TECNOLÓGICA DEL ORIENTE
INSTITUCIÓN DE EDUCACIÓN SUPERIOR

SISTEMAS DISTRIBUTIVOS

CONCEPTO DE CONSISTENCIA EN SISTEMAS DISTRIBUIDOS

CONCEPTO DE CONSISTENCIA EN SISTEMAS DISTRIBUIDOS

La consistencia en sistemas distribuidos, se refiere a la propiedad que asegura que todos los nodos que almacenan una copia de un mismo dato, reflejan la misma información en un momento determinado, manteniendo coherencia en el estado del sistema, incluso ante operaciones concurrentes o fallos.

¿Qué es la consistencia?

La consistencia implica que, tras una operación de escritura en un sistema distribuido, todas las lecturas posteriores de ese dato deben reflejar el valor actualizado, independientemente del nodo donde se consulte. Este concepto es fundamental para garantizar que los usuarios y procesos interactúen con datos correctos y actualizados, evitando decisiones basadas en información obsoleta o incoherente (Muñoz Escoí, 2013).

Consistencia en el modelo CAP

En el contexto del Teorema CAP, la consistencia es uno de los tres pilares (junto con la disponibilidad y la tolerancia a particiones) y establece que:

- Todos los nodos ven los mismos datos al mismo tiempo.

Por ejemplo, en un sistema de banca en línea, si un cliente transfiere dinero de una cuenta a otra, el saldo actualizado debe reflejarse de inmediato en todos los nodos que gestionan las cuentas, evitando inconsistencias que puedan causar errores financieros.

Tipos de consistencia en sistemas distribuidos

Existen diferentes niveles de consistencia que un sistema distribuido puede implementar, según sus necesidades de rendimiento y disponibilidad:

- **Consistencia fuerte.** Todos los nodos devuelven el mismo valor tras una actualización.
  - **Ejemplo.** En sistemas de archivos distribuidos como Google File System (GFS), una vez confirmado el éxito de una escritura, cualquier lectura posterior mostrará el nuevo valor (Boronat Seguí, 2012).
- **Consistencia eventual.** Los nodos pueden no mostrar el dato actualizado de inmediato, pero eventualmente se sincronizan para reflejar el valor correcto.
  - **Ejemplo.** DNS, donde una actualización puede tardar en propagarse entre servidores.
- **Consistencia causal.** Asegura que las operaciones que están causalmente relacionadas se vean en el mismo orden en todos los nodos.
  - **Ejemplo.** En sistemas de mensajería, si un mensaje de respuesta depende de uno anterior, ambos se mostrarán en orden.

Retos en mantener la consistencia

Los sistemas distribuidos enfrentan desafíos como:

- **Latencia de red.** Puede generar que algunos nodos reciban actualizaciones antes que otros.
- **Fallas de red o nodos.** Interrumpen el proceso de sincronización.
- **Operaciones concurrentes.** Pueden generar conflictos si dos procesos actualizan el mismo dato al mismo tiempo.

Para manejar estos retos, se emplean mecanismos como:

- Protocolos de consenso (Paxos, Raft). Aseguran que los nodos acuerden el estado correcto.
- Bloqueos distribuidos. Controlan el acceso concurrente a los datos.
- Vector clocks. Ayudan a rastrear el orden de las actualizaciones.

Ejemplo. En Amazon DynamoDB, se implementa consistencia eventual para mantener alta disponibilidad, pero permite configuraciones para consistencia fuerte en operaciones críticas, equilibrando rendimiento y coherencia, según las necesidades de la aplicación.