas réplicas se incluirán, dónde estarán ubicadas, qué nivel de consistencia se exigirá entre el estado de las distintas réplicas, etc. Todas estas opciones, que influirán en

el comportamiento ofrecido por el servidor, configuran un modelo de replicación. Los más importantes son la replicación activa y la replicación pasiva. En la primera, las peticiones enviadas al servicio se dirigen a todas las réplicas, y cada una de ellas realiza idéntica tarea, actualiza su estado y responde. En el segundo modelo, en cambio, solo una de las réplicas procesa las peticiones y se encarga de actualizar el estado de las otras. Existen múltiples variantes de cada modelo, así como modelos intermedios. La elección de uno u otro depende del entorno en que se trabaje y del comportamiento deseado para el servicio.

También hay diferentes posibilidades para dar soporte a replicación en un sistema. Por un lado, es posible replicar procesos o bien objetos. La segunda alternativa, más flexible, es sin embargo más compleja. Por otro lado, existen diversas aproximaciones para implementar una u otra opción, desde un sistema operativo distribuido específico que proporcione directamente el soporte a replicación (*Amoeba*), hasta una capa de middleware que proporcione las interfaces y protocolos necesarios susceptibles de ser empleados por las aplicaciones independientemente del sistema operativo empleado en cada máquina (estándar CORBA), pasando por otras opciones, como la utilización de bibliotecas específicas que puedan utilizarse sobre diversos sistemas operativos (*Isis, Relacs, Phoenix,...*), o la integración del soporte a objetos replicados en un lenguaje de programación distribuido.

Escalabilidad

Las diferentes técnicas de replicación proporcionan, además, distintas características en cuanto a la escalabilidad del sistema. Así, puede verse que la replicación también sirve para aumentar la accesibilidad de la información, proporcionando a los diferentes nodos del sistema la independencia necesaria para convertirse en una unidad computacional activa en el sistema.

La forma en que se integre cada unidad computacional en el sistema influye también en las prestaciones y en la escalabilidad de los servicios que ofrezca este. Además, la elección del modelo de replicación que se implante en el sistema deberá tener en cuenta el tipo de uso que las aplicaciones del sistema distribuido vayan a hacer de este.

Por ejemplo, si se implanta el modelo de replicación pasivo, la escalabilidad del sistema será mejor que la del modelo activo si las operaciones que se realizan sobre la información replicada son mayoritariamente de escritura. Esto es porque el modelo pasivo permite que cada nodo se responsabilice de los accesos a un subconjunto de objetos, actuando así de réplica primaria para dicho subconjunto. De esta forma, todos los nodos pueden hacer el papel de primario para diferentes porciones de la información que replican.

Existen multitud de técnicas para mejorar la escalabilidad y prestaciones del sistema, basadas en diferentes aspectos y principios. El uso de diferentes métodos para la difusión de las actualizaciones es una de las fuentes más comunes para ello. Por ejemplo, si cuando un objeto es modificado se notifican los cambios inmediatamente al resto de nodos, los costes serán mayores que si se retardan las difusiones en el tiempo, con la esperanza de que se realicen más modificaciones de ese objeto antes de difundir los cambios primeros (en este caso, solo sería necesario realizar la última difusión, puesto que la primera sería oculta por las siguientes).

Otras técnicas se basan en los mecanismos de comunicación entre los distintos nodos del sistema. Sobre todo para modelos de replicación del estilo del activo, una adecuada elección de las primitivas de comunicación (incluyendo garantías y características en cuanto a prestaciones y escalabilidad) puede determinar las características del sistema.

Consistencia

La replicación, como ya se ha visto, está palpablemente presente en el ámbito de los sistemas distribuidos. Gracias a ella es posible, por ejemplo, proporcionar tolerancia a fallos, e incrementar las prestaciones y escalabilidad del sistema. Sin embargo, las técnicas de replicación deben incluir los mecanismos necesarios para proporcionar garantías acerca de la consistencia de la información que se replica.

