Diseño de Sistemas Distribuidos

Virginia Padilla

Universidad Nacional Experimental de Guayana

virginiapadillas@gmail.com

8 de abril de 2025

Contenido

Diseño de Sistemas Distribuidos

Principios

La siguiente sección describe los objetivos o aspectos que deben tomarse en cuenta cuando se diseña un sistema sistema distribuido.

Concurrencia

Concurrencia

Tanto los servicios como las aplicaciones proporcionan recursos que los clientes pueden compartir en un sistema distribuido. Por tanto, existe la posibilidad de que varios clientes intenten acceder a un recurso compartido al mismo tiempo. La concurrencia se hace más compleja cuando las actividades paralelas interactúan o comparten los mismos recursos.

Compartir Recursos

- Los recursos, como periféricos, bases de datos, bibliotecas, no se pueden replicar por completo en todos los sitios porque no resulta práctico ni rentable.
- Los recursos no se pueden colocar en un solo sitio porque el acceso a ese sitio puede resultar en un cuello de botella. Por lo tanto, estos recursos se distribuyen normalmente por todo el sistema.
- Por ejemplo, bases de datos distribuidas como MySQL Cluster o Ethereum, particionar los conjuntos de datos en varios servidores, y los replican para lograr un acceso rápido y confiabilidad en el servicio. Otro ejemplo es el trabajo cooperativo asistido por computadora (CSCW).

Transparencia en Sistemas Distribuidos

- La transparencia en los sistemas distribuidos es el ocultar del usuario y programador de la aplicación, la separación de los componentes en un sistema distribuido, de manera que el sistema se percibe como uno solo en vez de la colección de componentes independientes.
- Por ejemplo, no necesitan preocuparse por la ubicación o los detalles de cómo otros componentes acceden a sus operaciones, o si serán replicados o migrados.

Tipos de Transparencia

Transparencias

- Acceso
- Ubicación
- Red

Transparencia en el Acceso es permitir el acceso con las mismas operaciones de los recursos locales y remotos.
Transparencia en la ubicación es que los recursos sean alcanzados sin conocer su ubicación física ni de red
Transparencia en la red combina ambas transparencias: en el acceso y en la ubicación.

Tipos de Transparencia

Transparencias

- Concurrencia
- Replicación
- Fallas

La concurrencia permite que varios procesos operen simultaneamente usando recursos compartidos sin interferencia
En la replicación se permite que múltiples instancia de recursos sean usados para aumentar la confiabilidad y rendimiento sin que el usuario de aplicaciones conozca de ellas
Las fallas se ocultan, permitiendo que los usuarios y las aplicaciones culminen sus tareas sin percatarse de ellas.

Tipos de Transparencia

Transparencias

- Movilidad
- Rendimiento
- Escalamiento

La movilidad permite el movimiento de recursos y clientes dentro del sistema sin afectar la operación de usuarios o programas
El rendimiento permite que el sistema sea reconfigurado para mejorar el rendimiento cuando su carga varía
El escalamiento permite que el sistema y la aplicación se incremente sin cambios en la estructura del sistema o en los algoritmos de aplicación

Preguntas

Fallas en Sistemas Distribuidos

La transparencia en las fallas es uno de los principios que promete un sistemas distribuido. Investigue las técnicas que usan los sistemas distribuidos para ocultar las fallas

Sistemas Abiertos

Sistemas Abiertos

- Los sistemas abiertos se caracterizan por el hecho de que sus interfaces clave están publicadas
- Se basan en la provisión de un mecanismo de comunicación uniforme e interfaces publicadas para el acceso a recursos compartidos
- Pueden construir a partir de hardware y software heterogéneo, posiblemente de diferentes proveedores.

Sistemas Simples

Simplicidad

- Es importante que un diseño sea lo más simple posible y al mismo tiempo pueda satisfacer las necesidades del servicio.
- Los sistemas crecen y se vuelven más complejos con el tiempo. Comenzar con un sistema que ya es complejo significa comenzar en desventaja.
- Un sistema demasiado complejo da como resultado una situación en la que ninguna persona lo entiende todo al mismo tiempo.

