SCD.

Actividad de cierre V y VI

Padilla Perez Jorge Daray

23/10/2024

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

Prof. Gutiérrez Salmerón Martha del Carmen

Contenido

Introducción 4

Contenido 5

Que es la tolerancia a fallas (Tipos de fallas, generalidades, ejemplos) 5

¿Qué es la tolerancia a fallas? 5

Tipos de fallas 5

1. Fallas de hardware 5
2. Fallas de software 5
3. Fallas de red 5
4. Errores humanos 5

Generalidades de la tolerancia a fallas 6

- Redundancia 6
- Replicación 6
- Reconfiguración dinámica 6
- Degradación controlada 6

Ejemplos de tolerancia a fallas 6

- RAID 6
- Clústeres de alta disponibilidad: 6
- Algoritmos de consenso: 7

Relaciona como se aplica la tolerancia a fallas en los sistemas distribuidos y brinda un breve ejemplo. 7

¿Cómo se aplica la tolerancia a fallas en sistemas distribuidos? 7

1. Replicación de datos y servicios 7
   - Aplicación en bases de datos distribuidas 7
2. Algoritmos de consenso 7
   - Aplicación en sistemas de control distribuido 7
3. Reconfiguración dinámica: 8
   - Aplicación en sistemas de microservicios 8
4. Monitoreo y detección de fallas 8
   - Aplicación en clústeres de alta disponibilidad 8

5. Degradación controlada 9
- Aplicación en sistemas de tiempo real 9

Ejemplo de aplicación en un sistema distribuido: Dropbox 9

Conclusión 10

Referencias 11

Introducción

En esta actividad se habla sobre la tolerancia a fallas y la administración de procesos y el cómo se interrelacionan para garantizar la disponibilidad, confiabilidad y eficiencia en sistemas distribuidos. La capacidad de un sistema para detectar y corregir fallos, junto con una planificación eficiente de los procesos, asegura que los servicios sigan operando sin interrupciones, incluso en entornos con múltiples puntos de falla.

Contenido

Que es la tolerancia a fallas (Tipos de fallas, generalidades, ejemplos)

¿Qué es la tolerancia a fallas?

La tolerancia a fallas es la capacidad de un sistema, ya sea hardware, software o una combinación de ambos, para seguir funcionando adecuadamente en presencia de errores o fallos. Su objetivo es garantizar la continuidad de las operaciones de un sistema, incluso cuando algunos de sus componentes fallan o funcionan incorrectamente. Esto es fundamental en sistemas que requieren alta disponibilidad, como los servicios bancarios, la nube, sistemas de control de tráfico aéreo o sistemas industriales críticos.

Tipos de fallas

1. Fallas de hardware: Estas fallas se refieren a malfuncionamientos en los componentes físicos del sistema. Pueden ser causadas por envejecimiento del hardware, defectos de fabricación, o daños físicos (por ejemplo, sobrecalentamiento, fallos de energía, etc.).

Ejemplo: Un servidor que sufre una falla en su disco duro. Si el sistema implementa RAID (Redundant Array of Independent Disks), los datos se distribuyen entre varios discos, y si uno de ellos falla, la redundancia permite que el sistema siga funcionando sin pérdida de datos.

2. Fallas de software: Ocurren cuando hay errores en el código o en la ejecución del software que provocan un mal comportamiento o bloqueo del sistema. Los errores de software pueden ser causados por errores de programación (bugs), problemas de compatibilidad o mal manejo de excepciones.

Ejemplo: Un servidor web que se bloquea debido a un error en la aplicación. Si el sistema tiene configurada una estrategia de reinicio automático o si está replicado, otro servidor puede tomar el control de la aplicación sin interrumpir el servicio.

3. Fallas de red: Estas ocurren cuando la comunicación entre diferentes partes del sistema se interrumpe o se degrada. Pueden ser causadas por problemas en los cables, interrupciones en la señal, o fallos en los dispositivos de red como routers o switches.

Ejemplo: En un sistema de bases de datos distribuido, una interrupción en la red puede hacer que algunos nodos no se comuniquen correctamente. Sin embargo, con algoritmos de consenso como Paxos o Raft, los nodos restantes pueden seguir operando y procesando datos mientras se soluciona el problema de la red.

