UMA PROPOSTA DE IMPLEMENTAÇÃO DE ÁRVORES RUBRO-NEGRAS EM HASKELL DE FORMA CONCORRENTE

THAIS SILVEIRA HÜBNER; RAISSA TELES PINTO;
RODRIGO M. DUARTE; ANDRÉ R. DU BOIS

UFPEL – UNIVERSIDADE FEDERAL DE PELOTAS
LUPS – Laboratory of Ubiquitous and Parallel Systems
{tshubner, rtpinto, rmduarte, dubois}@inf.ufpel.edu.br

1. INTRODUÇÃO

Processadores multicore refletem o mais recente avanço em arquitetura de processadores. Um computador multicore possui dois ou mais núcleos completos de processamento no mesmo chip, hoje essa arquitetura está presente em grande parte dos computadores (desktops ou laptops).

Para se conseguir explorar o máximo desempenho desta nova arquitetura, é necessário que a programação seja realizada de forma concorrente, ou seja, é preciso mais de um fluxo de execução (thread) trabalhando para chegar a um resultado consistente. Porém obter a correta sincronização das threads no acesso a memória compartilhada do computador não é algo trivial.

O modelo de sincronização mais usual á a utilização de bloqueios (locks), que são de difícil utilização e são propensos a erros de programação e.g. (deadlocks) (RAJWAR, 2003).

A *memória transacional* é uma nova abstração para programação baseada na ideia de transações de banco de dados, e que visa facilitar a programação concorrente. Este novo método fornece maior facilidade na programação, ausência de deadlock e a composabilidade de código, muito importante na engenharia de software (RIGO, 2007). STM Haskell é uma biblioteca da linguagem funcional Haskell que fornece a abstração de memórias transacionais para programação paralela.

Este trabalho traz como proposta a implementação de árvores rubro negras concorrentes em Haskell utilizando os diferentes métodos de sincronismo existentes nesta linguagem, a fim de realizar a análise de desempenho em diferentes contextos de execução como também a facilidade de implementação.

2. METODOLOGIA

2.1 - Metodos de Sincronismo em Haskell

Haskell possui diferentes abstrações para realizar o sincronismo entre threads, como variáveis de sincronização (MVars), abstrações de alto nível como STM Haskell (TVars) e acesso a instruções de baixo nível (IORef + AtomicModifyIORef). Estas abstrações aliadas a facilidade da linguagem funcional Haskell fornecem uma excelente ferramenta para o desenvolvimento de programas paralelos (MARLOW, 2013). A seguir, apresentamos uma breve descrição dessas abstrações.

MVar: MVars são tipos especiais de variáveis que podem assumir dois estados: cheia ou vazia. Podem ser criadas com um valor inicial ou vazias através de duas funções (newMVar e newEmptyMVar). Contam com duas funções para manipulação: takeMVar retorna o valor de MVar se estiver cheia, ou bloqueia se estiver vazia e putMVar que opera de forma contrária, bloqueia caso esteja cheia e escreve se estiver vazia. MVars operam como os tradicionais mutex (locks).

IORef: Um IORef é uma referência a uma posição de memória e possui operações de criação, leitura e escrita (newIORef, readIORef e writeIORef). Haskell fornece uma função que permite a modificação atômica da referência (atomicModifyIORef), permitindo que uma IORef seja utilizada para implementar algoritmos não bloqueantes.

STM Haskell: STM Haskell é uma extensão da linguagem Haskell que forneçe a abstração de memórias transacionais para a programação concorrente. Nela é definida um novo tipo de variável (TVar), que pode ser criada, lida e escrita através das funções newTVar, readTVar e writeTVar, respectivamente. Esses comandos só podem ser utilizados dentro de uma chamada a função atomically, garantindo assim, que as operações que modificam uma TVar sejam utilizadas somente dentro de uma transação.

2.2 - Árvores Rubro Negras

As árvores rubro-negras são estruturas de dados do tipo árvores binária que inserem e removem de forma inteligente para assegurar que a árvore permaneça aproximadamente balanceada. Cada nó desse tipo de árvore possui: cor, valor, filho esquerdo, filho direito e pai (LEISERSON, 2012).

O objetivo neste tipo de estrutura de dados é garantir que as operações básicas demorem O (lg n) no pior caso, como auxílio para tal, existem cinco propriedades:

- Todo nó da árvore ou é vermelho ou é preto.
- A raiz é preta.
- Toda folha é preta.
- Se um nó é vermelho, então ambos os filhos são pretos.
- Para todo nó, todos os caminhos do nó até as folhas descendentes contêm o mesmo número de nós pretos.

3. PROPOSTA E DISCUSSÃO

Este trabalho propõe implementar as árvores rubro-negras na linguagem funcional Haskell utilizando os três métodos de sincronismo citados no trabalho (MVar, IORef + atomicModifyIORef e STM Haskell). Para a realização deste trabalho, serão estudados alguns algoritmos de árvores rubro-negras concorrentes como os apresentados em (KIM, 2006) e (PARK, 2001)

A implementação será feita com cada um dos métodos de sincronismo, os resultados serão analisados para determinar qual método apresentará melhor desempenho e facilidade na implementação.

4. CONCLUSÕES

Este texto apresenta uma proposta inicial de estudo das diferentes formas de sincronização existentes em Haskell e sua aplicação no desenvolvimento de uma estrutura de dados do tipo árvores rubro-negras. Os próximos passos do trabalho incluem um estudo mais aprofundado das abstrações de programação e dos algoritmos para árvores rubro-negras. Quando terminada a fase de estudo, serão desenvolvidos testes que serão executados nas máquinas disponíveis no laboratório LUPS da Computação/CDTec.

5. REFERÊNCIAS BIBLIOGRÁFICAS

RIGO, S.; CENTRODUCATT, P.; BALDASSIN, A. Memórias transacionais uma nova alternativa para programação concorrente. **Anais do WSCAD,** Porto Alegre, 2007.

RAJWAR, R.; GOODMAN, J. Transactional execution: Toward reliable, high-performance multithreading. **IEEE Micro**, 23:117–125, 2003.

KIM, J. H.; CAMERON, H.; GRAHAM, P. Lock-Free Red-Black Trees Using CAS. **Concurrency and Computation: Practice and Experience**, 1—40. 2006.

PARK, H.; PARK, K.Parallel algorithms for red--black trees.**Theoretical Computer Science**, 415—435, 2001.

LEISERSON, C. E.; RIVEST, R. L.; STEIN, C.; CORMEN, T.H. Introduction to algorithms, USA: **MIT Press**, 2012

MARLOW, S. Parallel and Concurrent Programming in Haskell, USA: **O'Reilly**, 2013.

HERLIHY, M.; SHAVIT, N. The Art of Multiprocessor Programming, USA: **Elsevier**, 2012.