DEPARTAMENTO DE

INGENIERÍA INFORMÁTICA

UNIVERSIDAD DE SANTIAGO DE CHILE

DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

REPLICACIÓN Y CONSISTENCIA

13169

SISTEMAS DISTRIBUIDOS

Motivación

- Un tema importante dentro de los Sistemas Distribuidos es la **replicación de datos**.
- En general, los datos se replican para incrementar la **confiabilidad** o mejorar el **rendimiento del sistema**.
- Uno de las principales consideraciones en la replicación es la **consistencia**.

Replicación

- La replicación sirve para incrementar la confiabilidad en un sistema. Es posible continuar trabajando con el sistema ante la falla de una réplica. También ayuda a realizar una mejor protección contra datos corruptos.

- Otra razón para replicar es el rendimiento. Cuando el sistema necesita escalar en número de solicitudes que puede responder y en área geográfica, se puede mejorar el rendimiento replicando el servidor y, posteriormente, dividendo el trabajo.

- El costo tras la replicación es la **consistencia**. La consistencia se hace necesaria para mantener todas las réplicas con la misma información. El precio de la replicación lo determinan exactamente el cuándo y el cómo deben realizarse dichas modificaciones.

Replicación

- La replicación se utiliza, también, como una técnica de escalamiento.
- Colocar réplicas cerca de los procesos que la utilizan puede mejorar el rendimiento mediante la reducción del tiempo de acceso.
- Como contraparte, se necesita mayor ancho de banda de la red para mantener las copias actualizadas.
- El cómo y cuando sincronizar las réplicas, para mantener una consistencia entre las copias, es tema de interés y análisis.
- Mantener réplicas consistentes puede ser costoso y puede derivar en accesos aún más lentos que cuando no se consideran réplicas. Se puede relajar esta consistencia dependiendo del uso y finalidad de los datos.

Consistencia

- La consistencia se genera en el **almacén de datos**, el cual está físicamente distribuido en varias máquinas.
- Se asume que todo proceso que puede acceder a datos del almacén, tiene una copia local (o en las cercanías) disponible de todo el almacén.
- Las operaciones de escritura se propagan hacia el resto de copias.

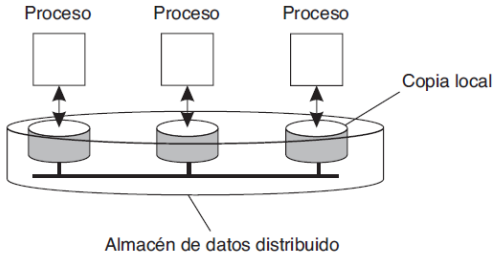


Figura 7-1. Organización general de un almacén de datos lógicos, físicamente distribuido y replicado a través de múltiples procesos.

Consistencia

- Un **modelo de consistencia** es básicamente un contrato entre los procesos y el almacén de datos.
- Dicho contrato indica que si los procesos involucrados siguen ciertas reglas, el almacén funcionará de manera correcta.
- En ausencia de un reloj global, es difícil determinar precisamente cuál es la última operación de escritura realizada.
- Como alternativa, es necesario proporcionar otras definiciones, lo cual lleva a distintos modelos de consistencia.

Consistencia

Consistencia Continua

- No existen reglas generales para relajar la inconsistencia. Lo que se puede tolerar exactamente depende, en gran medida, de las aplicaciones.

- Existen distintas formas en que las aplicaciones especifican las inconsistencias que pueden tolerar.

- La **consistencia continua** considera un método general para diferenciar tres ejes independientes para definir inconsistencias: desviación en valores numéricos entre réplicas, desviación en el deterioro entre réplicas y desviación con respecto al ordenamiento de operaciones de actualización.

Consistencia

Consistencia Continua

- La inconsistencia en términos de desviaciones numéricas puede utilizarse en aplicaciones para que las que los datos tienen semántica numéricas. Por ejemplo, en la replicación de registros que contienen precios de acciones.

- Se pueden considerar **desviaciones numéricas absolutas** o **desviaciones numéricas relativas**.

- La desviación numérica también se puede comprender en términos del número de actualizaciones que se han aplicado en una réplica dada. Por ejemplo, una caché web puede no haber visto un lote de operaciones realizadas por un servidor web. En este caso, la desviación asociada se conoce como **ponderación**.