4. Errores humanos: Este tipo de fallas es causado por acciones incorrectas realizadas por operadores o usuarios, ya sea por desconocimiento, descuido, o malentendidos. Aunque son difíciles de prevenir por completo, muchos sistemas implementan mecanismos de protección para minimizar su impacto.

Ejemplo: Un administrador de sistema borra accidentalmente archivos críticos. Un sistema con copias de seguridad automáticas o versiones anteriores almacenadas en la nube puede recuperar rápidamente la información eliminada.

Generalidades de la tolerancia a fallas

La tolerancia a fallas se basa en varios principios y técnicas que permiten que un sistema maneje errores de manera eficaz. Entre las técnicas más comunes se encuentran:

- Redundancia: Implica la duplicación de componentes clave del sistema para asegurar que, si uno de ellos falla, otro pueda asumir su función. Esta es una técnica esencial en sistemas críticos de misión, como los sistemas de control de vuelos o los sistemas bancarios. La redundancia puede aplicarse a hardware, software e incluso a redes enteras.

Ejemplo: En los centros de datos de grandes proveedores como Google o Amazon, los servidores están replicados en múltiples ubicaciones geográficas. Si un centro de datos falla, otro puede asumir inmediatamente la carga sin interrupción visible para los usuarios.

- Replicación: Mantener varias copias de los datos o de los servicios en diferentes ubicaciones o nodos. La replicación asegura que, si un nodo falla, otros pueden continuar procesando solicitudes sin pérdida de información.

Ejemplo: Bases de datos distribuidas como MongoDB o Cassandra implementan replicación de datos en múltiples nodos. Si un nodo falla, las consultas se redirigen a otro nodo que contiene una réplica de los datos.

- Reconfiguración dinámica: Esta técnica implica la capacidad de un sistema para detectar fallas y, en respuesta, reorganizar los recursos disponibles para mantener la operatividad. Se utilizan técnicas de monitoreo y detección de fallos para identificar componentes fallidos y proceder a desconectarlos o repararlos automáticamente.

Ejemplo: En un clúster de cómputo distribuido, si un servidor detecta que una de sus máquinas está fallando, puede automáticamente redistribuir la carga a otras máquinas del clúster sin intervención humana.

- Degradación controlada: Algunos sistemas implementan lo que se conoce como degradación controlada, donde el sistema sigue operando con capacidades reducidas, en lugar de fallar por completo. Esto es común en sistemas críticos, donde es preferible ofrecer un servicio limitado antes que una interrupción total.

Ejemplo: En un sistema de streaming de video, si una falla de red afecta la transmisión de alta calidad, el sistema puede reducir temporalmente la calidad del video en lugar de interrumpir completamente el servicio.

Ejemplos de tolerancia a fallas

- RAID: Un sistema común de almacenamiento en discos que utiliza técnicas de redundancia para garantizar que los datos permanezcan accesibles, incluso si uno o varios discos fallan.
- Clústeres de alta disponibilidad: Estos sistemas distribuyen aplicaciones y cargas de trabajo entre varios servidores. Si un servidor falla, otro puede tomar el control sin interrumpir los servicios. Esto es común en los servicios de la nube como Amazon Web Services (AWS) o Microsoft Azure.

- Algoritmos de consenso: Como Paxos o Raft, que permiten que sistemas distribuidos tomen decisiones correctas incluso en presencia de fallos de algunos de sus componentes. Estos algoritmos son fundamentales en bases de datos distribuidas y servicios como Apache Kafka, que requieren consistencia en entornos donde algunos nodos pueden fallar o desconectarse.

Relaciona como se aplica la tolerancia a fallas en los sistemas distribuidos y brinda un breve ejemplo.

En los sistemas distribuidos, la tolerancia a fallas es fundamental porque estos sistemas están compuestos por múltiples nodos que trabajan juntos para lograr un objetivo común, pero a menudo están sujetos a fallas en alguno de sus componentes debido a su naturaleza interconectada y distribuida. A diferencia de los sistemas monolíticos, en los que un fallo puede interrumpir todo el sistema, los sistemas distribuidos están diseñados para manejar fallos de manera eficiente y continuar operando, asegurando alta disponibilidad, confiabilidad y resistencia ante fallos.

