Marco Teórico

1. Sistemas Distribuidos: Consistencia y Coordinación

Los sistemas distribuidos se caracterizan por la interacción de múltiples nodos o procesos que trabajan de manera conjunta para ofrecer un servicio o realizar una tarea. Esta cooperación entre nodos es crucial para la escalabilidad, eficiencia y disponibilidad de los sistemas modernos. Sin embargo, uno de los mayores desafíos es garantizar la consistencia y coordinación entre estos nodos, especialmente en entornos donde la posibilidad de fallos de red, caídas de procesos o incluso desconexiones temporales es muy alta.

En un sistema distribuido, la consistencia se refiere a que todos los nodos deben compartir una misma visión del estado del sistema. La coordinación entre estos nodos es fundamental para realizar tareas como la replicación de datos, la respuesta a solicitudes de clientes y la toma de decisiones conjuntas. Para manejar estos aspectos, los sistemas distribuidos implementan mecanismos de consenso que permiten a los nodos ponerse de acuerdo sobre las decisiones clave, incluso si algunos nodos experimentan fallos.

2. Problema de la Elección de Líder

Uno de los problemas más comunes y cruciales que deben resolver los sistemas distribuidos es la elección de líder. En muchas arquitecturas distribuidas, un nodo debe asumir el rol de líder, ya que este líder es el encargado de tomar decisiones importantes y coordinar a los nodos restantes (conocidos como seguidores o followers). El líder gestiona operaciones de escritura en una base de datos distribuida, coordina la replicación de datos y asegura que las actualizaciones se propaguen correctamente a los nodos seguidores.

El problema surge cuando el líder falla o se desconecta. Dado que los sistemas distribuidos deben ser tolerantes a fallos, es fundamental que los nodos restantes detecten la caída del líder y elijan uno nuevo de manera rápida, segura y coordinada. Esta elección debe garantizar que solo un nodo sea seleccionado como líder en cualquier momento, evitando la posibilidad de tener múltiples líderes (una situación conocida como división de cerebro o split-brain), lo que podría comprometer la consistencia del sistema.

3. Algoritmos de Consenso

Los algoritmos de consenso son una clase de soluciones diseñadas específicamente para resolver el problema de la coordinación entre nodos en sistemas distribuidos, incluyendo la elección de líder. A través del consenso, los nodos pueden acordar el estado actual del sistema y actuar de forma sincronizada, incluso en presencia de fallos. Dos de los algoritmos de consenso más conocidos son Raft y Paxos, ambos ampliamente utilizados en sistemas de bases de datos

distribuidas, sistemas de archivos y otros servicios que requieren alta disponibilidad y consistencia.

3.1. Algoritmo Raft

Raft es un algoritmo de consenso desarrollado con el objetivo de ser más fácil de entender y de implementar en comparación con Paxos, sin sacrificar las garantías de seguridad y consistencia. Este algoritmo está orientado a la elección de líder y a la replicación consistente de los datos en un conjunto de nodos. Raft se basa en tres conceptos clave:

3.1.1 Elección de líder: En Raft, los nodos inician una elección de líder cuando detectan que el líder actual ha fallado. Para elegir un nuevo líder, los nodos restantes se postulan como candidatos y solicitan votos de los demás nodos. El nodo que obtiene la mayoría de los votos se convierte en el nuevo líder. Esta fase de elección garantiza que solo haya un líder en cada momento.

3.1.2 Replicación de log: El líder es responsable de replicar las operaciones en un registro (*log*) distribuido. Cada vez que el cliente realiza una solicitud, el líder registra la operación en su propio log y luego la replica en los seguidores. Solo cuando la mayoría de los nodos confirman que han replicado la operación, esta se considera comprometida.

3.1.3 Compromiso de entradas: Una vez que una entrada ha sido replicada en una mayoría de nodos, el líder la considera comprometida y la aplica al estado de la base de datos. Este enfoque asegura que el sistema mantenga la consistencia, incluso si el líder falla antes de que todas las entradas sean aplicadas.

Raft divide claramente el proceso de consenso en estas etapas, lo que hace que sea más comprensible y fácil de depurar en comparación con otros algoritmos.

