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, será coletado dados sobre esta aplicação para então tirar as conclusões sobre o seu desempenho, comparando com a mesma aplicação desenvolvida de forma sequencial.

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 executions, to write the conclusions about their performance, compared with the same application developed sequentially.

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

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
DECOM	Departamento de Computação

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 a CPU junto com aceleradores em um mesmo chip, como por exemplo o *Intel Graphics*.

Por mais que esse tipo de arquitetura aumente o desempenho e diminua o gasto energético, 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 explorará a programação de aplicações paralelas baseadas em tarefas, 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.

Como aplicação, será desenvolvido uma simulação de transferência de calor (decomposição cartesiana) em uma placa metálica bidimensional de forma sequencial. Posteriormente, será implementado uma versão paralela da aplicação, para ser executada no ambiente *StarPU*. Ao final desse trabalho, será coletado dados sobre as duas implementações e realizado uma análise de desempenho, para então, tirar as conclusões sobre o trabalho.

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 as conclusões.

2 OBJETIVOS

O presente trabalho tem como principal objetivo a implementação de uma aplicação paralela com o paradigma de Grafo de Dependência de Tarefas.

2.1 Objetivos Específicos

A seguir são listados os objetivos específicos para este projeto:

- Estudo sobre Arquiteturas Heterogêneas e Computação de Alto Desempenho;
- Implementação de uma simulação de transferência de calor (decomposição cartesiana) de um placa metálica bidimensional de forma sequencial e paralela;
- Configurar um ambiente para execução do StarPU;
- Execução da aplicação paralela no ambiente StarPU;
- Coletar dados e analisar o desempenho de ambas implementações.

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 visão 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 funcionar melhor na CPU, enquando que em outra máquina pode funcionar melhor da GPU. Existe também alguns problemas no escalonamento, se uma aplicação tem desempenho melhor na GPU, mas esta mesma está sobrecarregada e a CPU estiver ociosa, pode ser necessário balancear a carga de trabalho entre os dois recursos (PHOTHILIMTHANA et al., 2013).

Atualmente existem ambientes de execução que dão suporte a CPUs e GPU simultaneamente. Esses ambientes se encarregam de distribuir as tarefas de maneira igualitária entre os recursos computacionais CPUs e GPUs. Um fator importante é que esses ambientes implementam escalonadores que são responsáveis pelo desempenho final da aplicação, libertando o desenvolvedor da necessidade de adaptar a aplicação especificamente à máquina destino e unidades de processamento (KUMAR, 2017).

É importante também dizer que a utilização destes ambientes mantêm o código

portável entre diferentes modelos e ou fabricantes de hardware (DOLBEAU; BIHAN; BODIN, 2007 apud PINTO, 2011)

4 REFERÊNCIAL TEÓRICO

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.

Primeiramente será necessário a implementação de forma sequencial da simulação. Esse passo é interessante para entender como o *kernel* implementa a transferência de calor e as suas dependências com os vizinhos da decomposição cartesiana, assim também como ajustar os *timestemps* que a simulação terá.

Apos a implementação sequencial será necessário a configuração do ambiente StarPU. Para a configuração do mesmo, será estudado a documentação do ambiente e implementado alguns scripts do próprio tutorial do StarPU para validar a configuração.

Apos a configuração e estudo do ambiente, será buscado trabalhos relacionados para entender como funciona o paradigma de grafo de dependências de tarefas. Ao final dessa pesquisa, será necessário implementar a versão paralela da simulação para ser executada no ambiente StarPU.

Por fim, será realizado testes de desempenho sobre as execuções das duas implementações e tirado as conclusões do trabalho.

6 CRONOGRAMA

O quadro (**??**) 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	X	X							
2. Entrega versão preliminar do projeto			13						
3. Entrega versão final do projeto			27						
4. Defesa pública dos projetos				9 - 13					
5. Revisão dos apontamentos da banca				X					
6. Entrega projeto com ajustes da banca				18					
7. Implementação sequencial da simulação				X					
8. Redação da Monografia de TCC			X	X	X	X	X	X	
9. Estudo sobre arquiteturas heterogêneas			X	X					
10. Configuração e estudo StarPU			X	X	X				
11. Trabalhos relacionados						X			
12. Implementação paralela da simulação						X	X		
13. Execução no ambiente StarPU						X	X		
14. Testes e coleta de dados							X	X	
15. Revisão final e entrega da monografia									X
16. Elaboração da apresentação final									X
16. Defesa da monografia									X
17. Entrega da versão final									X

REFERÊNCIAS

DOLBEAU, R.; BIHAN, S.; BODIN, F. Hmpp: A hybrid multi-core parallel programming environment. In: **Workshop on General Purpose Processing on Graphics Processing Units (GPGPU 2007)**. [S.l.: s.n.], 2007. v. 28. Citado na página 4.

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 2 vezes nas páginas 3 e 4.

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