¿Cómo se aplica la tolerancia a fallas en sistemas distribuidos?

1. Replicación de datos y servicios: La replicación es uno de los mecanismos más efectivos para lograr la tolerancia a fallas en sistemas distribuidos. En este enfoque, los datos o servicios se duplican y distribuyen entre varios nodos o servidores. Esto asegura que, si uno de los nodos falla, los datos o servicios todavía están disponibles en otros nodos que poseen copias de la información o que pueden ejecutar la misma funcionalidad.
    - Aplicación en bases de datos distribuidas: En sistemas distribuidos de bases de datos como Cassandra, Couchbase, o MongoDB, los datos se replican en diferentes nodos ubicados en distintas partes del sistema o incluso en diferentes centros de datos. Esto asegura que, si uno de los nodos deja de responder debido a un fallo de hardware o red, otro nodo con una copia de los mismos datos puede asumir el control y procesar las consultas de los usuarios.
    - Ejemplo: Supongamos que un banco utiliza una base de datos distribuida para almacenar la información de sus clientes. Si un nodo de la base de datos que contiene los datos de las transacciones financieras deja de funcionar por una falla, el sistema automáticamente redirige las solicitudes de los clientes a otro nodo con la réplica de esos datos, garantizando que las transacciones continúen sin interrupciones.
2. Algoritmos de consenso: Los sistemas distribuidos requieren que los nodos tomen decisiones colectivas para mantener la consistencia del sistema, incluso cuando algunos nodos fallan o no están disponibles temporalmente. Los algoritmos de consenso, como Paxos, Raft o Zab, son diseñados para garantizar que los nodos puedan llegar a un acuerdo sobre un estado o una acción, incluso cuando un subconjunto de los nodos ha fallado.
    - Aplicación en sistemas de control distribuido: En sistemas como Apache Zookeeper (utilizado para coordinar procesos en sistemas distribuidos), el algoritmo de consenso garantiza que, incluso si algunos nodos en el clúster fallan, los nodos restantes pueden

llegar a un acuerdo sobre el estado del sistema, como la gestión de servicios o la asignación de recursos.

   - Ejemplo: Supongamos que, en un sistema de votación distribuido, varios nodos deben votar para determinar si una transacción financiera debe ser aprobada o rechazada. Si uno o más nodos fallan durante el proceso de votación, los nodos restantes, mediante un algoritmo de consenso como Raft, pueden seguir adelante con la transacción y mantener la consistencia del sistema.

3. Reconfiguración dinámica: La capacidad de los sistemas distribuidos para reconfigurarse automáticamente en respuesta a fallos es crucial para la tolerancia a fallas. La reconfiguración dinámica permite que, cuando un nodo o un enlace de comunicación falla, el sistema redistribuya las cargas de trabajo entre los nodos activos. Esto asegura que las operaciones continúen con un impacto mínimo, ajustándose de manera automática y rápida.

   - Aplicación en sistemas de microservicios: Los sistemas basados en microservicios, como los utilizados por Netflix o Uber, implementan reconfiguración dinámica para gestionar fallas en tiempo real. Los microservicios están distribuidos en múltiples servidores y, si uno de estos servidores falla, el sistema puede redirigir automáticamente las peticiones de los usuarios a otros servidores que ejecutan los mismos microservicios.

   - Ejemplo: En una plataforma de streaming como Netflix, si un servidor que gestiona la transmisión de películas falla, los usuarios que están viendo una película no experimentan interrupciones porque la plataforma automáticamente redirige las solicitudes de transmisión a otro servidor que tiene una réplica del servicio de streaming. Esta capacidad de redireccionar y reconfigurarse dinámicamente garantiza una experiencia ininterrumpida.

4. Monitoreo y detección de fallas: Los sistemas distribuidos a menudo emplean herramientas de monitoreo que permiten detectar fallas de manera proactiva y actuar en consecuencia. Los sistemas de monitoreo detectan si un nodo está respondiendo lentamente o ha dejado de responder, lo que permite que el sistema redistribuya las cargas de trabajo antes de que los usuarios noten el fallo.

