# UM ESTUDO DE CASO SOBRE LISTAS NÃO-BLOQUEANTES PARA O SISTEMA ANAHY

Juan da Silva Rios [1] ; Gerson Geraldo Homrich Cavalheiro[2]

*[1]Universidade Federal de Pelotas – jdsrios@inf.ufpel.edu.br*
*[2]Universidade Federal de Pelotas – gerson.cavalheiro@inf.ufpel.edu.br*

## 1. INTRODUÇÃO

Com o avanço da tecnologia nos últimos anos têm – se notado uma demanda cada vez maior por processamento de dados de maneira rápida, sejam elas de próposito acadêmico ou até mesmo de próposito geral. Devido a este acontecimento foi - se descoberto inúmeras técnicas de aumento de processamento, sendo a mais utilizada atualmente o processamento paralelo. Então foi proposto nesse artigo uma técnica de estruturas não-bloqueante para escalonamento de tarefas.

É comum dizer que uma estrutura não-bloqueante é uma forma de programar sem o uso de exclusão mútua(mutex), técnica essa que trata – se de uma forma de programar sem que haja a necessidade de saber muito sobre o código, isso é dado devido a capacidade de threads não se bloquearem entre si, ou seja, terem acesso simultaneamente a um recurso compartilhado sem que haja um erro, caractéristica essa que não pode ocorrer em estruturas bloqueantes, pois é necessário o uso de locks para que não se entre simultanamente em regiões criticas do código.

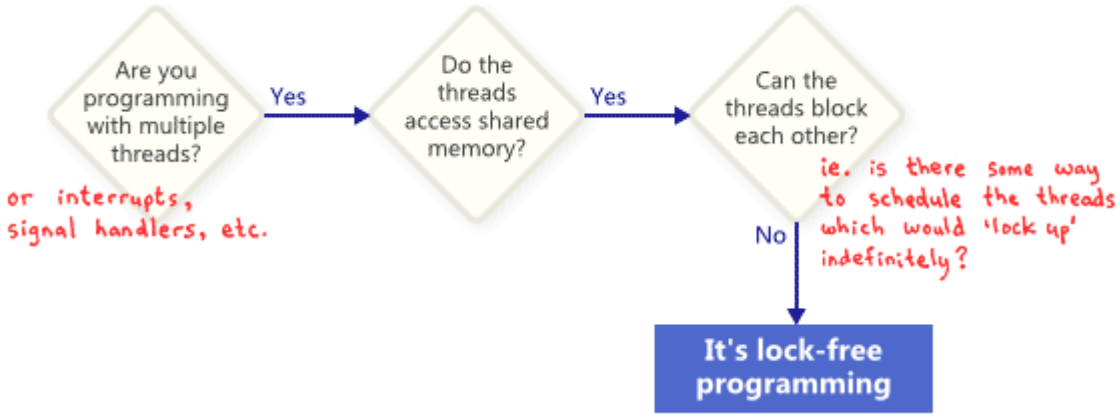
Listas encadeadas são uma das mais simples estruturas de dados desenvolvidos para um sistema computacional. Porém, devido a sua limitação por ser um sistema linear não é possível mexer, alterar determinada parte da estrutura ao mesmo tempo, pois esse recurso tem que ser alocado para o processador que está em execução no momento. Esse trabalho tenta propor uma ideia de como contornar tais limitações dessa estruturas através do uso de programação não-bloqueante(lock-free programming) e com isto a construção de um sistema onde sua estrutura de dados baseia – se no uso de listas não-bloqueantes com o uso de instruções atômicas como o compare-and-swap(CAS). A linguagem de programação utilizada para implementação do algoritmo foi o C++.

Na sessão 2 é apresentado a metodologia utilizada neste trabalho para a construção dessa estrutura. Na sessão 3 é apresentado os resultados obtidos bem como uma discussão a respeito dos dados coletados. Por fim, na sessão 4 é feita a conclusão do artigo e sugestões para trabalhos futuros.

