Desarrollo de Aplicaciones Web Empresariales

Exequiel Fuentes Lettura
exequiel.fuentes@ucn.cl

Información de contacto

- Exequiel Fuentes Lettura
  - Email: exequiel.fuentes@ucn.cl
  - Horario de Atención: Jueves y Viernes, bloque C

- Departamento de Ingeniería de Sistemas y Computación
  - Oficina: Y1 - 329
  - http://www.disc.ucn.cl

Arquitectura de aplicaciones Web

Arquitectura: escalabilidad horizontal

- Expande la capacidad agregando más instancias.
- La arquitectura de escalabilidad vertical hace una instancia más grande
  - Su límite está relacionado con el tamaño de la instancia
- Los beneficios de la escalabilidad horizontal:
  - Se escala cuando es necesario, sólo basta agregar o quitar instancias.
  - La hace más tolerante a fallos. Si una instancia deja de funcionar, las otras siguen funcionando.
- Desafío: Se requiere administrar múltiples instancias y distribuir el trabajo entre ellas.

Escalabilidad en servidores web

- Los navegadores se comunican con los servidores usando HTTP.
- Se usa un balanceador de carga para distribuir las solicitudes HTTP.
- DNS (Domain Name System):
  - Especificar varios destinos dado un nombre.
  - Administrar el sistema a través de distribución geográfica.
  - Rotar los servidores DNS entre los destinos, asistir cuando sea necesario.

Balanceando carga usando switch

- Una forma de balancear carga se puede hacer utilizando un switch en la red
  - Los paquetes pasan a través de un balanceador de carga en el switch.
  - El balanceador dirige las conexiones TCP a uno de varios servidores web.
  - El balanceador enviará todos los paquetes de esa conexión al mismo servidor.
- En algunos casos, los switches son lo suficientemente inteligentes para inspeccionar las cookies, así una misma sesión siempre va al mismo servidor.
- Es más fácil balancear carga en servidores sin estado. Diferentes solicitudes pueden ser manejadas por diferentes servidores.
- Selecciona el servidor web aleatoriamente o estimando su carga.

nginx ("Engine X")

- Un servidor web muy eficiente
  - Maneja decenas de miles de conexiones HTTP.
  - URL: https://www.nginx.com/
- Usos:
  - Balanceador de carga.
  - Administra servidores web y es tolerante a fallos.
  - Maneja solicitudes simples, como archivos estáticos, etc.
  - Mitiga ataques DOS limitando el ritmo de las solicitudes.
- Es un enfoque bastante usado con Node.js

Escalamiento horizontal

- La suposición es que cualquier servidor es capaz de realizar la solicitud.
- Como sabemos, un servidor sin estado es más fácil de balancear.
- Qué sucede con aquellos que mantienen sesiones?
  - El acceso a cada solicitud debe ser tan rápida como sea necesario (memcache?)
- Ahora, los WebSockets unen los navegadores con el servidor web
  - No es posible balancear carga por cada solicitud.

Escalamiento horizontal

- Qué sucede con el almacenamiento en este tipo de arquitectura?
- Tradicionalmente, las aplicaciones Web comenzaron con base de datos relacionales.
- Una sola instancia de base de datos no muy escalable!
- Data sharding, extiende la base de datos sobre instancias en una arquitectura horizontal
  - Cada pieza se le llama data shard (fragmento de datos)
  - Tolerante a fallos a través de replicación.

