Predição de Desempenho de Aplicações CUDA utilizando Aprendizado de Máquina e Características de Pré-Execução

Luan Siqueira, Marcos Amaris

[1]Universidade Federal do Pará
Faculdade de Engenharia de Computação, Tucuruí - Pará

luansiqueira18@gmail.com, amaris@ufpa.br

Resumo. *Com a evolução das unidades de processamento gráfico (GPU), as aplicações de computação paralela estão se tornando cada vez mais complexas. Prever o desempenho dessas aplicações ajuda desenvolvedores a otimizar seus algoritmos e a escalonadores na distribuição de seus trabalhos. Neste trabalho, foi realizado uma predição do desempenho de aplicações CUDA usando aprendizado de máquina e características de pré-execução. Foram usados dados de execuções de 9 aplicações CUDA sobre 8 GPUs. Utilizou-se dois técnicas de aprendizado de máquina RandomForest e Decision Tree, mostrou-se que os modelos criados apresentaram uma média ente 1,41% e 2,14% de erro de predição no tempo de execução, respectivamente.*
Palavras–chave: *Máquina de Aprendizado, Aplicações CUDA, Computação Paralela.*

Abstract. *With the evolution of graphics processing units (GPUs), parallel applications are becoming increasingly complexes. to predict the performance of parallel applications can support developers to optimize their algorithms and schedulers to distribute their jobs. In this work, a performance prediction of CUDA application using machine learning and pre-execution features was performed. Executions of 9 CUDA-based applications over 8 different GPUs were used. Two machine learning techniques were used, they were Random Forest and Decision Tree, they obtained an average MAPE of 1.41% and 2.14%, respectivelly.*
Keywords: *Machine Learning, CUDA Applications, Parallel Computing.*

1. INTRODUÇÃO

Com o advento da computação de alto desempenho, o poder de processamento das unidades de processamento gráfico (GPU) teve um aumento considerável nos últimos anos, o que resultou em uma maior complexidade de modelagem. Prever com precisão o desempenho de aplicações paralelas [Amaris et al. 2016] é um grande desafio. Para usufruir o poder computacional das GPUs, a Nvidia introduziu em 2006 a CUDA *(Compute Unified Device Architecture)*, suportando uma sintaxes semelhante a C, fornecendo um compilador que converte em um código pseudo-assembly PTX (*Parallel Thread Execution*) [NVIDIA and CUDA 2015].

O Tempo de execução das aplicações implementadas em infraestruturas de alto desempenho é fundamental com a finalidade de escalonar os processos adequadamente. Esse tempo de execução é requerido ao usuário de plataformas paralelas para demarcar

um *walltime*, um tempo máximo de execução de seus trabalhos. Dessa forma, este trabalho trata-se de uma predição de desempenho realizado em aplicações CUDA utilizando técnicas de aprendizado de máquina com características de pré-execução das aplicações. As informações que serão usadas pelas técnicas de aprendizagem de máquina serão coletadas a priori da execução das aplicações.

O restante do artigo está organizado como segue. Os fundamentos são discutidos na Seção 2. A Seção 3 descreve os trabalhos relacionados. A metodologia é apresentada na Seção 4. Na Seção 5 se apresentam os resultados. Por fim, as conclusões e trabalhos futuros são apresentados no Seção 6.

2. FUNDAMENTOS

2.1. Predição de Desempenho

Prever com precisão o desempenho de uma aplicação é uma tarefa complexa pois, as aplicações são frequentemente incompletas e não determinísticas. Dentre os algoritmos preditivos inseridos na área da inteligência artificial, a aprendizagem de máquinas apresentam potencial de identificar relações complexas e não lineares presentes nos dados com consequências positivas no desempenho preditivo desses modelos, dentre eles, existem algumas técnicas de aprendizado de máquina simples: Regressão Linear, *Support Vector Machine*, *Random Forest*, entre outros [Susto et al. 2014].

2.2. Características Pré-execução CUDA

Um passo essencial para o processo de predição de desempenho é a seleção dos dados. Em [NVIDIA and CUDA 2015] é disponibilizado especificações de hardware de diversas arquitetura GPU e através de suas configurações, o *Nvidia Cuda Compiler* (NVCC) e *Cuda Calculator Occupancy* apresenta características inerente ao código-fonte, sem a necessidade de execução do algoritmo desenvolvido.

