Replicación y consistencia

Tema 5

Modelos de consistencia

Índice

Bibliografía

[CJF09] Vicent Cholvi, Ernesto Jiménez, Antonio Fernández Anta: “Interconnection of distributed memory models”. J. Parallel Distrib. Comput. 69(3): 295-306 (2009)

[DGS85] Susan B. Davidson, Hector Garcia-Molina, Dale Skeen: “Consistency in Partitioned Networks”. ACM Comput. Surv. 17(3): 341-370 (1985)

[HW90] Maurice Herlihy, Jeannette M. Wing: “Linearizability: A Correctness Condition for Concurrent Objects”. ACM Trans. Program. Lang. Syst. 12(3): 463-492 (1990)

[Lam78] Leslie Lamport: “Time, Clocks, and the Ordering of Events in a Distributed System”. Commun. ACM 21(7): 558-565 (1978)

[Lam79] Leslie Lamport: “How to Make a Multiprocessor Computer That Correctly Executes Multiprocess Programs”. IEEE Trans. Computers 28(9): 690-691 (1979)

[Lam86] Leslie Lamport: “On Interprocess Communication”. Distributed Computing 1(2): 77-101 (1986)

[Mos93] David Mosberger: “Memory Consistency Models”, Operating Systems Review, 27(1):18-26, 1993.

[TDP+94] Douglas B. Terry, Alan J. Demers, Karin Petersen, Mike Spreitzer, Marvin Theimer, Brent B. Welch: “Session Guarantees for Weakly Consistent Replicated Data”. PDIS 1994: 140-149

[Vog09] Werner Vogels: “Eventually consistent”. Commun. ACM 52(1): 40-44 (2009)

Índice

1. Consistencia de datos

- Cuando se replica información entre diferentes nodos, un modelo de consistencia especifica qué divergencias se llegarán a admitir entre el estado de cada una de las réplicas de un mismo dato.
- Los modelos de consistencia se definieron inicialmente para arquitecturas multiprocesador con memoria compartida.
  - Pero los modelos también son aplicables a entornos distribuidos con información replicada.

1. Consistencia de datos

- En un entorno distribuido asumiremos replicación total (una réplica de cada dato en cada nodo):
  - Los clientes solicitan la escritura a un solo nodo.
  - Este nodo realiza la escritura localmente y después difunde el nuevo valor a las restantes.
  - Las lecturas sólo se realizan en un nodo y no se informa a los demás.

1. Consistencia de datos

- Tomamos como base a [Mos93]:
  - La consistencia depende de los retardos en la propagación de esas escrituras.
  - A menor grado de consistencia, mayor esfuerzo por parte de las aplicaciones.
  - Hay ciertos modelos de consistencia que no son implantables, ni siquiera en ese entorno más favorable.
  - Para especificar un modelo de consistencia hay que establecer un conjunto de “condiciones” a cumplir por el sistema.
    - Esto es aplicable tanto a los modelos centrados en datos como a los centrados en el usuario o cliente (a estudiar en la sección 2).

Índice

1.1. Consistencia estricta

- Cada operación (así como sus efectos) ocurre en una determinada unidad de tiempo.
  - La propagación es inmediata.
  - Las operaciones no pueden solaparse.
  - Es equivalente a un modelo centralizado.
  - También conocida como consistencia atómica.
  - Definida inicialmente por Lamport [Lam86].
  - La consistencia "*linearizable*" [HW90] es similar, pero asume objetos en lugar de posiciones de memoria.

