UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA

PROPOSTA DE TRABALHO DE GRADUAÇÃO

ÁREAS: PROGRAMAÇÃO FUNCIONAL, CONCORRENTE

ANÁLISE E IMPLEMENTAÇÃO DE ESTRUTURAS DE DADOS SEGURAS PARA THREADS, NÃO-BLOQUEANTES EM HASKELL

Aluno:

Paulo Vitor Julião Lieuthier <pvjl@cin.ufpe.br>

Orientador:

Fernando José Castor de Lima Filho <castor@cin.ufpe.br>

RECIFE

2016

SUMÁRIO

1 CONTEXTUALIZAÇÃO

Com a popularização de disposivitos de computação portáteis e a crescente dificuldade com os limites físicos de tamanho dos semicondutores, a corrida pela maior frequência de operação dos processadores perdeu sentido. Dispositivos portáteis dependem de fontes de energia limitadas, e alta frequência de processadores levam ao alto consumo de energia e dissipação de calor.

A solução encontrada foi a fabricação de processadores mulltinúlceo, componentes de computação com duas ou mais unidades de processamento que funcionam paralela e independentemente. Esse paralelismo, contudo, não é trivialmente aproveitável. O processo de desenvolvimento do software precisa levar em conta a capacidade de paralelização dos componentes de hardware.

A complicação de tal tarefa se deve principalmente à necessidade de sincronização de tarefas paralelas ou concorrentes: duas unidades de processamento podem ser programadas para fazer uso de um mesmo espaço de memória, potencialmente executadas sobrepostas. Essa sobreposição (interleaving) pode causar inconsistências na memória sendo compartilhada, gerando problemas de sincronização e comportamentos inesperados.

Pesquisa extensiva tem sido feita para encontrar soluções para problemas de sincronização. A solução mais comum é o uso de monitores (locks) para forçar a sequencialização do acesso à porções da memória compartilhada, chamadas de regiões críticas. O uso de monitores não é trivial, e o descuido leva a bugs díficeis de identificar e reproduzir, assim como deadlocks.

Além disso, o uso de travamento para sincronização pode acarretar alto custo de performance, especialmente quando muitas tarefas concorrentes são programadas para fazer uso da mesma região crítica, o que é chamado de contenção (lock contention).

Com a evolução dos componentes de hardware, processadores modernos são capazes de executar instruções já levando em consideração a execução de componentes de software paralelos ou concorrentes. Instruções atômicas podem ser utilizadas para desenvolver programas concorrentes seguros sem o uso de travamento, apesar de aumentar ainda mais as complexidades de implementação.

Uma vez que estruturas de dados são componentes fundamentais no desenvolvimento de software, existem muitas implementações de estruturas de dados concorrentes e paralelas. Contudo, há ainda relativamente poucos resultados de implementações sem travamento.

2 RESUMO DA PROPOSTA

Apesar de desejáveis em alguns cenários, algoritmos bloqueantes podem causar grandes penalidades de performance, especialmente com a ocorrência de contenção de travas (lock contention). Alternativas ao uso de travas explícitas têm sido sugeridas, como a concorrência probabilística (utilizando, por exemplo, memória transacional). Naturalmente, existe desvatagens em todas as alternativas, como a baixa eficiência na pararelização das tarefas. O objetivo desse trabalho é implementar, em Haskell, estruturas de dados concorrentes seguras para threads e não-bloqueantes, que são uma outra alternativa. A desvantagem nesse caso é a alta complexidade de implementação e a maior frequência de problemas sutis. Para satisfazer as propriedades de segurança e de não-travamento, instruções de máquina CAS (compare-and-swap) serão usadas. As estruturas de dados a serem implementadas inicialmente são Pilhas, Filas Prioritárias e Skiplists.

3 POSSÍVEIS AVALIADORES

Possíveis availadores, levando em consideração as áreas de atuação e pesquisa, são:

- André Luis de Medeiros Santos (alms@cin.ufpe.br)
- Francisco Miranda Soares da Silva Neto (fmssn@cin.ufpe.br)
- Márcio Lopes Cornélio (mlc2@cin.ufpe.br)
- Paulo Borba (phmb@cin.ufpe.br)

4 REFERÊNCIAS INICIAIS

1 Maurice Herlihy and Nir Shavit. 2008. The Art of Multiprocessor Programming. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

2 Sulzmann, M., Lam, E.S., Marlow, S.: Comparing the performance of concurrent linked-list implementations in Haskell. In: Proceedings of the 4th workshop on Declarative aspects of multicore programming (DAMP '09). ACM, New York, NY, USA, 37-46.

3 Fomitchev, M., Tuppert, E.: Lock-free linked lists and skip list. In: Preecedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing, PODC 2004, pp.50-59. ACM, New York (2004)

4 Simon Marlow. 2011. Parallel and concurrent programming in Haskell. In Proceedings of the 4th Summer School conference on Central European Functional Programming School (CEFP'11), Viktória Zsók, Zoltán Horváth, and Rinus Plasmeijer (Eds.). Springer-Verlag, Berlin, Heidelberg, 339-401

5 Anthony Discolo, Tim Harris, Simon Marlow, Simon Peyton Jones, and Satnam Singh. 2006. Lock free data structures using STM in haskell. In Proceedings of the 8th international conference on Functional and Logic Programming (FLOPS'06), Masami Hagiya and Philip Wadler (Eds.). Springer-Verlag, Berlin, Heidelberg, 65-80

6 DUARTE, Rodrigo Medeiros. Implementação de um algoritmo de tabela hash não bloqueante em Haskell. 2015. Tese de Doutorado. Universidade Federal de Pelotas.

7 ASSINATURAS

_____

Paulo Vitor Julião Lieuthier

Orientando

_____

Fernando José Castor de Lima Filho

Orientador