Teorema CAP

CAP

El teorema CAP (Consistency, Avalaibility, Partition) establece que en un sistema distribuido, se puede cumplir como máximo con dos de estas propiedades: consistencia, disponibilidad y tolerancia de particion.

Consistencia

Ejemplo de consistencia en aplicaciones distribuidas [Sullivan, 2015]: si una empresa usa un servidor de base de datos de respaldo, cuando un usuario actualiza la cuenta de un cliente, esos mismos cambios se harían en el servidor de respaldo.

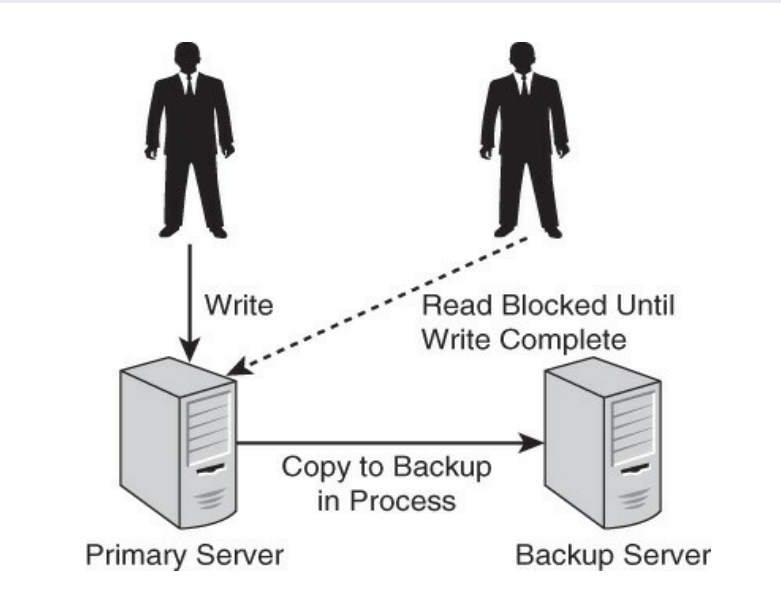
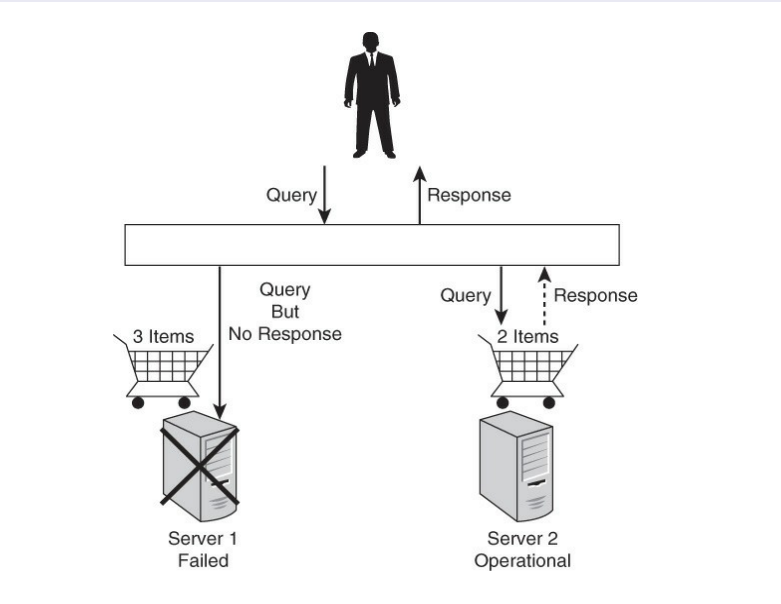


Figura: Consistencia

Disponibilidad

En el carrito de compras del e-commerce es posible tener una copia de seguridad de datos del carrito, no sincronizada con la copia principal. Los datos seguirían disponibles si el servidor primario falla, pero los datos del servidor de respaldo serían inconsistentes con los datos del servidor primario si el servidor primario falla antes de actualizar el servidor de respaldo.



Tolerancia a la Partición

El ejemplo más simple de tolerancia de particion es cuando el sistema continúa funcionando incluso si las máquinas involucradas en la prestación del servicio pierden la capacidad de comunicarse entre sí debido a que un enlace de red se cae.