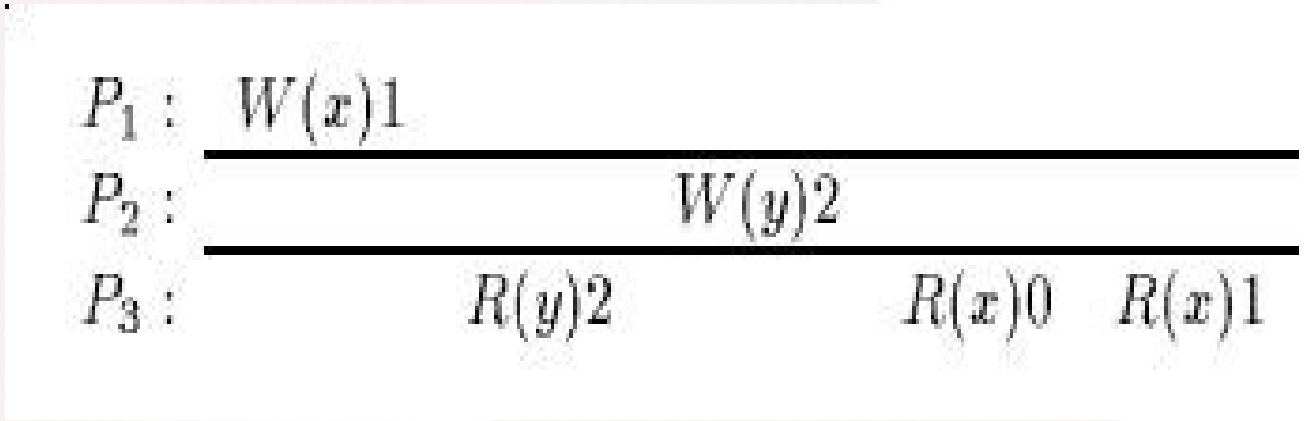
Índice

1.2. Consistencia secuencial

- Definida por Lamport en [Lam79]:
  - “El resultado de una ejecución es el mismo que si las operaciones de todos los procesos se ejecutaran en algún orden secuencial, y las operaciones de cada proceso aparecieran en dicha secuencia en el orden que dicta su respectivo programa”.
  - Podría implantarse difundiendo las escrituras en orden total FIFO.
  - Es el modelo que se proporciona en las especificaciones iniciales de los modelos de replicación activo y pasivo.

1.2. Consistencia secuencial

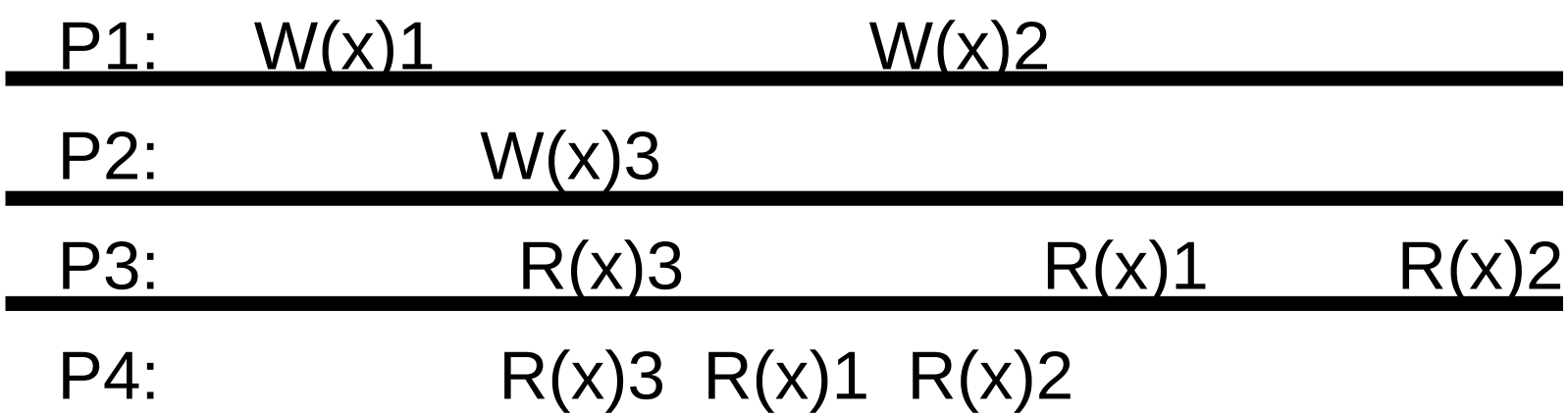
- Ejemplo de ejecución secuencial:



  - Como ambas escrituras se realizan sobre diferentes variables, no hay ningún orden entre ellas.

1.2. Consistencia secuencial

- Otro ejemplo:



- En este caso todas las escrituras se realizan sobre la misma variable.
- Todos los lectores deben observar las tres escrituras en el mismo orden.
- Nada evita que un proceso (P4) haya leído la última escritura antes de que otro proceso (P3) lea la anterior.
- Además, el orden seguido no coincide con el orden real de escritura.

Índice

1.3. Consistencia causal

- Sus ejecuciones respetan el orden causal de eventos definido por Lamport en su relación de orden "*happens before*" [Lam78]. Para ello:
  - Las escrituras se consideran envíos de mensaje.
  - Las lecturas se consideran recepciones de tales mensajes de escritura.