3. TRABALHOS RELACIONADOS

Pouco tempo atrás, surgiram vários estudos sobre o desempenho de aplicações CUDA usando diferentes estratégias de estatísticas e de aprendizado de máquina. O trabalho de [Alavani et al. 2018] realizou uma análise estática para previsão de tempo de duração para o *kernel* CUDA, eles conseguiram um erro médio absoluto de 26,86% em comparação ao tempo de execução real. [Hayashi et al. 2015] desenvolveu um modelo preditivo para estimar o tempo de execução de aplicativos paralelos se baseando em um modelo de predição binária com máquinas de vetores de suporte (SVM).

Neste artigo, implementamos um modelo de aprendizado de máquina afim de predizer o tempo de execução do *kernel* em GPUs NVIDIA. Embora alguns trabalhos utilizam métodos de aprendizado de máquina e análise estática, muitos extraem os dados de informação através de *profiling* no momento da execução da aplicação e, neste trabalho, a extração de dados é realizada no momento da compilação.

4. METODOLOGIA

Para a realização dos testes, foram utilizados os seguintes algoritmos:

- **Multiplicação de matriz:** foram utilizados quatro algoritmos, sendo eles: memória global com acessos não coalescidos (MMGU); memória compartilhada com acessos não coalescidos a memória global (MMSU); memória global com acessos coalescidos (MMGC) e e memória compartilhada com acessos coalescidos a memória global (MMSC).
- **Soma de matriz:** utilizou-se duas aplicações diferentes: memória global com acessos não coalescidos (MAU); memória global com acessos coalescidos (MAC).
- **Adição de vetor:** (vAdd) em aplicações de *GPU*, o algoritmo de adição de vetor consiste em realizar em cada *thread* uma adição de uma posição de vetores A + B armazenando o resultado em C.
- **Produto escalar:** (dotP) o algoritmo de produto escalar executa em cada *thread* a multiplicação de um posição de vetores A x B e armazena a variável compartilhada do resultado.
- **Problema de submatriz máxima:** (MSA) O problema de submatriz máxima consiste em encontrar o contíguo submatriz dentro de uma sequência de N números inteiros (x1,x2...,xn) que tem a maior soma de elementos.

Na primeira fase, foi extraído informações utilizando o *Nvidia CUDA Compiler* (NVCC), aplicando diferentes versões de arquitetura de GPUs e tamanho de *threads* por blocos, 8^2, 16^2, 32^2 para aplicações bidimensionais e unidimensional. Para o caso da aplicação MSA, foi utilizado 128 *threads* por bloco. Com o *NVCC*, foi extraído a quantidade de registradores, memória constante e memória compartilhada de cada algoritmo testado. Também foi extraído o nível de ocupação de uma *GPU* para cada *kernel CUDA* através da *CUDA Occupancy Calculator* [NVIDIA and CUDA 2015].

Com a ideia de adicionar mais características da arquitetura das GPUs CUDA, realizou-se a mesclagem dos dados extraídos na primeira fase com os dados da pesquisa [Amaris et al. 2016], totalizando 8459 amostras no conjunto de dados. A linguagem de programação Python foi utilizada para a realizar a extração das informações e junto com a biblioteca sklearn foi criado o modelo de máquina de aprendizado e, como métrica de avaliação, foi definido o erro de percentual médio absoluto (MAPE em inglês).

5. RESULTADOS

Para o desenvolvimento de cada modelo, foi utilizado os dados das oito aplicações para o conjunto de treino e uma aplicação para o conjunto de teste. Isso foi realizado para cada uma das aplicações. Para o conjunto de treino e teste, foi aplicado a transformação logarítmica das variáveis para se obter uma distribuição mais simétrica dos dados. Os parâmetros configurados para os dois modelos consistiu em uma profundidade máxima da árvore em 10 ramificações e a divisão aleatória de maneira fixa.

