**Modelos de consistencia sistemas operativos**

Un modelo de consistencia es un contrato entre el software y la memoria. Dice que si el SW acepta seguir ciertas reglas, la memoria promete trabajar en forma correcta. Si el SW viola estas reglas, todo acaba y ya no se garantiza que la operación de memoria sea correcta.

Existen varios modelos básicos de consistencia:

1. Consistencia estricta  
2. Consistencia secuencial  
3. Consistencia causal  
4. Consistencia PRAM y del procesador  
5. Consistencia débil  
6. Consistencia de liberación  
7. Consistencia de entrada  
  
A continuación se presenta una explicación detallada de cada una:  
  
**CONSISTENCIA ESTRICTA**

Está definido por la siguiente condición:

Cualquier lectura a una localidad de memoria x regresa el valor guardado por la operación de escritura más reciente en x.

Esta definición supone la existencia de un tiempo global absoluto (como en la física de Newton), para que la determinación del evento "más reciente" no sea ambigua.

Esta consistencia es típica de sistemas uniprocesador, donde nunca se pueden presentar dos eventos simultáneamente.

Sin embargo es prácticamente imposible de lograr en sistemas multiprocesador (distribuidos o no), debido a que pueden existir escrituras y lecturas en memoria separadas por espacios mínimos de tiempo que no alcanzan a cumplir la condición exigida.

Por ejemplo: Supongamos que una máquina A escribe algo en su memoria en el momento T2 y existe una máquina B que envió un mensaje en el instante T1 para leer que había en la memoria antes del instante T2. Si las máquinas están separadas una distancia de 3 metros y  T2 - T1 es de 1 nanosegundo, la señal de lectura debería viajar 10 veces más rápido que la velocidad de la luz para cumplir con la consistencia estricta, lo que es imposible.

**CONSISTENCIA SECUENCIAL**

Es un modelo de consistencia un poco más débil que la consistencia estricta. Fue definida por Lamport (1979). Está definido por la siguiente condición:

El resultado de cualquier ejecución es el mismo que si las operaciones de todos los procesos fueran ejecutadas en algún orden secuencial, y las operaciones de cada proceso individual a parecen en esta secuencia en el orden especificado por su programa.

Esto quiere decir que en este modelo sólo importa que los eventos generados por un programa sean vistos por otro en el mismo orden, sin importar que se intercalen con los eventos de otros programas.

Por ejemplo, tres procesos concurrentes podrían ejecutar las siguientes instrucciones:   
    
P1   
a=1;    
print(b, c);  
P2   
b=1;   
print(a, c);  
P3   
c=1;   
print(a, b);

Y cada proceso podría recibir los eventos ordenados y considerarlos válidos:   
  
(a)   
a=1;    
b=1;   
print(a, c);   
print(b, c);   
c=1;   
print(a, b);  
(b)   
c=1;   
print(a, b);   
a=1;    
b=1;   
print(a, c);   
print(b, c);  
(c)   
a=1;    
b=1;   
c=1;   
print(a, c);

print(b,c);   
print(a, b);

La consistencia secuencial no garantiza que una lectura regrese el valor escrito por otro proceso un nanosegundo antes, un microsegundo antes, o incluso un minuto antes. Sólo garantiza que todos los procesos vean todas las referencias a memoria en el mismo orden.

**CONSISTENCIA CASUAL.**

Es un debilitamiento de la consistencia secuencial que hace distinción entre eventos potencialmente relacionados por causalidad y aquellos que no lo están (Hutto y Ahamad, 1990).

La memoria con consistencia causal obedece las siguientes condiciones.

Las escrituras potencialmente relacionadas de forma causal son vistas por todos los procesos en el mismo orden. Las escrituras concurrentes pueden ser vistas en un orden diferente en máquinas diferentes.

Supongamos, por ejemplo, que el proceso PA escribe una variable x. Después, un proceso PB lee x y escribe y. Aquí la lectura de x y la escritura de y están relacionadas de forma casual.

Por otro lado, si dos procesos escriben de forma espontánea y simultánea en dos variables, no están relacionados en forma casual. Las operaciones que no están relacionadas en forma causal son concurrentes.

Esto quiere decir que en este modelo sólo importa que los eventos generados por un programa sean vistos por otro en el mismo orden, sin importar que se intercalen con los eventos de otros programas.

La consistencia causal mantiene un registro de cuáles procesos han visto cuáles escrituras. Esto significa que debe construirse y mantenerse una gráfica de dependencia con las operaciones que dependen de otras, lo que implica cierto costo.

**CONSISTENCIA DEBIL**

Fue propuesta por Dubois en 1986 y basa su funcionamiento en que los modelos anteriores de consistencia se consideran aún restrictivos porque requieren que las escrituras de un proceso se vean en orden.

Esto no siempre es necesario. Por ejemplo, cuando se está en una región crítica no es necesario propagar valores intermedios sino los valores finales.

Para poder implantar este modelo se necesita de una variable de sincronización.

Los accesos a las variables de sincronización son secuencialmente consistentes: todos los procesos ven todos los accesos a las variables de sincronización en el mismo orden.

No se permite el acceso a ninguna variable de sincronización hasta que todas las escrituras previas se hayan completado: Hacer una sincronización después de operaciones de escritura obliga a que los nuevos valores se propaguen a todas las memorias.

El hacer una operación de sincronización antes de leer los datos, le garantiza a un proceso que leerá los últimos valores.

Además, la operación de sincronización garantiza que las escrituras locales sean propagadas a todas las otras máquinas y se actualiza la memoria actual con escrituras hechas remotamente.  
Existen variantes a este modelo como los modelos de consistencia relajada, de liberación y de entrada por mencionar algunos.

**CONSISTENCIA DE LIBERACION.**

Distingue dos tipos de acciones de sincronización:  
Las que preceden a la entrada en secciones críticas.  
Las que se desarrollan después de completarse una sección crítica.   
Estas acciones de sincronización son conocidas comúnmente como:  
acquire access (acceso de adquisición).  
release access (acceso de liberación). 

**CONSISTENCIA DE ENTRADA.**

Mejora de la consistencia de liberación.

Las variables compartidas que han cambiado ya no se determinan de manera empírica.  
Variables de sincronización independientes.

1. No se permite realizar un acceso de adquisición a una variable de sincronización con respecto a un proceso hasta que se realicen todas las actualizaciones de los datos compartidos protegidos con respecto a ese proceso.   
2. Antes de permitir la realización de un acceso en modo exclusivo a unas variables de sincronización por un proceso, ningún otro proceso debe poseer la variable de sincronización, ni siquiera en modo no exclusivo.   
3. Después de realizar un acceso en modo exclusivo a una variable de sincronización, no se puede realizar el siguiente acceso en modo no exclusivo de otro proceso a esa variable de sincronización hasta haber sido realizado con respecto del propietario de esa variable.