1.3. Consistencia causal

- Relación "*happens-before*":
  - Un evento "a" ocurre antes que un evento "b" ("a → b") cuando:
    - "a" y "b" se han ejecutado en un mismo proceso "p" en ese orden.
    - "a" es el evento de envío de un mensaje "m" y "b" es el evento de recepción de ese mismo mensaje "m".
    - Existe un evento "c" tal que "a → c" y "c → b".
  - Dos eventos "a" y "b" son concurrentes cuando:
    - No se da "a → b" ni "b → a".

1.3. Consistencia causal

- Ejemplo de ejecución causal:

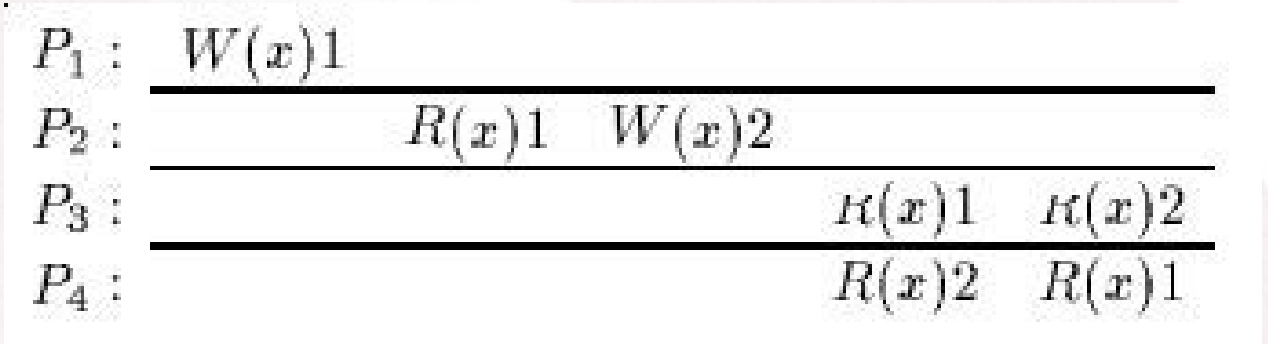
P_1 :	$W(x)1$		$W(x)3$	
P_2 :	$R(x)1$	$W(x)2$		
P_3 :	$R(x)1$		$R(x)3$	$R(x)2$
P_4 :	$R(x)1$		$R(x)2$	$R(x)3$

- El valor 1 precede causalmente al valor 2 y al valor 3.
- Pero los valores 2 y 3 son concurrentes.

Índice

1.4. Consistencia FIFO

- También conocida como consistencia PRAM.
- Todos los procesos deben observar las escrituras de cada proceso en el orden en que éste las realizó.
- Pero no tienen por qué respetar ninguna restricción al ordenar escrituras realizadas por distintos procesos.



1.4. Consistencia FIFO

- El orden de escritura de cada proceso afecta a todas las variables sobre las que haya escrito.
- Ejemplo:
  - P1:W(x)2, P1:W(y)3, P1:W(x)1, P2:R(x)2, P2:R(y)3, P2:R(x)1.
  - P2 no puede recibir la escritura del valor 1 sobre “x” antes de haber recibido la escritura del valor 3 sobre “y”.

Índice

1.5. Consistencia caché

- Relajación de la consistencia secuencial en la que cuando se acceda a diferentes variables ya no tendrá por qué respetarse el orden impuesto por el programa.
- Es decir, sólo hay que respetar un mismo orden cuando se consideren múltiples accesos de los procesos sobre la misma variable.

$$\begin{array}{lll} P_1: & W(x)1 & R(y)0 \\ \hline P_2: & W(y)1 & R(x)0 \end{array}$$

1.5. Consistencia caché

- La consistencia caché también respeta el “orden de programa” para las escrituras realizadas por un mismo proceso sobre una variable determinada.
- Ejemplo:
  - P1:W(x)1, P1:W(x)3, P2:W(x)2...
    - Los lectores pueden acordar que la secuencia de escritura sobre “x” es:
      - 1, 3, 2, o...
      - 1, 2, 3, o...
      - 2, 1, 3.
      - El 1 siempre antes que el 3.

