Francisco Torres-Rojas, Mustaque Ahamad y Michel Raynal

Timed Consistency for Shared Distributed Objects

CARLOS MARTÍN FLORES GONZÁLEZ, Carné: 2015183528

Instituto Tecnológico de Costa Rica
Maestría en Computación
Sistemas Operativos Avanzados
Profesor: Francisco Torres Rojas, Ph.D

Ordenamiento y tiempo son dos aspectos diferentes de consistencia de objectos compartidos en sistemas distribuidos. Uno evita conflictos entre operaciones, el otro dice que tan rápido los efectos de una operación son percibidos por el resto del sistema.

Consistency Criteria: La historia global H de un sistema distribuido es un conjunto parcialmente ordenado de todas las operaciones que ocurren en todos los sitios del sistema. H_i es la secuencia de operaciones que son ejecutadas en el sitio i. Si a ocurre antes que b en H_i decimos que a precede a b en orden de programa. Se asume que todas las operaciones en H son **read** o **write**. A estas operaciones les toma un tiempo finito ejecutarse, por lo tanto, hay un intervalo que va desde el momento cuando un **read** o un **write** "inicia" al momento cuando alguna de estas operaciones "finaliza". Sin embargo, para los propósitos de la *timed consistency*, se asocia un instante a cada operación llamada el tiempo efectivo (*effective time*) de la operación. El *effective time* de una operación corresponde a algún instante entre su inicio y si fin. Si el tiempo efectivo de a es t, se dice que a fue "ejecutado" en el tiempo t. Si D es un conjunto de operaciones, entonces la serialización de D es una secuencia linear S conteniendo exactamente todas las operaciones de D tal que cada operación **read** para un objecto particular retorna el valor que fue escrito por la operación **write** más reciente. Sea $\sim$ una relación de orden parcial arbitraria definida sobre D, decimos que la serialización S "respeta $\sim$" si $\forall a, b \in D$ tal que $a \sim b$, pasa que a precede a b en S. La historia H satisface linealización (**LIN**) si hay una serialización de H que respeta el orden inducido por los tiempos efectivos de las operaciones. Serialización estricta se define sobre historias formadas por transacciones y requere de la existencia de una serialización de H que respete el orden de tiempo real de las transacciones. Normalidad es equivalente a **LIN** cuando operaciones en objectos son unarias, pero es estrictamente más débil que **LIN** cuando las operaciones pueden abarcar varios objectos. Un nivel más débil de consistencia es ofrecido por consistencia secuencial (**SC**). Este modelo, no garantiza que una operación **read** retorna el valor más reciente con respecto al tiempo real sino solo que el resultado de cualquier ejecución sea el mismo como si las operciones en todos los sitios fuera ejecutados en orden secuencial, y las operaciones de cada sitio individual aparecen en esta secuencia en el orden especificado por su programa. La historia H satisface **SC** si hay una serialización de **H** que respete el orden del programa para cada sitio en el sistema. La relación de causalidad "$\rightarrow$" para sistemas de paso de mensajes puede ser modificada para ordernar las operaciones de H. Sea a, b y $c \in H$, se dice que $a \rightarrow b$ (a precede causalmente a b) si se da una de las siguientes: (i) a y b son ejecutados en el mismo sitio y a es ejecutado antes que b, (ii) b lee el valor de un objeto escrito por a, (iii) $a \rightarrow c$ y $c \rightarrow b$. Dos operaciones distintas a y b son concurrentes is ninguna de estas condiciones se dan entre ellas. Sea H_{i+w} el conjunto de todas las operaciones en H_i más todas las operaciones **write** en H. La historia H satisface consistencia causal (**CC**) si para cada sitio i existe una serialización de H_{i+w} que respete el orden causal "$\rightarrow$". Así, si $a, b, c \in H$ son tales que a escribe un valor **v**

en el objeto X, c lee el mismo valor **v** del objeto X, y b escribe el valor **v'** en el objeto X, nunca se da el caso que $a \rightarrow b \rightarrow c$. **CC** require que todas las relaciones relacionadas con causalidad sean vistas en el mismo order por todos los sitios, mientras que diferentes sitios pueden percibir operaciones concurrentes en órdenes diferentes. **CC** es un modelo de consistencia más débil que **SC** pero puede ser implementado eficientemente.

