07 - Consistencia y replicación

IIC2523 - Sistemas Distribuidos

Cristian Ruz – cruz@ing.puc.cl

Departamento de Ciencia de la Computación
Pontificia Universidad Católica de Chile

Semestre 2-2020

Contenidos

- Modelos de consistencia *data-centric*
- Modelos de consistencia *client-centric*

Rendimiento y escalabilidad

- Réplicas aumentan la disponibilidad (y la escalabilidad)
- Réplicas traen el problema de la consistencia

Operaciones conflictivas

Para mantener consistencia, se debe asegurar que todas las **operaciones conflictivas** se ejecuten en el mismo orden en cada réplica.

Operaciones conflictivas

- **Conflicto read-write**: operaciones de lectura y escritura concurrentes
- **Conflicto write-write**: dos operaciones de escritura concurrentes
- Ausencia de un coordinador global

¿Soluciones?

Garantizar orden global de operaciones conflictivas puede ser costoso, y afecta la escalabilidad.

Solución propuesta: relajar requisitos de consistencia

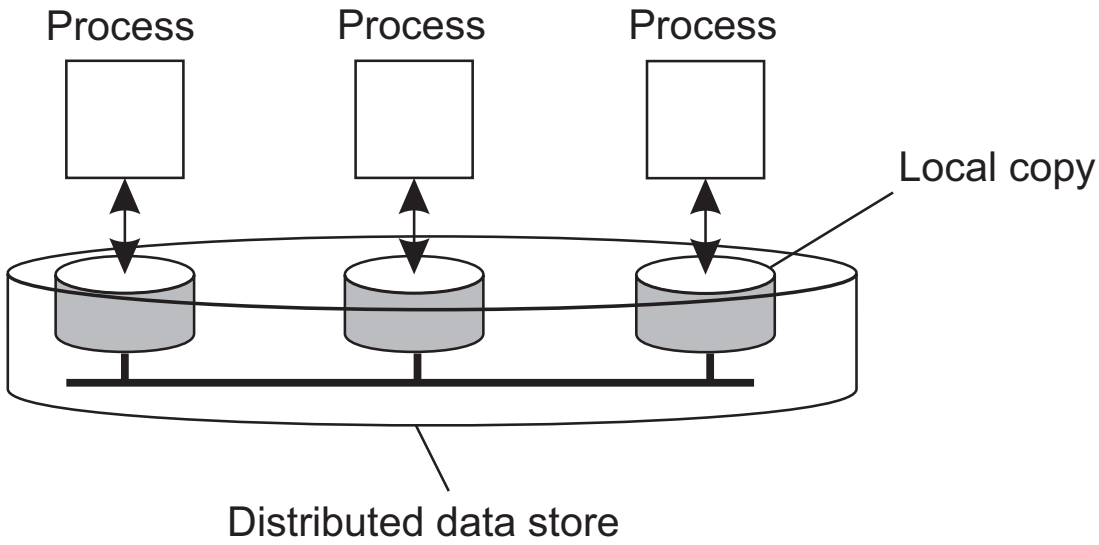
Modelos de consistencia *data-centric*

Modelo de consistencia

Modelo de consistencia

Contrato entre base de datos (distribuida) y procesos.
La base de datos determina de manera precisa como se comportan los resultados de operaciones conflictivas.

La base de datos es una colección distribuida de almacenamiento:



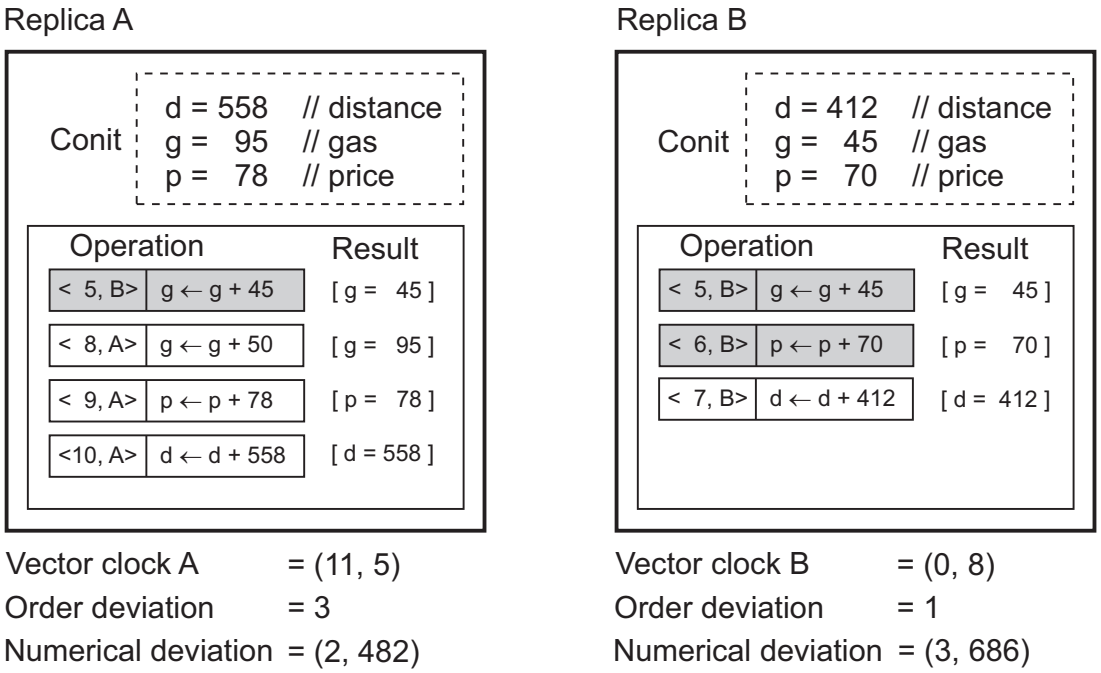
Consistencia contínua

Hay **grados de consistencia entre réplicas**

- Diferencias en valores numéricos
- Diferencias en periodos de actualización (*staleness*)
- Diferencias en el orden de ejecución de operaciones

Conit: unidad de consistencia (*consistency unit*). Unidad de datos sobre los cuales analizaremos consistencia.

Consistency unit



Las variables g, p, d forman un **conit**.

- Cada réplica tiene su **vector clock**
- B envía a A la operación $[\langle 5, B\rangle : g \leftarrow g + 45]$
- Operaciones en gris están *commited*
- A tiene tres operaciones pendientes de *commit*
- A no ha visto dos operaciones de B

Consistencia Secuencial

Consistencia Secuencial

El resultado de cualquier ejecución debe ser el mismo que si las operaciones de todos los procesos se ejecutasen en algún orden secuencial, y las operaciones de cada proceso individual deben ocurrir en el mismo orden que están especificadas en el programa

Consistencia Secuencial

Cualquier *interleaving* es válido, pero todos deben ver el mismo orden.

P1:	W(x)a			
P2:		W(x)b		
P3:			R(x)b	R(x)a
P4:			R(x)b	R(x)a

P1:	W(x)a			
P2:		W(x)b		
P3:			R(x)b	R(x)a
P4:			R(x)a	R(x)b

El primero **cumple consistencia secuencial**

El segundo **NO cumple consistencia secuencial**

Consistencia Secuencial

Process P_1	Process P_2	Process P_3
x ← 1;	y ← 1;	z ← 1;
print(y,z);	print(x,z);	print(x,y);

3 procesos, 2 operaciones cada uno. 90 ejecuciones mezcladas posibles.

Execution 1	Execution 2	Execution 3	Execution 4
P_1: x ← 1;	P_1: x ← 1;	P_2: y ← 1;	P_2: y ← 1;
P_1: print(y,z);	P_2: y ← 1;	P_3: z ← 1;	P_1: x ← 1;
P_2: y ← 1;	P_2: print(x,z);	P_3: print(x,y);	P_3: z ← 1;
P_2: print(x,z);	P_1: print(y,z);	P_2: print(x,z);	P_2: print(x,z);
P_3: z ← 1;	P_3: z ← 1;	P_1: x ← 1;	P_1: print(y,z);
P_3: print(x,y);	P_3: print(x,y);	P_1: print(y,z);	P_3: print(x,y);
Prints: 001011	*Prints:* 101011	*Prints:* 010111	*Prints:* 111111
Signature: 00 10 11	*Signature:* 10 10 11	*Signature:* 11 01 01	*Signature:* 11 11 11
(a)	(b)	(c)	(d)

Aquí se muestran cuatro de las 90 posibles. Menos de 64 resultados distintos.

Consistencia Causal

Consistencia Causal

Los *writes* que están **potencialmente causalmente relacionados** debe ser vistos en el mismo orden por todos los proceos. Los *writes* **concurrentes** puede ser visto en orden distinto por procesos diferentes.

