Caso de Estudio: Aplicaciones en Arquitectura

Virginia Padilla

Universidad Nacional Experimental de Guayana

virginiapadillas@gmail.com

8 de junio de 2021

Aplicaciones en Arquitectura

1. Conceptos
   - Web Server
   - Web Service de tres niveles

Web Server

Web Server de [Limoncelli, 2014]

- Una máquina que proporciona un servicio web (ver Figura 1) .
- La máquina ejecuta un software que usa protocolo HTTP, recibe solicitudes, las procesa, genera resultado y envía respuesta.
- Sitios web pequeños y aplicaciones basadas en web la utilizan.

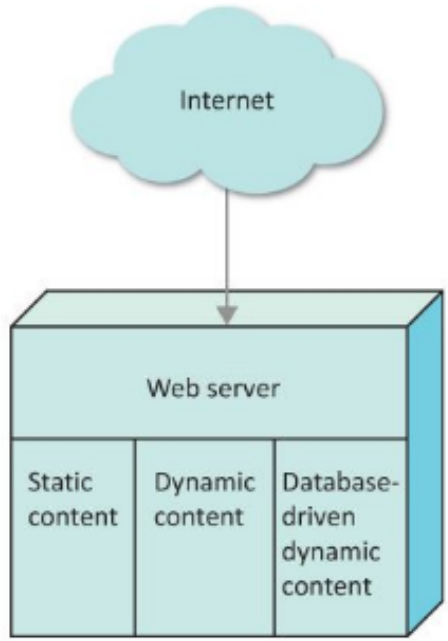

Figura: Arquitectura Web Server de una sola máquina.

Web Server

- Contenido estático: los archivos se leen desde el almacenamiento local y se envían al usuario sin cambios. Pueden ser páginas HTML, imágenes u otro contenido.
- Contenido dinámico: los programas que se ejecutan en el servidor web generan páginas HTML y posiblemente otros resultados que se envían al usuario. Pueden hacerlo de forma independiente o según la información recibida del usuario.
- Contenido dinámico basado en la base de datos: los programas que se ejecutan en el servidor web consultan una base de datos para obtener información y la utilizan para generar la página web.

Web Server

Características

- No todos los servidores web tienen los tres tipos de tráfico.
- Un servidor web estático no tiene contenido dinámico.
- Los servidores de contenido dinámico pueden necesitar o no una base de datos.
- Suficiente para muchas aplicaciones pequeñas.

Web Server

Limitaciones

- No puede almacenar ni acceder a más datos de los que caben en una sola máquina.
- La cantidad de usuarios simultáneos que puede atender está limitada por la capacidad de la CPU, la memoria y la capacidad de E/S de la máquina.
- El sistema también es tan confiable como una máquina. Si la máquina se bloquea o muere, el servicio web se detiene hasta que se repare la máquina.
- Es difícil realizar el mantenimiento de la máquina. Las actualizaciones de software, los cambios de contenido, etc., son interrupciones que pueden requerir tiempo de inactividad.

Web Service de tres niveles

Web Service de [Limoncelli, 2014]

- Patrón construido a partir de tres capas: la capa del equilibrador de carga, la capa del servidor web y la capa del servicio de datos (ver Figura 2).
- Todos los servidores web dependen de un servidor de datos *back-end* común, a menudo una base de datos SQL.

