CC5303 – Sistemas Distribuidos

7.- Replicación

Parte 2

Sebastián Blasco V.

Consistencia Momentánea

Hasta qué punto los procesos en realidad operan de manera concurrente, y hasta qué punto la consistencia necesita garantizarse, puede variar

- Ej. Operaciones de **updates de una BD**
- Ej. DNS, nunca hay **conflictos W/W** por naturaleza del servicio
  - A lo más conflictos W/R que son aceptables
- Ej. La web, **cacheo de versiones** viejas que son (en cierta medida) aceptables.

Estos ejemplos pueden considerarse como casos de bases de datos replicadas y distribuidas (de gran escala) que toleran un relativamente alto grado de inconsistencia.

Consistencia Momentánea

Los ejemplos anteriores pueden considerarse como casos de bases de datos replicadas y distribuidas (de gran escala) que **toleran un relativamente alto grado de inconsistencia**.

Tienen en común que, si no ocurren actualizaciones durante mucho tiempo, **todas las réplicas gradualmente se volverán inconsistentes**.

Esta forma de consistencia se conoce como **consistencia momentánea**.

Consistencia Momentánea

Los almacenes de datos que son momentáneamente consistentes tienen la propiedad de que:

- En ausencia de actualizaciones, todas las réplicas convergen en copias idénticas unas de otras.

La consistencia momentánea sólo requiere la garantía de que las actualizaciones se propaguen a todas las réplicas.

Los conflictos de escritura-escritura con frecuencia son fáciles de resolver cuando se asume que **sólo un pequeño grupo de procesos puede realizar actualizaciones**.

La implementación de la consistencia momentánea es **barata**.