Timed Consistency. Ni en **SC** ni en **CC**, el tiempo real es explícitamente capturado. En **SC**, operaciones puede aparecer fuer de orden en relación con sus tiempos efectivos. En **CC**, cada sitio puede ver operciones de escritura concurrente en diferente orden. Por otro lado, **LIN** requere que las operaciones sean observadas en un orden que respete su ordenamiento de tiempo real. *Timed consistency* requiere que si el tiempo efectivo de un **write** es t, el valor escrito por esta operación tiene que estar visible a todos los sitios en el sistema distribuido en el tiempo $t + \Delta$, donde Δ es un parámetro de ejecución. Se puede ver que cuando Δ es 0, *timed consistency* se convierte en **LIN**[1]. *Reading on Time:* en modelos basados en tiempo, el conjunto de los valores que un **read** puede retorna está restringuido por la cantidad de tiempo que ha transcurrido a partir de los **write** que le preceden. Un **read** ocurre a tiempo (*on time*) si no devuelve datos obsoletos cuando hay valores más recientes que han estado disponibles por más de Δ unidades de tiempo. Esta definición depende las propiedades del reloj usando para assignar marcas de tiempo (*timestamps*) a las operaciones en ejecución. Primero, se asume relojes físicos perfectamente sincronizados en donde $T(a)$ es el instante en tiempo real correspondiente al tiempo efectivo de la operación a. **Definición 1:** Sea $\mathtt{D} \subseteq \mathtt{H}$ un conjunto de operaciones y S una serialización de D. Sea $w, r \in \mathtt{D}$ tal que w escribe un valor dentro de un objeto X que luego es leído por r (w es la operación **write** más cercana dentro del objeto X que aparece a la izquierda de r en la serialización S). Se define el conjunto $\mathtt{W}_r$ asociado con r como: $\mathtt{W}_r = \{w' \in \mathtt{D} : (w' \text{ escribe un valor dentro del objeto X}) \wedge (T(w) < T(w') < (T(r) - \Delta))\}$. Se dice que la operación r ocurre o lee a tiempo en la serialización S si $\mathtt{W}_r = \emptyset$. S es *timed* si cada operación **read** en S occurre a tiempo. *Approximately-syncronized real-time clocks:* En relojes de tiempo real aproximadamente sincronizados, resincronizaciones periódicas garantizan que no hay dos relojes difieran en más de ε unidades de tiempo. Típicamente también se garantiza que la diferencia entre cualquier reloj y el tiempo "real"[2] nunca es mayor que $\varepsilon/2$. De esta forma, si el tiempo efectivo de una operación a fue reportada por un sitio en particular como $T(a)$, entonces, desde el punto de vista del servidor de tiempo, este tiempo efectivo corresponde a algún instance en el intervalo $[T(a) - \varepsilon/2, T(a) + \varepsilon/2]$. Se dice que $\forall a, b \in \mathtt{H}$, a definitivamente ocurrió antes de b si $T(a) + \varepsilon/2 < T(b) - \varepsilon/2$, o, de forma equivalente si $T(a) + \varepsilon < T(b)$. Si ni a ni b definitivamente ocurrieron antes que el otro, se se dice que sus marcas de tiempo son no comparales o concurrentes. Sea $w, r \in \mathtt{D} \subseteq \mathtt{H}$ tal que w escribe un valor dentro del objecto X que luego es leído por r. Dejar que $w' \in \mathtt{D}$ actualice también el valor del objeto X. Si $T(w')$ definitivamente ocurrió antses que $T(w)$, o si $(T(r) - \Delta)$ ocurrió definitivamente antes que $T(w')$, entonces el claro que w' no afecta el hecho que r ocurra a tiempo. Ahora, si ambos $T(w')$ y $T(w)$ son concurrentes, o si $(T(r) - \Delta)$ y $T(w)$ son concurrentes, entonces no hay evidencia que w' sea más reciente que w, o de w' ocurriendo en más de Δ unidades de tiempo real antes que r respectivamente. En estos casos, aun se puede argumentar que r ocurre a tiempo. **Definición 2:** sea $\mathtt{D} \subseteq \mathtt{H}$ un conjunto de operaciones y S una serialización de D. Sea $w, r \in \mathtt{D}$ como se presenta en la definición 1. Se asume un conjunto $\mathtt{W}_r$, asociado con r como: $\mathtt{W}_r = \{w' \in \mathtt{D} : (w' \text{ escribe un valor dentro del objeto X}) \wedge (T(w) + \varepsilon < T(w')) \wedge (T((w')) \wedge (T(w') + \varepsilon < T(r) - \Delta)\}$. Se dice que una operación r ocurre o lee a tiempo en la serialización S si $\mathtt{W}_r = \emptyset$. S es *timed* si cada operación de **read** en S ocurre a tiempo.

[1] *Timed Consistency* es una generalización (o un debilitamiento) de **LIN**
[2] Mantenido por un servidor de tiempo

1. ¿CUÁL ES EL PROBLEMA QUE PLANTEA EL *PAPER*?
2. ¿POR QUÉ EL PROBLEMA ES INTERESANTE O IMPORTANTE?
3. ¿QUÉ OTRAS SOLUCIONES SE HAN INTENTADO PARA RESOLVER ESTE PROBLEMA?
4. ¿CUÁL ES LA SOLUCIÓN PROPUESTA POR LOS AUTORES?
5. ¿QUÉ TAN EXITOSA ES ESTA SOLUCIÓN?