Práctica 1

Laboratorio de Sistemas Distribuidos

Integrantes del equipo

Juan Carlos Olvera González 151408

Ricardo Rosas Juarez 150371

Francisco Javier Zaragoza Zepeda 151603

Fecha

25/01/2017

Índice

Objetivo

Tener un primer acercamiento a los sistemas distribuidos, conceptos básicos, problemas y herramientas que pueden encontrarse al enfrentar a una estructura informática de dicha índole.

En este reporte se presentan los resultados del análisis colaborativo sobre las diversas situaciones propuestas en la práctica realizada en el salón de clases. Por tanto, el siguiente documento está dividido en 5 secciones principales en las cuales se profundiza en los siguientes temas.

1. Recursos de un sistema distribuido, problema de heterogeneidad y ejemplo de la World Wide Web
2. Problema de objeto BLOB
3. Fallas en la arquitectura cliente/servidor
4. Benchmarking de herramientas
5. Conclusión

Recursos de los Sistemas Distribuidos, heterogeneidad y la World Wide Web

En un sistema distribuido, diferentes elementos autónomos trabajan en conjunto para dar la apariencia de ser un solo sistema coherente al usuario. A continuación se ejemplifican tres elementos de hardware, software y datos.

Hardware:

1. Impresoras: en una red hogar (por ejemplo) todas las personas conectadas a dicha red pueden utilizar la misma impresora.
2. Procesador: en un servidor puede tenerse un procesador lo suficientemente poderoso para solventar las necesidades de varias computadoras.
3. RAM: de la misma manera que el ejemplo anterior, un servidor puede tener suficiente memoria RAM para que diversos usuarios sean capaces de trabajar con sus ordenadores mientras estén conectados al servidor en cuestión.

Software:

1. Archivos en la nube: como por ejemplo en la utilización de Google drive.
2. Bases de datos: en un banco con diversas sucursales, todas deben tener acceso a los mismos clientes.
3. Juegos multijugador masivos: se componen de varios clientes conectados a un servidor que tiene que realizar los cambios a las estadísticas de todos los jugadores conectados en tiempo real.

Datos:

1. Clientes en una sucursal bancaria
2. Imágenes: por ejemplo las que podemos encontrar en la web, ya que todos los usuarios que se

encuentren conectados tienen acceso a ellas.

3. Videos: En youtube, una gran cantidad de usuarios es capaz de reproducir un video al mismo tiempo.

Una vez analizados los elementos presentados anteriormente, se discuten de manera subsecuente las problemáticas y una propuesta de solución que existen al tener la necesidad de la sincronización de relojes en una red de equipos interconectados.

En el caso de dos equipos conectados por una red local y sin utilizarse una referencia externa de tiempo, podrían sincronizarse las horas utilizando una computadora como "servidor" maestro y comparándola con los demás equipos conectados para ajustar el tiempo según sea conveniente. Se toma como parámetro de comparación la hora del ordenador seleccionado como *master*. No obstante, cabe tomar en cuenta que una limitación sería la cantidad de equipos conectados a la red, debido a que la solución previamente propuesta se volvería inexacta entre más computadoras se tengan conectadas. A su vez, una manera muy utilizada para resolver esta situación es usando el NTP (Network Time Protocol) que puede reducir los desfases de sincronización de los relojes a milésimas de segundo a través del internet y a fracciones de milésimas de segundo en redes de área local.

Por otra parte si se toma como ejemplo el de la World Wide Web, puede explicarse el cómo funciona la compartición de recursos para integrar un sistema distribuido.

En la World Wide Web existen múltiples y muy variados elementos (imágenes, videos, texto, etc…) que son accesados de manera concurrente por diversos usuarios. De manera general, varios servidores le dedican cierto tiempo a cada uno de los usuarios o clientes que intenta ocupar el contenido que observan en su buscador mediante el ingreso de una dirección URL específica que se envía utilizando un protocolo de transferencia denominado como http, de esta forma, el navegador puede obtener la información solicitada del cliente al servidor. para que todos los contenidos se muestren de manera transparente.

Problema de objeto BLOB

A continuación se presenta un caso específico para analizar las problemáticas de transparencia contra heterogeneidad que pueden haber al considerar la implementación de un objeto BLOB.