P1:	W(x)a							W(x)c			
P2:			R(x)a		W(x)b						
P3:				R(x)a						R(x)c	R(x)b
P4:		R(x)a								R(x)b	R(x)c

Causal, pero no secuencial

Consistencia Causal

Consistencia Causal

Los *writes* que están **potencialmente causalmente relacionados** debe ser vistos en el mismo orden por todos los proceos. Los *writes* **concurrentes** puede ser visto en orden distinto por procesos diferentes.

P1: W(x)a				
P2:	R(x)a	W(x)b		
P3:			R(x)b	R(x)a
P4:			R(x)a	R(x)b

P1: W(x)a				
P2:		W(x)b		
P3:			R(x)b	R(x)a
P4:			R(x)a	R(x)b

El primero **NO cumple consistencia causal**
El segundo **cumple consistencia causal**

Consistencia Causal

Consistencia Causal

Los *writes* que están **potencialmente causalmente relacionados** debe ser vistos en el mismo orden por todos los proceos. Los *writes* **concurrentes** puede ser visto en orden distinto por procesos diferentes.

P1: W(x)a				
P2:	R(x)a	W(y)b		
P3:			R(y)b	R(x)?
P4:			R(x)a	R(y)?

¿Qué deberían indicar $R_3(x)$ ó $R_4(x)$?

Agrupamiento de operaciones

Podemos agrupar operaciones usando *locks*

- Acceso a *locks* usa **consistencia secuencial**
- No se puede acceder al *lock* hasta que todos los *write* se hayan completado
- No se puede acceder a los datos hasta que todos los acceso a *locks* han sido realizados.

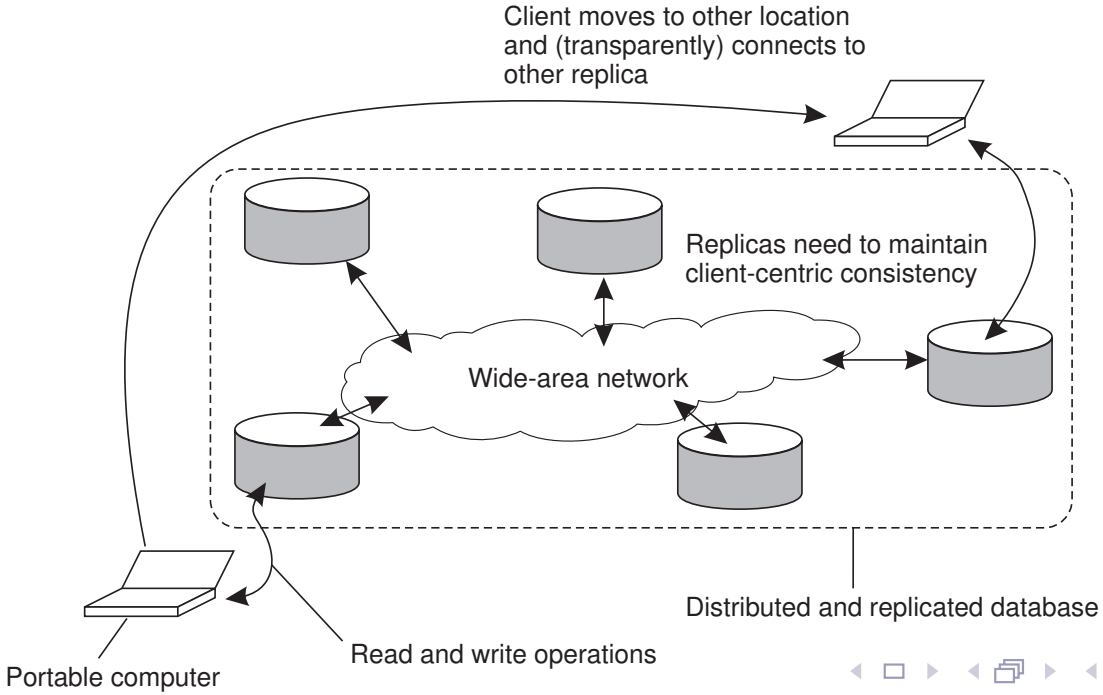
Entry Consistency

P1:	L(x)	W(x)a	L(y)	W(y)b	U(x)	U(y)					
P2:							L(x)	R(x)a		R(y)	NIL
P3:									L(y)	R(y)b	

Consistencia en usuarios móviles

Ejemplo: base de datos distribuida

- Se ejecutan lectura y escrituras desde la ubicación A
- Al llegar a la ubicación B puede pasar varias cosas:
  - Actualizaciones de A pueden no haber sido propagadas aún
  - Pueden haber entradas más recientes que las de A
  - Las actualizaciones en B pueden causar conflicto con las de A