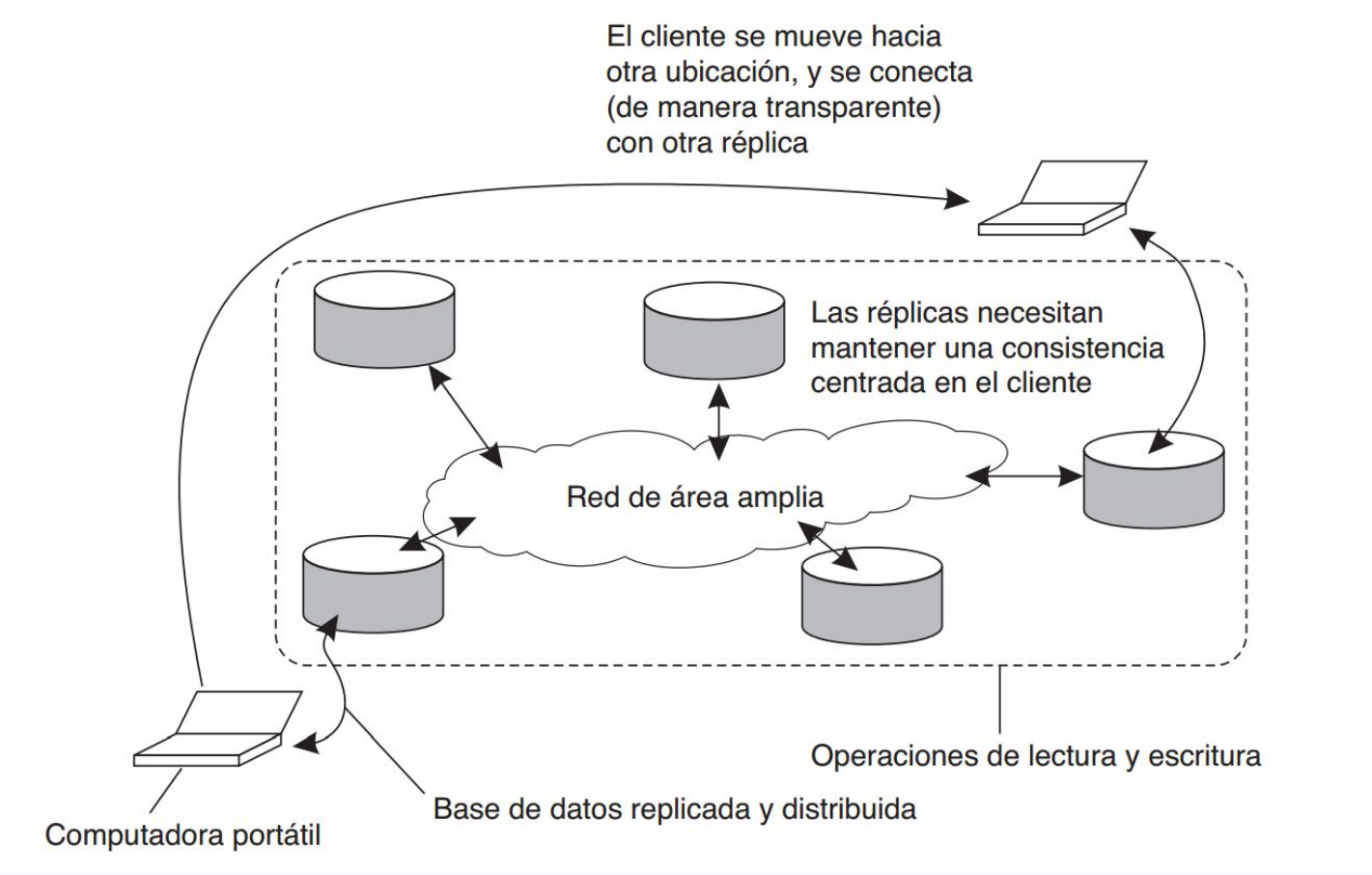
Consistencia Momentánea

Pero!

- Los almacenes de datos consistentes momentáneamente funcionan bien siempre y cuando los clientes siempre accedan a la misma réplica. Los problemas surgen cuando en un periodo corto se accede a réplicas diferentes

Ej. *clientes móviles* - si las actualizaciones realizadas antes aún no se han propagado, el usuario notará un comportamiento inconsistente

Consistencia Momentánea



Consistencia Momentánea

- Este problema se origina porque los usuarios en ocasiones pueden operar sobre réplicas diferentes.
- El problema puede aligerarse introduciendo la **consistencia centrada en el cliente**.

- A considerar:
  - Almacén de datos físicamente distribuido en varias máquinas.
  - Un proceso accede al almacén de datos, se conecta a la copia local (o más cercana) disponible.
  - Todas las operaciones de lectura y escritura se realizan en esa copia local.
  - Las actualizaciones se propagan, en algún momento, a las otras copias.
  - Suponemos que los elementos de datos tienen un propietario asociado, el cual es el único proceso que tiene permitido modificar ese elemento. (Evita conflicto W/W)

Consistencia Momentánea

- Notación
  - Sea ***xi[t]*** la versión del ítem de dato *x* en la copia local *Li* al tiempo *t*.
  - La versión *xi[t]* es es el resultado de una serie de operaciones de escritura sobre *Li* desde la inicialización, denotada ***WS(xi[t])***.
  - Si *WS(xi[t1])* fue además realizada sobre lo copia local *Lj* al tiempo *t2* entonces la serie se escribirá ***WS(xi[t1];xj[t2])***.
  - Si el orden de las operaciones o el tiempo son claros por el contexto, el índice del tiempo se omite.

Modelos de consistencia

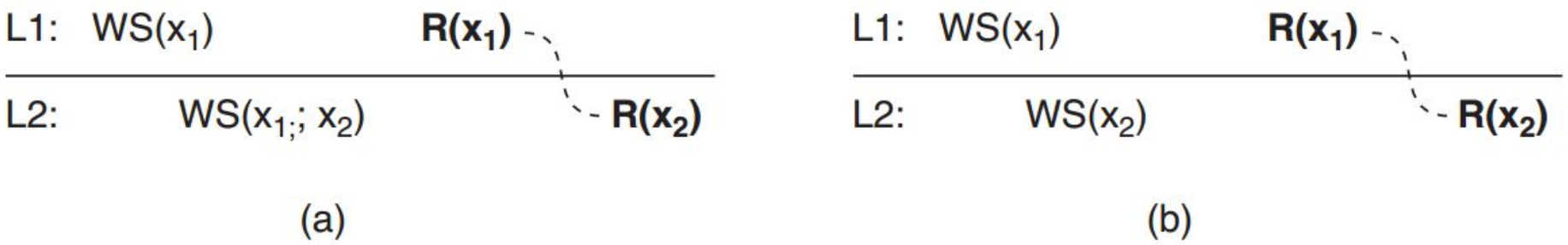
En la ausencia de un reloj global, es difícil definir precisamente cuál es la última operación de escritura. Como alternativa, necesitamos proporcionar otras definiciones, lo que nos lleva a una gama de modelos de consistencia.