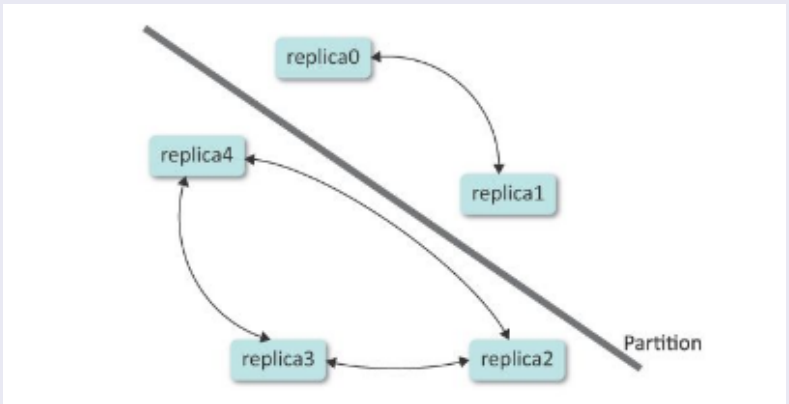


Figura: Tolerancia a la Partición

Relación entre los conceptos del Teorema CAP

La figura esquematiza estos conceptos y la relación entre ellos. Alli se resalta los puntos de intersección entre las propiedades definidas en el teorema `CAP`. Establece que no es posible construir un sistema distribuido que garantice los tres conceptos. Se pueden lograr uno o dos de ellos, pero no los tres simultáneamente.

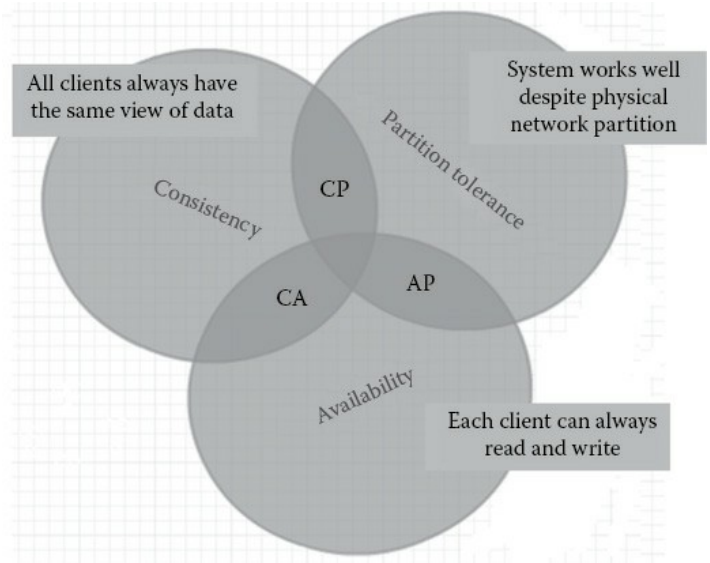


Figura: Relación entre conceptos del Teorema CAP

Ejemplo de Aplicación del Teorema CAP

Para una aplicación de red social, o de comercio electrónico, la solución es mantener la disponibilidad incluso si se sacrifica cierta consistencia entre los usuarios.
En el sistema financiero, requiere mantener la consistencia de datos sobre la disponibilidad de los mismos, por tanto las actualizaciones podrían requerir más tiempo para su ejecución.