Consistencia

Consistencia Continua

- Las desviaciones viejas guardan relación con la última vez que actualizó una réplica. Para algunas aplicaciones, es tolerable que una réplica proporcione datos viejos siempre y cuando no sean *demasiado* viejos. Por ejemplo, las aplicaciones sobre informes de clima pueden presentar datos un poco desactualizados respecto al servidor original.

- También existen aplicaciones en que se permite que el ordenamiento de actualizaciones sea diferente en varias réplicas, siempre que las diferencias sean limitadas. Por ejemplo, se pueden aplicar los cambios a una copia local en espera de un acuerdo global entre las réplicas.

Consistencia

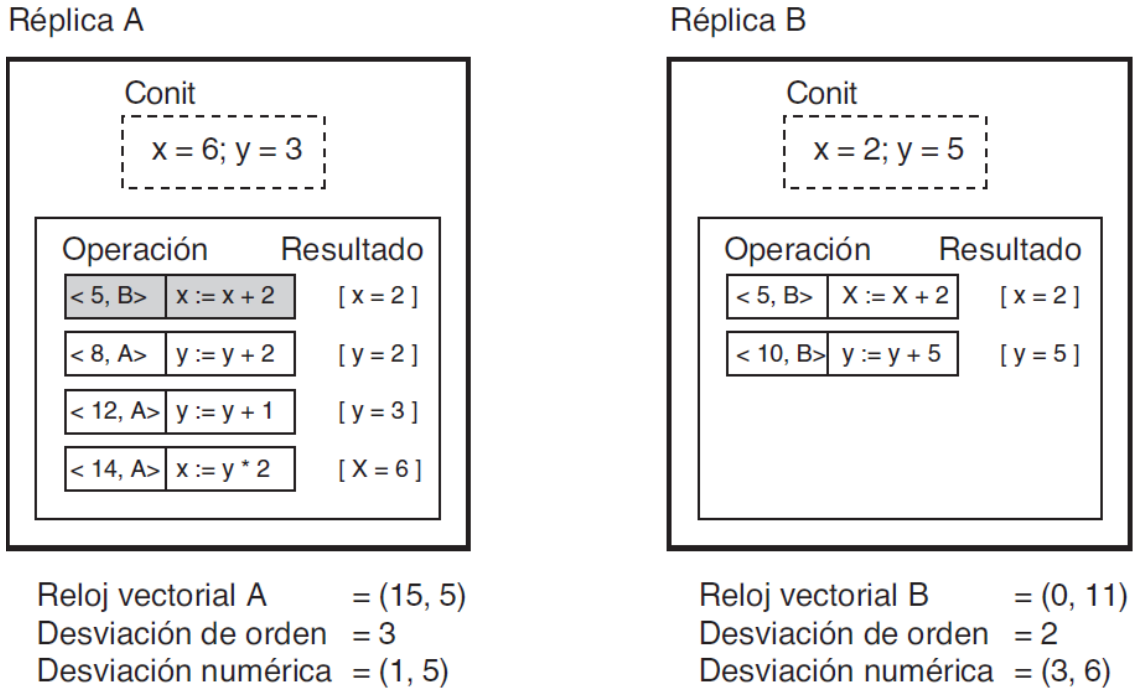
Conit

- Para definir la inconsistencia, se establece una unidad de consistencia, abreviada como **conit**, la cual especifica la unidad con la que se medirá la consistencia.

- Para dar un ejemplo, consideremos una situación de dos réplicas. Cada réplica *i* mantiene un reloj vectorial bidimensional VC_i. Se utiliza la notación *t,i* para expresar una operación que fue realizada por la réplica *i* en su tiempo lógico *t*.

Consistencia

Conit



Reloj vectorial A = (15, 5)
Desviación de orden = 3
Desviación numérica = (1, 5)

Reloj vectorial B = (0, 11)
Desviación de orden = 2
Desviación numérica = (3, 6)

