Proceeding Series of the Brazilian Society of Applied and Computational Mathematics, Vol. 5, N. 1, 2017.

Trabalho apresentado no CNMAC, Gramado - RS, 2016.

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics

Proposta de um Escalonador de Transações para uma Linguagem Funcional Pura

Rodrigo M. Duarte[1]
Engenharia de Computação, CDTEc, UFPel, Pelotas, RS
André R. Du Bois[2]
Maurício L. PIlla[3]
Renata H.S. Reiser[4]
Curso de Ciência da Computação, CDTEc, UFPel, Pelotas, RS

1 Introdução

O paradigma de programação funcional se utiliza de funções matemáticas para expressar a solução de problemas computacionais. Programar em linguagens que usem este paradigma é basicamente definir funções matemáticas e utilizar computação para avaliar estas funções e ou expressões. Este estilo declarativo de programação é simples de aprender e utilizar [1] pois faz o programador focar no que fazer e não em como fazer, como é feito em linguagens de programação imperativas (C,C++, etc...)

Outro item importante das linguagens funcionais é que, assim como nas funções matemáticas, sempre é definido um mesmo valor quando fornecido o mesmo conjunto de argumentos, ou seja, as funções sempre retornam o mesmo valor não importando a ordem de avaliação das expressões. Essa característica facilita a geração de provas matemáticas para garantir a corretude dos programas desenvolvidos [6].

É notada que a necessidade de computação para a realização de cálculos matemáticos de forma eficiente e correta tem aumentado devido à requisição cada vez mais rápida de informação [3],(simulações de computação quântica, previsão do tempo etc...). Com o objetivo de reduzir o tempo gasto no processamento de cálculos cada vez mais complexos, a programação paralela e de alto desempenho tem sido essencial neste campo. Hoje em dia um simples computador doméstico possui muitas unidades de processamento (CPUs), fornecendo assim muito poder computacional. Para se conseguir explorar todo este recurso disponível é necessário que se utilize de programação paralela e concorrente. Com isso muito se tem pesquisado para facilitar e aumentar a abstração para o desenvolvimento de programas paralelos.

[1]rmduarte@inf.ufpel.edu.br
[2]dubois@inf.ufpel.edu.br
[3]pilla@inf.ufpel.edu.br
[4]reiser@inf.ufpel.edu.br

2 STM Haskell

Haskell é uma linguagem de programação funcional pura que possui várias ferramentas para o desenvolvimento de programas paralelos e concorrentes. Entre estas ferramentas está a de Memórias transacionais (MT) [5], na qual as computações são realizadas como transações, parecidas com as transações presentes em bancos de dados.

STM Haskell é uma extensão da linguagem Haskell que fornece primitivas para a programação usando MT [2] e que apresenta uma maior abstração para o desenvolvimento de programas paralelos. Essa característica, somada ao poder declarativo do paradigma funcional de Haskell, permite desenvolver programas concorrentes de forma fácil, rápida e segura [4].

Apesar dos grandes avanços na implementação da biblioteca STM Haskell, esta ainda pode ser otimizada para aumentar ainda mais o desempenho da mesma.

3 Proposta

Este trabalho traz como proposta a criação de um escalonador de transações para Haskell. O algoritmo proposto é a criação de filas de execução de transações para cada unidade de processamento, onde transações conflitantes são colocadas em uma mesma fila. Este algoritmo pretende evitar que transações que concorrem a um mesmo dado compartilhem de unidades de processamento distintas, evitando assim que uma transação que está fadada a ser cancelada ocupe recurso de processamento de forma fútil.

Referências

[1] T. H. C. Castro, J. A. N. Castro, C. S. Menezes, M. C. Silva. e M. C. P. V. Rauber, Utilizando programação funcional em disciplinas introdutórias de computação, UFMG, 2011.

[2] T. Harris, S. Marlow, S. P. Jones e M. Herlihy, Composable memory transactions, *Commun. ACM*, 2008. DOI: 10.1145/1378704.1378725.

[3] J. Jeffers e J. Reinders, *High Performance Parallelism Pearls: Multicore and Many-core Programming Approaches*, Morgan Kaufmann, Massachusetts - USA, 2015.

[4] S. Marlow, *Parallel and Concurrent Programming in Haskell: Techniques for Multicore and Multithreaded Programming*, O'Reilly Media, Inc.,California - USA, 2013.

[5] S. Rigo, P. Centoducatte e A. Baldassin, Memórias Transacionais: Uma Nova Alternativa para Programação Concorrente, *Minicursos do VIII Workshop em Sistemas Computacionais de Alto Desempenho, WSCAD*, 2007.

[6] R. W. Sebesta, *Conceitos de linguagens de programação*, Bookman Editora, Porto Alegre, 2011.