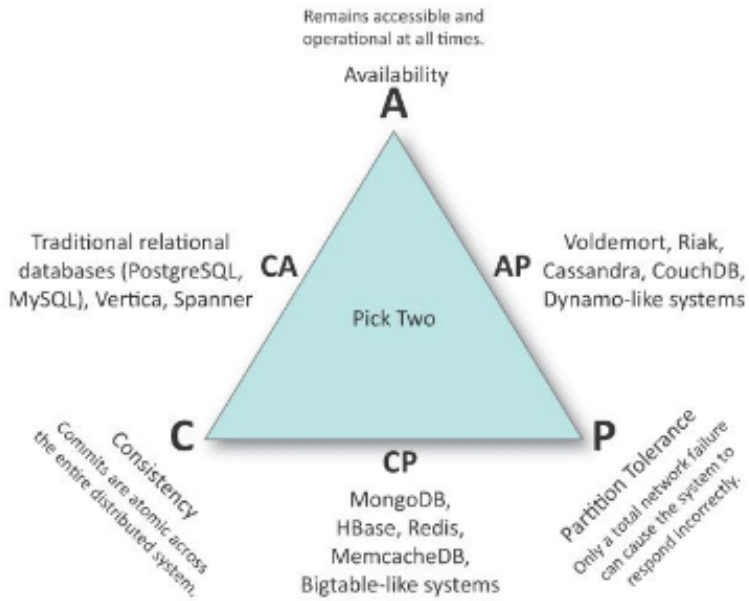


Figura: Significado del Teorema CAP

Teorema Cap

Ejemplos

1. Sistema Bancario (CP - Consistencia ante Particiones)
   Suponga: Un banco usa una base de datos distribuida para manejar saldos de cuentas. Hay 3 servidores (1, 2, 3) replicando datos. Un fallo en la red divide el sistema en 2 particiones:
   - Particìon A: 1 y 2 (conectados entre sí).
   - Particiòn B: 3 (aislado).
   - Si un cliente intenta retirar dinero: Si un cliente intenta retirar dinero en un cajero conectado a 3 (aislado), el sistema rechaza la operación porque no puede verificar el saldo actualizado en 1 y 2. (consistenci: evita saldos incorrectos.)

Teorema Cap

Ejemplos

1. Red Social (AP - Disponibilidad ante Particiones) Una app como Twitter o Facebook necesita estar siempre accesible, incluso si algunos servidores fallan. Hay 3 servidores (1, 2, 3) almacenando publicaciones. Ocurre una falla: 1 y 2 quedan aislados de 3. Si se hace una publicación:
   - Si el usuario está conectado a 1 o 2, su tweet se guarda solo en ese grupo.
   - Si otro usuario consulta desde 3, no verá el tweet nuevo hasta que la red se recupere. Aunque haya inconsistencia, la app responde siempre.

Teorema Cap

Ejemplos

1. Juego Online: Un videojuego multijugador necesita equilibrio entre consistencia y disponibilidad. Algunos servidores de juego pierden conexión con la base de datos central.
   - CP (Consistencia): Si un jugador compra un arma, el sistema bloquea la transacción hasta confirmar en todos los nodos (evita duplicados o fraudes).
   - AP (Disponibilidad): El juego sigue funcionando, pero algunos jugadores podrían ver items que luego desaparecen cuando se sincronizan los datos.

   Muchos juegos usan híbridos: Para datos crítico CP (para evitar fraudes) (ej: compras). Datos no críticos, AP para mantener activo el servicio (ej: chat del juego).

Teorema Cap

Ejemplos

1. Blockchain (Tolerancia a Particiones Extrema) Bitcoin y Ethereum son sistemas distribuidos globalmente.
   - Si Internet se divide en dos (América vs Asia), cada región sigue minando bloques.
   - - Solución (Consenso): Cuando la red se recupera, la cadena más larga gana (se descartan bloques de la partición "perdedora").
   - Prioriza: Tolerancia a Particiones (P) + Consistencia eventual (no inmediata).

   Blockchain sacrifica disponibilidad inmediata para mantener seguridad y descentralización.

Teorema Cap

Ejemplos

1. ¿ Y la propiedad AC? ¿ Es aplicable en Sistemas Distribuidos?

Acoplamiento

Acoplamiento

- El acoplamiento es la fuerza de la interconexión entre dos módulos de software: cuanto mayor es la fuerza de la interconexión, mayor es el acoplamiento.
- Acoplado débil si cada componente tiene poco o ningún conocimiento de las partes internas de los otros componentes.
- Acoplamiento fuerte:cada componente de un sistema tiene conocimiento de partes de otros componentes, como memoria común, almacenamient secundario,etc.

Preguntas

Acoplamiento débil vs acoplamiento Fuerte

1. Diferencias entre el acoplamiento fuerte versus el acoplamiento débil. Vea enlace sugerido en `https://www.geeksforgeeks.org/difference-between-loosely-coupled-and-tightly-coupled` ¿ Cuáles son estas diferencias?