- Centrados en los datos
- Centrados en el cliente
  - Lecturas monotónicas
  - Escrituras monotónicas
  - Lea sus escrituras
  - Las escrituras siguen a las lecturas

Lecturas monotónicas

- Un almacén de datos proporciona consistencia de lectura monotónica si se cumple la siguiente condición:
  - Si un proceso lee el valor de un elemento de datos x, cualquier operación de lectura sucesiva sobre x que haga ese proceso devolverá siempre el mismo valor o un valor más reciente.
- Más fácil:
  - "Si un proceso ha visto un valor de x al tiempo t, nunca verá una versión más vieja de x en un tiempo posterior."
- Ej. Correo Electrónico
  - Se descarga de L1,
  - Me muevo a usar L2
  - Usando L2, aún dispongo de lo obtenido por L1

Lecturas monotónicas



Si leí R(x_1) desde L1 me debo asegurar que leeré R(x_1) en L2.

Operaciones de lectura realizadas por un solo proceso, P, en dos diferentes copias del mismo almacén de datos.

- (a) Almacén de datos con consistencia de lectura monotónica.
- (b) Almacén de datos que no proporciona lecturas monotónicas.

Escrituras monotónicas

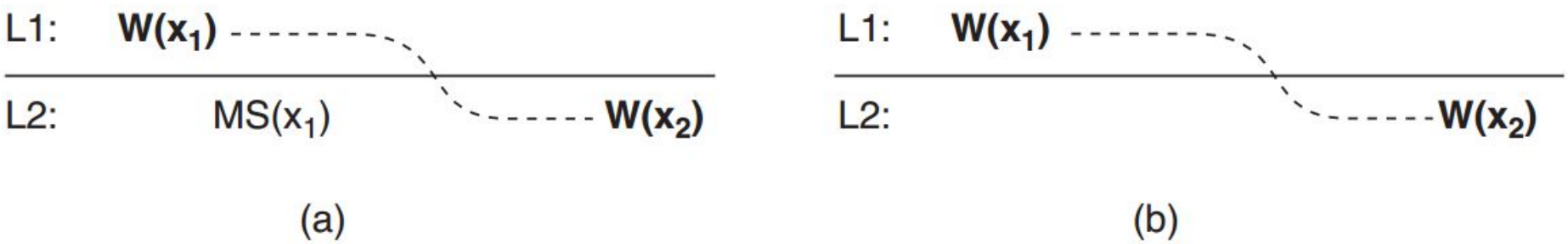
A veces es importante que las operaciones de escritura se propaguen en el orden correcto hacia todas las copias del almacén de datos.

- En un almacén con consistencia de escritura monotónica, se cumple la siguiente condición:
  - Una operación de escritura hecha por un proceso sobre un elemento x se completa antes que cualquier otra operación sucesiva de escritura sobre x realizada por el mismo proceso.
- Más fácil:
  - "Una operación de escritura sobre una copia del elemento x se realiza sólo si esa copia se ha actualizado mediante cualquier operación de escritura previa, la cual pudo haber ocurrido en otras copias de x. (Si es necesario, la nueva escritura debe esperar a que terminen otras escrituras anteriores.)"

Escrituras monotónicas

- Ej.
  - Consideremos una biblioteca de software. En muchos casos, la actualización de una biblioteca de este tipo se hace reemplazando una o más funciones, lo que deriva en una siguiente versión.
  - Con la consistencia de escritura monotónica se proporcionan garantías de que si una actualización se realiza sobre una copia de la biblioteca, todas las actualizaciones anteriores se realizarán primero.
  - La biblioteca resultante se volverá entonces la versión más reciente, e incluirá todas las actualizaciones que han dado pie a versiones anteriores de la biblioteca.

Escrituras monotónicas



Operaciones de escritura realizadas por un solo proceso P en dos copias locales diferentes del mismo almacén de datos.

