

**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**

Elemento obrigatório em tese, dissertação, monografia e TCC. É a versão do resumo em português para o idioma de divulgação internacional. Deve ser antecedido pela referência do estudo. Deve aparecer em folha distinta do resumo em língua portuguesa e seguido das palavras representativas do conteúdo do estudo, isto é, das palavras-chave. Sugere-se a elaboração do resumo (Abstract) e das palavras-chave (Keywords) em inglês; para resumos em outras línguas, que não o inglês, consultar o departamento / curso de origem.

**Keywords:** Word. Second Word. Another word.

## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO . . . . .</b>	<b>1</b>
<b>2</b>	<b>OBJETIVOS . . . . .</b>	<b>2</b>
<b>2.1</b>	<b>Objetivos Específicos . . . . .</b>	<b>2</b>
<b>3</b>	<b>JUSTIFICATIVA . . . . .</b>	<b>3</b>
	<b>REFERÊNCIAS . . . . .</b>	<b>5</b>

## 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, enquanto 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)

## 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 atualização em Informática (JAI)**, 2012. Citado na página 3.