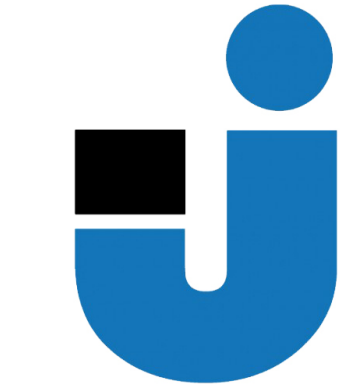

Universidad Nacional
ARTURO JAURETCHE

Sistemas Distribuidos

Práctica nº 1

Introducción

Autor
Emiliano Salvatori

11 de septiembre de 2020

Índice

1. Introducción

En el siguiente informe se detalla lo realizado como parte del laboratorio de la materia **Sistemas Distribuidos** para la **Comisión nº 1**. El presente trabajo se basa en los temas que abordan la Internet de las Cosas.

2. Cuestionario

2.1. ¿Por qué existe una relación tan fuerte entre la definición de un sistema distribuido y el término *middleware*. Explique la función este último

Definición de *Middleware*: Con el objeto de dar soporte a computadoras y redes heterogéneas mientras se ofrece la vista de un sistema único, los sistemas distribuidos se organizan a menudo en términos de una capa de software, esto es, vienen colocados de manera lógica entre una capa de alto nivel que consta de usuarios y aplicaciones, y una capa subyacente constituida por sistemas operativos y recursos básicos de comunicación.

A cada aplicación se le ofrece la misma interfaz. El sistema distribuido proporciona los medios para que los componentes de una sola aplicación distribuida se puedan comunicar ente sí, pero también para permitir la comunicación entre las diferentes aplicaciones. Al mismo tiempo, oculta, lo mejor y más razonablemente posible, las diferencias que se presentan entre el hardware y los sistemas operativos para cada aplicación [1]

La razón por la que existe una fuerte relación entre los Sistemas Distribuidos y la capa middleware es porque esta última es la que permite que todos los sistemas heterogéneos que conforman un Sistema Distribuido permitan presentarse de forma lógica y ordenada ante los usuarios. Si esto no fuera posible, para poder gestionar cada sistema se deberían enforcar en las características particulares de cada dispositivo que lo integra.

En pocas palabras, permite ocultar, lo mejor posible, las diferencias que se presentan entre el hardware y los sistemas operativos para cada aplicación.

2.2. Busque ejemplos de recursos informáticos (hardware y datos) que puedan compartirse, dando ejemplos del sistema distribuido que hace posible que se compartan

El primer ejemplo que se puede citar y que está a la vista de todos es la Word Wide Web. Esta permite que distintos usuarios distribuidos alrededor del mundo, ingresando con hardware y software de distintas características, puedan acceder, visualizar, e interactuar de formas similares con las páginas HTML de la misma manera que lo hace cualquier otro internauta. Las páginas que ofrecen video juegos on line, permiten evidenciar esto aún más, ya que los distintos usuarios que integran plataforma pueden visualizar e interactuar con otros usuarios el mismo juego de una forma en que pare-

ce que están jugando en la misma consola, haciendo caso omiso a las características particulares de cada dispositivo con el que se accede.

Otro ejemplo de software es Google Drive, el cual nos permite guardar en los servidores de Google los documentos que querramos, sin tener nociones de cómo están conformados los distintos servidores en los que se hospeda la información suministrada, siendo para el usuario totalmente transparente al abordar la aplicación, ya que sin importar la plataforma o la arquitectura con la que se acceda, se presenta de una manera semejante a cada cliente.

Otro ejemplo es BitTorrent, el cual tiene por finalidad el facilitar la transferencia de archivos entre diferentes pares en la red sin tener que pasar por un servidor principal. Al usar un cliente BitTorrent, se puede conectar a múltiples computadoras en todo el mundo para descargar un archivo. Una computadora que actúa como coordinadora ayuda a mostrar los nodos en la red que tienen el archivo que se desea. BitTorrent permite alojar archivos de forma voluntaria y subirlos para otros usuarios que los deseen.

2.3. ¿Cómo relaciona una página HTML transmitida mediante el protocolo HTTP con varios recursos referenciados mediante URLs con un sistema distribuido? Describa el funcionamiento del cliente (navegador). Dé tres ejemplos de URLs que denoten diferentes recursos ¿Hasta qué punto podemos decir que la URL brinda transparencia de ubicación?

2.4. ¿Por qué es complicado ocultar la concurrencia y recuperación ente fallos en los Sistemas Distribuidos?

Una de las tantas características que identifican a los Sistemas Distribuidos es el de facilitar a los usuarios un mismo recurso que es compartido junto a otros usuarios. El problema que se tiene es el acceso concurrente a un recurso compartido, el cual puede quedar inconsistente si no se tienen las precauciones necesarias para impedirlo. Quizás una de las maneras de poder brindar el servicio denominado *transparencia de la concurrencia* es mediante la replicación (la cual es otra cualidad de los Sistemas Distribuidos es lo denominado como *transparencia de la replicación*), y que consta de tener varias copias de un mismo recurso para que el usuario tenga la sensación que el recurso se encuentra disponible sólo para él.

