

Diseño y Arquitectura de Servicios Escalables

Unidad 2

Consistencia. Teorema CAP

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 eventual

4. Teorema CAP

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)
- [VV16] Paolo Viotti, Marko Vukolić: “Consistency in Non-Transactional Distributed Storage Systems”, ACM Comput. Surv. 49(1):19:1-19:34 (2016)

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 eventual

4. Teorema CAP

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 [Mos93,VV16]:
 - La consistencia depende de los retardos y el algoritmo utilizado en la propagación de esas escrituras.
 - A menor grado de consistencia, mayor esfuerzo por parte de las aplicaciones.
 - 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 cliente (a estudiar en la sección 2).
- 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.

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 eventual

4. Teorema CAP

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. 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 eventual

4. Teorema CAP

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:

$P_1 :$	$W(x)1$			
$P_2 :$		$W(y)2$		
$P_3 :$		$R(y)2$	$R(x)0$	$R(x)1$

- Como P_3 observa el orden $y=2, x=0, x=1$, entonces todos los demás procesos también tendrán que seguir ese mismo orden.

1.2. Consistencia secuencial

- Otro ejemplo:

P1:	W(x)1	W(x)2		
P2:	W(x)3			
P3:	R(x)3	R(x)1	R(x)2	
P4:	R(x)3	R(x)1	R(x)2	

- 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 tiene por qué coincidir con el orden cronológico de escritura.

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 eventual

4. Teorema CAP

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 esos mensajes de escritura.

1.3. Consistencia causal

- Relación “*happens-before*”:
 - Un evento “a” ocurre antes que un evento “b” (“a \rightarrow 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 \rightarrow c” y “c \rightarrow b”.
 - Dos eventos “a” y “b” son concurrentes cuando:
 - No se da “a \rightarrow b” ni “b \rightarrow 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. 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 eventual

4. Teorema CAP

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.

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

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. 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 eventual

4. Teorema CAP

1.5. Consistencia caché

- Relajación de la consistencia secuencial en la que cuando se acceda a diferentes variables ya no tendrá por qué acordar un orden común entre todos los procesos.
- Es decir, es un orden secuencial “por variable”.

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

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. 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 eventual

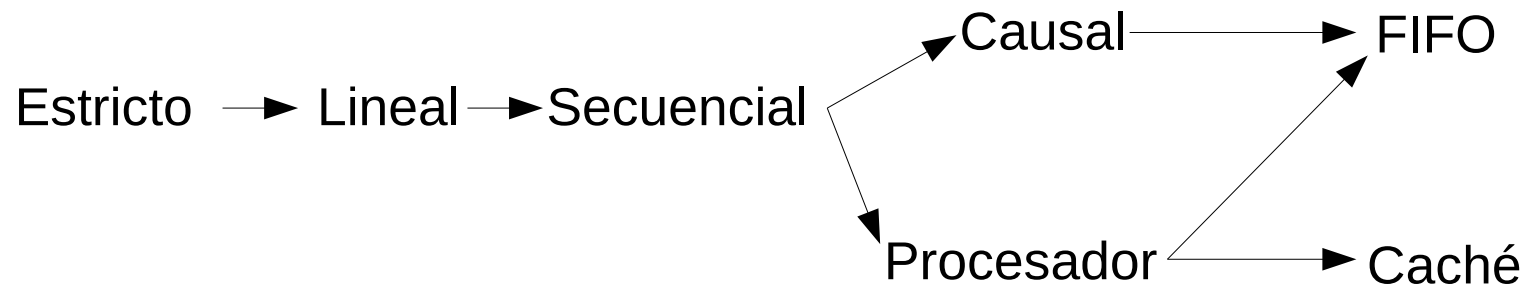
4. Teorema CAP

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 eventual

4. Teorema CAP

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 y FIFO.
- No son modelos rápidos: lineal, secuencial y procesador.
- Al interconectar modelos rápidos distintos se obtiene una consistencia global igual a la más relajada de las dos.
 - Ejemplo:
 - Causal + FIFO \rightarrow FIFO

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 eventual

4. Teorema CAP

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 el 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

- Si se considera la imagen percibida por cada proceso cliente como la imagen de un proceso servidor en un sistema con modelos de consistencia centrados en datos, se puede establecer la siguiente correspondencia:
 - $MW+RYW+MR = \text{FIFO}$
 - $MW+RYW+MR+WFR = \text{causal}$

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 eventual**

4. Teorema CAP

3. Consistencia eventual

- La consistencia eventual (“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 eventualmente si no hay más escrituras nuevas.
 - Y las ya iniciadas se propagan a todas las réplicas.
 - Para cumplirlo, se necesita:
 - Propagación perezosa de todas las escrituras a todas las réplicas.
 - Replicación multi-máster.
 - Aplicación en idéntico orden de los pares de escrituras no conmutativas.

3. Consistencia eventual

- La consistencia eventual 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].

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 eventual

4. Teorema CAP

Capítulo 2. Consistencia Teorema CAP

4. Teorema CAP



¡Teorema CAP!!

- Los servicios escalables están replicados:
 1. Hay que mantenerlos consistentes
 2. Hay que garantizar su disponibilidad
 3. Habría que soportar todas las situaciones de fallo
 - Incluyendo particionado de la red
 - Pérdida de conectividad entre subsistemas
- ¡Los protocolos necesarios suelen ser bloqueantes!!

4. Teorema CAP

- Teorema CAP [GL02]
 - No se puede garantizar simultáneamente:
 - Consistencia fuerte
 - Modelos “no rápidos”.
 - Disponibilidad de servicio
 - Tolerancia a las situaciones de partición de la red

4. Teorema CAP

- Solución
 - Las particiones se dan en sistemas de gran escala
 - Aunque se adopten todas las precauciones posibles para evitarlas
 - La disponibilidad es un requisito en un sistema escalable
 - ¡Hay que sacrificar la consistencia!
 - En lugar de consistencia fuerte, se garantizará consistencia eventual

4. Teorema CAP

- Consistencia eventual:
 - La propagación perezosa permite comunicación asíncrona
 - Las particiones de la red no suponen ningún problema
 - Esta consistencia relajada no introduce bloqueos en las actividades
 - Es justamente lo que se buscaba
 - Rendimiento óptimo
 - Para que sea viable...
 - Operaciones de escritura conmutativas entre sí

4. Teorema CAP

- En situaciones de particionado...
 - Cada subsistema se mantendrá disponible
 - Aceptará y ejecutará todas las modificaciones que soliciten los clientes
 - Al recuperar la conectividad...
 - No se necesita un protocolo de reconciliación intrusivo
 - Basta con que cada subsistema propague a los demás todo lo que haya hecho
 - Se aceptan y aplican estas operaciones en todos los subsistemas
 - Como las operaciones son conmutativas, el resultado eventual es idéntico en todo el sistema