## 2. METODOLOGIA

Foi desenvolvido um estudo de caso para a implementação de uma lista não-bloqueante no sistema Anahy. Sistema este desenvolvido no laborátorio onde este trabalho foi desenvolvido. Anahy é uma ferramenta que tem a pretenção de auxiliar o programador dando mais liberdade de programação e poder ao programador. Onde ele visa ter uma interface simples com o programador não necessitando saber o quão paralelizável a aplicação pode ser, pois um dos seus parametros de entrada consiste na função que deseja que o sistema execute de maneira paralela. Sua interface ainda conta com o uso da bibloteca pthread e sua estrutura de dados anteriormente eram baseadas em uma lista encadeada, onde

basiava – se no metodo de roubo de tarefas, técnica está que consiste na ideia ter uma lista de tarefas dinâmica, onde um processador rouba uma tarefa que está há mais tempo na lista para auxiliar  outro processador, resultando em uma diminuição significativa na migração de tarefas entre processadores. Então nesse trabalho foi proposto uma nova metodologia de escalonamento de tarefas, o uso de lock-free programming. Esse metodo é um novo jeito de se programar em paralelo, visando o uso de compartilhamento de recursos de forma simultanea sem a necessidade de lock de recursos.




Fonte: Preshing on Programming[1].

Então a implementação dessa lista foi utilizado métodos atômicos nátivos do compilador GCC, como exemplo o CAS, onde seus argumentos são um ponteiro *p* para a variável a ser comparada e dois booleanos, um para indicar o valor de comparação e outro para indicar quando for verdadeiro o teste qual será o novo valor, e então setar o valor no ponteiro apontado pelo *p.* Suas funções são as comuns de uma lista encadeada linear, contendo os metodos de inserção, remoção da cabeça e remoção da cola. Tratando – se de uma lock-free list de tarefas, para melhorar o desempenho de compartilhamento de recursos, ou seja, duas ou mais tarefas podendo utilizar o mesmo recurso simultaneamente.

Para a realização dos testes de desempenho do algoritmo foi utilizado um computador com um processador Intel i5 de 3.5 Ghz com 4 núcleos, 8 Gb de memória RAM e seu sistema operacional foi o Ubuntu 10.04.1 LTS com 64 bits.

## 3. RESULTADOS E DISCUSSÃO

Como o trabalho se encontra em desenvolvimento ainda, foram encontrados diferentes resultados divergentes, e até alguns discrepantes. Com a execução dos testes no algoritmo de lista lock-free foram descobertas diferentes situações a respeito do uso deste método, tendo em vista que a implementação dessa lista possívelmente irá resultar no aumento de desempenho do sistema, uma vez que a proposta do uso deste metódo é para aumentar ainda mais a capacidade do programador. Apesar de estar em desenvolvimento o trabalho apresenta bastante s resultados positivos. Então como proposta de trabalhos futuros é a inserção dessas listas de forma efetiva no sistema, de forma que possa tirar o máximo proveito desse metodo, e apresentar uma interface mais simples ao programador que utilizar o sistema Anahy, contando com heuristicas do mesmo modelo utilizado pela linguagem de programação C++11.

1 – Disponível em: <http://preshing.com/images/its-lock-free.png> Acessado em Ago 2016.

## 4. CONCLUSÕES

Com a execução deste trabalho pode se observar de maneira positiva o uso dessa técnica para a diminuição do overhead de escalonamento de tarefas. Um possível trabalho futuro e a implementação deste mesmo sistema de forma a optimizar ainda mais a diminuição do overhead no escalonamento.

## 5. REFERÊNCIAS BIBLIOGRÁFICAS

HERLIHY, M. e SHAVIT, N. **The Art of Multiprocessor Programming.** Elsevier Inc. 2012. 1v.

HARRIS T. An Pragmatic implematation of Non-blocking Linked-Lists. **DISC.** Cambridge. 2001.

Preshing. **Introduction to Lock Free Programming**. 12 junho. 2012. Acessado em 27 julho. 2016. Online. Disponível em: http://preshing.com/20120612/an-introduction-to-lock-free-programming/

VALOIS, J. Lock-Free Linked Lists Using Compare-and-Swap. **Rensselaer Polytechnic Institute.** Troy. 2002.

GIBSON, J e GRAMOLI, V. Why Non-Blocking Operations Should Be Selfish. **DISC**. Sydney. 2015

MICHAEL, M. SCOTT M. Simple, Fast, and Practical Non-Blocking and Blocking-concurrent queue algorithms. **Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing Concurrent Queue Algorithms.** New York. 1996. p. 267-275.

WILLIAMS, A. **C++ Concurrency in Action: Practical Multithreading.** Manning Publications. 2012. 1v.

1 – Disponível em: <http://preshing.com/images/its-lock-free.png> Acessado em Ago 2016.