Podríamos definir la consistencia como las garantías que se proporcionan acerca de la visibilidad de los cambios realizados en la información que se replica. Idealmente, cuando en un sistema replicado uno de los nodos realiza una modificación, esta modificación debería poder ser observada en todos y cada uno de los nodos antes de poder realizar ninguna otra operación.

La necesidad de proporcionar estas garantías, sin embargo, entra en conflicto directo con la escalabilidad del sistema, y puede llegar a comprometer las prestaciones alcanzables con un sistema distribuido.

Es por esto que se han definido múltiples relajaciones de la consistencia, que proporcionan al sistema las garantías suficientes para su buen funcionamiento sin comprometer más de lo imprescindible sus prestaciones.

Algunas de estos modos de consistencia son:

- **Consistencia Total.** Es la más estricta, garantizando que todas las operaciones se realizan de forma atómica en cada uno de los nodos del sistema, difundiendo sus efectos antes de que ninguna otra operación se inicie. De este modo, todos los nodos observan los cambios en el mismo orden, que coincide con el orden global en que se realizan.

- **Consistencia Secuencial.** Es similar al anterior, con la diferencia de que el orden en que se aplican los cambios no tiene por qué coincidir con el orden real en que se realizaron. Sin embargo, la consistencia Secuencial continúa exigiendo que todos los nodos coincidan en dicho orden.

- **Consistencia Causal.** Es una nueva relajación de las anteriores, en la que se permite que operaciones iniciadas en diferentes nodos sean difundidas en órdenes distintos, si no existe entre ellas una dependencia "causa – efecto". Así, el orden en que se aplican las operaciones puede variar en dos nodos distintos si las diferencias no afectan la causalidad de las operaciones.

- **Consistencia Débil.** Hace uso de variables de sincronización, a las que los diferentes nodos acceden siguiendo una consistencia secuencial, garantizando que no se obtiene acceso a una variable de sincronización hasta que todas las escrituras previas han sido difundidas a todos los nodos. En otras palabras, las variables de sincronización permiten hacer un volcado de las escrituras de cada nodo, respetando una consistencia secuencial.

La gestión de la consistencia no es pues una tarea simple, y supone un sobreesfuerzo en el diseño de un sistema distribuido,

así como un sobrecoste en cuanto a las prestaciones que el sistema pueda ofrecer.

Aplicaciones Distribuidas

Las aplicaciones distribuidas son aquellas construidas sobre sistemas distribuidos, con las consiguientes ventajas potenciales en cuanto a rendimiento, disponibilidad y escalabilidad. En general es deseable que los aspectos de mayor complejidad del sistema distribuido queden ocultos a la aplicación:

- **Localización.** El usuario no necesita conocer la localización física de un objeto en el sistema. Asimismo, el acceso a objetos locales o remotos se realiza del mismo modo.
- **Replicación.** El usuario no precisa conocer el grado de replicación de un objeto (ni siquiera si está replicado), ni el modelo en que tal replicación está implementada.
- **Fallos.** El sistema debe enmascarar (al menos en cierto grado) los fallos de alguno de sus componentes de modo que la prestación del servicio continúe.
- **Concurrencia.** El acceso a recursos compartidos paralelamente por diversos usuarios requiere una sincronización por parte del sistema.

Si el sistema distribuido cumple estos requisitos de transparencia, el programador de aplicaciones no ha de preocuparse de tales detalles, sino de utilizar los recursos que le ofrece el sistema. Del mismo modo, la aplicación no precisaría ser modificada por el hecho de que el sistema esté compuesto por máquinas heterogéneas, ni por el hecho de que se decida añadir nuevos nodos para su extensión.

Dependiendo del soporte a replicación adoptado, las aplicaciones pueden ser específicas para el sistema operativo distribuido sobre el que se construyen o bien aprovechar las características de alta disponibilidad facilitadas por una capa de middleware.