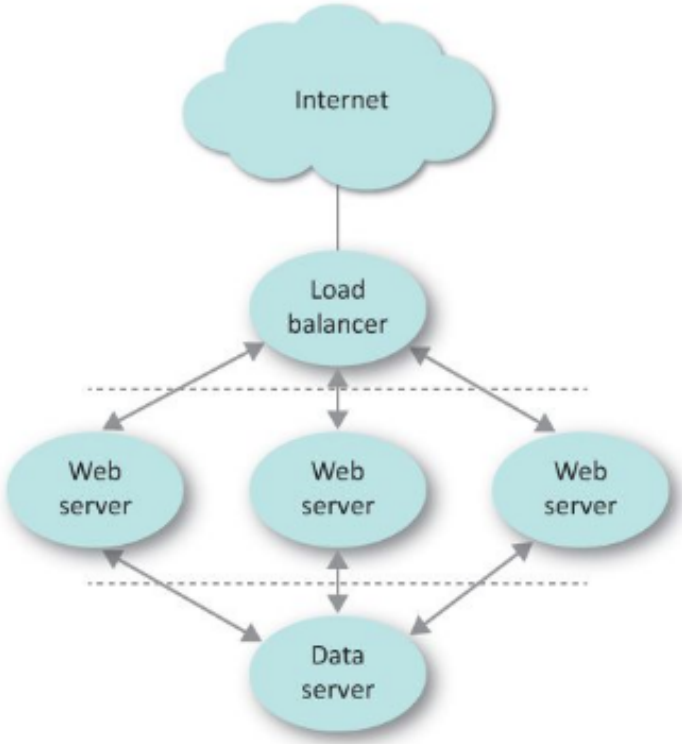

Figura: Arquitectura Web Service de tres capas.

Web Service de tres niveles

Un balanceador de carga funciona al recibir solicitudes y reenviarlas a una de las muchas réplicas, Es una interfaz para muchas réplicas.

- Las solicitudes ingresan al sistema yendo al balanceador de carga.
- El balanceador de carga elige una de las máquinas de la capa intermedia y transmite la solicitud a ese servidor web.
- El servidor web procesa la solicitud, posiblemente consultando la base de datos para ayudarlo a hacerlo.
- La respuesta se genera y se envía a través del equilibrador de carga.

Tipos de Balanceadores de Carga

DNS Round Robin

- Enumeran las direcciones IP de todas las réplicas en la entrada DNS para el nombre del servidor web.
- Los navegadores web recibirán todas las direcciones IP, pero elegirán aleatoriamente una de ellas para probar primero.
- Cuando una multitud de navegadores web visitan el sitio, la carga se distribuirá casi de manera uniforme entre las réplicas
- Los beneficios de esta técnica es que resulta fácil de implementar y es gratuita.
- Limitaciones: Si una réplica muere inesperadamente, los clientes seguirán intentando acceder a ella, ya que el DNS se almacena en caché. El sitio parecerá estar inactivo hasta que esos cachés caduquen.

Tipos de Balanceadores de Carga

Balanceadores de carga de capa 3 y 4 - parte 1

- Nombre proviene de la Pila del protocolo ISO: La capa 3 es la capa de red (IP), capa 4 es la capa de sesión (TCP)
- Reciben cada sesión de TCP y la redirigen a una de las réplicas.
- Cada paquete de la sesión pasa primero por el balanceador de carga y luego por la réplica.
- Los paquetes de respuesta de la réplica vuelven a través del balanceador de carga.
- Los balanceadores de carga L3 rastrean las sesiones TCP en función de las direcciones IP de origen y destino (capa de red).

Tipos de Balanceadores de Carga

Balanceadores de carga de capa 3 y 4 - parte 2

- Todo el tráfico de una dirección de origen determinada se enviará al mismo servidor independiente del número de sesiones TCP generadas.
- Los balanceadores de carga L4 rastrean los puertos de origen y destino además de las direcciones IP (la capa de sesión).
- Beneficio: son simples y rápidos. Además, si una réplica falla, el equilibrador de carga enrutará el tráfico a las réplicas restantes.

Tipos de Balanceadores de Carga

Balanceador de carga de capa 7. parte 1

- Funcionan similar a los balanceadores de carga L3/L4, pero toman decisiones de lo que se puede observar en la capa de aplicación (Capa 7).
- Pueden examinar qué hay dentro del protocolo HTTP en sí (cookies, encabezados, URL, etc.) y tomar decisiones basadas en lo que encontraron.
- Puede verificar si se ha establecido una cookie y enviar tráfico a un conjunto diferente de servidores según ese criterio.
- Así es como empresas manejan a los usuarios especiales: establecen una *cookie* especial cuando sus clientes más importantes inician sesión, configuran el equilibrador de carga para detectar esa cookie y enviar su tráfico a servidores más rápidos.