1.5. Consistencia caché

- Ejemplo:
  - Caché no FIFO:
    - P1:W(x)1, P1:W(y)2, P2:W(x)3, P1:W(x)5, P2:W(y)1, P3:R(y)1, P3:R(x)1, P4:R(x)1, P4:R(x)5, P4:R(y)1, P3:R(x)5, P3:R(x)3, P3:R(y)2, P4:R(y)2, P4:R(x)5.
    - P3 y P4 acuerdan que:
      - Lecturas de 'x' en orden 1, 5, 3.
      - Lecturas de 'y' en orden 1, 2.
    - Pero la secuencia seguida en las lecturas no respeta el orden FIFO de P2.

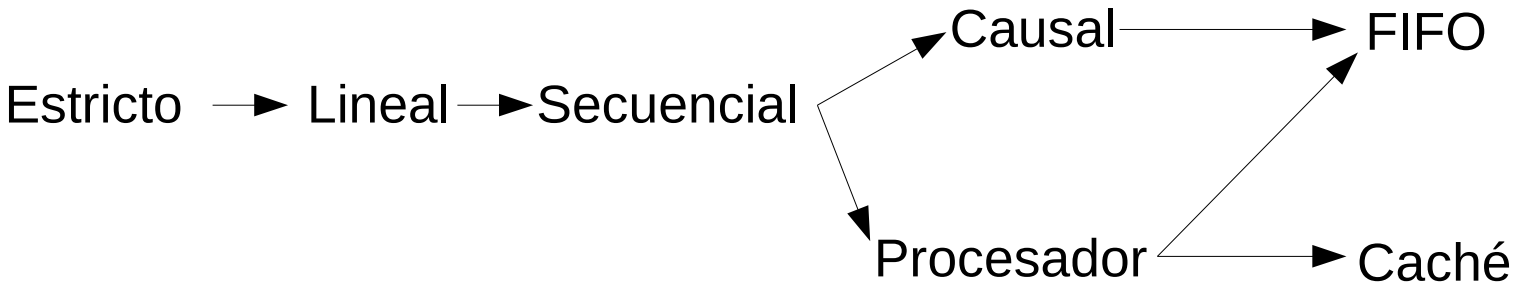
Índice

1.6. Otros modelos

- Basados en primitivas de sincronización:
  - Consistencia débil.
  - Consistencia “release”.
  - Consistencia “entry”.
- Modelos intermedios a los explicados.
  - “*Linearizable*”: entre el estricto y el secuencial.
  - Procesador: entre el secuencial y el caché. Cumple simultáneamente las condiciones del caché y el FIFO.

1.6. Otros modelos

- Jerarquía de modelos:



Índice

1. Consistencia centrada en datos
    1. Estricta
    2. Secuencial
    3. Causal
    4. FIFO
    5. Caché
    6. Otros modelos
    7. **Composición**
2. Consistencia centrada en cliente
3. Consistencia final

1.7. Composición

- Un modelo de consistencia se considera rápido ("*fast*") cuando la realización de un acceso no necesita consultar lo que hayan realizado otros procesos del sistema.
- Los modelos de consistencia rápidos pueden componerse entre sí [CJF09].
  - Es decir, podremos interconectar subsistemas que utilicen estos modelos para formar un sistema global con la misma consistencia.
  - Los protocolos de consistencia empleados en cada subsistema se podrán mantener.
  - Sólo se necesitará propagación FIFO de las escrituras entre subsistemas.

1.7. Composición

- Son modelos rápidos: causal, FIFO y caché.
- No son modelos rápidos: estricto, lineal, secuencial y procesador.
- Al interconectar modelos rápidos distintos se obtiene una consistencia global igual a la más relajada de las dos.
  - Ejemplos:
    - Causal + FIFO → FIFO
    - Caché + Causal → Caché
    - Causal + Causal → Causal

Índice

1. Consistencia centrada en datos
    1. Estricta
    2. Secuencial
    3. Causal
    4. FIFO
    5. Caché
    6. Otros modelos
    7. Composición
2. **Consistencia centrada en cliente**
3. Consistencia final

2. Consistencia cliente

- Si en lugar de centrarnos en los datos mantenidos en los servidores, nos centramos en las garantías de consistencia que espera un usuario, habrá que estudiar los “modelos de consistencia centrados en el cliente”.
- La “consistencia cliente” fue definida por Douglas B. Terry et al. en [TDP+94].
  - Está basada en el concepto de “sesión”.