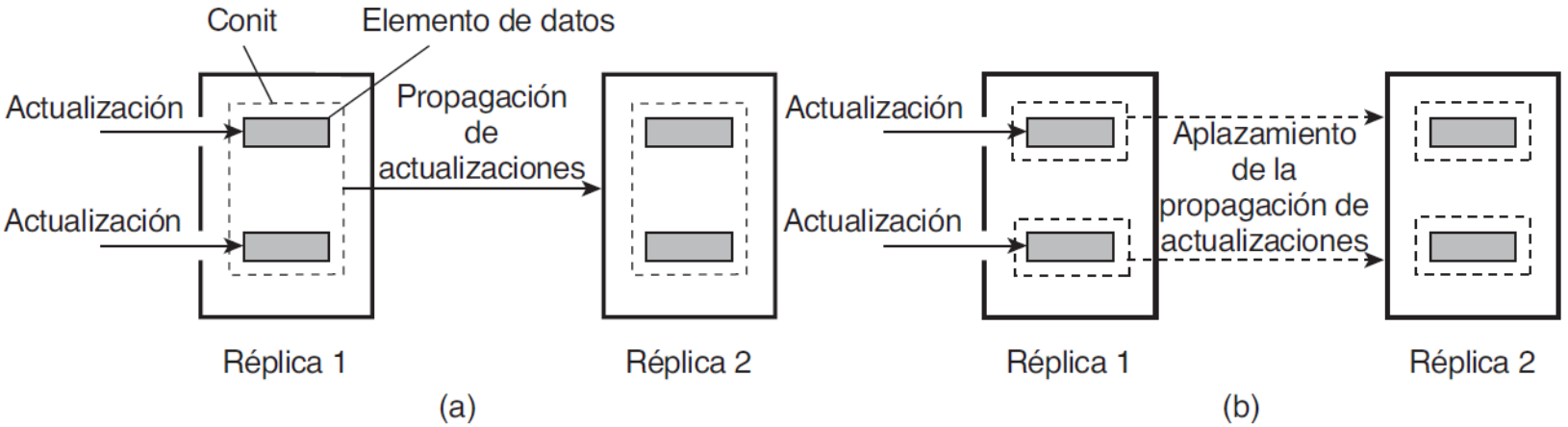
Consistencia

Conit

- Existen ventajas y desventajas tras la granularidad de los conit.
- Una granularidad gruesa, puede causar que las réplicas entren en inconsistencia rápidamente.
- Una granularidad fina y, por tanto, más conits, puede implicar una sobrecarga al sistema y una baja del rendimiento total.

Consistencia

Conit



Consistencia

Ordenamiento Consistente de Operaciones

- Es necesario analizar operaciones de ordenamiento consistente sobre datos compartidos y replicados.
- Cuando es necesario confirmar las actualizaciones en réplicas, éstas tendrán que acordar un ordenamiento global de las actualizaciones.
- Básicamente, necesitan acordar un ordenamiento consistente de dichas actualizaciones.

Consistencia

Ordenamiento Consistente de Operaciones – Consistencia Secuencial

- Se dice que un almacén de datos es secuencialmente consistente, si las operaciones de todos los procesos se ejecutan en algún orden secuencial y las operaciones de cada proceso individual aparecen en esa secuencia en el orden especificado en el programa.

- Esto quiere decir que cualquier interpolación de operaciones es válida, pero todos los procesos deben ver la misma interpolación de operaciones.

Consistencia

Ordenamiento Consistente de Operaciones – Consistencia Secuencial

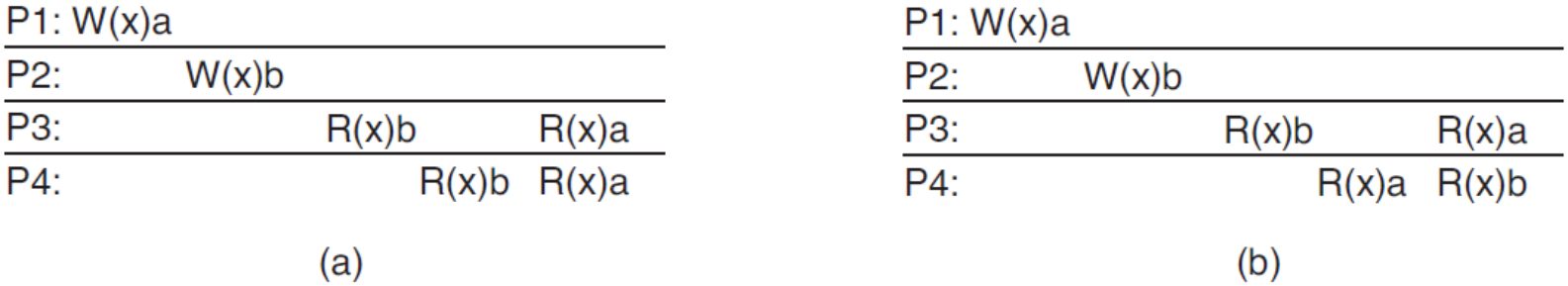


Figura 7-5. (a) Almacén de datos secuencialmente consistente. (b) Almacén de datos que no es secuencialmente consistente.