3.1.4 Paso a Paso

Contexto del Ejemplo

Supongamos que tenemos un sistema distribuido con cinco nodos: A, B, C, D y E. El sistema necesita elegir un líder que coordine las operaciones de escritura y garantice la consistencia de los datos replicados.

1. Elección de Líder

El proceso de elección de líder se activa cuando:

- El sistema se inicia.

- El líder actual falla y necesita ser reemplazado.

Fase de elección:

Los nodos comienzan en el estado de seguidor. Si un nodo no ha recibido comunicación del líder dentro de un periodo de tiempo llamado "timeout", pasa a ser candidato y comienza una elección.

Supongamos que el nodo A no recibe señales de un líder, así que A cambia su estado a candidato y envía solicitudes de voto a los otros nodos (B, C, D, E).

Solicitud de votos:

Cada nodo seguidor recibe la solicitud de voto de A. Si no han votado por otro nodo en la misma elección, le otorgan su voto.

Si A recibe la mayoría de los votos (al menos 3 de los 5 nodos), se convierte en el nuevo líder.

Finalización de la elección:

Una vez elegido el líder, A empieza a enviar mensajes de latido ("heartbeats") a los otros nodos para mantener su liderazgo y evitar que otros nodos inicien una nueva elección.

2. Replicación de Log

Una vez que hay un líder, las operaciones que los clientes envían al sistema deben ser replicadas en todos los nodos.

Proceso de replicación:

Supongamos que un cliente envía una solicitud de escritura a A (el líder). A registra la solicitud en su log y luego envía la solicitud a los demás nodos (B, C, D, E).

Los nodos seguidores replican la entrada en sus propios logs y confirman al líder que han recibido la operación.

Compromiso de la operación:

La entrada en el log no se considera comprometida hasta que una mayoría de los nodos (por ejemplo, 3 de 5) haya replicado la operación.

Una vez que A recibe confirmaciones de al menos 3 nodos, se compromete la entrada y A la aplica al estado de la base de datos.

3. Compromiso de Entradas y Fallos

Tolerancia a fallos:

Si el líder A falla, los otros nodos detectarán que no han recibido mensajes de latido y comenzarán una nueva elección.

Supongamos que A falla, y C se convierte en el nuevo líder. El nuevo líder C consulta los logs de los demás nodos para asegurarse de que tiene todas las entradas replicadas. Si faltan entradas, C replica las entradas pendientes para garantizar la consistencia antes de aceptar nuevas operaciones.

Reconexión de nodos:

Si un nodo que había fallado (por ejemplo, B) vuelve a unirse al sistema, el líder actual (C) enviará las entradas de log que le faltan para que B pueda ponerse al día.

3.2 **Algoritmo Paxos**

Paxos es uno de los algoritmos de consenso más influyentes y complejos, diseñado por Leslie Lamport. Paxos garantiza que un conjunto de nodos distribuidos pueda llegar a un acuerdo sobre un valor propuesto, incluso en presencia de fallos en algunos nodos. Paxos se compone de tres roles principales:

1. **Proponente**: Es el nodo que propone un valor que será acordado por el resto de los nodos.

2. **Aceptadores**: Son los nodos que reciben las propuestas del proponente. Cada aceptador decide si acepta o rechaza la propuesta basándose en ciertas reglas que garantizan la consistencia.

3. **Aprendices**: Son los nodos que observan las decisiones de los aceptadores y aplican los resultados finales.

El proceso de Paxos se divide en dos fases:

- En la fase de propuesta, un nodo propone un valor a ser acordado.
- En la fase de aceptación, los nodos aceptan la propuesta si cumple con los criterios de consistencia.

Una vez que la mayoría de los aceptadores aceptan la propuesta, el valor se considera elegido y se puede aplicar. Paxos asegura que, sin importar cuántos fallos ocurran, mientras exista una mayoría de nodos funcionales, el sistema podrá llegar a un acuerdo sin perder consistencia.

3.2.1 Paso a paso

Contexto del Ejemplo

Tenemos de nuevo cinco nodos: A, B, C, D y E. El objetivo es que todos se pongan de acuerdo sobre una única propuesta, como la escritura de un valor en la base de datos.

