Proposta de Machine Learning aplicado na otimização de Kernel para GPUs *

Bruno Machado Morales[1], **Cristiano Künas** [1], **Thiago Araujo** [1], **João Vitor Vargas Oliveira**[1], **Philippe O. A. Navaux**[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

{bmmorales, cakunas, tsaraujo, jvvoliveira, navaux}@inf.ufrgs.br

Resumo. *A Inteligência Artificial (IA) está avançando e solucionando problemas cada vez mais desafiadores e transformando diversas áreas, como classificação de imagens, reconhecimento de fala e processamento de linguagem natural. No campo do Deep Learning (DL), as redes neurais profundas tem se destacado por sua capacidade de lidar com grandes volumes de dados e aprender representações complexas. A otimização do kernel desempenha um papel crucial, pois envolve aprimorar o desempenho das operações de baixo nível realizadas diretamente na unidade de processamento gráfico. A presente pesquisa foca na otimização do kernel como um passo fundamental para lidar com os desafios cada vez maiores impostos pelas aplicações modernas de IA.*

1. Introdução

Com o avanço contínuo da tecnologia, a inteligência artificial tem emergido como uma das áreas mais transformadoras do século. Por meio de uma ampla diversidade de técnicas e abordagens, ela busca replicar ou até mesmo superar a capacidade humana de raciocinar, aprender e tomar decisões. A otimização de modelos de *machine learning* pode ocorrer em várias etapas, desde o pré-processamento dos dados até a melhoria do desempenho em execução, utilizar *machine learning* para otimizar o *kernel* faz com que sua eficiência seja aprimorada. O processo de otimização envolve a utilização de técnicas avançadas para ajustar e refinar a execução de código, explorando diferentes estratégias e abordagens para maximizar o uso eficiente dos recursos da *Graphics Processing Unit (GPU)* [Jam et al. 2025].

2. Proposta

A ferramenta Atlas [Whaley 2011] é projetada para realizar operações de álgebra linear, como multiplicação de matrizes, e também realiza ajustes antes da execução do programa *offline*. Isso reduz o impacto no desempenho em tempo de execução, já que as otimizações são pré-carregadas. No entanto, ajustes *offline* podem ser limitados devido à falta de informações disponíveis apenas em tempo real.

Os ajustes online são realizados durante a execução, e utilizam informações em tempo real para otimizar o desempenho. Em [Tillet and Cox 2017], a técnica de hill-climbing é empregada durante a execução com base em um modelo que antecipa resultados. Já em [Gadioli et al. 2018], sistemas realizam o acompanhamento contínuo de

*O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001, pela Petrobras sob número 2020/00182-5 e pelo edital CNPq/MCTI/FNDCT - Universal 18/2021 sob número 406182/2021-3.

kernels, modificando-os para cumprir exigências de performance predefinidas. Um ponto crítico nessa estratégia reside na necessidade de garantir que os ganhos obtidos no desempenho superem o esforço computacional exigido pelo processo de otimização.

O objetivo é realizar uma análise detalhada, utilizando *machine learning* para identificar automaticamente os parâmetros que mais impactam o desempenho e o consumo de energia, o uso de *machine learning* permitirá otimizar configurações como tamanho e distribuição de blocos e *threads*, uso de memória compartilhada, alocação de registradores, visando encontrar as melhores combinações para otimizar a execução.

3. Metodologia

Serão realizados diferentes testes de execução dos *kernels*, variando parâmetros como o tamanho e a distribuição de blocos e *threads*, o uso de memória compartilhada, alocação de registradores e o balanceamento da ocupação dos multiprocessadores. Para avaliar o desempenho, será utilizado o benchmark *HPL (High-Performance Linpack)*, para medir a capacidade computacional de sistemas paralelos. Os testes serão conduzidos em múltiplas *GPUs*, abrangendo tanto arquiteturas NVIDIA quanto AMD, a fim de analisar o comportamento dos *kernels* em diferentes plataformas de hardware. Para cada configuração, serão coletadas métricas de desempenho e consumo energético, permitindo a construção de um conjunto de dados sobre o impacto de cada parâmetro na execução dos *kernels*.

Com base nos dados coletados, um modelo de aprendizado de máquina será treinado para prever o desempenho e o consumo energético das diferentes combinações de parâmetros e identificar automaticamente aquelas que resultam em maior eficiência. A partir das predições geradas, serão selecionadas as melhores configurações para cada *kernel*, visando a otimização do tempo de processamento e a redução do consumo de energia.

Por fim, serão implementadas as otimizações indicadas pelo modelo. Isso incluirá ajustes específicos nos *kernels*, como a reorganização da distribuição de *threads* e blocos, a otimização do uso da memória compartilhada para minimizar acessos à memória global, a redefinição da alocação de registradores para evitar sobrecarga na memória local e o balanceamento da ocupação dos multiprocessadores para reduzir gargalos de execução.

Referências

Gadioli, D., Nobre, R., Pinto, P., Vitali, E., Ashouri, A. H., Palermo, G., Cardoso, J., and Silvano, C. (2018). Socrates—a seamless online compiler and system runtime autotuning framework for energy-aware applications. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1143–1146. IEEE.

Jam, M., Petit, E., Castro, P. d. O., Defour, D., Henry, G., and Jalby, W. (2025). Mlkaps: Machine learning and adaptive sampling for hpc kernel auto-tuning. *arXiv preprint arXiv:2501.05811*.

Tillet, P. and Cox, D. (2017). Input-aware auto-tuning of compute-bound hpc kernels. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, pages 1–12.

Whaley, R. C. (2011). Atlas (automatically tuned linear algebra software). *http://www. netlib. org/atlas/index. html*.