Consistencia

Ordenamiento Consistente de Operaciones – Consistencia Causal

- La consistencia causal, establece que las escrituras que están potencialmente relacionadas por la causalidad, deben ser vistas por todos los procesos en el mismo orden. Las escrituras concurrentes pueden verse en un orden diferente en diferentes máquinas.

- Implementar esta consistencia requiere dar seguimiento a cuáles procesos han visto cuáles escrituras, lo cual se puede lograr mediante un registro de tiempo vectorial.

Consistencia

Ordenamiento Consistente de Operaciones – Consistencia Causal

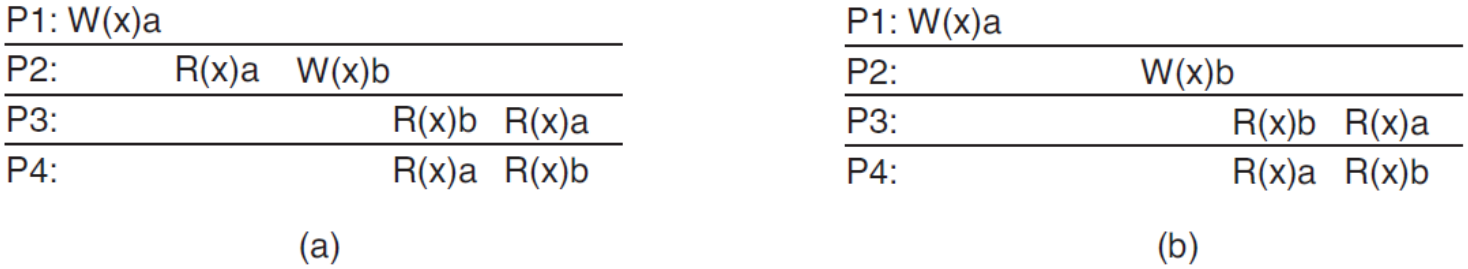


Figura 7-9. (a) Violación a un almacén causalmente consistente. (b) Secuencia de eventos correcta en un almacén causalmente consistente.

Consistencia

Ordenamiento Consistente de Operaciones – Consistencia de Entrada

- La granularidad fina provista por la consistencia causal y secuencial, no coincide en muchos casos con la granularidad provista por las aplicaciones.
- Lo que ocurre en estas aplicaciones, es que la concurrencia se mantiene bajo control a través de mecanismos de sincronización para exclusión mutua y transacciones.
- Para esto, se utilizan **variables de sincronización** compartidas

Consistencia

Ordenamiento Consistente de Operaciones – Consistencia de Entrada

P1:	Acq(Lx)	W(x)a		Acq(Ly)		W(y)b		Rel(Lx)		Rel(Ly)			
P2:									Acq(Lx)	R(x)a		R(y)	NIL
P3:											Acq(Ly)	R(y)b	

Figura 7-10. Secuencia de eventos válida para consistencia de entrada.

Consistencia Centrada en el Cliente

Consistencia momentánea

- En aplicaciones con bases de datos replicadas y distribuidas (de gran escala) que toleran un relativo alto grado de consistencia, se tiene en común que, si no ocurren actualizaciones durante mucho tiempo, todas las réplicas gradualmente se volverán inconsistentes. A esto se le conoce como **consistencia momentánea**.

- Dicha consistencia sólo requiere que las actualizaciones se propaguen a todas las réplicas.

- Esto funciona bastante bien si se asegura que los clientes siempre acceden a las mismas réplicas.

Consistencia Centrada en el Cliente

Consistencia momentánea

- Se puede producir un problema si la actualización no se propaga antes de que el cliente acceda a una nueva réplica.
- Este problema se puede aligerar introduciendo **consistencia centrada en el cliente**.
- En este caso, se garantiza consistencia para un cliente en particular.