Acoplamiento

Acoplamiento

- El acoplamiento es la fuerza de la interconexión entre dos módulos de software: cuanto mayor es la fuerza de la interconexión, mayor es el acoplamiento.
- Acoplado débil si cada componente tiene poco o ningún conocimiento de las partes internas de los otros componentes.
- Acoplamiento fuerte:cada componente de un sistema tiene conocimiento de partes de otros componentes, como memoria común, almacenamient secundario,etc.

Preguntas

Acoplamiento débil vs acoplamiento Fuerte

1. Diferencias entre el acoplamiento fuerte versus el acoplamiento débil. Vea enlace sugerido en `https://www.geeksforgeeks.org/difference-between-loosely-coupled-and-tightly-coupled` ¿ Cuáles son estas diferencias?

Escalabilidad

Sistema Escalable

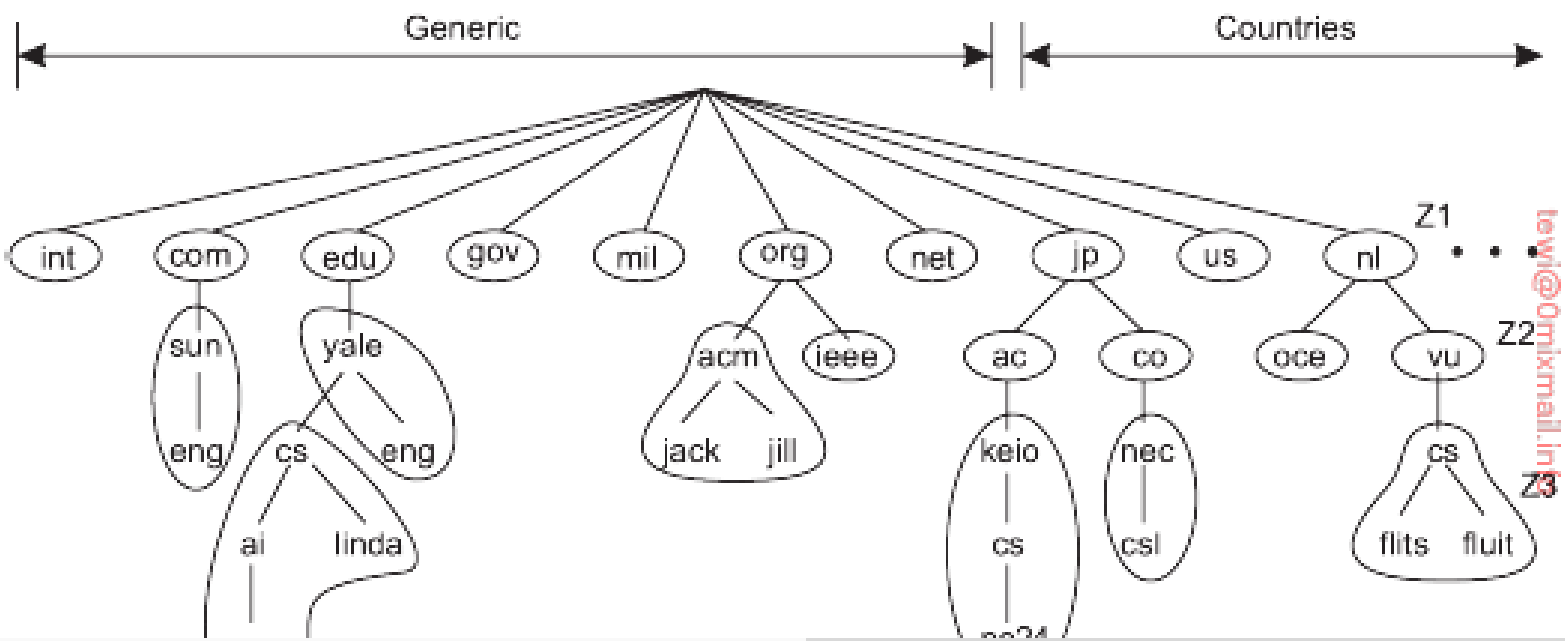
Un sistema es escalable si permanece efectivo cuando hay un aumento significativo en el número de recursos y usuarios, lo cual representa varios desafíos.