El problema que esto contrae, es que todas las copias deben estar sincronizadas, ya que ante el cambio en alguno de ellas, esa misma modificación debe verse reflejada en las demás. También cabe destacar que ante la modificación por varios usuarios de un recurso, se puede ver deteriorado el recurso, ya que se debe asegurar que este sea modificado de forma consistente.

Por otro lado, la principal dificultad que se presenta en el enmascaramiento de las fallas (denominado *transparencia a fallas*)radica en la falta de habilidad para distinguir entre un recurso muerto y un recurso penosamente lento. Por ejemplo, cuando hace contacto con un servidor web ocupado, el navegador genera un error (de time out) e

indica que la página web no está disponible. En ese punto, el usuario no puede concluir si el servidor se encuentra fuera de servicio.

También existe cierto intercambio entre un alto grado de transparencia y el rendimiento del sistema. Por ejemplo, muchas aplicaciones de internet tratan repetidamente de contactar a un servidor antes de darse por vencidas. En consecuencia, intentar enmascarar la falla de un servidor transitorio antes de buscar el contacto con otro servidor, pudiera volver más lento a todo el sistema. En tal caso, podría ser preferible desistir antes, o al menos permitir que el usuario intente hacer contacto.

Otro ejemplo es cuando necesitamos garantizar que las diversas réplicas, localizadas en diferentes continentes, sean consistentes (coherentes) todo el tiempo. En otras palabras, si modificamos una copia, ese cambio se debe propagar a todas las copias antes de permitir cualquier otra operación. Queda claro que una sencilla operación de actualización podría no tomar ni siquiera un segundo para llevarse a cabo, algo que no podemos ocultar de los usuarios.

2.5. ¿Qué significa que un Sistema Distribuido sea abierto?

Un sistema distribuido abierto es un sistema que ofrece servicios de acuerdo con las reglas estándar que describen la sintaxis y la semántica de dichos servicios. Por ejemplo, en las redes de computadoras, las reglas estándar gobiernan formato, contenido, y significado de los mensajes enviados y recibidos. Tales reglas se formalizan mediante protocolos. En los sistemas distribuidos, por lo general, los servicios se especifican a través de interfaces, las cuales a menudo se definen como lenguaje de definición de interfaz (IDL, por sus siglas en inglés). Por lo general, las definiciones de interfaz escritas en IDL solamente capturan la sintaxis de los servicios. En otras palabras, especifican de manera precisa los nombres de las funciones disponibles junto con los tipos de parámetros, valores de retorno, posibles excepciones que se pueden alcanzar, entre otros elementos. La parte difícil es especificar de modo preciso lo que hacen esos servicios, esto es, la semántica de las interfaces. En la práctica, tales especificaciones siempre están dadas de manera informal por medio de un lenguaje natural.

2.6. ¿Qué significa que determinado hardware sea *big endian* o *little endian*? ¿Cómo relaciona el intercambio de datos entre arquitecturas con diferentes *endianness* y la propiedad de los Sistemas Distribuidos abiertos?

El término inglés endianness (“extremidad”) designa el formato en el que se almacenan los datos de más de un byte en un ordenador. El problema es similar a los idiomas en los que se escriben de derecha a izquierda, como el árabe, o el hebreo, frente a los que se escriben de izquierda a derecha, pero trasladado de la escritura al almacenamiento en memoria de los bytes [1]

Al momento de guardar un dato de un byte en la memoria no hay mucho de qué preocuparse, pero cuando se desean almacenar datos de 2 o mas bytes surge un problema, debido a que cada posición de memoria es capaz de guardar un solo byte. Entonces

[1] Obtenido de: `https://es.wikipedia.org/wiki/Endianness`

¿Cómo es ordenado un dato de 16 bits en memoria? Pues bien, es de suma importancia tener en cuenta el formato de almacenamiento que posea la arquitectura a utilizar. [2]

Big endian

En este formato, los bits menos significativos se almacenan en la posición más baja de memoria, mientras que los más significativos se guardan en las posiciones altas consecutivas

Este formato que puede parecer una forma más "natural" de escritura es utilizado por procesadores usados en máquinas Apple entre otras arquitecturas.

Little Endian

En este formato, los bits más significativos se almacenan en la posición más baja de memoria, mientras que los menos significativos se guardan en las más altas consecutivas. Este tipo de formato es adoptado por la mayoría de los procesadores Intel.

Existen ciertas arquitecturas que permiten trabajar con ambos formatos, de manera que pueda escogerse con cuál de ellos trabajar. Éstas son referidos como sistemas que utilizan un formato "Middle Endian".

Mediante lo que se denomina como *Bit Order Mask*, se puede conocer la codificación de caracteres Unicode de marcar si los datos han sido guardados en Little Endian o Big Endian.