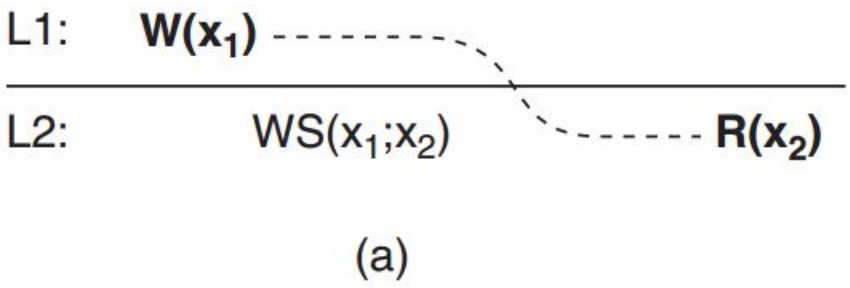
- (a) Almacén de datos con consistencia de escritura monotónica.
- (b) Almacén de datos que no proporciona consistencia de escritura monotónica.

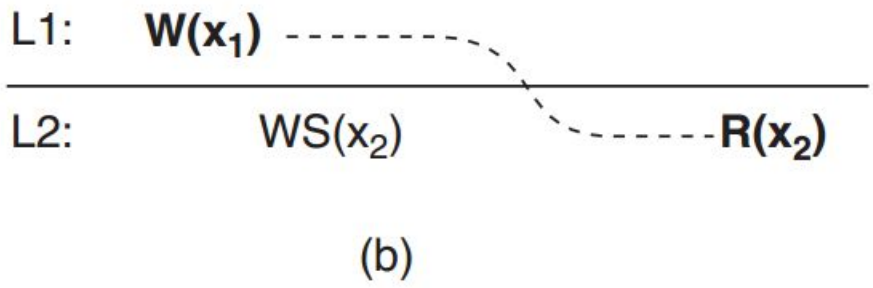
Lea sus escrituras

- Se dice que un almacén de datos proporciona consistencia lea sus escrituras si cumple la siguiente condición:
  - El efecto de una operación de escritura hecha por un proceso sobre un elemento de datos x siempre será visto por una operación de lectura sucesiva sobre x hecha por el mismo proceso.
- Más fácil:
  - "Una operación de escritura siempre se completa antes de una operación de lectura sucesiva del mismo proceso, independientemente del lugar donde ocurra la operación de lectura."

Ej. cambio de Contraseñas

Lea sus escrituras





Si escribí x1 en L1 me debo asegurar que lo lea desde L2.

- (a) Almacén de datos que proporciona consistencia lea sus escrituras.
- (b) Almacén de datos que no la proporciona.

