Avaliação de Memórias Transacionais para Máquinas NUMA *

Jerônimo da Cunha Ramos, André Rauber Du Bois, Maurício Lima Pilla

Programa de Pós-Graduação em Computação – Universidade Federal de Pelotas
Campus Porto – Rua Gomes Carneiro, 1 – Pelotas – RS – Brasil

{jdcramos, dubois, pilla}@inf.ufpel.edu.br

Resumo. *Memória transacional é uma abstração para programação concorrente baseada na ideia de transações, parecidas com as de banco de dados. As Máquinas NUMA, quando comparadas aos multi-core comuns, possuem o diferencial do tempo de acesso a memória variável. Motivado por isto, esta proposta tem como principal objetivo avaliar o comportamento de implementações de memórias transacionais já existentes em máquinas NUMA.*

1. Memórias Transacionais

Para tirar proveito do paralelismo disponibilizado pelos computadores atuais, os programas devem conter atividades que possam executar concorrentemente. Normalmente, em arquiteturas de memória compartilhada, isto é feito com o uso de *threads*, variáveis compartilhadas e *locks*. Porém, este modelo de programação impõe quase toda a complexidade de exploração do paralelismo e de tratamento das condições de corrida ao programador, tornando-o propenso a erros, como *deadlocks*. A comunidade científica vem propondo alternativas a este modelo para facilitar e popularizar a programação concorrente, como é o caso das memórias transacionais [Rigo et al. 2007].

Uma transação de memória é uma sequência atômica de operações que modificam a memória e que podem ser executadas completamente ou podem ser abortadas. Dessa maneira, transações em memória executam como se estivessem isoladas, no sentido de que as *threads* executam as operações como se estivessem modificando uma área de memória isolada do acesso de outras *threads*. Estas só conseguem ver o resultado após o *commit*. Este modelo possui várias vantagens sobre o modelo clássico de programação, sendo as principais: ausência de deadlock, possibilidade de abortar e retornar ao estado original do programa, maior possibilidade de exploração do paralelismo, facilidade de programação, escalabilidade e composabilidade [Rigo et al. 2007].

2. Máquinas NUMA

As máquinas NUMA (*Non-Uniform Memory Access*) são compostas por módulos de processador e memória. O acesso é dito "não uniforme" devido ao fato de que acessar a memória de outro módulo é mais demorado do que a local. Como elas possuem um único espaço de endereçamento e um único sistema operacional, diferentemente dos *clusters*, programas feitos para máquinas UMA (*Uniform Memory Access*) funcionam corretamente em NUMA, desde que estas mantenham coerência de cache. Mesmo funcionando, pode haver perda de desempenho se a localidade dos dados não for avaliada. Por outro lado, arquiteturas NUMA possuem a grande vantagem de não apresentarem

*Este trabalho é financiado pelos projetos FAPERGS/PRONEX/CNPq GREEN-GRID e FAPERGS/PESQUISADOR GAUCHO.

aumento no gargalo de acesso ao barramento, quando se aumenta o número de processadores [Caríssimi et al. 2007].

3. Proposta

As pesquisas sobre Memórias Transacionais têm sido bastante focadas em máquinas multi-core, deixando outras arquiteturas, como *clusters* e máquinas NUMA, quase inexploradas. Existem algumas propostas para *clusters*, como é o caso de Cluster-STM [Bocchino et al. 2008] e de DiSTM [Kotselidis et al. 2008]. Porém, ainda não existem muitos trabalhos sobre memórias transacionais que abordem as particularidades de máquinas NUMA [Lu et al. 2010]

Motivado por isto, este estudo tem como objetivo geral analisar o comportamento de várias implementações de memórias transacionais em uma máquina NUMA. Para tal, foram estudados os conceitos de Memórias Transacionais, Transações Distribuídas e trabalhos relacionados. Atualmente está sendo executado o *benchmark* Stamp [Cao Minh et al. 2008] com diferentes implementações de memórias transacionais, tais como TinySTM [Felber et al. 2008] e SwisSTM [Dragojević et al. 2009], em uma máquina NUMA, para compreender como elas se comportam em máquinas com esta arquitetura.

4. Considerações Finais

O modelo de memórias transacionais tem se mostrado bastante promissor, porém, devido às latências variadas das máquinas NUMA, os modelos de memória transacional já propostos podem perder muito desempenho. Este trabalho está em etapa inicial, porém, espera-se que os estudos que estão sendo realizados auxiliem a modelar e validar a programação com memória transacional para a arquitetura NUMA.

Referências

Bocchino, R. L., Adve, V. S., and Chamberlain, B. L. (2008). Software transactional memory for large scale clusters. In *Proceedings of the 13th ACM SIGPLAN PPoPP*.

Cao Minh, C., Chung, J., Kozyrakis, C., and Olukotun, K. (2008). STAMP: Stanford transactional applications for multi-processing. In *Proceedings of the IISWC*.

Caríssimi, A., Dupros, F., Méhaut, J.-F., and Polanczyk, R. V. (2007). Aspectos de programação paralela em arquiteturas NUMA. In *Minicursos do VIII WSCAD*.

Dragojević, A., Guerraoui, R., and Kapalka, M. (2009). Stretching transactional memory. In *Proceedings of the 2009 ACM SIGPLAN PLDI*, pages 155–165.

Felber, P., Fetzer, C., and Riegel, T. (2008). Dynamic performance tuning of word-based software transactional memory. In *Proceedings of the 13th ACM SIGPLAN PPoPP*.

Kotselidis, C., Ansari, M., Jarvis, K., Luján, M., Kirkham, C., and Watson, I. (2008). Distm: A software transactional memory framework for clusters. In *Proceedings of the 2008 37th International Conference on Parallel Processing*, pages 51–58.

Lu, K., Wang, R., and Lu, X. (2010). Brief announcement: Numa-aware transactional memory. In *Proceedings of the 29th ACM SIGACT-SIGOPS*, PODC '10, pages 69–70.

Rigo, S., Centoducatte, P., and Baldassin, A. (2007). Memórias transacionais: Uma nova alternativa para programação concorrente. In *Minicursos do VIII WSCAD*.