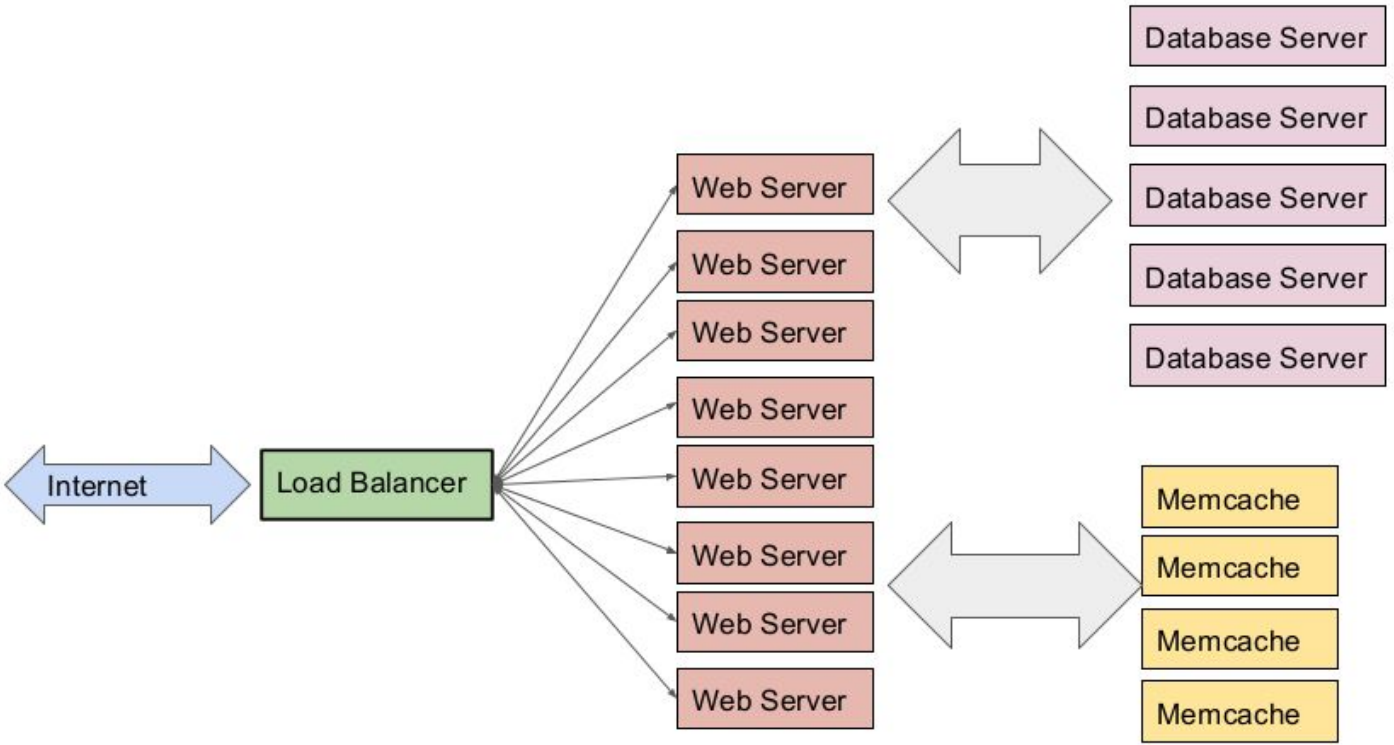
Escalamiento horizontal

- Las aplicaciones deben particionar los datos entre múltiples base de datos independientes.
- Facebook inicialmente tenía una instancia de base de datos por universidad.
- En el año 2009 Facebook tenía 4000 servidores MySQL, usaba una función hash para seleccionar los data shard.
- Puede encontrar un ejemplo implementado en Java en este link: http://sleeplessinslc.blogspot.cl/2008/09/hibernate-shards-maven-simple-example.html

Memcache: main-memory caching system

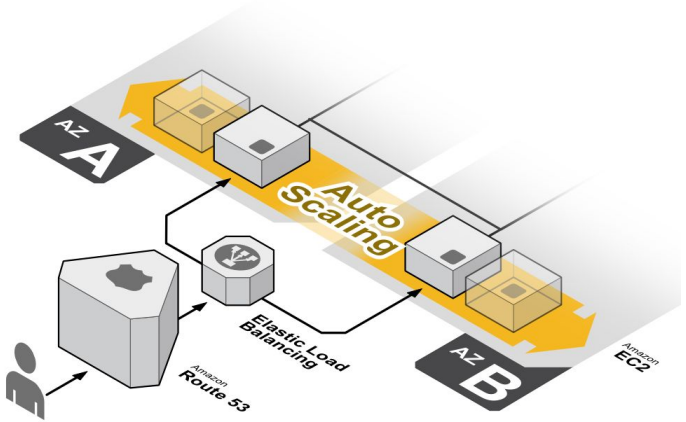
- Enfoque clave-valor (ambos son datos arbitrarios)
- Usado para mantener los resultados de consultas recientes a base de datos.
- Es más rápido que base de datos:
  - 500 microsegundos versus 10 milisegundos
- Ejemplo: Facebook tenía 2000 servidores memcache en el año 2009