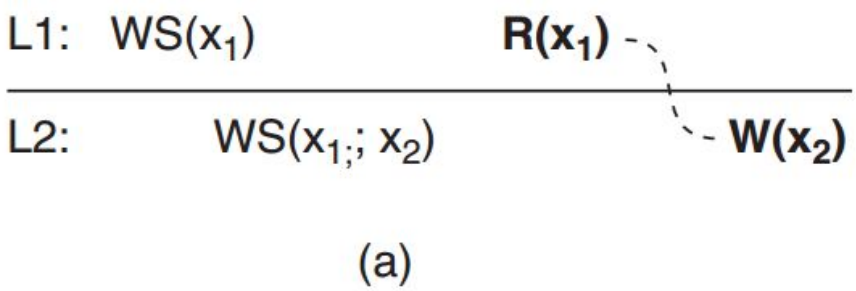
Las escrituras siguen a las lecturas

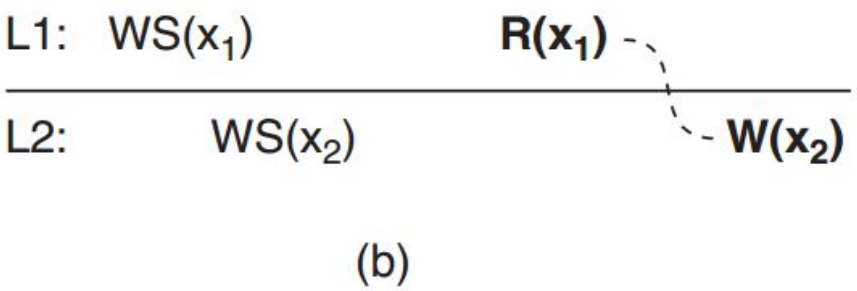
- Se dice que un almacén de datos proporciona consistencia las escrituras siguen a las lecturas si cumple con lo siguiente:
  - Se garantiza que la operación de escritura de un proceso sobre un elemento de datos x que sigue a una operación de lectura previa sobre x efectuada por el mismo proceso ocurrirá en el mismo o en el más reciente valor de x que se leyó.
- Más fácil:
  - "Cualquier operación sucesiva de un proceso sobre un elemento de datos x se realizará sobre una copia de x que está actualizada con el valor más recientemente leído por ese proceso."

Las escrituras siguen a las lecturas

- Ej. Grupo de noticias
  - Los usuarios de una red de un grupo de noticias vean el anuncio de una respuesta a un artículo sólo después de que han visto el artículo original.
  - Suponga que un usuario lee primero el artículo A; después, reacciona publica la respuesta B.
  - Al requerir consistencia las escrituras siguen a las lecturas, B será escrita en cualquier copia del grupo de noticias sólo después de que A también sea escrita.
  - Los usuarios que sólo leen los artículos no necesariamente requieren un modelo específico de consistencia centrada en el cliente. La consistencia las escrituras siguen a las lecturas garantiza que las reacciones a los artículos se almacenan en una copia local sólo si el original también se almacena ahí.

Las escrituras siguen a las lecturas





Si leí x1 en L1 me debo asegurar que toda escritura posterior al menos lleve x1 en cualquier copia.

- (a) Almacén de datos con consistencia las escrituras siguen a las lecturas.
- (b) Almacén de datos que no proporciona consistencia las escrituras siguen a las lecturas.

Administración de Réplicas

Un punto clave para cualquier sistema distribuido que soporta la replicación es decidir **dónde, cuándo, y por quién** deben ubicarse las réplicas, y posteriormente cuáles **mecanismos utilizar para mantener consistentes** a dichas réplicas.

- El problema de ubicación, por sí mismo, debe dividirse en dos subproblemas:
  - Ubicación de servidores de réplicas
  - Ubicación de contenido

Ubicación de servidores de réplicas

- Tiene que ver con encontrar los mejores lugares para colocar un servidor que pueda hospedar (parte de) un almacén de datos.
- Existen varias formas de calcular la mejor ubicación de servidores de réplicas, pero todas se reducen a un **problema de optimización**.
  - Se necesita seleccionar la mejor K de entre N ubicaciones (K < N). Se sabe que estos problemas son de cómputo complejo, y que sólo pueden resolverse

