Sistemas Distribuidos. Temario.

1. Introducción
2. Comunicación
3. Procesos
4. Nombrado y localización
5. Sincronización
6. Consistencia y replicación
7. Tolerancia a fallos

Tema 6.- Consistencia y replicación

1. Replicación
2. Modelos de consistencia
3. Protocolos de consistencia y replicación.

Bibliografía: Capítulo 6 de Tanenbaum

1.- Replicación

- Los datos se replican para aumentar la disponibilidad o el rendimiento.
- Un aspecto fundamental es la consistencia entre las diferentes réplicas: cuando se actualiza una copia, también se actualicen las demás.
- La replicación está muy relacionada con la escalabilidad. Relación opuesta.
- Hay diferentes semánticas de consistencia que proporcionan diferentes garantías: desde consistencia fuerte a débil.
- En general, cuanto más débil sea la consistencia, más escalable la replicación.

1.- Replicación

Replicación en objetos distribuidos

- **Concurrencia:** antes de tratar replicación, se debe considerar cómo controlar la concurrencia en el acceso a un objeto por parte de diferentes clientes:
  - El objeto mismo la controla: métodos *synchronized* de Java.
  - El soporte a objetos la controla: p.ej: permitiendo una sola tarea por cada objeto.
- **Replicación**: si un objeto tiene más de una réplica, las operaciones deben realizarse en determinado orden en cada réplica.
  - El objeto mismo la controla (el programador con soluciones ad-hoc).
  - El soporte a objetos la controla: el sistema de soporte a objetos garantiza que las invocaciones alcanzan a las diferentes réplicas en el orden adecuado.

Tema 6.- Consistencia y replicación

1. Replicación
2. Modelos de consistencia
3. Protocolos de consistencia y replicación.

Bibliografía: Capítulo 6 de Tanenbaum

2.- Modelos de consistencia

- Modelo básico de sistema:
  - Existen dos operaciones: write y read.
    - $W_i(x)a$: el proceso i realiza write sobre X con valor a
    - $R_j(y)b$: el proceso j realiza read sobre Y, obteniendo el valor b.
  - Un nodo que accede a los datos, realiza read o write sobre una copia de los datos.
  - Existen múltiples copias de los datos, estructuradas en lo que llamaremos **almacén de datos.**
- Un **modelo de consistencia** consiste en un **contrato** entre los procesos que acceden a los datos y el almacén de datos.
- El contrato estipula que: si los procesos siguen ciertas reglas, el almacén de datos proporciona ciertas garantías.
  - Normalmente, si un proceso realiza una operación read, obtendrá la última actualización del dato.
  - Suele ser difícil/costoso saber cuál es la última actualización
- **Una ejecución distribuida es un conjunto de secuencias de operaciones R y W. Cada elemento del conjunto es la secuencia de operaciones ejecutadas por un nodo.**

2.1.- Consistencia estricta

- **Una operación de lectura sobre el dato X, retorna el valor de la última operación write sobre X.**
- Este modelo asume la existencia de un reloj físico global.
- Es el modelo de consistencia que se utiliza en sistemas centralizados.
- En sistemas distribuidos sin reloj global es prácticamente imposible.

2.2.- Consistencia secuencial

- El resultado de cualquier ejecución distribuida, es el mismo que obtendríamos al ejecutar las operaciones R y W en algún orden secuencial, respetando que las operaciones R y W de un mismo proceso se ejecutan en el orden que impone el proceso.
- Es decir, cualquier intercalación entre lecturas y escrituras es admisible, pero todos los procesos las ven en el mismo orden.
- Ejemplo:
  - P1: Wx (P1 ejecuta Wx), P2: Wx, P3: Rx, Rx; P4: Rx, Rx
  - **SI**: W1(x)a, W2(x)b, R3(x)b, R4(x)b, R3(x)a, R4(x)a
  - **NO**: W1(x)a, W2(x)b, R3(x)b, R4(x)a, R3(x)a, R4(x)b

2.3.- Consistencia linealizable

- Igual que la consistencia secuencial, pero se asume que existe un reloj global sincronizado de **precisión finita** entre los diferentes nodos.
- Este tipo de relojes se pueden construir con algoritmos de sincronización de relojes y proporcionan un reloj global en el que varios eventos pueden ejecutarse en el mismo instante: dependiendo de la precisión.
- **Consistencia linealizable**: Como la secuencial, pero si el tiempo de la operación 1 es menor que el tiempo de la operación 2, entonces la operación 1 precederá a la 2.

2.4.- Consistencia causal

- Las escrituras que tengan relación causal potencial, deben verse en el mismo orden. Las escrituras concurrentes pueden verse en órdenes diferentes.
- Ejemplo:
  - **Causal y no secuencial**: P1Wa, P2Ra, P3Ra P4Ra, P2Wb, P1Wc, P3Rc, P4Rb, P3Rb, P4Rc.
  - **No causal**: P1Wa, P2Ra, P2Wb, P3Rb, P4Ra, P3Ra, P4Rb

2.5.- Consistencia FIFO

- Las escrituras que realiza un único proceso, son vistas por todos los demás en el mismo orden, pero las escrituras que realizan procesos diferentes, pueden verse en órdenes diferentes.
- Ejemplo:
  - **FIFO**: P1Wa, P2Ra, P2Wb, P2Wc, P3Rb, P4Ra, P3Ra, P4Rb, P3Rc, P4Rc

2.6.- Otros modelos de consistencia

- Consistencia débil
- Consistencia *release*
- Consistencia *entry*.

- Introducen operaciones de acceso a variables de sincronización, en lugar de únicamente operaciones read y write.

Tema 6.- Consistencia y replicación

1. Replicación
2. Modelos de consistencia
3. Protocolos de replicación

Bibliografía: Capítulo 6 de Tanenbaum

3.- Protocolos de replicación

1. Ubicación de las réplicas
2. Propagación de las modificaciones
3. Protocolos de replicación para invocaciones

3.1.- Ubicación de las réplicas

- Réplicas permanentes
  - La ubicación de las réplicas está preconfigurada.
  - Generalmente pocas réplicas.
- Réplicas iniciadas por los servidores
  - Los servidores deciden crear nuevas réplicas cuando se excede cierto umbral (carga, distancia, etc)
  - Se la conoce como cachés *push.*
- Réplicas iniciadas por los clientes
  - El cliente decide pedir una copia de los datos.
  - Se la conoce como cachés *pull.*

3.2.- Propagación de las modificaciones

Diferentes tópicos a considerar:

1. Estado vs operaciones
    1. Se puede transmitir sólo una notificación de cambio: protocolos de invalidación.
    2. Se puede transmitir el estado de una copia a otra: protocolos de checkpoint.
    3. Se puede transmitir la operación que supone el cambio: protocolos activos.
2. Pull vs Push
    1. Pull en general para réplicas iniciadas por clientes
    2. Push para réplicas permanentes e iniciadas por el servidor.
3. Unicast vs multicast

3.3.- Replicación para invocaciones

- Replicación activa
  - La invocación se dirige a todas las réplicas en determinado orden (utilizando protocolos de difusión que garanticen cierto orden).
- Replicación pasiva
  - Las invocaciones se dirigen a la copia primaria. La copia primaria ejecuta la operación y redirige la operación o el estado a las secundarias.
  - Cuando las secundarios terminan, se responde a la primaria.
  - La copia primaria responde al cliente.