Arquitectura web con escalamiento horizontal

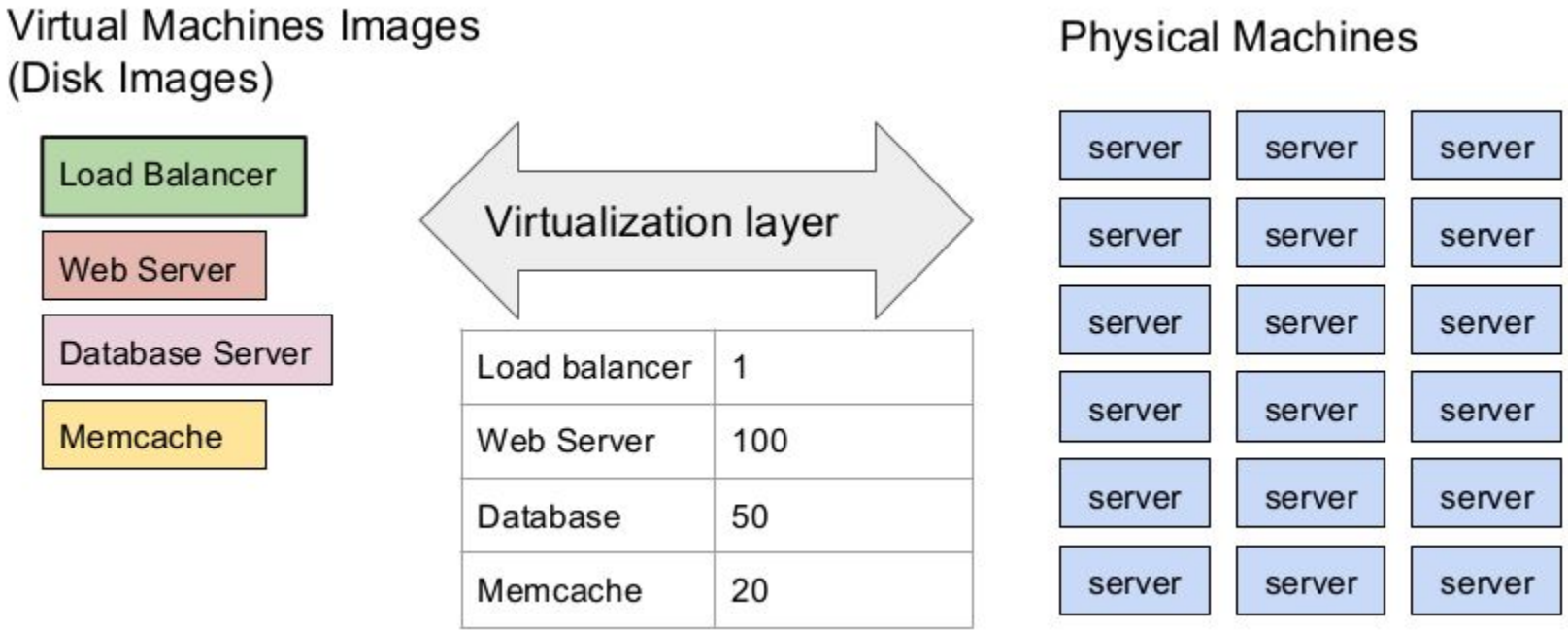


Construir esta arquitectura no es fácil

- Se requiere dinero y tiempo para comprar e instalar el equipo.
- Debes convertirte en un experto en la administración de datacenters.
- Calcular el número correcto de los diferentes componentes no es tarea fácil
  - Depende de la demanda



Virtualización - Máquinas virtuales y físicas



Computación en la nube

- Idea: Usar servidores alojados y administrador por alguien más
  - Usar Internet para acceder a ellos.
- La virtualización es la clave
  - Se especifica el almacenamiento, la comunicación y el poder de cómputo y nube hace el resto.
- Ejemplos:
  - Google Cloud
  - Amazon EC2
  - Microsoft Azure
  - Y muchos otros

Las ventajas de la computación en la nube

- Clave: Pagar por los recursos que se utilizan
  - Sin costo inicial de capital.
  - Se necesitan 1000 máquinas ahora?
  - Es perfecta para start-ups.
- La facturación está basada en los recursos
  - CPUs, memoria, almacenamiento, red.
- Ejecución extremadamente eficiente
  - Al comprar en alto volúmenes se pueden obtener descuentos.
  - No se necesita contratar a muchos expertos.
  - Los servidores están localizados donde los costos son bajos.

Interfaz de alto nivel en servicios en la nube

- Administrar el backend de una aplicación web usando máquinas virtuales requiere de capacidades técnicas de un administrador de sistemas.
- Si no las tienes, puedes usar alguna plataforma escalable. Ejemplo: Google App Engine.

Google App Engine

- Se puede utilizar Python, Java, PHP o Go.
- Google realiza el resto:
  - Asignar máquinas para ejecutar tu código.
  - Organizar asignaciones de nombres para que las solicitudes HTTP encuentren su camino a tu código.
  - Escalar la asignación de las máquinas según demanda.
  - AppEngine también incluye un sistema de almacenamiento escalable.
- Link: https://cloud.google.com/appengine/

Computación en la nube y App Web

- El modelo pagar-por-recursos-usados funciona bien para muchas compañías
  - En algún punto si se requieren muchos recursos podría tener más sentido construir tu propio data center.
- Hay muchos servicios útiles disponibles:
  - Auto escalabilidad.
  - Distribución geográfica.
  - Monitorear y reportar.
  - Manejo de fallos.

Content Distribution Network (CDN)

- Una aplicación web tiene partes que sólo son de lectura
  - Los navegadores necesitan obtener el contenido pero no le importa de donde provenga.
- Red de distribución de contenido
  - Tiene muchos servidores localizados en muchos lugares del mundo.
  - Se proporciona el contenido (imágen, javascript, etc) y ellos la URL.
  - Agregas la URL en tu aplicación.
  - Cuando un usuario accede a la URL, ellos envían el archivo desde un servidor cercano (DNS)
- Beneficios:
  - Rápido para obtener contenido.
  - Reduce la carga del servidor.
- Sólo funciona cuando el contenido no cambia muy seguido.

¿Preguntas?