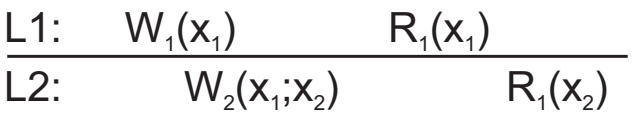


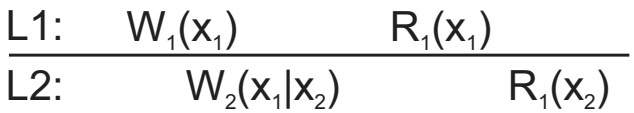
Lecturas monotónicas

Lecturas monotónicas

Si un proceso lee la variable `x`, cualquier operación sucesiva de lectura de `x` debe entregar el mismo valor o uno más reciente.

- Lectura en dos ubicaciones: L1 y L2
- Izq: con lecturas monotónicas
- Der: sin lecturas monotónicas





Notación

- $W_1(x_2)$: escritura de P_1 que lleva a versión x_2 de x
- $W_1(x_i; x_j)$: P_1 produce versión x_j basada en versión previa de x_i.
- $W_1(x_i|x_j)$: P_1 produce versión x_j de manera **concurrente** a x_i.

Ej: lectura de correo, lectura de agenda, desde distintos servidores

Escrituras monotónicas

Escrituras monotónicas

Una escritura de x por un proceso es completa antes de cualquier escritura posterior de x por el mismo proceso.

- (a) Consistente con escritura monotónica
- (b) No escritura monotónica
- (c) No escritura monotónica, pues $W_s(x_1|x_2)$ y $W_s(x_1|x_3)$
- (d) Consistente. $W_s(x_1; x_3)$ aunque x_1 ha sobreescrito aparentemente x_2

(a)

L1:	$W_1(x_1)$	
L2:	$W_2(x_1;x_2)$	$W_1(x_2;x_3)$

(b)

L1:	$W_1(x_1)$	
L2:	$W_2(x_1\|x_2)$	$W_1(x_1\|x_3)$

(c)

L1:	$W_1(x_1)$	
L2:	$W_2(x_1\|x_2)$	$W_1(x_2;x_3)$

(d)

L1:	$W_1(x_1)$	
L2:	$W_2(x_1\|x_2)$	$W_1(x_1;x_3)$

Ejemplos: mantener archivos replicados en el orden correcto en todos los repositorios, aunque implique enviar versiones antiguas

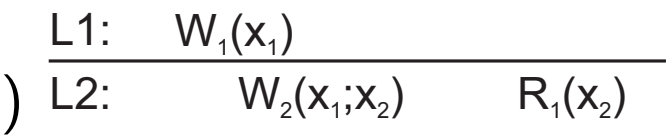
Read your writes

Read your writes

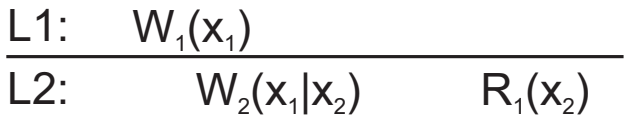
El efecto de un *write* de un proceso en `x`, siempre será visto por un *read* posterior de `x` en el mismo proceso.

- (a) Consistencia *read-your-writes*
- (b) No consistente bajo *read-your-writes*

(a)



(b)



Ejemplo: actualizar una página web, y asegurarse que el navegador muestre la última versión en lugar de una copia en caché

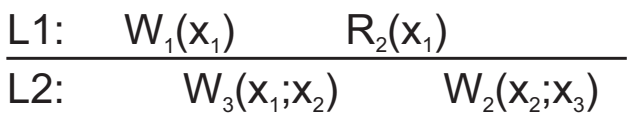
Writes follow reads

Writes follow readas

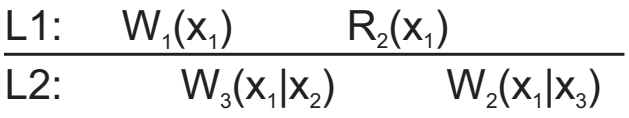
Una operación de escritura de un proceso en `x` que sigue a un *read* de `x` en el mismo procso, debe ocurrir garantizadamente sobre un valor de `x` igual o más reciente al que acaba de ser leido.

- (a) Consistencia *writes-follow-reads*
- (b) No consistente bajo *read your writes*

(a)



(b)



Ejemplo: ver reacciones a artículos solamente si se está viendo la publicación original