A Figura 1 apresenta a métrica MAPE para os modelos de máquina de aprendizado criados. Pode-se notar que os modelos *Random Forest* e *Decision Tree* apresentaram resultados semelhantes. É possível observar que para todas as aplicações, os dois modelos obteve resultados satisfatórios, tendo apenas uma pequena alta na taxa de erros nas 4 aplicações de matrizes de multiplicação.

6. CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho avaliamos a predição do desempenho de 9 aplicações *CUDA* usando técnicas de aprendizado de máquina, especificamente os modelos de *Random Forest* e *Decision*

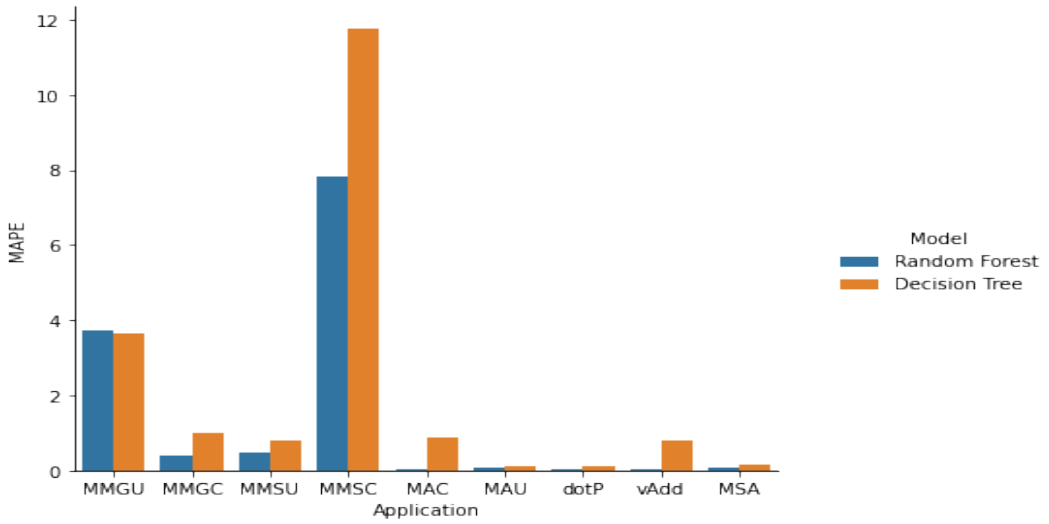

Figura 1. MAPE (%) do tempo de execução predito

Tree. levantou-se um conjunto de dados contendo características da arquitetura das GPU's Nvidia e das aplicações antes de ser executadas.

Os resultados se mostraram promissores, obtendo resultados muitos similares em comparação um com outro, no entanto, na maioria dos casos, o modelo *Random Forest* obteve resultados significativamente melhor, apresentando uma média de 1,41% de erro em suas predições. Como trabalhos futuros, pretende-se explorar mais modelos, como a regressão linear, para verificar o comportamento perante aos conjuntos de dados. Além disso, é de grande valia extrair mais recursos do código-fonte da aplicação e, para isso, consideramos utilizar o compilador LLVM e posteriormente selecionar os recursos mais relevantes e testá-los com os modelos de regressão estudados neste trabalho.

Referências

[Alavani et al. 2018] Alavani, G., Varma, K., and Sarkar, S. (2018). Predicting execution time of cuda kernel using static analysis. In *(ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, pages 948–955. IEEE.

[Amaris et al. 2016] Amaris, M., de Camargo, R. Y., Dyab, M., Goldman, A., and Trystram, D. (2016). A comparison of gpu execution time prediction using machine learning and analytical modeling. In *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*, pages 326–333. IEEE.

[Hayashi et al. 2015] Hayashi, A., Ishizaki, K., Koblents, G., and Sarkar, V. (2015). Machine-learning-based performance heuristics for runtime cpu/gpu selection. In *Proceedings of the principles and practices of programming on the Java platform*.

[NVIDIA and CUDA 2015] NVIDIA, C. C. and CUDA, C. (2015). Programming guide, version 7.

[Susto et al. 2014] Susto, G. A., Schirru, A., Pampuri, S., McLoone, S., and Beghi, A. (2014). Machine learning for predictive maintenance: A multiple classifier approach. *IEEE Transactions on Industrial Informatics*, 11(3):812–820.