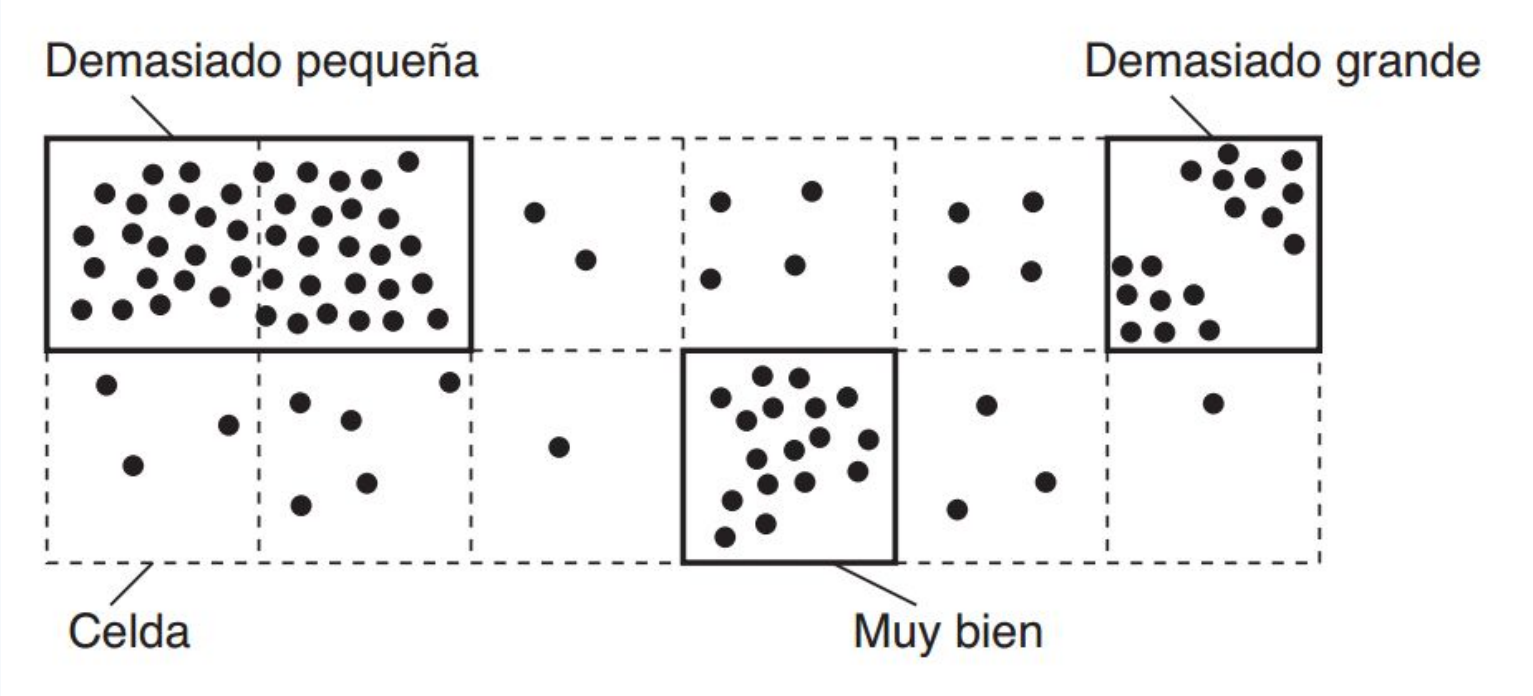
Ubicación de servidores de réplicas

- Szymaniak y colaboradores (2006)
  - Método mediante el cual puede identificarse rápidamente una región para ubicar réplicas.
  - Una región se identifica para ser una colección de nodos que accede al mismo contenido, pero para la cual la latencia internodal es baja.
  - El objetivo del algoritmo es seleccionar primero las regiones con más demanda y después dejar que uno de los nodos de esa región actúe como servidor de réplicas. Con este fin, se asume que los nodos están posicionados en un espacio geométrico m dimensional.

Ubicación de servidores de réplicas

- Szymaniak y colaboradores (2006)
  - La idea básica es identificar los K cluster más grandes y asignar un nodo de cada cluster para que hospede el contenido replicado.
  - Para identificar estos cluster, todo el espacio se divide en celdas.
  - Las K celdas más densas se eligen entonces para colocar un servidor de réplicas.