Se considera un programa escrito en por ejemplo C++, que implementa un objeto BLOB (Binary Large Object). La aplicación está destinada a ser solicitada por componentes clientes que pueden estar escritos en un lenguaje diferente (por ejemplo Java). Los equipos cliente y servidor pueden tener un hardware diferente, pero están conectados a través de un servicio de internet. Las fallas por lo tanto que son producidas debido a cada uno de los aspectos de heterogeneidad son:

1. Redes: La conexión de los clientes puede fallar
2. Hardware de computadora: Cada uno de los clientes presenta diferente Hardware
3. Sistema operativo: Al haber diferente Hardware, el sistema operativo puede variar
4. Lenguajes de programación: Se puede consultar en varios lenguajes (Java, C, etc.)
5. Implementación por diferentes desarrolladores: Al haber diferentes lenguajes de programación se pueden utilizar diferentes algoritmos para resolver el mismo problema.

Fallas posibles en la arquitectura cliente/ servidor

Los tres principales componentes de software que pueden fallar cuando un proceso del lado del cliente invoca al método objeto ubicado en el servidor son:

1. Servidor remoto.- Una persona que intenta acceder a su perfil en una página web, no es capaz de contactar al servidor responsable de responder a las solicitudes de los usuarios.
2. Datos corruptos (red).- Puede darse una falla en la obtención de los datos (por ejemplo al descargar una imágen) lo cual provoca que no pueda accederse correctamente al contenido deseado.
3. Base de datos.- Si no se maneja bien la concurrencia y el nivel de aislamiento para las transacciones, pueden haber problemas de coherencia en los datos cuando varios clientes intentan hacer modificaciones y consultas a la vez sobre la base de datos.

Benchmarking de herramientas

Subsecuentemente se presenta una tabla con 3 herramientas y 5 criterios de comparación con el objetivo de conocer las ventajas y desventajas de las mismas.

Criterios / Herramientas	RPC gen	Angular	Servicios web
Seguridad	Sus protocolos de seguridad son buenos	Permite la encriptación de los datos a manipular	La información del usuario puede ser encriptada
Localización de recursos	Tienen un acceso y localización transparentes	Se deben programar métodos para acceder a los archivos de la BD	Cuenta con un frameworks que posibilitan la realización de operaciones CRUD
Ejecución remota	En esta herramienta tanto las aplicaciones locales como las distribuidas pueden trabajar entre ellas en un mismo nivel.	Permite que cuando se encuentran conectadas a un mismo servidor se pueden realizar varias acciones desde diferentes ordenadores	Te redirige al error al hacer una petición desde otra computadora
Conectividad	Las solicitudes de conexión fluye en ambos sentidos ya sea cliente servidor o servidor cliente.	Esta herramienta tiene que tener una conexión permanente.	Esta herramienta tiene que tener una conexión permanente.
Compatibilidad con diferentes lenguajes de programación	No se escribe en el lenguaje C sino que en el lenguaje de definición XDR, pero proyecta los datos XDR a tipos de datos C.	Ofrece alternativas de lenguajes basadas en JS, EcmaScript 5, ES6 o incluso TypeScript.	Pueden ocuparse diversos lenguajes de programación para crear web services como: java, ruby, c#, etc… siendo el más eficiente para ello javascript.

			Posteriormente pueden exponerse via REST o SOAP

Conclusión

Después de realizar esta práctica se adquirieron conocimientos básicos acerca de la resolución de problemas relacionados con los sistemas distribuidos además de conceptos básicos y diferentes herramientas que nos permiten realizar un ambiente distribuido analizando los diferentes recursos que los componen y evaluando diferentes criterios de cada una. A su vez, pudimos observar y analizar los diferentes problemas de heterogeneidad y cómo se comparten recursos a varios usuarios, al igual que identificar las diferentes fallas que se pueden presentar en una arquitectura cliente/servidor .

Referencias

George Couluris, Jean Dollimore, Tim Kindberg, Gordon Blair. (2012). Distributed Systems Concepts and Designs. United States of America: Addison-Wesley.

desarrolloweb.com. (2016). Qué lenguaje usar para desarrollar en Angular 2 #programadorIO. 2017, de desarrolloweb.com Sitio web: http://www.desarrolloweb.com/en-directo/desarrollar-angular2-programadorio-8974.html

UPM. (2016). Recomendaciones para la programación con RPC de Sun. 2017, de UPM Sitio web: laurel.datsi.fi.upm.es/_media/docencia/asignaturas/sod/recomrpc.html