Tipos de Balanceadores de Carga

Balanceador de carga de capa 7. parte 2

- Son transparentes: la dirección IP de origen de la solicitud no se modifica.
- La mayoría no lo son: la dirección IP de origen de cada solicitud que ve el *backend* es la dirección IP del balanceador de carga. Desde el *backend*, las solicitudes parecen provenir de una sola fuente.
- Cuando todas las solicitudes parecen provenir de la misma dirección IP, la depuración y el análisis de registros pueden ser imposibles en el peor de los casos y confusos en el mejor de los casos.
- Para abordar este problema el equilibrador de carga inyecta un encabezado adicional con la dirección IP del solicitante original.

Método de Balance de Carga

Algoritmos. Parte 1

Por cada solicitud, un balanceador de carga L3, L4 o L7 tiene que decidir a qué backend enviarlo. Existen algoritmos para tomar esta decisión:

- Round Robin (RR): las máquinas giran en un bucle. Si hay tres réplicas, la rotación se parecería a A-B-C-A-B-C.
- RR ponderado: similar al RR pero da más consultas a los *back-end* con más capacidad.
- Menos cargado (LL): el equilibrador de carga recibe información de cada backend que indica qué tan cargado está. Las solicitudes entrantes siempre van al backend menos cargado.
- Menos cargado con inicio lento: Similar a LL, pero cuando un nuevo *backend* entra en línea, no se inunda de inmediato con

Método de Balance de Carga

Algoritmos - parte 2

- Límite de utilización: cada servidor estima cuántos QPS (Query Per Seconds) más puede manejar y se lo comunica al equilibrador de carga. Las estimaciones pueden basarse en el rendimiento actual o en los datos recopilados de las pruebas de carga sintética.
- Latencia: el equilibrador de carga deja de reenviar solicitudes a un backend en función de la latencia de las solicitudes recientes. Por ejemplo, cuando las solicitudes tardan más de 100 ms, el balanceador de carga asume que este backend está sobrecargado.
- Cascada: la primera réplica recibe todas las solicitudes hasta que alcanza su capacidad. Cualquier desbordamiento se dirige a la siguiente réplica, y así sucesivamente.

Equilibrador de carga con estado compartido

- Un problema con el equilibrador de carga entre muchas réplicas es el estado compartido.
- Suponga que una solicitud HTTP genera información que necesita la siguiente solicitud HTTP.
- Un solo servidor web puede almacenar esa información localmente para que esté disponible cuando llegue la segunda solicitud HTTP.
- Pero, ¿qué pasa si el equilibrador de carga envía la siguiente solicitud HTTP a un *backend* diferente? No tiene esa información (estado).

Equilibrador de carga con estado compartido

- Considere el caso en el que una solicitud HTTP toma el nombre y la contraseña de un usuario y los valida, permitiendo que el usuario inicie sesión.
- El servidor almacena el hecho de que el usuario ha iniciado sesión y lee su perfil en la base de datos.
- Esto se almacena localmente para un acceso rápido. Las futuras solicitudes HTTP a la misma máquina saben que el usuario ha iniciado sesión y tiene el perfil de usuario a mano, por lo que no es necesario acceder a la base de datos.
- ¿Qué pasa si el balanceador de carga envía la siguiente solicitud HTTP a un backend diferente?
- Este backend no sabrá que el usuario ha iniciado sesión y le pedirá el inicio de sesión. Es molesto para el usuario y trabajo adicional para la base de datos.