2. Consistencia cliente

- Una sesión [TDP+94] es...:
  - "...una abstracción de la secuencia de lecturas y escrituras realizadas durante la ejecución de una aplicación".
  - En lugar de centrarse en los datos mantenidos por los servidores, su objetivo es la presentación a cada aplicación de una visión de la base de datos que sea consistente con sus propias acciones, aunque lean y escriban en diferentes servidores inconsistentes.

2. Consistencia cliente

- Para especificar las condiciones de consistencia, se asume que:
  - Existe una base de datos DB mantenida por un conjunto de servidores.
  - Sólo se consideran dos operaciones: R (lectura) y W (escritura).
  - Las transacciones se realizan en un único servidor, que después propagará las escrituras a las demás réplicas de manera perezosa.
  - DB(S,t) representa la secuencia ordenada de escrituras aplicadas en el servidor S hasta el instante t.

2. Consistencia cliente

- “Writeset” completo:
  - Un writeset WS se dice que es completo para la lectura R y DB(S,t) si y sólo si WS es un subconjunto de DB(S,t) y para cualquier conjunto WS2 que incluya a WS, el resultado de R aplicada a WS2 es idéntico al resultado de R aplicada a WS.
- La función *RelevantWrites(S,t,R)* retorna el writeset más pequeño que sea completo para una lectura R y DB(S,t).
- Todos los servidores deben aplicar las escrituras que no sean conmutativas en un mismo orden.
  - WriteOrder(W1,W2) es un predicado booleano que indica si la escritura W1 debe ser ordenada antes que W2.
  - El sistema asegurará que si WriteOrder(W1,W2) es cierto, entonces W1 será ordenada antes que W2 en DB(S) para todo servidor S que haya recibido ambas escrituras.

2. Consistencia cliente

- Las condiciones de consistencia son:
  - “Read your writes” (RYW): Si la lectura R sigue a la escritura W en una sesión y R se ejecuta en el servidor S en el instante t, entonces W está incluida en DB(S,t).
  - “Monotonic reads” (MR): Si la lectura R1 ocurre antes que R2 en una sesión y R1 accede al servidor S1 en el instante t1 y R2 accede al servidor S2 en el instante t2, entonces RelevantWrites(S1,t1,R1) es un subconjunto de DB(S2,t2).

2. Consistencia cliente

- “Writes follow reads” (WFR): Si la lectura R1 precede a la escritura W2 en una sesión y R1 se da en el servidor S1 en el instante t1, entonces, para cualquier servidor S2, si W2 está en DB(S2) entonces cualquier escritura W1 en RelevantWrites(S1,t1,R1) estará también en DB(S2) y WriteOrder(W1,W2) será cierto.
- “Monotonic writes” (MW): Si la escritura W1 precede a la escritura W2 en una sesión, entonces para todo servidor S2, si W2 se da en DB(S2) entonces W1 también estará en DB(S2) y WriteOrder(W1,W2) será cierto.

2. Consistencia cliente

- No hay una correspondencia exacta entre las cuatro condiciones de la consistencia cliente y los modelos de consistencia de datos.
- Una aproximación sería:
  - RYW y MR: No hay correspondencia directa. Con la consistencia lineal podría cumplirse, pero es un modelo “exigente”.
  - WFR: Consistencia causal.
  - MW: Consistencia FIFO.
- Pero además la consistencia cliente exige el mismo orden de aplicación en todas las réplicas entre los pares de escrituras no conmutativas.

Índice

1. Consistencia centrada en datos
    1. Estricta
    2. Secuencial
    3. Causal
    4. FIFO
    5. Caché
    6. Otros modelos
    7. Composición
2. Consistencia centrada en cliente
3. **Consistencia final**

3. Consistencia final

- La consistencia final (“eventual consistency”) fue definida de manera informal por Terry et al. en [TDP+94].
  - Aquella que se da cuando para cualquier par de servidores S1 y S2, DB(S1,t) no es necesariamente idéntico a DB(S2,t), pero los estados de S1 y S2 convergerán finalmente si no hay más escrituras.
  - Para cumplirlo, se necesita:
    - Propagación perezosa de todas las escrituras a todas las réplicas.
    - Aplicación en idéntico orden de los pares de escrituras no conmutativas.

3. Consistencia final

- La consistencia final ha sido identificada en [Vog09] como una característica clave para desarrollar sistemas distribuidos escalables.
- Como admite inconsistencias durante intervalos potencialmente largos, es también una forma de sobrellevar el teorema CAP [DGS85].