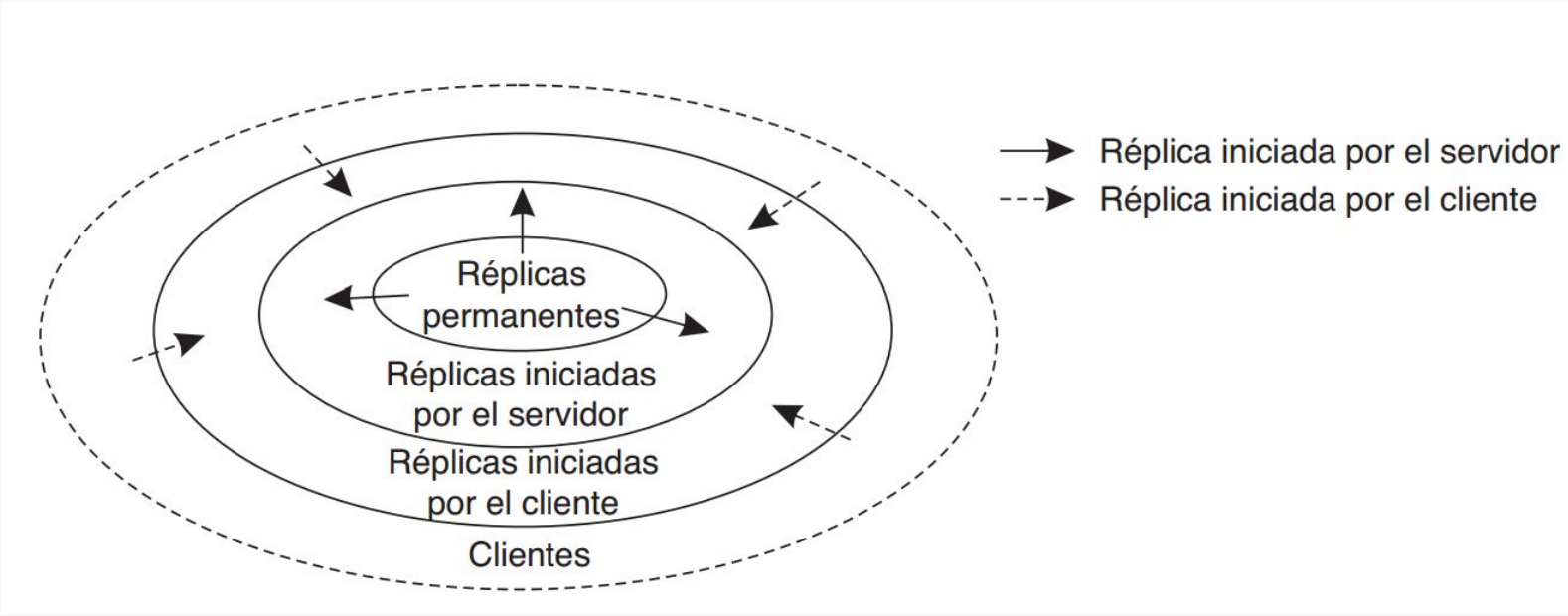
Ubicación de servidores de réplicas



Elección del tamaño adecuado de una celda para ubicar un servidor.

Ubicación de contenido

- Se relaciona con encontrar a los mejores servidores para colocar el contenido.
- Es posible diferenciar tres tipos de réplicas lógicamente organizadas:

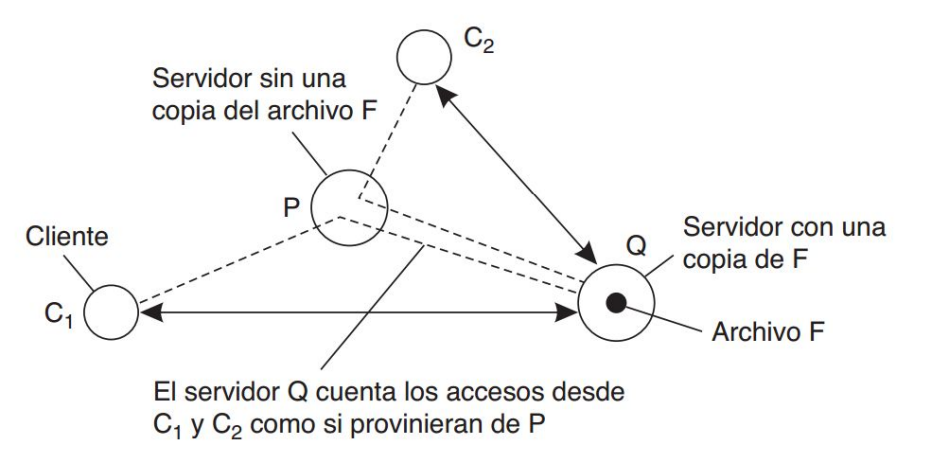


Ubicación de contenido

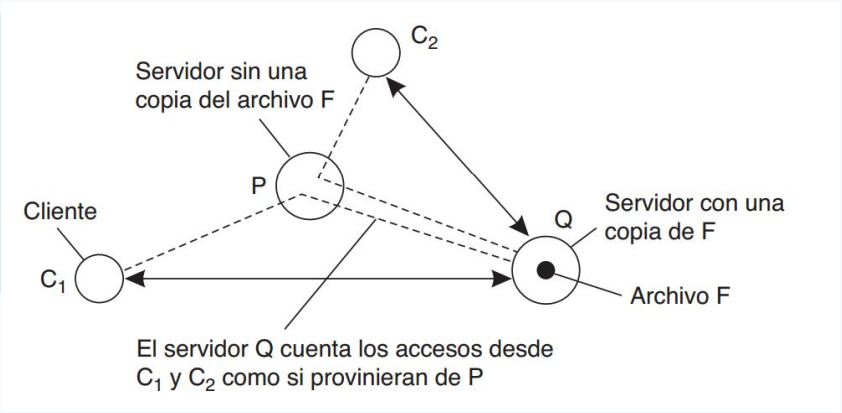
- Réplicas Permanentes
  - Es el primer subconjunto de réplicas.
  - Comúnmente son un número pequeño de réplicas.
  - Ejemplos:
    - Múltiples copias cercanas y los requerimientos son dirigidos a cada una a la vez mediante algún esquema (por ejemplo round-robin).
    - Múltiples copias lejanas a las cuales se elige entrar (mirror).

Ubicación de contenido

- Réplicas iniciadas por el Servidor
  - Copias del almacén de datos para mejorar la performance.
  - Réplicas creadas y actualizadas bajo la iniciativa del dueño del almacén de datos (servidor)
  - Problema
    - ¿Dónde poner las réplicas? (cerca/lejos)
    - ¿Cuándo actualizarlas?
  - Solución
    - Umbral de replicación



Ubicación de contenido



- Umbral de replicación
  - Cada servidor cuenta el número de accesos.
  - Cada cliente accede al servidor más cercano.
    - Si C1 y C2 comparten el servidor más cercano (P), y éste no tiene réplica, se toma como si P fuese quien realiza la consulta.
    - Si Q determina que el número de pedidos de F es menor a un cierto límite, se borra F de esa réplica (a menos que sea la última).
    - Si P tiene más de la mitad de los accesos a F, entonces se lleva F de Q a P.
    - Si P no puede recibir F (por ejemplo, no tiene espacio de disco suficiente), y el número de accesos es mayor que un cierto límite, F se replica a otros servidores: el más lejano que supere un cierto % de accesos a F.

Ubicación de contenido

- Réplicas iniciadas por el Cliente
  - Más conocidas como caché de cliente.
  - Copia temporal de datos para el uso del cliente.
  - Mejora el tiempo de acceso a datos.
  - Útil si la mayoría de las operaciones son de lectura.
  - Caché puede ser compartido entre un grupo de clientes cercanos.

Distribución (o propagación) de contenido

¿Qué propagar? hay 3 opciones:

1. Propagar solamente la **notificación de una actualización**.
   a. Protocolos de invalidación
   b. Mínimo uso de la red
   c. Necesidad de re obtención de datos
2. **Transferir datos** de una copia a otra.
   a. Uso intenso de la red
   b. Integridad y efectividad
3. **Propagar la operación de actualización** hacia otras copias.
   a. Replicación activa
   b. Menor uso de la red
   c. Al enviar orden de trabajo, implica trabajo en cada réplica

Distribución (o propagación) de contenido

- Protocolos *pull* versus protocolos *push*
  - Método basado en push (o protocolos basados en servidor), las actualizaciones se propagan hacia otras réplicas sin que éstas lo soliciten.
    - Se utilizan entre **réplicas permanentes y las iniciadas por servidor**.
    - Se aplican cuando las réplicas necesitan mantener un grado de consistencia relativamente alto, es decir, cuando se necesita que las réplicas se mantengan idénticas.
  - Método basado en pull (o protocolos basados en el cliente), un servidor o un cliente solicita a otro servidor que le envíe cualquier actualización que tenga hasta el momento.
    - Utilizados por **cachés cliente**.
    - Un método basado en pull es eficiente cuando la relación lectura-actualización es relativamente baja.

Distribución (o propagación) de contenido

- Protocolos *pull* versus protocolos *push*

Cuestión	Basado en push	Basado en pull
Estado en el servidor	Lista de réplicas cliente y cachés	Ninguno
Mensajes enviados	Actualiza (y posiblemente trae la actualización después)	Sondea y actualiza
Tiempo de respuesta en el cliente	Inmediato (o busca el tiempo de actualización)	Busca el tiempo de actualización

Comparación entre los protocolos basados en push y los basados en pull para el caso de sistemas de un solo servidor y varios clientes.