UMA INTERFACE PARA PROGRAMAÇÃO CONCORRENTE BASEADA EM ESQUELETOS±

DEIVES MESQUITA KIST[1*]; ANDRÉ RAUBER DU BOIS[1];
GERSON GERALDO H. CAVALHEIRO[1]

[1]Programa de Pós-Graduação em Computação
Centro de Desenvolvimento Tecnológico - Universidade Federal de Pelotas (UFPel).
E-mails: {dmkist, dubois, gerson.cavalheiro}@inf.ufpel.edu.br.

1. INTRODUÇÃO

A crescente utilização das arquiteturas de computadores com várias unidades de processamento tem motivado o crescimento da demanda por *software* paralelo. No entanto, o desenvolvimento deste tipo de *software* possui alto grau de dificuldade, uma vez que o programador deve não apenas codificar seu algoritmo, mas também decompô-lo em termos de atividades concorrentes e gerir o bom uso dos recursos de *hardware* disponíveis. Devido a isto, nota-se um esforço da comunidade em disponibilizar ferramentas para a programação paralela com níveis mais elevados de abstração.

A programação paralela consiste na divisão de uma aplicação em partes, de maneira que essas partes possam ser executadas simultaneamente nos núcleos do processador. Para a execução paralela destas partes são utilizados os conceitos de processos e *threads*, que devem cooperar entre si utilizando primitivas de comunicação e sincronização fornecidas pela ferramenta de programação.

É por meio das ferramentas de programação paralela que o programador deve superar o desafio de converter um programa sequencial em paralelo e distribuir as partes paralelas entre os núcleos de processamento. O desenvolvedor também deve se preocupar com as formas de acesso aos dados pelos processos, pois se o acesso não for controlado poderá gerar uma inconsistência de dados. Outra importante questão que o programador deve resolver são as formas de comunicação e sincronização entre os fluxos de execução, pois ele deve indicar claramente a dependência entre eles, para coordenar a manipulação concorrente dos dados. Somando a essas questões existe o fato que a depuração se torna mais difícil, pois muitos dos *bugs* só se manifestam em tempo de execução, em função da natureza não determinística desta execução.

Neste trabalho utilizamos o ambiente de programação *multithread* Anahy, implementado pela ferramenta Athreads CAVALHEIRO; GASPARY e CARDOZO et al. (2006) como um subconjunto dos serviços previstos pelo padrão *POSIX Threads*. Este ambiente oferece além dos recursos para descrição e controle da concorrência, facilidades para o desenvolvimento de aplicativos paralelos por meio de mecanismos de escalonamento. Estes mecanismos são executados de forma automática pelo ambiente de execução deixando o programador livre da responsabilidade de mapear as atividades entre os núcleos do processador. Porém, esta ferramenta oferece poucos mecanismos para auxiliar o programador na descrição da concorrência de sua aplicação.

A interface de programação de Athreads propõe o uso de duas primitivas, *create* e *join*. Com uso deste limitado número de recursos o programador deve

±FAPERGS/PqG (11/1065-1), PRONEX/FAPERGS/CNPq (10/0042-8).
*Bolsista CAPES-DS.

descrever toda manifestação de concorrência no seu algoritmo. Embora seja possível descrever uma grande variedade de estruturas concorrentes com estas primitivas, o nível de abstração oferecido ao programador é bastante baixo, uma vez que construtores paralelos mais abstratos não estão disponíveis. Como consequência, o tempo de desenvolvimento de programas é longo, bem como sua robustez sujeita a exaustivos testes de verificação.

A ideia deste trabalho, ilustrada na Figura 1, é estender a interface de programação de Athreads por meio de recursos de programação paralela de mais alto nível. Esta figura mostra a arquitetura do ambiente Anahy em módulos e está destacado o módulo que este trabalho irá desenvolver chamado de Interface de Esqueletos.

Para implementar este novo conjunto de recursos de programação, encontramos na literatura o conceito de *Skeletons* (esqueletos) que de acordo com GONZÁLEZ; LEYTON (2010) é uma alternativa para especificação de construtores paralelos de alto nível. Os esqueletos são abstrações para construtores paralelos que encapsulam padrões de algoritmos recorrentes de programação paralela. Estes esqueletos implementam padrões simples que podem ser combinados para a construção de padrões mais complexos. Os esqueletos capturaram, organizam e mascararam do programador os detalhes envolvidos na estrutura da computação paralela.

