Modelos de Consistencia y Protocolos de Coherencia en DVSM

J. Echaiz, R. B. García, J. R. Ardenghi

Laboratorio de Investigación de Sistemas Distribuidos LISiDi
Dpto. de Ciencias de la Computación
Universidad Nacional del Sur
E-mail: {je, rbg, jra}@cs.uns.edu.ar

Palabras clave: Distributed Virtual Shared Memory (DVSM), Scope Consistency (ScC)

1 Introducción

Los sistemas de Memoria Compartida Distribuida (DSM) son el vehículo ideal para la programación paralela debido a las facilidades de programación que brinda la memoria compartida y a la escalabilidad de los sistemas distribuidos. El reto de construir un DSM es lograr una buena performance sobre un amplio espectro de programas paralelos sin requerir que los programadores reestructuren sus programas de memoria compartida.

Por su parte, en la implementación por software de estos sistemas, del tipo DVSM, se tiene la tendencia a una gran cantidad de comunicación que se debe realizar entre procesadores para mantener consistente la memoria. Desde la creación de los primeros DVSM se han aplicado diversas alternativas para aliviar este cuello de botella en la performance. La mayoría de ellas se concentran en los *modelos de consistencia de memoria*, i.e. se encargan de definir como se ve la memoria compartida frente al programador, determinan la interface entre el programador y el sistema [11]. Una tendencia en estas alternativas es el empleo de modelos relajados, los cuales aumentan la complejidad del protocolo pero reducen el tráfico en la red mientras siguen manteniendo consistente la memoria. Ejemplo de ello es la *lazy release consistency* (*LRC*) [1] en TreadMarks [7] o la *scope consistency* (*ScC*) [2] en JIAJIA v1.1 [5]. Otras implementaciones tratan de reducir el tráfico refinando protocolos de coherencia de memoria, como el *protocolo de migración de home* en JIAJIA v2.1 [8] y el de *home migratorio* en JUMP [4].

Uno de los modelos más recientes de consistencia es la *Scope Consistency* (*ScC*), el cual se desarrolló [2] con dos objetivos precisos: (1) programabilidad (no se necesita ligar explícitamente los datos con las variables de sincronización, como por ejemplo en la *consistencia entry*) y (2) performance (disminuye el false sharing y la cantidad de comunicación).

En la implementación de la ScC se pueden emplear dos tipos de protocolos:

- *Basados en Home*: (también llamados protocolos de *home fijo*) un nodo del cluster es designado para mantener la copia más actual de cada página de memoria compartida en el sistema. Como consecuencia, cuando un procesador solicita una página (mediante una operación *request*), solamente necesita comunicarse con el home de la misma.
- *Homeless*: no existe el concepto de home, por lo tanto ningún procesador es responsable de tener la copia más actual de una página. Como consecuencia, para atender un page fault, un procesador debe comunicarse con todos aquellos que hayan modificado dicha página para obtener las actualizaciones (por ejemplo utilizando diffs [7]). Luego, el procesador que causó el fault aplica las actualizaciones en orden (según las estampillas de tiempo asociadas a los diffs, las cuales indican cuándo se hicieron las modificaciones) para obtener una copia válida de la página. Esta alternativa, si bien evita transferir páginas completas, presenta los inconvenientes de aumentar el overhead de procesamiento y la polución de cache.

Nuestra línea de investigación se orienta al primer esquema, esto es *Basado en Home*, dado que la dificultad de mayores volúmenes de información transmitida se ve atenuada por los crecientes anchos de bandas de las redes actuales.

2 Análisis y Comparación de Protocolos de Coherencia

Un protocolo de coherencia especifica como se implementará el conjunto de reglas definidas por el modelo de consistencia de memoria. Diferentes protocolos sobre un mismo modelo de consistencia conducirán a distinta performance.

El *protocolo de home migratorio* utilizado en JUMP [4] tiene como objetivo mantener la consistencia de memoria en un contexto de mínimo overhead de comunicación en la red. Hemos comparado este protocolo con el *homeless* empleado por TreadMarks, el *basado en home*, usado en JIAJIA v1.1, y el *de migración de home* propuesto en JIAJIA v2.1.

Las características relevantes de estos protocolos son:

- En el *protocolo basado en home (fijo)* cada página de memoria compartida tiene asociado un procesador para mantener su copia más actual. Este procesador es el *home* de la página y las modificaciones hechas por cualquier otro procesador se propagan al procesador home.
- En el *protocolo homeless*, como su nombre lo indica, no existe el concepto de home, por lo cual ningún procesador es responsable de mantener la copia más actual de una página.
- En el *protocolo de migración de home*, se permite que el home de una página migre si existe solamente un único procesador que escribe en la página entre dos barreras (*intervalo*).
- En el *protocolo de home migratorio* se permite al home de una página migrar de un procesador a otro cuando este último le hace un pedido de una página al primero (bajo ciertas condiciones).

Los protocolos basados en el concepto de home aseguran que al momento de sincronización el home contiene todas las modificaciones realizadas, de esta forma se garantiza que ésta sea la copia maestra, i.e. que sea la más actual.

Atender un *page fault* en un sistema bajo el protocolo homeless no es una tarea trivial. Un procesador debe comunicarse con sus pares que hayan modificado esa página para obtener las actualizaciones (por ejemplo mediante diffs [7]). Luego, para obtener la copia de la página, el procesador que causó el fault aplica las actualizaciones en orden (según las estampillas de tiempo asociadas a los diffs, las cuales indican cuándo se hicieron las modificaciones). En [6] se muestra que el protocolo de coherencia de cache basado en home es más eficiente que su contrapartida homeless. Más aún, el protocolo basado en home es más sencillo de implementar debido a que no necesita manejar estampillas de tiempo.