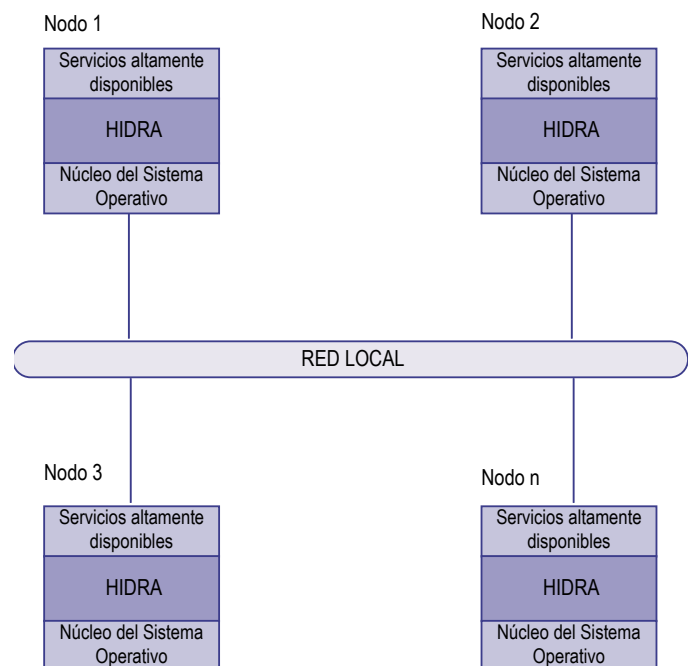


Figura 2: Cluster HIDRA.

Sistemas Distribuidos en el ITI

El grupo de Sistemas Distribuidos (SiDi) del ITI investiga en todos los aspectos relacionados con los sistemas distribuidos, incluyendo en particular soporte para alta disponibilidad, sistemas de almacenamiento fiable, consistencia en memoria compartida, arquitectura de aplicaciones distribuidas, soporte para alta disponibilidad, etc.

En los últimos años, el SiDi ha participado en diferentes proyectos, tanto a escala nacional como soportados por la Unión Europea, relativos a alta disponibilidad, sistemas en cluster, y bases de datos distribuidas. El grupo tiene asimismo una amplia experiencia en la investigación y el desarrollo de soluciones particulares para integrar las ventajas propias de los sistemas distribuidos como soluciones particulares.

Algunos de los últimos proyectos desarrollados, como GlobData, HIDRA, MADIS, y HARL han sido principales para el desarrollo de nuevas tecnologías, y su implantación en ámbitos tan dispares como las bases de datos distribuidas, sistemas de redes, o sistemas en cluster.

El proyecto HIDRA consistió en el diseño de una arquitectura para dar soporte a objetos replicados en un sistema en cluster. Tales sistemas se caracterizan por proporcionar hacia el exterior la imagen de un sistema único. El objetivo de la arquitectura HIDRA es servir como base para el desarrollo de aplicaciones y sistemas altamente disponibles (ver Figura 2). El modelo de programación ofrecido es el de objetos distribuidos, por lo que el componente principal de la arquitectura es un gestor de invocaciones a objetos (ORB) con soporte a replicación. Además del ORB, HIDRA está compuesta por una serie de niveles que proporcionan los servicios complementarios necesarios para dar el soporte a replicación: monitor de pertenencia, nivel de transporte no fiable (el nivel más básico de comunicación, sin garantías en cuanto a la entrega u orden de los mensajes) y transporte fiable (que garantice la entrega de los mensajes enviados siempre que el destinatario no haya fallado). Estos componentes se ubican en parte en el núcleo del sistema operativo que sirve de base, ya que HIDRA pretende extender el núcleo, de modo que sea posible construir también servicios altamente disponibles del propio sistema operativo y ser base para el desarrollo de sistemas cluster. Por ello ofrece un diferente soporte a nivel de sistema operativo y a nivel de procesos de usuario.