1. Fase de Propuesta

Un nodo actúa como proponente, es decir, el nodo que inicia el proceso de consenso proponiendo un valor.

Proponer un valor:

Supongamos que el nodo A es el proponente y quiere escribir el valor "X" en la base de datos distribuida.

A envía una propuesta a todos los nodos (B, C, D, E) con un número de secuencia (por ejemplo, el número 1) para ser aceptada.

2. Fase de Aceptación

Los nodos restantes actúan como aceptantes, quienes reciben la propuesta y deciden si aceptarla o no.

Primera fase de aceptación:

Cada nodo que recibe la propuesta de A (B, C, D, E) responde si no han aceptado una propuesta con un número de secuencia mayor.

Si el nodo B no ha aceptado ninguna otra propuesta o ha aceptado una con un número menor, responde positivamente a A.

Segunda fase de aceptación:

Si A recibe respuestas positivas de una mayoría de nodos (al menos 3 de 5), A envía un segundo mensaje para aceptar formalmente la propuesta "X".

Los nodos que reciben este segundo mensaje aceptan oficialmente la propuesta y lo registran en su log.

3. Fase de Aprendizaje

Después de que una mayoría de nodos haya aceptado la propuesta, los nodos actúan como aprendices, registrando y aprendiendo el valor aceptado.

Propagación de la decisión:

- Supongamos que los nodos B, C y D han aceptado la propuesta de A para escribir "X". Estos nodos ahora comunican a los demás nodos (A, E) que "X" ha sido aceptada.
- Todos los nodos acuerdan que "X" es el valor decidido y lo aplican al estado de la base de datos.

3.3. Comparación entre Raft y Paxos

Si bien tanto Raft como Paxos son algoritmos robustos que garantizan la consistencia y disponibilidad en sistemas distribuidos, existen diferencias clave entre ambos:

- **Simplicidad**: Raft fue diseñado específicamente para ser más sencillo que Paxos. La separación clara entre las fases de elección de líder y replicación de datos en Raft lo hace más fácil de implementar y depurar.

- **Popularidad**: Aunque Paxos es el algoritmo de consenso más estudiado teóricamente, Raft ha ganado popularidad en la industria debido a su simplicidad y eficiencia en implementaciones prácticas.

- **Uso en la industria**: Sistemas distribuidos modernos como Consul, Etcd y RethinkDB utilizan Raft como su algoritmo de consenso debido a su simplicidad y capacidad para manejar de manera eficiente la elección de líder y la replicación de logs.

4. Aplicación del Consenso en la Elección de Líder

El consenso no solo asegura que los nodos lleguen a un acuerdo sobre los valores propuestos, sino que también permite coordinar la elección de líder en caso de fallas. La elección de un líder en un sistema distribuido debe cumplir con los siguientes principios:

- **Tolerancia a fallos**: El sistema debe ser capaz de elegir un nuevo líder incluso si uno o más nodos fallan.

- **Consistencia**: Solo un nodo debe ser elegido como líder en un momento dado, y el nuevo líder debe mantener la consistencia del estado de los datos.

- **Disponibilidad**: El proceso de elección de líder debe completarse rápidamente para minimizar el tiempo de inactividad.

5. Importancia de los Algoritmos de Consenso en Sistemas Distribuidos

Los algoritmos de consenso, como Raft y Paxos, son fundamentales para la robustez de los sistemas distribuidos. Estos algoritmos permiten que los nodos colaboren de manera segura y eficiente, asegurando que las operaciones distribuidas se realicen de manera correcta, incluso cuando algunos nodos fallan. Sin estos mecanismos de consenso, los sistemas distribuidos estarían más expuestos a fallas catastróficas que podrían comprometer tanto la consistencia como la disponibilidad.

Los algoritmos de consenso son esenciales para cualquier sistema distribuido que necesite garantizar la consistencia de los datos, la tolerancia a fallos y la alta disponibilidad, ya sea en bases de datos, sistemas de archivos distribuidos o servicios en la nube. Raft y Paxos representan enfoques robustos y confiables para resolver el problema de la elección de líder y asegurar que los sistemas distribuidos puedan operar correctamente incluso en condiciones adversas.