   - Aplicación en clústeres de alta disponibilidad: Sistemas distribuidos como los clústeres de alta disponibilidad implementan monitoreo continuo para asegurarse de que todos los nodos y servicios estén activos. Si uno de los nodos deja de responder, el sistema automáticamente lo saca de la rotación de trabajo y redirige las solicitudes a otros nodos activos.

   - Ejemplo: En un clúster de servidores que maneja el tráfico web de una tienda en línea, si uno de los servidores experimenta una sobrecarga o falla en su hardware, el sistema de monitoreo lo detecta inmediatamente. Como respuesta, las peticiones de los usuarios son redirigidas a otros servidores en el clúster sin afectar la disponibilidad de la tienda en línea.

5. Degradación controlada: En algunos casos, en lugar de intentar continuar operando a plena capacidad después de una falla, un sistema distribuido puede implementar una estrategia de degradación controlada. Esto significa que el sistema puede seguir funcionando, pero con capacidades limitadas. Es preferible ofrecer un servicio reducido que interrumpir completamente el sistema.
    - Aplicación en sistemas de tiempo real: En sistemas de comunicación en tiempo real, como videollamadas o transmisiones de video, la degradación controlada permite reducir la calidad del servicio (por ejemplo, bajar la calidad de video) en lugar de perder la conexión por completo.
    - Ejemplo: Durante una videollamada en una plataforma como Zoom, si la red se ve afectada y no puede soportar la calidad de transmisión HD, el sistema puede degradar la calidad del video a resolución más baja para mantener la conexión y la llamada en curso, en lugar de cortar la comunicación por completo.

Ejemplo de aplicación en un sistema distribuido: Dropbox

En un servicio de almacenamiento en la nube como **Dropbox**, los archivos de los usuarios están distribuidos en múltiples servidores ubicados en diferentes partes del mundo. Para garantizar la disponibilidad continua de los datos, Dropbox implementa replicación de datos. Esto significa que cada archivo subido por el usuario se guarda en varios servidores en distintas ubicaciones geográficas. Si uno de los servidores en una región falla (por ejemplo, un servidor en Europa sufre una caída por un fallo de hardware), Dropbox redirige automáticamente las solicitudes de acceso a los archivos a servidores en otras regiones (por ejemplo, América o Asia), donde existen copias replicadas de esos archivos.

Esto garantiza que el usuario aún pueda acceder a sus archivos sin interrupciones, incluso si un servidor está fuera de servicio. Además, si el servidor fallido se recupera, el sistema de replicación se asegura de que los archivos estén actualizados y sincronizados correctamente. Esta combinación de replicación de datos, detección automática de fallos y reconfiguración dinámica es un ejemplo clásico de cómo la tolerancia a fallas se implementa en sistemas distribuidos para garantizar la disponibilidad y la resiliencia ante fallas.

Conclusión

La tolerancia a fallas es una propiedad esencial en los sistemas distribuidos, dado que estos sistemas dependen de la colaboración entre múltiples nodos, redes y servicios que, debido a su complejidad, son propensos a fallos. A través de técnicas como la replicación de datos, la reconfiguración dinámica, el monitoreo de fallos y los algoritmos de consenso, los sistemas distribuidos pueden continuar operando de manera efectiva, minimizando el impacto de las fallas en el servicio.

Ya sea que se trate de aplicaciones en la nube, servicios de bases de datos distribuidas o sistemas críticos de tiempo real, la tolerancia a fallas asegura que los sistemas distribuidos sean altamente disponibles, confiables y capaces de gestionar fallas de manera eficiente.

Referencias

Tanebaum Andrew. (1995). Sistemas Operativos Distribuidos. España. Prentice-Hall Hisp.

Shooman, M. (2002). Fault-Tolerant Computing: Systems (2nd ed.). John Wiley & Sons.

Jalote, P. (1994). An Introduction to Fault-Tolerant Distributed Systems. Addison-Wesley.

Arenas Selee, D. (1998). Metodología para Análisis y Diseño de Sistemas Distribuidos. Instituto Tecnológico y de Estudios Superiores de Monterrey.

Burns, A., & Wellings, A. (2003). Real-Time Systems and Programming Languages (3rd ed.). Addison-Wesley.