Tarea-Examen Final

Cova Pacheco Felipe de Jesús

Cómputo Concurrente

Profesor: Luis German Pérez Hernández
Ayudante: Daniel Michel Tavera
Ayudante de Laboratorio: Fernando Michel Tavera

1. En un sistema distribuido, ¿qué algoritmos existen para la escritura y lectura de relojes?

R. Hablando de los algoritmos de sincronización, éstos dependen más que nada del reloj usado.
Es así como podemos categorizar los algoritmos de acuerdo a su reloj, como se muestra a continuación :

I. Relojes Físicos:

Cuando utilizamos una base de tiempo Universal, se dice que utilizamos Relojes Físicos. Son útiles en aplicaciones que se manejan eventos secuenciados a alta velocidad, de tal suerte que no hay recursos disponibles para organizar la sincronización; En ciertos sistemas (por ejemplo, los sistemas de tiempo real), es importante la hora real del reloj. Para estos sistemas se necesitan relojes físicos externos. Por razones de eficiencia y redundancia, por lo general son recomendables varios relojes físicos.

Los relojes físicos deben ser iguales (estar sincronizados), no deben desviarse del tiempo real más allá de cierta magnitud. Debido a la distorsión del tiempo y las derivas en los relojes de los distintos CPUs, en ciertos sistemas es importante la hora real del reloj, por lo que se precisan relojes físicos externos (más de uno) y se deben sincronizar con los relojes del mundo real.
Para mantener estos relojes sincronizados entre sí podemos usar los siguientes algoritmos:

1) Algoritmo de Cristian

• Es probabilístico: Solo se puede usar si los tiempos de ida y vuelta entre el cliente y el servidor son menores que la precisión mínima requerida.
• Servidor único de tiempo
• Puede conectarse a UTC
  - El proceso P envía mensaje m_r solicitando el tiempo al servidor S
  - S envía el tiempo en m_t
  - P ajusta su reloj utilizando el tiempo recibido en m_t

Ventajas:

- Adecuado para sincronización con UTC.
- Tiempo de transmisión del mensaje: (T1-T0)/2
- Tiempo en propagar el mensaje: (T1 - T0 - I)/2
- Valor que devuelve el servidor se incrementa en (T1 - T0 - I)/2
- Para mejorar la precisión se pueden hacer varias mediciones y descartar cualquiera en la que T1-T0 exceda de un límite

Problemas:

- Duración variable del tiempo de transmisión de los mensajes en la red, y del tiempo de respuesta del servidor

- Servidor único: sobrecarga y caída.
  - Solución: grupo de servidores sincronizado

- Servidor impostor: seguridad
  - Solución: autenticación del servidor

- Servidor con reloj que falla: tiempo erróneo en el SD V
  - Solución: algoritmo de Berkeley

2) Algoritmo de Berkeley

- Desarrollado para el UNIX de Berkeley
- No admite sincronización con el UTC
- Una computadora coordinador es elegido para actuar como maestro
- El maestro interroga a los esclavos, que deben estar sincronizados
- El maestro estima el valor de los relojes de los esclavos (como en el de Cristian)
- Con los valores estimados y el suyo propio, el maestro realiza un promedio de los tiempos
  - ✓ Se cancelan en media los efectos de la deriva
- Este promediado es tolerante a fallos: elimina lecturas de relojes que han derivado mucho

• El maestro envía a los esclavos la cantidad (positiva ó negativa) con que debe ajustar su reloj
  - ✓ Se elimina la influencia del tiempo de transmisión
• Si el maestro falla
  - ✓ Un esclavo debe ser elegido como maestro

2. Relojes Lógicos

La idea de un reloj lógico consiste en crear un sistema de convergencia del tiempo mediante la medición de las derivas, de tal suerte que la noción de tiempo universal se sustituye por la noción de un tiempo global auto-ajustable.

Los relojes lógicos son útiles para ordenar eventos en ausencia de un reloj común. Cada proceso P mantiene una variable entera LCP (su reloj lógico), cuando un proceso P genera un evento, LCP = LCP +1, entonces cuando un proceso envía un mensaje incluye el valor de su reloj; si otro proceso Q recibe un mensaje, éste debe revisar su reloj lógico para determinar si existe secuencialidad con el mensaje en turno o su puede trabajar en concurrencia. Los relojes lógicos sólo representan una relación de orden parcial, y funcionan mediante la alteración del conteo de tiempo para mantener la sincronización mediante variables que afectan la hora de referencia. El reloj lógico establece un orden total entre eventos si se añade el número del procesador, pero esto implica pagar un precio por la comunicación y por mantener el registro la tabla de CPUs.

Se Pueden utilizar los siguientes algoritmos para mantener los relojes sincronizados:

1) Algoritmo de Mattern Fidge
   Nos deja evaluar siempre si dos marcas de tiempo tienen o no relación de precedencia.

2) Algoritmo de Lamport

Si "a" y "b" son eventos en el mismo proceso y "a" ocurre antes de "b", entonces "a –> b" es verdadero.
"Ocurre antes de" es una relación transitiva:
Si "a –> b" y "b –> c", entonces "a –> c".
Si dos eventos "x" e "y" están en procesos diferentes que no intercambian mensajes, entonces "x –> y" no es verdadero, pero tampoco lo es "y –> x":
Se dice que son eventos concurrentes.

