Sistemas Distribuidos: Control de Concurrencia

Karina M. Cenci[1], Jorge R. Ardenghi[2]
Laboratorio de Investigación en Sistemas Distribuidos
Departamento de Ciencias de la Computación
Universidad Nacional del Sur – Bahía Blanca

Introducción

Los sistemas distribuidos son requeridos para incrementar la potencialidad de los servicios, aumentando la performance (velocidad), la tolerancia y la seguridad. Estos funcionan sobre grandes redes físicas, cuyo comportamiento puede no ser predecible.

Este tipo de sistemas está formado por procesos y recursos / servicios que son compartidos por los mismos. Para el correcto funcionamiento de los procesos es necesario contar con mecanismos que sincronicen los procesos cooperantes, o que controlen la utilización de recursos / servicios que sólo un número acotado de procesos pueden utilizar en un determinado momento.

Por ende, el desarrollo de sistemas distribuidos es muy complejo.

En todos los tipos de sistemas es necesario poder entender el dominio del problema y que este se encuentre documentado; en especial en este tipo de sistemas que son tan complejos y que se están convirtiendo en indispensables para varias aplicaciones y organizaciones. Una buena manera para manejar / controlar el incremento de la complejidad es a través de formas estructurales, observándolo en diferentes niveles de abstracción y como descomposición paralela de los componentes que interactúan.

Para poder especificar las propiedades de los sistemas distribuidos se utilizan modelos. El propósito del modelo es definir precisamente las propiedades o características de un sistema a construir o analizar y proveer las bases para la verificación de las propiedades.

Trabajo

En los sistemas distribuidos es importante analizar y verificar el funcionamiento de los procesos cooperativos y de aquellos que comparten recursos y/o servicios. Los procesos cooperativos pueden directamente compartir un espacio de dirección lógico (código y datos), o solamente comparte los datos a través de archivos.

El acceso concurrente a los datos compartidos puede resultar en una inconsistencia de los mismos. Se necesitan mecanismos para asegurar un ordenamiento en la ejecución de procesos cooperativos que comparten un espacio de direccionamiento lógico, tal que la consistencia en los datos sea respetada.

[1] e-mail: kmc@cs.uns.edu.ar
[2] e-mail: jra@cs.uns.edu.ar

Los mecanismos de concurrencia para asegurar el ordenamiento en la ejecución de procesos: sincronización y exclusión mutua y alocación de recursos en un entorno de memoria compartida asincrónica son las herramientas que se están considerando en el trabajo. Las condiciones que debe respetar un protocolo para soportar estos mecanismos son:

- Buena formación
- Exclusión (exclusión mutua)
- Progreso

Las características para considerar bueno a un protocolo son: libre de interbloqueo, libre de inanición e imparcialidad.

En la primera etapa se comienza con el estudio de algoritmos de exclusión mutua utilizando el paradigma de memoria compartida asincrónica. La presentación de estos algoritmos se basa inicialmente en el formato tradicional de un algoritmo, como el mismo puede presentar ambigüedades se lo rescribe utilizando el modelo de autómata de I/O.

Las motivaciones para utilizar *autómata de I/O* es que modela el comportamiento de un sistema distribuido que pueden interactuar con otros componentes del sistema. Es un simple tipo de máquina de estados en el cual las transiciones son asociadas con los nombres de acciones. Las acciones son clasificadas como entrada, salida ó internas. Las entradas y salidas son utilizadas para la comunicación con el ambiente de autómata, mientras que las acciones internas son sólo visibles por él mismo. Este modelo soporta un conjunto de métodos de prueba, incluyendo invariant assertion técnicas para probar que una propiedad es verdadera en todos los estados alcanzables. (reachable)

Los algoritmos utilizan variables de control para garantizar la exclusión mutua entre los procesos. De acuerdo al modo de acceso de las variables se los puede clasificar:

- Múltiple-lectura y múltiple-escritura. Casos de estudio son los algoritmos de Peterson 2P, Peterson NP, Tournament.
- Múltiple-lectura y simple-escritura. Casos de estudio son los algoritmos de Panadero, Burns.

En la segunda etapa se analizan los algoritmos para exclusión mutua utilizando el paradigma de pasaje de mensajes. Los algoritmos se pueden clasificar en:

- Basados en un ficha (token). La ficha representa un punto de control y es pasada entre todos los procesos. Algunos casos de estudio son: Ricart y Agrawala's segundo algoritmo, un algoritmo simple basado en ficha.
- No basados en una ficha. Todos los procesos se comunican entre sí para determinar cuál es el próximo que puede ejecutar la sección crítica. Algunos casos de estudio son: Lamport, Ricart y Agrawala y Maekawa.

En estos algoritmos es importante considerar la cantidad de mensajes necesarios para autorizar que un proceso acceda a la sección crítica.

En la tercera etapa, se trabajará sobre la problemática de alocación de recursos. El problema de exclusión mutua, es un extracto del problema de alocación de recurso conteniendo accesos de usuarios concurrentes a un único recurso no compartido. La generalización del problema incluye varios recursos en vez de uno. Se considerará la especificación de recursos explícitos y de exclusión para resolver el problema de alocación mediante la utilización de memoria compartida.

En las diferentes etapas, se trabajará con el mismo modelo (autómata de I/O), por las características presentadas facilita la verificación de las condiciones necesarias que debe mantener un buen algoritmo.

En la comparación de los diferentes protocolos se tiene que considerar:

- La complejidad de implementación, en el caso de memoria compartida: tiene un elevado costo la implementación de algoritmos de múltiple-escritura y múltiple lectura.
- Libre de interbloqueo: un protocolo en cualquiera de los casos puede soportar la exclusión, pero el algoritmo en algún caso puede bloquear el sistema.
- Libre de inanición: un proceso permanezca indefinidamente en la región de entrada a la sección crítica.

Los algoritmos presentados están diseñados para trabajar sobre un modelo de sistema distribuido. En el caso aquellos que fueron especificados para un modelo asincrónico, ¿es posible implementarlos en otro modelo manteniendo las mismas propiedades?. La respuesta a esta pregunta forma parte del trabajo que se va a realizar durante el período.

Bibliografía

[1] Jie Wu, *Distributed System Design*, 1999.

[2] Gary L. Peterson, Myths about the mutual exclusion problem. *Information Processing Letters*, Junio 1981.

[3] Nancy A Lynch. *Distributed Algorithms*, 1997.

[4] Sape Mullender. *Distributed Systems*, 2da. Ed. 1993.

[5] Gary L. Peterson y Michael Fischer. Economical solutions for the critical section problem in a distributed system. *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing*, Mayo 1977.

[6] Michael Raynal. *Algorithms for Mutual Exclusion*. MIT Press, Cambridge, 1986.

[7] M. Ben Ari. *Principles of Concurrent Programming*. Prentice Hall, Englewood Cliffs, 1982.

[8] Sandeep Lodha y Ajay Kshemkalyani. *A fair Distributed Exclusion Algorithm*, 2000.