

Implementação de um algoritmo de tabela hash não bloqueante em Haskell

Rodrigo Medeiros Duarte¹

Prof. Dr. André R. Du Bois (Orientador)

Prof. Dr. Maurício Lima Pilla (Co-orientador)

Prof. Dr. Renata H. S. Reiser (Co-orientadora)

Engenharia de Computação
LUPS Laboratory of Ubiquitous and Parallel Systems
Universidade Federal de Pelotas
rmduarte@inf.ufpel.edu.br

Abril de 2015

¹Bolsista AF-PBIP UFPEL



- 1 **Introdução**
- 2 **Algoritmo não bloqueante**
- 3 **Metodos de Sincronismo em Haskell**
- 4 **Testes**
- 5 **Resultados**
- 6 **Conclusão**
- 7 **Trabalhos Futuros**



Introdução

Tabelas *Hash* Concorrentes

- Paralelismo Natural.
- Acesso aos dados são suscetíveis de serem disjuntos.

Dificuldades

- Conseguir desempenho não é algo trivial.
- Sincronizar o acesso a tabela para inserção, remoção e consulta.
- Duplicar a tabela de forma que ela continue consistente.

Motivação

- Haskell não possui implementações de *hash* concorrente.



Algoritmo de tabela *hash* não bloqueante utilizado

Algoritmo de tabela hash usando lista encadeada não bloqueante

- Os dados são armazenados em uma lista encadeada não bloqueante
- Cada posição da tabela é uma referencia a um dado ponto na lista (Guarda)
- Dados são ordenados na tabela pelo valor reverso dos bits da chave de entrada.
 - Método que evita a copia dos dados para uma tabela nova na necessidade de crescimento da mesma.



Tabela *Hash* não bloqueante

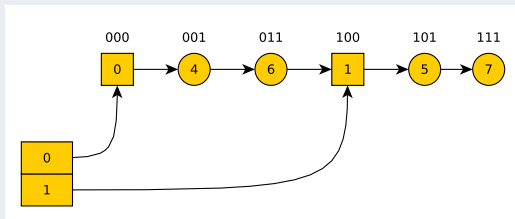


Figura: Estrutura da Tabela não bloqueante

Algoritmos de tabela *hash* não bloqueante

Algoritmo de tabela hash usando lista encadeada não bloqueante

- No inicio somente a posição zero é inicializada
- Cada guarda é inicializada conforme a necessidade.
- Método de ordenação de bits faz o re-hash automaticamente.



Inicialização de uma guarda

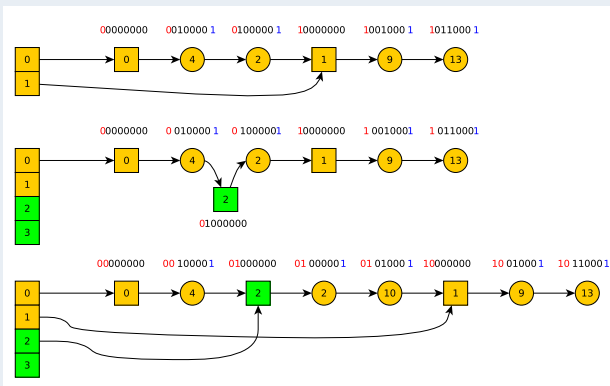


Figura: Passos para inserção da guarda de valor 2

IRef + atomicModifyIRef

- `newIRef :: a -> IO IRef a.`
- `readIRef :: IRef a -> IO a.`
- `writeIRef :: IRef a -> a -> IO ().`
- `atomicModifyIRef :: IRef a -> (a -> (a,b)) IO b.`
- Ideal para implementar algoritmos não bloqueantes.

STM Haskell

- `newTVar :: a -> STM TVar a.`
- `readTVar :: TVar a -> STM a.`
- `writeTVar :: TVar a -> a -> STM ().`
- `atomically :: STM a -> IO a.`
- Facilidade no uso, Sistema transacional garante o sincronismo

UPS

Testes

A Tabela foi implementada com dois diferentes tipos de sincronização

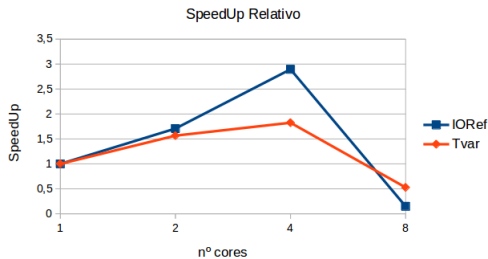
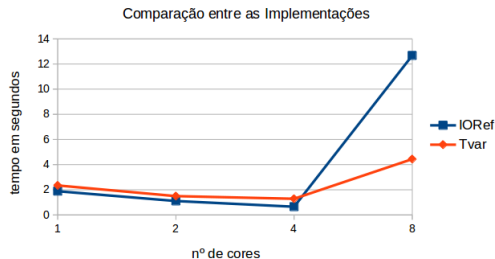
- Uma usando IOREf
- Outra usando TVar com STM Haskell

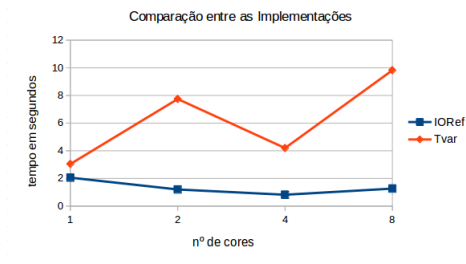
Máquina de teste

- Core i7 (4 cores físicos + 4 lógicos) de 3.0Ghz com 8Gb RAM
- Ubuntu 14.04 64bits.
- GHC 7.6.3 com STM Haskell versão 2.4.2

Testes realizados

- 1 milhão de operações (10% inserções, 10% remoções e 80% consultas)
- (80% inserções, 10% remoções e 10% consultas)
- 30 execuções de cada implementação com 1,2,4 e 8 processadores





Conclusão

STM apresentou o maior tempo de execução

- A implementação usando *IORef* teve o melhor desempenho.
 - Tanto para 10% como para 80% de inserções
 - Salientando o caso do uso de HyperThreading
 - Comportamento anômalo com 8 cores
- A implementação usando STM foi mais simples e apresentou desempenho muito semelhante até 4 threads (10% inserções).
- Ambas implementações apresentaram comportamento semelhante quando os números de inserções e remoções foram baixos.



Trabalhos Futuros

- Comparar os resultados com as implementações de um trabalho anterior.
- Realizar estudo estatístico para validar os resultados.
- Montar uma biblioteca para o Hackage afim de fornecer uma hash concorrente.



Agradecimentos

- UFPel pela bolsa AF-PBIP
- FAPERGS (PqG 002/2014, Processo 309533/2013-9,)
- MCTI/CNPQ (Universal/2014, Processo 448766/2014-0)



Implementação de um algoritmo de tabela hash não bloqueante em Haskell

Rodrigo Medeiros Duarte²

Prof. Dr. André R. Du Bois (Orientador)

Prof. Dr. Maurício Lima Pilla (Co-orientador)

Prof. Dr. Renata H. S. Reiser (Co-orientadora)

Engenharia de Computação

LUPS Laboratory of Ubiquitous and Parallel Systems

Universidade Federal de Pelotas

`rmduarte@inf.ufpel.edu.br`

Abril de 2015