2. ¿Qué es un sistema de cómputo en tiempo real? Y ¿Qué algoritmos o métodos existen en dichos sistemas?

Un sistema en tiempo real son aquellos que deben producir respuestas correctas dentro de un intervalo de tiempo definido. Si el tiempo de respuesta excede ese límite, se produce una degradación del funcionamiento y/o un funcionamiento erróneo.

Son utilizados dos tipos de algoritmos basados en la localización de recursos para sistemas distribuidos en tiempo real asíncronos y tolerantes a fallas:

a) Hablando del análisis de la sobrecarga del sistema, y una manera clara de ejemplificar esta categoría es mediante el algoritmo periódico HILO que realiza un balanceo de cargas al utilizar la métrica de disponibilidad del nodo para eso, y así se logra un desempeño de lo más óptimo.
b) También se puede ver desde el punto de vista de el análisis de tiempos.

Siguiendo con la misma idea, también se puede utilizar un algoritmo de planificación que pueda manejar los recursos del sistema distribuido de acuerdo a dos características principales, el manejo de eventos y el manejo de tiempos. Ejemplos de estos casos son los siguientes:

a) SJF: Es un algoritmo no apropiativo que supone que los tiempos de ejecución se conocen de antemano. Cuando hay varios trabajos de igual importancia esperando a ser iniciados en la cola de entrada, el planificador selecciona el trabajo más corto primero

b) SRTN: Es un algoritmo apropiativo donde el planificador siempre selecciona el proceso cuyo tiempo restante de ejecución sea el más corto. De nuevo, se debe conocer el tiempo de ejecución de antemano. Cuando llega un nuevo trabajo, su tiempo total se compara con el tiempo restante del proceso actual. Si el nuevo trabajo necesita menos tiempo para terminar que el proceso actual, éste se suspende y el nuevo trabajo se inicia. Ese esquema permite que los trabajos cortos nuevos obtengan un buen servicio.

*En los dos algoritmos, las tareas pueden ser expulsadas del procesador en cualquier momento y son óptimos.

** Hay una cantidad exorbitante de algoritmos para la planificación y un sistema de computo real depende mucho del modelo elegido, por lo que es incomprensible detallar cada caso.

3. **En un sistema distribuido ¿Qué es exclusión mutua distribuida? ¿Qué algoritmos o métodos existen? y ¿Cuál es su utilización?**

La exclusión mutua distribuida se produce cuando los procesos y el recurso no se encuentran en el mismo equipo, por lo que en este caso, para coordinar el acceso al recurso las variables compartidas mencionadas anteriormente no pueden ser utilizadas. Es por esto que el único medio para controlar la sección crítica es la comunicación mediante el paso de mensajes.

Hay distintos grupos de algoritmos que se pueden utilizar para resolver esta exclusión mutua distribuida, y según su funcionamiento podrían clasificarse en tres categorías:

- Basados en tokens: un elemento único es compartido por todos los nodos, permitiendo el acceso a la sección crítica únicamente al nodo que lo posee.
- No basados en tokens: el nodo con acceso a la sección crítica se establece mediante el intercambio de mensajes en dos o más rondas.
- Basados en quorum: para que un nodo pueda acceder a la sección crítica tiene que tener permiso de todos los nodos de su subconjunto.

A continuación se mostrarán 3 algoritmos diferentes, uno por cada categoría mencionada anteriormente:

a) Algoritmo de cola de prioridad compartida:

Es algoritmo que ayuda a crear la exclusión mutua. En este algoritmo, (basado en los tiempos lógicos de Lamport) un nodo i cualquiera mantendrá de forma local una copia de parte de la cola de prioridad compartida para poder saber si tiene acceso a la sección crítica o no.

b) Algoritmo de Anillo

En este algoritmo la entrada a la sección crítica se basa en la posesión o no de un token, que se van pasando todos los nodos en forma circular. Cuando un nodo consigue el testigo, puede entrar a la sección crítica debiendo liberarlo y pasarlo cuando sale de ella hacia el siguiente nodo.

c) Algoritmo de Maekawa

El algoritmo de Maekawa pertenece a los basados en quorum y para entenderlo hay que pensar en la red o sistema distribuido como un subconjunto de nodos (Si), y que para que un nodo pueda acceder a la sección crítica este debe haber bloqueado anteriormente a los demás pertenecientes al mismo subconjunto.

Cuando el nodo i trate de bloquear a todos los demás nodos si lo consigue podrá acceder a la sección crítica, pero si no tendrá que esperar a que todos los demás nodos estén libres para poder volver a intentarlo.

Bibliografía:

- Gunnar Wolf, Esteban Ruiz, Federico Bergero, Erwin Meza. "Fundamentos de sistemas operativos". Recuperado el 1 de junio de 2018 en: https://sistop.org/pdf/sistemas_operativos.pdf

- Departamento de arquitectura y tecnología de computadoras Universidad de Sevilla. "Sistemas en tiempo real". Recuperado el 1 de junio en: http://icaro.eii.us.es/descargas/Tema6-parte1.pdf

- Silberschatz Abraham, Baer Galvin Peter, Greg Gagne."Operating system concepts" Wiley: Ninth edition,

- Paul Krzyzanowski. Lectures on clock Synchronization. 2009 Extraido de: http://soft.vub.ac.be/~tvcutsem/distsys/clocks.pdf