En GlobData, se proporciona un middleware Java para la construcción, sobre bases de datos relacionales, de una base de datos distribuida geográficamente que proporciona, además, una visión orientada a objetos de las bases de datos subyacentes. La arquitectura de este middleware, llamada COPLA (ver Figura 3), se basa en tecnología CORBA para integrar las diferentes componentes con las aplicaciones usuarias. COPLA replica la base de datos

proporcionando diferentes modos transaccionales, permitiendo incluso el uso del sistema (con ciertas restricciones) cuando un nodo se encuentra aislado del resto. La consistencia es gestionada incluyendo diferentes protocolos de consistencia intercambiables, que proporcionan diferentes características adecuadas a los diferentes tipos de aplicaciones que pueden existir. La tolerancia a fallos, además, es tratada con especial detalle por COPLA, proporcionando incluso protocolos que permiten la reconexión de nodos fallidos sin necesidad de costosos procesos de "puesta al día" de su información.

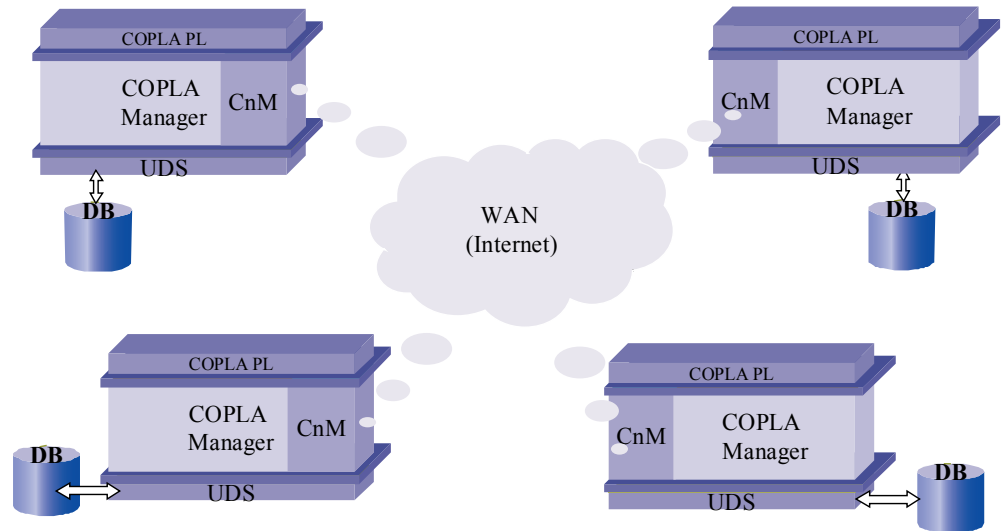


Figura 3: GLOBDATA. Esquema del sistema COPLA.

En el proyecto MADIS se retoma el problema de las bases de datos distribuidas iniciado en GlobData, centrándose más en la integración con Java, a través de interfaces JDBC estándar, y proporcionando protocolos de consistencia más adecuados para entornos de cluster. El objetivo de MADIS incluye, además, la inclusión de las tecnologías desarrolladas en HIDRA, como soporte para la integración en bases de datos en cluster, y las características obtenidas en GlobData.

Otros proyectos, como HARL, han desarrollado soluciones particulares en el ámbito de los sistemas operativos, así como en los sistemas de redes. HARL es un router tolerante a fallos, capaz de proporcionar alta disponibilidad a las comunicaciones de una organización, a bajo coste. El diseño de HARL está basado en tecnología de PC, ejecutando una modificación del sistema operativo Linux. El sistema está compuesto por una serie de PC's que replican el papel de router. Mediante una cuidadosa infraestructura en el kernel, capaz de detectar fallos en el router principal, el sistema es capaz de reemplazar en muy breve espacio de tiempo el papel del router caído, manteniendo activas las conexiones que existían antes del fallo. De este modo, el error se hace imperceptible a todos los efectos, pudiendo sustituir en caliente la unidad dañada, sin interrumpir el funcionamiento del router altamente disponible.

Autores: M^a Carmen Bañuls, Luis Irún
 Para más información sobre Sistemas Distribuidos:
 sidi@iti.upv.es