Los Desafíos:

- Controlar el costo de los recursos físicos
- Controlar la pérdida de rendimiento
- Evitar cuellos de botella en el rendimiento

Evitar Cuellos de botella en el Rendimiento

- Ejemplo: predecesor del sistema de dominio de nombres DNS
  - La tabla de nombres se mantuvo en un archivo maestro único que se descarga a cualquier computadora que lo necesite. Bien para solo unos cientos de computadoras en internet.
  - Luego, se eliminó este cuello de botella al dividir la tabla de nombres entre servidores ubicado en Internet y administrado localmente.



Heterogeneidad

Heterogeneidad

- Los sistemas distribuidos son heterogéneos, ya que se pueden construirse a partir de una variedad de redes, sistemas operativos, hardware informático y lenguajes de programación.
- También los codigo móviles o las máquinas virtual pueden formar parte de ellos.
- El middleware se usa para ocultar estar diferencias y permitir la comunicación y administración de datos entre aplicaciones distribuidas.

Manejo de Fallas

Fallas

- Las fallas en un sistema distribuido son parciales, es decir, algunos componentes fallan mientras otros continúan funcionando
- Existen técnicas para lidiar con fallas:
  - detección de la ocurrencia de fallas
  - enmascaramiento de las fallas
  - tolerancia a las fallas
  - recuperación luego de la ocurrencia de fallas
  - redundancia de rutas, caminos o servidores

Pregunta

Fallas en los Sistemas Distribuidos

Las fallas en los sistemas distribuidos puede atribuirse a la complejidad de la ingeniería de los mismos. En este documento puede revisar aspectos de este tema:
`https://aws.amazon.com/es/builders-library/challenges-with-distributed-systems/`

Pregunta

¿ Cuáles aspectos se resaltan en el documento?

Manejo de Fallas

Los ingenieros no pueden combinar las condiciones de error. En su lugar, deben considerar muchas combinaciones de fallas.

El resultado de cualquier operación de red puede ser DESCONOCIDO. En cuyo caso es posible que la solicitud haya fallado, se haya procesado correctamente o se haya recibido pero no procesado.

Los problemas distribuidos se producen en todos los niveles. No solo en los equipos físicos de nivel bajo, sin también, en los niveles lógicos del sistema distribuido.

Manejo de Fallas

Recursividad. Los problemas distribuidos empeoran en los niveles superiores del sistema, debido a la recursividad.

Aparición del error. Los errores distribuidos suelen aparecer mucho después de su implementación en un sistema.

Propagación del error. Los errores distribuidos se pueden propagar en todo el sistema.

Origen de la falla. Muchos de estos problemas provienen de las leyes físicas de las redes, que no se pueden cambiar.

Seguridad

Garantías de Seguridad

Es importante proporcionar garantías con respecto a cualidades asociadas al acceso del servicio, entre ellas el rendimiento, la seguridad y la confiabilidad.

- En un sistema distribuido, el cliente envia solicitudes de acceso a través de la red a un conjunto de servidores, o almacena sus datos en servicios en la nube.
- Ambos ejemplos requiere que se construyan apliaciones seguras con la información protegida en el tránsito del mensaje y en su sitio de almacenamiento.
- Si el almacenamiento está en la nube, es necesaria el uso de criptografía de datos en bases de datos en la nube.

References

Maarten Van Steen and Andrew Tanenbaum (2017)
Distributed Systems
Pearson Education, Inc.

Coulouris, George and Dollimore, Jean and Kindberg, Tim and Blair, Gordon (2011)
Distributed Systems: Concepts and Design
Addison-Wesley Publishing Company.

Andrew Tanenbaum and Van Marteen (2007)
Distributed Systems: Principles and Paradigms
Pearson Prentice Hall.

Veríssimo, Paulo and Rodrigues, Luís (2012)
Distributed Systems for System Architects
Springer.

Dan Sullivan (2015)
NoSQL for Mere Mortals
Pearson.

Fin