Aunque más eficiente, para el caso de home fijo, el sistema no podría adaptarse correctamente al patrón de acceso de la aplicación, incrementando considerablemente el overhead de red.

Por su parte, el protocolo de home migratorio, al mover la ubicación del home de una página hacia el procesador que va a accederla, reducirá el número de page faults. El protocolo manejará *migration notices* para notificar a los otros procesadores del cluster sobre los cambios de ubicación del home al momento de sincronización. Debido al alto número de notices transmitidos se podrían concatenar varios mensajes en un mensaje único. Para optimizar aún más el protocolo se puede recurrir a librerías de comunicación a nivel usuario, por ejemplo Socket-DP (JUMP-DP) basado en el *Modelo de Abstracción Directed Point* (*DP*) [3].

Los protocolos de home migratorio (JUMP) y de migración de home (JIAJIA v2.1) comparten el mismo objetivo: adaptarse a los patrones de acceso a memoria de las aplicaciones en un sistema DVSM. En el de migración de home podemos reducir el overhead de enviar la ubicación del home a los nodos incorporando esta información en el mensaje de otorgamiento de barrera. A su vez la regla de migración del home es demasiado estricta, en el sentido de restringirlo a aquellas páginas que han sido modificadas por un único procesador en el intervalo entre dos barreras. Por otro lado si las aplicaciones solo emplean locks como mecanismo de sincronización no podrían beneficiarse con la migración del home. Estos dos problemas no se presentan en el protocolo de home migratorio.

3 Trabajos Futuros

El protocolo utilizado en la implementación del modelo de consistencia de memoria es tan importante como el modelo mismo ya que tiene un efecto directo sobre el comportamiento global del DVSM.

El sistema JIAJIA, tomado como ejemplo puntual de un protocolo de migración de home (fijo) que implementa la ScC, es más eficiente que el protocolo homeless de TreadMarks, sin embargo no logra el desempeño del protocolo de home migratorio presente en JUMP.

El protocolo de home migratorio logra un menor tiempo de ejecución en los sistemas DVSM y es capaz de adaptarse mejor que el resto de los protocolos a los patrones de acceso a memoria de la mayoría de las aplicaciones. Esto se debe a una reducción dramática en el número de bytes transmitidos en las comunicaciones y a un menor procesamiento de los diffs.

Uno de los ítem fundamentales a tener en cuenta en la construcción de futuros sistemas DVSM es el protocolo de coherencia. Resulta evidente que deben invertirse esfuerzos en la creación de protocolos cada vez más eficientes, siendo en la actualidad el protocolo de *home migratorio* el que resulta más promisorio.

Nuestra línea de investigación se va a orientar a sistemas de home migratorio en un contexto de actualización automática.

Bibliografía

[1] P. Keleher, A. L. Cox y W. Zwaenepoel. Lazy release consistency for software distributed shared memory. *Proceedings of the 19th Annual International Symposium on Computer Architecture* (ISCA '92), p. 13-21, may. 1992.

[2] L. Iftode, J. P. Singh y K. Li. Scope consistency: A bridge between release consistency and entry consistency. *Proceedings of the 8th ACM Annual Symposium on Parallel Algorithms and Architectures* (SPAA '96), pp. 277-287, jun. 1996.

[3] W. Zhu, D. Lee y C. I. Wang. High Performance Communication Subsystem for Clustering Standard High-Volume Servers using Gigabit Ethernet. *HPC Asia 2000*, may. 2000.

[4] B. Cheung, C. I. Wang y K. Hwang. A Migrating-Home Protocol for Implementing Scope Consistency Model on a Cluster of Worksations. *1999 International Conference on Parallel and Distributed Processing Techniques and Applications* (PDPTA '99), Las Vegas, Nevada, USA.

[5] W. Hu, W. Shi y Z. Tang. A lock-based cache coherence protocol for scope consistency. *Journal of Computer Science and Technology*, 13(2):97-109, marzo 1998.

[6] Y. Zhou, L. Iftode y K. Li. Performance evaluation of two home-based lazy release consistency protocols for shared memory virtual memory systems. *Proceedings of the 2nd Symposium on Operating Systems Design and Implementation* (OSDI '96), pp. 75-88, oct. 1996.

[7] P. Keleher, S. Dwarkadas, A. L. Cox y W. Zwaenepoel. Treadmarks: Distributed Shared Memory on standard workstations and operating systems. *Proceedings of the Winter 1994 USENIX Conference*, pp. 115-131, enero 1994.

[8] W. Hu, W. Shi y Z. Tang. JIAJIA: An SVM system based on a new cache coherence protocol. *Proceedings of the High-Performance Computing and Networking Europe 1999* (HPCN '99), pp. 463-472, abril 1999.

[9] B. Cheung, C. L. Wang y K. Hwang. JUMP-DP: A Software DSM System with Low-Latency Communication Support, *CCTEA 2000*.

[10] W. Hu, W. Shi y Z. Tang. Home migration in home-based software DSMs. *Proceedings of the 1st Workshop on Software Distributed Shared Memory* (WSDSM '99), jun. 1999.

[11] S. V. Adve y K Gharachorloo. Shared Memory Consistency Models: A Tutorial. *IEEE Computer* 29(12):66-76, dic. 1996.