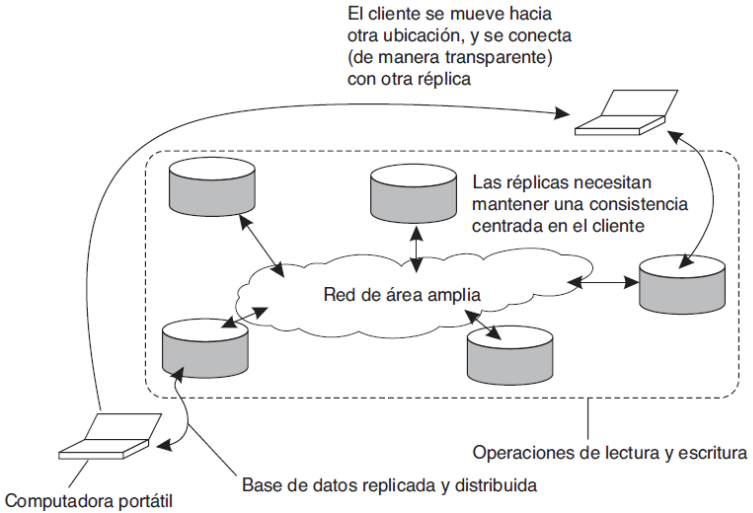


Figura 7-11. El principio referente a un usuario móvil que accede a diferentes réplicas de una base de datos distribuida.

Consistencia Centrada en el Cliente

Lecturas Monotónicas

- Un almacén de datos proporciona **lecturas monotónicas** si se cumple que, ante la lectura de un valor *x*, cualquier operación de lectura sucesiva sobre *x* que haga ese proceso devolverá siempre el mismo valor o un valor más reciente.

- Básicamente, la consistencia de lectura monotónica garantiza que si un proceso ha visto un valor *x* en el tiempo *t*, nunca verá una versión más vieja de *x* en un tiempo posterior.

- Por ejemplo, si consideramos una base de datos distribuida para un servicio de correo, el usuario necesita leer ciertos datos en Santiago y, si luego se mueve, debe leer los mismos datos en Calama. La consistencia de lectura monotónica asegurará esto.

Consistencia Centrada en el Cliente

Lecturas Monotónicas

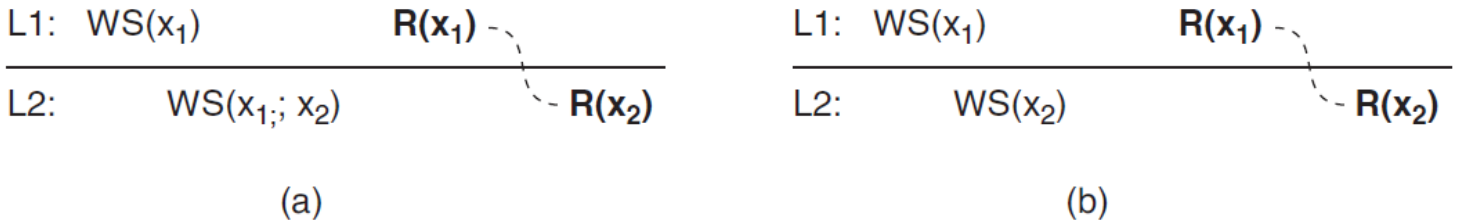


Figura 7-12. Operaciones de lectura realizadas por un solo proceso, *P*, en dos diferentes copias del mismo almacén de datos. (a) Almacén de datos con consistencia de lectura monotónica. (b) Almacén de datos que no proporciona lecturas monotónicas.

Consistencia Centrada en el Cliente

Escrituras Monotónicas

- Un almacén de datos con **consistencia de escritura monotónica**, se cumple que 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.

- Básicamente, 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 ocurrir en otra réplica de *x*.

- En caso de ser necesario, la nueva escritura debe esperar a que terminen otras escrituras anteriores.