Equilibrador de carga con estado compartido

Como lidiar con ello:

Conexiones fijas: los balanceadores de carga tienen una función llamada adherencia, significa que si la solicitud HTTP anterior de un usuario se dirigió a un *backend* en particular, la siguiente también debería ir allí.

- Si ese backend muere, el balanceador de carga enviará solicitudes de ese usuario a otro *backend*; no tiene elección.
- Este nuevo backend no sabrá que el usuario ha iniciado sesión. Por lo tanto, esta es solo una solución parcial.

Equilibrador de carga con estado compartido

Como lidiar con ello:

Estado compartido: el hecho de que el usuario haya iniciado sesión, la información del perfil del usuario se almacena en algún lugar al que puedan acceder todos los *backends*.

- Si ese backend muere, el balanceador de carga enviará solicitudes de ese usuario a otro *backend*; no tiene elección.
- Para cada conexión HTTP, el estado del usuario se obtiene de esta área compartida. Con este enfoque, no importa si cada solicitud HTTP va a una máquina diferente.
- No se le pide al usuario que inicie sesión cada vez que se cambian los backends.

Equilibrador de carga con estado compartido

Como lidiar con ello:

Híbrido: cuando el estado de un usuario se mueve de un backend a otro, generalmente crea un poco de trabajo adicional para el servidor web.

- A veces, esta carga es pequeña y tolerable. Para algunas aplicaciones es muy inconveniente y requiere mucho más procesamiento.
- En ese caso, la mejor solución es usar tanto conexiones fijas como estado compartido.

Equilibrador de carga con estado compartido

- Otros sistemas están diseñados para esto, y a menudo contienen toda la información en la *RAM* para un acceso más rápido.
- *Memcached* y *Redis* son dos ejemplos.
- En cualquier caso, debe evitar el uso de un directorio en un servidor NFS, ya que no proporciona conmutación por falla en caso de una interrupción del servidor o un bloqueo de archivos.

Identidad de Usuario

- Aunque al usuario le parece que la interacción con un sitio web es una sesión sin interrupciones, en realidad se compone de muchas solicitudes HTTP distintas.
- Los *backends* necesitan saber que las solicitudes HTTP son del mismo usuario.
- No pueden usar la dirección IP de origen de la solicitud HTTP: varias máquinas usan la misma dirección IP.
- Cuando un usuario inicia sesión en una aplicación web, se genera una palabra secreta y lo incluye con la respuesta.
- La palabra secreta se genera aleatoriamente y se le da solo a ese usuario en ese navegador web.
- Cada vez que ese navegador web envía una solicitud *HTTP* a esa misma aplicación web, también envía la palabra secreta.
- Este esquema se conoce como *cookie* y la palabra secreta a menudo se denomina **ID de sesión**.

Escala

El servicio web de tres niveles tiene muchas ventajas sobre el servidor web único :

- Grupos de réplicas: puede servir a muchos grupos de réplicas. Cada grupo tiene un sitio web diferente. El número de réplicas en cada grupo se puede aumentar de forma independiente según lo requiera el sitio web.
- Equilibradores de carga dual: puede haber varios equilibradores de carga, cada uno una réplica del otro. Si uno falla, el otro puede hacerse cargo.
- Múltiples almacenes de datos: puede haber muchos almacenes de datos diferentes. Cada grupo de réplicas puede tener su propio almacén de datos o se pueden compartir todos los almacenes de datos. Cada uno de los almacenes de datos puede utilizar la misma tecnología o una tecnología diferente.

References

Van Steen M , Tanenbaum, A (2017)
Distributed Systems
Pearson Education, Inc.

Coulouris G, Dollimore J, Kindberg T, Blair G (2011)
Distributed Systems: Concepts and Design
Addison-Wesley Publishing Company.

Veríssimo, P. and Rodrigues, L. (2012)
Distributed Systems for System Architects
Springer.

Limoncelli T, Strata R, Hogan C. (2014)
The Practice of Cloud System Administration: Designing and Operating Large Distributed Systems
Addison Wesley.

Fin