2.7. Ejemplo de Big Endian

Si se tiene el numero hexadecimal `759A6FD10E` (5 bytes) y se desea guardar en la posición de memoria 100, se hará de la siguiente manera:

Posición de Memoria	Dato Almacenado
100	75
101	9A
102	6F
103	D1
104	0E

2.8. Ejemplo de Little Endian

Posición de Memoria	Dato Almacenado
100	0E
101	D1
102	6F
103	9A
104	75

[2] Obtenido de `http://www.wikitronica.labc.usb.ve/index.php/Little_Endian_y_Big_Endian`

En ocasiones se tendrán que leer algún fichero de datos y se deberá conocer bajo qué formato ha sido creado, y modificarlo en caso de que no coincida con el de nuestra máquina para que los datos sean correctos al procesarlos.

2.9. Busque ejemplos de acceso simultáneo a un recurso en el cual pueda dejar inconsistente un dato

Existe una seria desventaja en el uso del caché y la replicación que pudiera afectar seriamente la escalabilidad. Debido a que por ahora tenemos copias múltiples de un recurso, modificar una copia la hace diferente del resto. En consecuencia, el caché y la replicación provocan problemas de consistencia (incoherencia en la información).

Hasta qué punto pueden tolerarse las inconsistencias, depende en gran medida del uso de un recurso. Por ejemplo, muchos usuarios de la web consideran aceptable que su navegador devuelva un documento almacenado en caché cuya verificación de validez no ha sido confirmada durante el último par de minutos. Sin embargo, existen muchos casos en los que se requiere cumplir con una fuerte garantía de consistencia, tal como en las transacciones electrónicas de inventario y subastas. El fuerte problema con la consistencia es que una actualización se debe propagar de inmediato a todas las demás copias. Más aún, cuando suceden dos actualizaciones de manera concurrente, a menudo se requiere que cada copia se actualice en el mismo orden. Situaciones como éstas requieren, por lo general, un mecanismo de sincronización global. Desafortunadamente, dichos mecanismos son en extremo difíciles o incluso imposibles de implementar de manera escalable, pues se insiste en que los fotones y las señales eléctricas obedecen a un límite de velocidad de 187 millas/ms (301 km/milisegundo) (la velocidad de la luz). Como consecuencia, el escalamiento mediante replicación podría introducir otras soluciones inherentes y no escalables.

Un ejemplo de ello puede ser cuando se quiere adquirir un ticket en alguna página web, cuando

2.10. Mencione dos ventajas y dos desventajas de los Sistemas Distribuidos respecto de los centralizados

Algunas de las ventajas que aporta un sistema distribuido son:

1. **Mayor eficacia**: Ante la posiblidad de conectar muchos dispositivos a una misma red y poder aumentar los recursos disponibles para cada usuario que se conecte a esa red, hace que la eficacia sea mucho mayor a la de los sistemas centralizados.

2. **Mayor tolerancia a fallos**: al estar distribuida la información en nodos, en caso de que se caiga un nodo, dicha información va a encontrarse replicada en otros nodos.

3. **Mayor velocidad y procesamiento distribuido**: cuando se realiza una consulta, lo procesamientos se dividen entre todos los nodos que forman el sistema distribuido, en lugar de enviarlos a un único nodo y que el mismo tenga que hacer todo el trabajo.

4. **Escalabilidad**: si, por ejemplo, se necesita más procesamiento o añadir más disco duro, en lugar de que los equipos crezcan de forma vertical añadiendo más almacenamiento, RAM o CPU, se añaden equipos de forma horizontal al clúster o sistema distribuido. Esto se puede ver en la sección 2.2 donde se habló de BitTorrent.

Las desventajas que puede presentar un Sistema Distribuido pueden ser:

1. **Gestión**: Como se mencionó en el apartado 2.1 , para que un Sistema Distribuido funcione de forma transparente para todos los usuarios, es necesario que se ejecute un Middleware. Este al tener que lidiar con diferentes tipos de hardware y ejecutar varias versiones distintas de sistemas operativos, puede representar un esfuerzo mucho mayor para gestionar y mantener en funcionamiento del sistema a diferencia de si se estuviese gestionando uno de tipo Centralizado.

2. **Inconsistencias**: Como se hizo mención en el apartado 2.9, cuando en un Sistema Distribuido intervienen muchos usuarios, se vuelve muy difícil mantener la integridad de los datos en el sistema y por ende que los servicios no se vean degradados. Se vuelve muy difícil mantener el concepto de *transparencia en la Concurrencia* a diferencia de los Sistemas Centralizados.

3. **Escalabilidad**: Uno de los problemas que se tiene para poder implementar datos consistentes a todo el Sistema Distribuido es que para tener disponibles varios recursos en la red, se invoca a tener varias copias de un mismo dato, el problema es que la escalabilidad es un problema si estos datos se replican de manera incremental, ya que para que un mínimo cambio sea impactado de forma automática en todas las demás copias, se requiere una sincronización global, siendo estos mecanismos extremo difíciles o incluso imposibles de implementar de una manera escalable [1].

Bibliografía

[1] Andrew S. Tanenbaum. *Distributed Operating System*. Pearson, 2004.