Consistencia Centrada en el Cliente

Escrituras Monotónicas

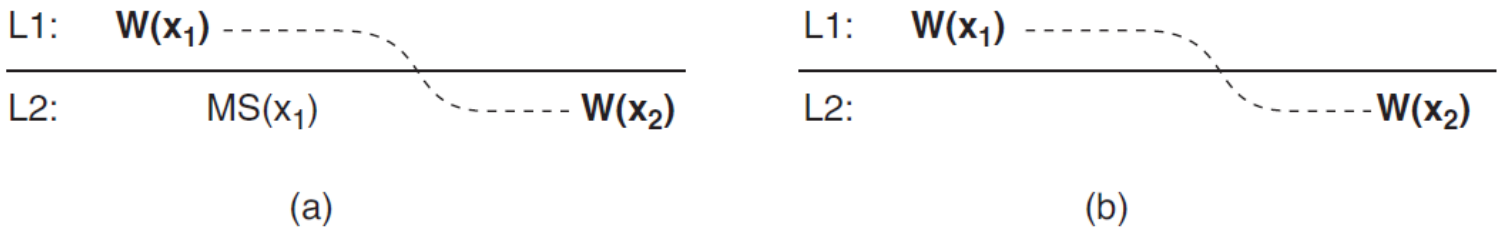


Figura 7-13. Operaciones de escritura realizadas por un solo proceso *P* en dos copias locales diferentes del mismo almacén de datos. (a) Almacén de datos con consistencia de escritura monotónica. (b) Almacén de datos que no proporciona consistencia de escritura monotónica.

Administración de Réplicas

- Se debe decidir dónde, cuándo y por quién deben ubicarse las réplicas y, posteriormente, cuáles mecanismos utilizar para mantener consistentes dichas réplicas.

- El problema de ubicación se divide en dos subproblemas, la ubicación de **servidores y réplicas** y la **ubicación de contenido**.

- La ubicación de servidores y réplicas tiene que ver con encontrar los mejores lugares para colocar un servidor que pueda hospedar un almacén de datos (o parte de dicho almacén).

- La ubicación de contenido se relaciona con encontrar a los mejores servidores para colocar el contenido.

Administración de Réplicas

Ubicación del Servidor de Réplicas

- En general, pasa más por una decisión económica que algorítmica o de optimización. Aún así, vale la pena destacar el análisis de propiedades del cliente y de la red para tomar las decisiones pertinentes.

- Se puede considerar como un problema de optimización en el que se deben seleccionar las mejores *K* ubicaciones de las posibles *N* ubicaciones.

- También se puede considerar la topología de la red y establecer criterios respecto a la respuesta de los clientes en términos de latencia, uso del ancho de banda y tiempos de respuesta.

Administración de Réplicas

Ubicación del Servidor de Réplicas

- Cada técnica a utilizar tiene ventajas y desventajas. Por sobre todo, son algoritmos de complejidad alta (del orden de N^2), siendo N la cantidad de ubicaciones por inspeccionar.

- Esto puede ser inaceptable si consideramos situaciones como las **flash crowds** (un súbito estallido de peticiones para un servicio específico).

Ubicación y Replicación de Contenido

- **Réplicas permanentes**: Se considera como el conjunto inicial de réplicas que constituyen un almacén de datos. Son un número pequeño de réplicas pequeñas.

Administración de Réplicas

Ubicación y Replicación de Contenido

- **Réplica iniciadas por servidores**: Son réplicas iniciadas por los servidores acorde a las necesidades del sistema. Estas réplicas se pueden convertir en permanentes si es necesario, pero en general existen sólo mientras la aplicación requiere de más recursos para su correcto funcionamiento.

- **Réplica iniciada por el cliente**: Se generan mediante los sistemas caché. Los datos se almacenan de manera local, con lo que se logra tener una mejora en el rendimiento del sistema que percibe el usuario. Los datos se mantienen por un tiempo limitado acorde a la administración del servidor.

Administración de Réplicas

Distribución de Contenido

- Trata sobre la actualización de contenidos en las réplicas del almacén de datos.
- Se puede decidir propagar sólo una notificación de actualización. Esto se conoce como **protocolos de invalidación**. Básicamente, se le indica a las réplicas qué datos están desactualizados, lo cual usa poco ancho de banda ya que sólo es necesario transmitir qué datos no son válidos y, luego, transferir dichos datos.
- Una segunda alternativa es propagar los datos modificados entre las réplicas.
- Como tercera opción, podemos considerar enviar las operaciones necesarias sobre los datos a actualizar. Esto se conoce como **replicación activa**, donde cada réplica tiene un proceso capaz de mantener actualizados “activamente” los datos mediante las operaciones necesarias.

DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Departamento de Ingeniería Informática
Universidad de Santiago de Chile

¿CONSULTAS?