Otimização do Mapeamento de *Threads* e Dados em Aplicações com Memória Transacional usando Aprendizado de Máquina

Tiago Perlin[1,2], **Andre Rauber Du Bois**[1]

[1]Centro de Desenvolvimento Tecnológico – Universidade Federal de Pelotas (UFPel)
Pelotas/RS

[2]Instituto Federal de Educação, Ciência e Tecnologia Farroupilha (IFFar)
Santa Maria/RS

{tiago.perlin,dubois}@inf.ufpel.edu.br

Resumo. *Memórias Transacionais facilitam a programação e evitam a ocorrência de deadlocks durante a execução da aplicação em sistemas com múltiplas threads. O mapeamento das threads entre os núcleos de processamento e a alocação de dados influenciam o desempenho das aplicações. Neste trabalho é proposta a utilização de algoritmos de aprendizado de máquina para a otimização em conjunto das políticas de mapeamento de threads e de dados, coletando-se métricas da Memória Transacional.*

1. Introdução

As Memórias Transacionais (TM - *Transactional Memory*) [Pasqualin et al. 2020] têm sido usadas como alternativas aos mutexes e semáforos, facilitando a programação e evitando a ocorrência de *deadlocks*. Neste paradigma, as regiões críticas do código são delimitadas na forma de transações, ideia semelhante às transações em bancos de dados, e podem ser executadas concorrentemente, pois, em caso de conflitos, deverão ser abortadas e reexecutadas. Em máquinas com arquitetura de memória do tipo NUMA (*Non-Uniform Memory Access*), a escolha da política de distribuição das *threads* da aplicação entre os núcleos de processamento, bem como, a política de alocação de páginas de memória entre os nós de memória, importam devido às diferenças nos tempos de acesso à memória entre os nós e à contenção na memória *cache*. A seleção das políticas de mapeamento de *threads* e de dados, de modo complementar [Denoyelle et al. 2019], podem ser exploradas para a otimização da execução da aplicação. As informações coletadas da própria TM podem ser usadas para guiar a otimização do mapeamento de *threads* [Castro et al. 2014] [Pasqualin et al. 2020]. Este trabalho tem o objetivo de investigar a otimização de aplicações com TM por meio da escolha em conjunto da melhor política de mapeamento de *threads* e de alocação de dados.

2. Mapeamento de *Threads* e Dados em Aplicações com Memória Transacional e Aprendizado de Máquina

Algoritmos de Aprendizado de Máquina (ML - *Machine Learning*) têm sido usados na otimização de parâmetros de configuração em TM [Di Sanzo et al. 2019]. Além disso, no trabalho de [Castro et al. 2014] a otimização da política de mapeamento de *threads* é realizada utilizando-se métricas de desempenho coletadas da TM e do ambiente de execução.

Neste trabalho está sendo proposta a utilização de algoritmos de ML para a seleção da melhor combinação de políticas de mapeamento de *threads* e de mapeamento de dados em aplicações com TM executadas em máquinas com arquitetura do tipo NUMA.

No trabalho de [Pasqualin et al. 2020] foi demonstrado que informações quanto às variáveis compartilhadas podem ser coletadas na própria TM e usadas para otimização do mapeamento das *threads*. De forma semelhante, neste trabalho, propõe-se a coleta tanto de métricas da TM [Pasqualin et al. 2020], quanto de contadores de *hardware* [Castro et al. 2014], como as faltas no acesso à *cache*. As métricas coletadas da TM podem indicar o nível de compartilhamento de dados entre as *threads*, enquanto os contadores de *hardware* servem para representar o nível de contenção no acesso à memória. Como a definição da melhor combinação das políticas de mapeamento de *threads* e dados a partir das métricas coletadas não é trivial, propõe-se o uso de modelos de ML supervisionada, como árvores de decisão [Castro et al. 2014]. O treinamento será *offline*, com um conjunto de exemplos formado por métricas coletadas previamente com *benchmarks*, tendo-se como variável alvo a combinação das políticas de mapeamento de *threads*, *default*, *compact* e *scatter*, e de dados, *first-touch*, *interleaved* e NUMA *Balancing*. Posteriormente, o modelo treinado deve ser integrado à biblioteca da TM para atuação *online*.

3. Conclusões

A seleção de políticas de mapeamento de *threads* e de alocação de dados é importante quando busca-se alto desempenho. Informações coletadas durante a execução de aplicações com TM podem ser usadas para guiar o mapeamento. Neste trabalho está sendo explorada a relação destas áreas, com a proposição de um mecanismo para otimização da execução de aplicações em máquinas do tipo NUMA usando-se ML.

Agradecimentos

O presente trabalho está sendo realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Referências

Castro, M., Góes, L., and Méhaut, J.-F. (2014). Adaptive thread mapping strategies for transactional memory applications. *Journal of Parallel and Distributed Computing*, 74.

Denoyelle, N., Goglin, B., Jeannot, E., and Ropars, T. (2019). Data and thread placement in NUMA architectures: A statistical learning approach. In *Proceedings of the 48th International Conference on Parallel Processing*, ICPP 2019, New York, NY, USA. Association for Computing Machinery.

Di Sanzo, P., Pellegrini, A., Sannicandro, M., Ciciani, B., and Quaglia, F. (2019). Adaptive model-based scheduling in software transactional memory. *IEEE Transactions on Computers*, PP:1–1.

Pasqualin, D. P., Diener, M., Du Bois, A. R., and Pilla, M. L. (2020). Online sharing-aware thread mapping in software transactional memory. In *2020 IEEE 32nd International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, pages 35–42.