A adição de esqueletos à Athreads facilitará a programação paralela pois eles serão utilizados na forma de simples funções, ou seja, o programador somente precisará passar os parâmetros de entrada que os esqueletos realizarão a execução paralela. Os esqueletos simplificarão os projetos de aplicações paralelas ao oferecerem soluções padrões de problemas recorrentes e poderão ser combinados para construir aplicações mais robustas. Além disto, os programas paralelos desenvolvidos utilizando estes esqueletos se beneficiarão dos mecanismos de escalonamento disponíveis em Athreads e terão seus níveis de robustez mais elevados.

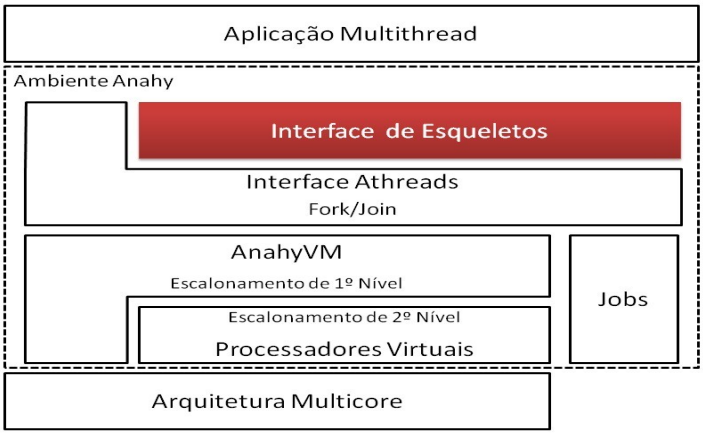

Figura 1 - Arquitetura do Ambiente Anahy em módulos.

2. METODOLOGIA

A primeira etapa do desenvolvimento deste trabalho foi a revisão bibliográfica dos esqueletos e uma análise de alguns *frameworks* que contenham implementações de esqueletos, como por exemplo: eSkel ESKEL (2013) e SkePU ENMYREN (2010). Nesta etapa foram analisadas questões como distribuição, sincronização e escalonamento das *threads* pelos esqueletos, além da forma como os dados são tratados e principalmente como os esqueletos podem ser combinados.

Na segunda etapa foram selecionados os esqueletos a serem implementados para compor a interface. Porém, antes disto foi feita análise de requisitos, ou seja, foram analisados os protótipos e os parâmetros necessários para implementações dos esqueletos. Além de verificar a necessidade da definição de novos tipos de dados, a forma que os dados de entrada do programador são tratados e como os dados de saída são retornados. Nesta etapa os estudos foram concentrados para que os parâmetros sejam simplificados de forma a facilitar a utilização dos padrões pelo programador.

Na implementação dos esqueletos está sendo utilizada a linguagem de programação C++, devido a última versão de Athreads ter sido implementada com esta linguagem. Assim, estes esqueletos utilizarão os mecanismos de escalonamento automáticos disponíveis em Athreads. Assim, está sendo possível desenvolver um código-fonte enxuto, de fácil compreensão e principalmente de fácil depuração.

A validação destas implementações utilizará aplicações sintéticas. Uma comparação qualitativa do código produzido, considerando, por exemplo, o número de linhas de código e o número de invocações aos serviços de Athreads, e uma comparação quantitativa do desempenho, em termos de tempos de execução para contabilizar a existência de sobre-custos diferentes, dos mesmos programas implementados com a atual interface e a interface proposta. Nesta etapa as implementações dos esqueletos serão analisados individualmente para obter resultados mais específicos. Além disso, alguns problemas com estas implementações poderão ser detectados.

3. RESULTADOS E DISCUSSÃO

Este trabalho se encontra na etapa de implementação, logo existem resultados parciais. O primeiro resultado deste trabalho é a lista de esqueletos que estão sendo implementados. Desta lista até este momento já foram implementados os seguintes esqueletos: Map, Reduce, Scan, MapReduce, Fork, If, Seq, While, For e Farm. Os que ainda faltam ser implementados são: *Branch and bound*, *Divide and Conquer* e *Pipeline*. Estes esqueletos estão sendo implementados de acordo as definições apresentado em GONZÁLEZ; LEYTON (2010). Estes esqueletos foram selecionados porque implementam padrões de programação tradicionais na computação paralela e que se enquadram em dois tipos diferentes de paralelismos, como: paralelismo de dados e de paralelismo de tarefas RAUBER; Rüger (2010). Foi então realizada uma modelagem dos esqueletos considerando sua implementação em uma linguagem de programação paralela com interface baseada em *multithreading*.

