UNIVERSIDADE REGIONAL INTEGRADA DO ALTO URUGUAI E DAS MISSÕES - CAMPUS DE ERECHIM
DEPARTAMENTO DE ENGENHARIAS E CIÊNCIA DA COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

MARCOS VINICIUS DE MOURA LIMA

PESQUISA E IMPLEMENTAÇÃO DE UMA APLICAÇÃO PARALELA UTILIZANDO O PARADIGMA DE GRAFOS DE DEPENDÊNCIAS DE TAREFAS

ERECHIM - RS
2018

MARCOS VINICIUS DE MOURA LIMA

PESQUISA E IMPLEMENTAÇÃO DE UMA APLICAÇÃO PARALELA UTILIZANDO O PARADIGMA DE GRAFOS DE DEPENDÊNCIAS DE TAREFAS

Projeto de Conclusão de Curso elaborado e apresentado na disciplina de Projeto de Conclusão de Curso, Curso de Curso de Ciência da Computação, Departamento de Engenharias e Ciência da Computação da Universidade Regional Integrada do Alto Uruguai e das Missões - Campus de Erechim.

Professor. Fábio A. Zanin

ERECHIM - RS
2018

RESUMO

O desenvolvimento de aplicações paralelas voltadas a arquiteturas heterogêneas pode vir a ser algo desafiador, pois nessas arquiteturas, os componentes são formados por processadores *multi-core* (CPUs) e placas gráficas (GPUs). O programador então, deverá distribuir as instruções que devem ser executadas em alguns destes componentes, para tentar alcançar o melhor desempenho da aplicação. Existem atualmente ambientes que se encarregam desta distribuição das instruções para os componentes da arquitetura, como por exemplo, o *StarPU*. Tendo em vista isto, este trabalho explorará o paradigma de Grafo de Dependência de Tarefa, aplicando-o a uma aplicação paralela de simulação de Decomposição Cartesiana sobre o ambiente de execução *StarPU*. Ao final deste trabalho, serão coletados dados sobre esta aplicação buscando avaliar seu desempenho neste ambiente, comparando com a versão sequencial da aplicação.

Palavras-chave: Aplicação Paralela. Arquiteturas Heterogêneas. Computação de Alto Desempenho.

ABSTRACT

The development of parallel applications for heterogeneous architectures can be somewhat challenging because, in these architectures, the components are made up of processors multi-core and accelerators such as the GPU. The developer should then distribute the instructions that must be performed on some of these components, to gain the best performance of the application. There are currently runtime environments that are responsible for distributing the instructions to the components of the architecture, for example, the StarPU. from this perspective this paper explores the Task Dependency Graph paradigm, applying it to a parallel application of Cartesian decomposition simulation on the StarPU environment. At the end of this work, data will be collected on this application, searching to evaluate their performance in this environment, comparing with the sequential version of the application.

Keywords: Parallel Application. Heterogeneous Architectures. High Performance Computing.

SUMÁRIO

1 INTRODUÇÃO

Arquiteturas Heterogêneas são compostas pelo uso de diferentes tipos de componentes em um mesmo nó computacional. Geralmente essas arquiteturas são compostas por CPUs (*Central Processing Unit*) e aceleradores, como por exemplo, a GPU (*Graphics Processing Unit*). Atualmente, muitos fabricantes de chips estão integrando os aceleradores junto com a CPU em um mesmo chip, um exemplo disso é o *Intel Graphics*.

Utilizar aceleradores como a GPU, junto com a CPU aumenta o desempenho e diminui o gasto energético, porém, o desenvolvimento de aplicações paralelas para arquiteturas heterogêneas é de certa forma visto como algo desafiador. Para o desenvolvedor conseguir aproveitar totalmente os componentes e recursos, o mesmo deverá identificar e distribuir as instruções mais adequadas em alguns destes componentes, para então tentar alcançar o melhor desempenho da aplicação. Atualmente existem ambientes que se encarregam desta distribuição das instruções para os componentes da arquitetura, como por exemplo, o *StarPU*.

Neste contexto, o trabalho proposto tentará avaliar o desempenho de uma aplicação quando executada em um destes ambientes. Para conseguir explorar os ambientes de execução em arquiteturas heterogêneas, será necessário desenvolver uma aplicação. Essa aplicação será a de uma simulação de transferência de calor em uma placa metálica bidimensional, utilizando o paradigma de Grafo de Dependência de Tarefas.

Nesse paradigma, o código da aplicação registra a criação de tarefas e a dependência de dados que existe entre elas. Esse registro de criação de tarefas é direcionado a um ambiente de execução, que se encarrega de distribuir as tarefas de maneira igualitária entre os recursos computacionais.