A Figura 2 destaca o modelo do esqueleto Map que possui quatro parâmetros: um vetor de entrada, uma função que deve ser aplicada a todos os elementos do vetor de entrada, e um vetor de saída, o qual conterá o resultado da aplicação da função sobre o vetor de entrada e o tamanho do vetor de entrada. Este esqueleto aplica a função sobre cada elemento do vetor paralelamente por meio de *threads e* retorna um vetor de dados do mesmo tamanho do de entrada.

O segundo resultado deste trabalho é o diagrama de classes que representa a estrutura do projeto da interface. O diagrama de classes esta sendo utilizado para facilitar o desenvolvimento da interface com a linguagem de programação orientada a objetos C++, porque permite adicionar ou remover classes da estrutura sem a necessidade de implementação.

Além destes resultados foi verificado, após as implementações iniciais dos esqueletos, que é viável desenvolver uma interface independente de tipos

específicos de dados. Isto é possível utilizando um mecanismo da própria linguagem C++. Logo, os tipos de dados com o qual cada esqueletos irão tratar serão definidos pelo programador. Mesmo com esses resultados o desenvolvimento desta interface tem encontrado dificuldades devido aos conceitos em tornos dos esqueletos serem originais das linguagens de programação funcionais que empregam conceitos bem diferentes da orientada objeto C++. Outra dificuldade presente neste trabalho é definir a forma de manipulação dos esqueletos para que sejam de fácil manuseio pelo programador.

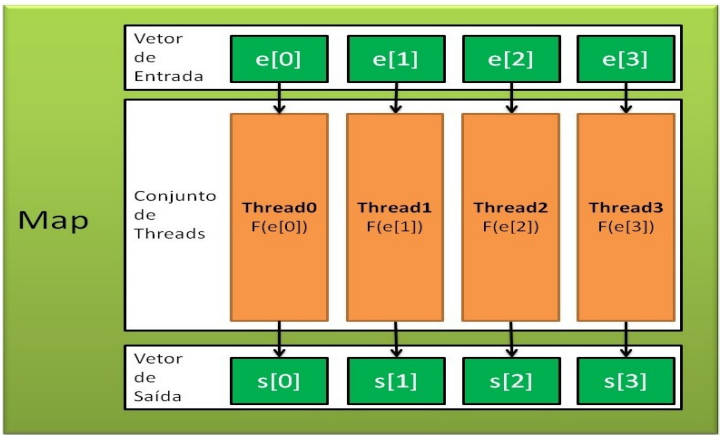

Figura 2 - Esqueleto Map paralelo.

4. CONCLUSÕES

Os esqueletos estão se mostrando instrumentos úteis para o desenvolvimento de uma interface com alto grau de abstração para o ambiente Anahy. Devido a este fato, a interface proposta fornecerá ao programador facilidades na programação paralela. Consequentemente o programador terá um aumento de produtividade e um aumento da robustez das aplicações que utilizam a ferramenta de programação paralela Athreads.

5. REFERÊNCIAS BIBLIOGRÁFICAS

CAVALHEIRO, G. G. H.; GASPARY, L. P.; CARDOZO, M. A.; CORDEIRO, O. C. Anahy: A Programming Environment for Cluster Computing. **VECPAR**. Anais. . . Heidelberg: Springer. LNCS 4395. p.198–211, 2007.

ENMYREN, J. **A Skeleton Programming Library for Multicore CPU and Multi-GPU Systems**. 2010. 103f. Tese (Doutorado em Ciência da Computação) - Curso de Pós-graduação em Ciência da Computação, Universidade Linköping.

ESKEL. **The Edinburgh Skeleton Library**. Acessado em 20 jan. 2013. Online. Disponível em: http://homepages.inf.ed.ac.uk/mic/eSkel/.

GONZALEZ, V., H.; LEYTON, M. A survey of algorithmic skeleton frameworks: high-level structured parallel programming enablers. **Softw: Pract. Exper**, [S.l.], v.40, n.12, p.1135–1160, 2010.

RAUBER, T.; RÜGER, G. **Parallel Programming: for Multicore and Cluster Systems.** Berlin: Springer, 2010. 2v.