Em um primeiro momento será implementado uma versão sequencial da aplicação e posteriormente, será implementado uma versão paralela, para ser executada no ambiente *StarPU*. Ao final desse trabalho, será coletado dados sobre as duas implementações e realizado uma análise de desempenho, com o objetivo de avaliar se a aplicação utilizando o paradigma de grafo de dependência de tarefas, terá maior desempenho quando executada no ambiente *StarPU*.

O presente trabalho está estruturado da seguinte maneira: a seção 2 apresenta os objetivos gerais e específicos, a sessão 3 a justificativa do projeto de pesquisa e a seção 4 o referencial teórico. Em seguida a seção 5 apresenta a metodologia e a sessão 6 o cronograma de atividades. Por fim a seção 7 apresenta os resultados esperados.

2 OBJETIVOS

O presente trabalho tem como principal objetivo, analisar o desempenho de uma aplicação paralela utilizando o paradigma de Grafo de Dependência de Tarefas no ambiente de execução StarPU.

2.1 Objetivos Específicos

A seguir são listados os objetivos específicos para este projeto:

- Implementar a aplicação sequencial da simulação de transferência de calor;
- Configurar o ambiente StarPU;
- Implementar a aplicação paralela da simulação de transferência de calor com o paradigma de grafo de dependências de tarefas;
- Verificar qual é melhor técnica de análise de desempenho sobre os dados das execuções das aplicações.

3 JUSTIFICATIVA

A maioria dos sistemas de processamento de alto desempenho tem atualmente a sua arquitetura heterogênea, composta por CPUs (*Central Processing Unit*) *multicore* e aceleradores como a GPU (*Graphics Processing Unit*). O desenvolvimento de aplicações para essas arquiteturas é feito através de APIs (*Application Programming Interface*), ou bibliotecas. Essas APIs são específicas para cada tipo de componente de uma arquitetura heterogênea. Por exemplo, nas aplicações que exploram os recursos das CPUs *multicore* pode se usar a API OpenMP (*Open Multi-Processing*) (OPENMP, 2018) ou a biblioteca Intel TBB (INTEL, 2018) (*Threading Building Blocks*). Já nas aplicações que visam utilizar a GPU pode se usar OpenCL(*Open Computing Language*) (NVIDIA, 2018b) ou CUDA (*Compute Unified Device Architecture*) (NVIDIA, 2018a).

De acordo com (STRINGHINI; GONÇALVES; GOLDMAN, 2012), caso o desenvolvedor queira extrair o melhor desempenho possível e utilizar ambos recursos de uma arquitetura heterogênea, é necessário que o mesmo saiba identificar quais são as tarefas mais adequadas para a execução em um acelerador e aquelas mais adequadas à execução em uma CPU *multicore*.

Outra questão que é observada é sobre a portabilidade de um código para uma arquitetura heterogênea. Segundo (PHOTHILIMTHANA et al., 2013), um dos grandes desafios para utilizar os recursos heterogêneos é a diversidade de dispositivos em diferentes máquinas. Pois um programa que tenha sido otimizado para um determinado acelerador, pode não funcionar tão bem na próxima geração de processadores ou em um dispositivo de um outro fornecedor. No caso da GPU a programação é feita em um nível próximo ao do *hardware*, fazendo com que muitas vezes o código fique restrito a um modelo e ou fabricante (PINTO, 2011).

Vale salientar que o desempenho das CPUs e GPUs também variam entre as máquinas. Uma execução específica pode ter um melhor desempenho na CPU, enquanto que em outra máquina o desempenho pode ser melhor na GPU. Existe também alguns problemas no escalonamento das tarefas de uma aplicação para os recursos da máquina, que podem ser resolvidas com balanceamento de carga. Como por exemplo, redistribuir a carga para uma CPU ociosa, quando a GPU estiver sobrecarregada. (PHOTHILIMTHANA et al., 2013).

Atualmente existem ambientes de execução que dão suporte a CPUs e GPU simultaneamente, como por exemplo, o StarPU (AUGONNET et al., 2011). Esses ambientes buscam distribuir as tarefas de maneira igualitária entre os recursos computacionais CPUs e GPUs. De acordo com (KUMAR, 2017), a utilização destes ambientes libera o desenvolvedor da necessidade de adaptar a aplicação especificamente à máquina destino e unidades de processamento.

4 REFERÊNCIAL TEÓRICO

4.1 Computação de Alto Desempenho

4.1.1 Programação Paralela

4.1.2 Classificação e Terminologia

4.2 Arquiteturas Heterogêneas

4.2.1 Processadores Multicore

4.2.2 Aceleradores

4.3 Programação Paralela em Arquiteturas Heterogêneas

4.3.1 CUDA

4.3.2 OpenCL

4.3.3 Intel tbb

4.3.4 OpenACC

4.4 Ambientes de Execução para Arquiteturas Heterogêneas

4.4.1 StarPU

4.4.2 Xkaapi

4.4.3 StarSs

4.5 Paradigma de Grafo de Dependência de Tarefas

5 METODOLOGIA

Para conseguir explorar os ambientes de execução em arquiteturas heterogêneas será necessário desenvolver uma aplicação. Essa aplicação que será desenvolvida será a de uma simulação de transferência de calor em uma placa metálica bidimensional, utilizando a decomposição cartesiana.

Inicialmente, será feita uma pesquisa sobre arquiteturas heterogêneas e como funcionam os ambientes de execução para estas arquiteturas, mais especificamente o ambiente StarPU. Também serão estudados alguns trabalhos relacionados, que utilizam este ambiente e o paradigma de grafo de dependências de tarefas.

Posterior a isso, será realizado a implementação sequencial da simulação. Este passo é interessante para entender como funciona a transferência de calor, utilizando a decomposição cartesiana e as dependências de dados que existem na aplicação.

Apos a implementação sequencial, será necessário estudar e configurar o ambiente StarPU. Para a configuração do mesmo, será estudado a documentação do ambiente e implementado alguns scripts do próprio tutorial. O tutorial do StarPU é importante tanto para entender como funciona o ambiente de execução, como para configurar de forma correta o ambiente.

Em seguida, será realizado a implementação paralela da simulação de transferência de calor utilizando o paradigma de grafos de dependências de tarefas. Apos isso, a aplicação será executada no ambiente StarPU.

Por fim, será verificado qual é a melhor técnica de análise de desempenho para os dados obtidos nas execuções, para então, ser concluído se aplicação paralela teve melhor desempenho, ou não, quando comparada com a aplicação sequencial.

6 CRONOGRAMA

O Quadro 1 apresenta o cronograma de atividades para este trabalho proposto.

Quadro 1 – Cronograma de Atividades.

Atividades	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
1. Revisão bibliográfica									
2. Redação da Monografia									
3. Estudo sobre as ferramentas									
4. Trabalhos relacionados									
5. Implementação sequencial da aplicação									
6. Configuração do ambiente StarPU									
7. Implementação paralela da aplicação									
8. Execução no ambiente StarPU									
8. Testes e coleta de dados									
9. Procurar a melhor técnica para a análise de desempenho									
10. Revisão final e entrega da monografia									
11. Elaboração da apresentação									
12. Defesa da monografia									
13. Entrega da versão final									

7 RESULTADOS ESPERADOS

Ao término deste trabalho espera-se saber se a aplicação paralela implementada com o paradigma de grafo de dependências de tarefas e executada no ambiente StarPU, terá melhor desempenho quando comparada com a implementação sequencial da aplicação.

REFERÊNCIAS

AUGONNET, C. et al. Starpu: a unified platform for task scheduling on heterogeneous multicore architectures. **Concurrency and Computation: Practice and Experience**, Wiley Online Library, v. 23, n. 2, p. 187–198, 2011. Citado na página 3.

INTEL. **Threading Building Blocks**. 2018. Disponível em: <https://www.threadingbuildingblocks.org/>. Acesso em: 18 de maio de 2018. Citado na página 3.

KUMAR, S. **Scheduling of Dense Linear Algebra Kernels on Heterogeneous Resources**. Tese (Doutorado) — Université de Bordeaux, abr. 2017. Disponível em: <https://tel.archives-ouvertes.fr/tel-01538516>. Citado na página 3.

NVIDIA. **Nvidia Accelerated Computing CUDA**. 2018. Disponível em: <https://developer.nvidia.com/cuda-zone>. Acesso em: 18 de maio de 2018. Citado na página 3.

NVIDIA. **Nvidia Accelerated Computing OpenCL**. 2018. Disponível em: <https://developer.nvidia.com/opencl>. Acesso em: 18 de maio de 2018. Citado na página 3.

OPENMP. **OpenMP The OpenMP API specification for parallel programming**. 2018. Disponível em: <https://www.openmp.org/>. Acesso em: 18 de maio de 2018. Citado na página 3.

PHOTHILIMTHANA, P. M. et al. Portable performance on heterogeneous architectures. In: ACM. **ACM SIGARCH Computer Architecture News**. [S.l.], 2013. v. 41, n. 1, p. 431–444. Citado na página 3.

PINTO, V. G. Ambientes de programação paralela híbrida. **Porto Alegre**, 2011. Citado na página 3.

STRINGHINI, D.; GONÇALVES, R. A.; GOLDMAN, A. Introdução à computação heterogênea. **XXXI Jornada de atualizaça em Informática (JAI)**, 2012. Citado na página 3.