Análise de Desempenho e de Requisitos Computacionais Utilizando o Modelo Roofline: Um estudo para Aplicações de Inteligência Artificial e do NAS-HPC

Vitor Sá, Vinícius Klôh, Bruno Schulze, Mariza Ferro

[1]Laboratório Nacional de Computação Científica – LNCC
Av. Getúlio Vargas, 333 – 25651-075 – Quitandinha, Petrópolis – RJ –Brasil

{vitorsa, viniciusk, schulze, mariza}@lncc.br

Abstract. *In this work, we propose the use of the Roofline model for an understanding of the computational requirements of the applications. NAS-HPC benchmarks and Artificial Intelligence algorithms were executed, analyzed, and classified according to their computational bottlenecks. The results were very positive for the understanding and classification of the different computational requirements of the applications, highlighting the different patterns among them.*

Resumo. *Neste trabalho é proposto a utilização do modelo Roofline para a compreensão dos requisitos computacionais de aplicações científicas. Foram analisados e caracterizados, de acordo com seus gargalos computacionais, aplicações do NAS-HPC e de Inteligência Artificial. Os resultados foram muito positivos para o entendimento e a caracterização dos diferentes requisitos computacionais das aplicações, destacando os diferentes padrões entre eles.*

1. Introdução

Para aumentar a capacidade computacional de forma a atender as necessidades das aplicações científicas, os sistemas de Computação de Alto Desempenho (*HPC*) recebem atualizações contínuas de *hardware* e *software*. Somado às constantes evoluções dos sistemas *HPC* está o uso de arquiteturas heterogêneas, pois aplicações diferentes têm necessidades diferentes em relação aos recursos computacionais. Assim, essas arquiteturas vêm possibilitando o desenvolvimento dos sistemas de forma a atenderem cada vez mais os requisitos computacionais das diferentes aplicações científicas. Entretanto, compreender e categorizar as diferentes necessidades das aplicações em relação aos recursos computacionais não é uma tarefa trivial.

Nessa tarefa, a avaliação de desempenho tem se mostrado útil na busca por maximizar o desempenho, na redução do custo, na aquisição e manutenção de sistemas computacionais, e principalmente, para obtenção de conhecimento a cerca dos requisitos computacionais das aplicações científicas. Uma maneira tradicional de se realizar as avaliações de desempenho são por meio dos *benchmarks*, um conjunto de aplicações que provêm um método de comparação do desempenho de vários subsistemas entre diferentes arquiteturas. Entretanto, em geral, eles não apontam onde estão os gargalos da aplicação e nem quais as limitações de *hardware*. Por meio dos limites de *hardware* pode-se avaliar o desempenho da aplicação perante o *hardware*, mapear áreas do sistema que são intensivas em memória, em processamento ou equilibradas entre ambas. Além disso, sugerir informações importantes sobre o quão otimizada a aplicação se encontra em relação

ao *hardware*. Por meio dos gargalos da aplicação pode-se sugerir possíveis otimizações para a aplicação, quais podem ser realizadas, e categorizar de maneira distinta os *loops* e funções da aplicação que são intensivos em memória, em processamento ou se possuem um equilíbrio entre memória e processamento. Portanto, mesclar o conceito de limites de *hardware* e gargalos da aplicação é útil para entender o desempenho de uma determinada aplicação em um determinado sistema, sem precisar instrumentar a aplicação, tornando um conceito flexível e independente.

Com a proposta de relacionar esses conceitos em um gráfico bidimensional, [Williams et al. 2009] apresenta o *Roofline*, que fornece informações sobre o sistema e o desempenho da aplicação, sendo valioso para auxiliar no desenvolvimento e otimização de aplicações e arquiteturas. O seu diferencial é integrar informações do sistema e da aplicação em um único ambiente, diferindo dos *benchmarks* tradicionais, que analisam separadamente o desempenho da aplicação e a capacidade computacional do *hardware*.

Neste trabalho é proposto o uso do modelo *Roofline* para a avaliação de desempenho e a obtenção de conhecimento sobre os requisitos computacionais de diversos algoritmos de duas áreas de aplicação. Foram executados seis *benchmarks* voltados para *HPC* (*NAS Parallel Benchmark* [1]) e quatro aplicações de Inteligência Artificial (IA). Os requisitos computacionais desses dois conjuntos de aplicações são analisados e categorizados de acordo com seus limites computacionais. Os resultados foram muito positivos para a compreensão e classificação dos diferentes requisitos computacionais entre as aplicações, demonstrado os diferentes padrões entre eles.

Este trabalho está organizado da seguinte forma: na Seção 2 são apresentados conceitos básicos sobre o modelo *Roofline* e trabalhos relacionados; na Seção 3 são apresentadas as arquiteturas, aplicações e a ferramenta utilizada para aplicar o modelo *Roofline*; na Seção 4 são apresentados e analisados os resultados obtidos durante esse estudo. Finalmente na Seção 5 são apresentadas as considerações finais e trabalhos futuros.

2. Modelo *Roofline*

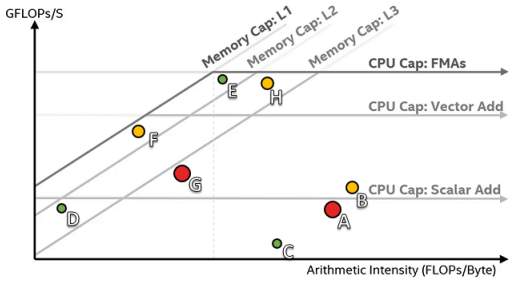


Figura 1. Gráfico Modelo *Roofline* (*Advisor*)
Fonte: https://software.intel.com/content/www/us/en/develop/tools/advisor/roofline.html

O *Roofline* traz um modelo visual, que relaciona o desempenho do processador ao tráfego de memória, apontando os gargalos da aplicação e possíveis formas de otimização. Reunir todas essas informações oferecem *insights* sobre o funcionamento do *hardware* e do *software*, vinculando a intensidade aritmética (operações por byte de tráfego), a largura de banda e o desempenho de ponto flutuante (operações por segundo) juntos em um

[1] `https://www.nas.nasa.gov/publications/npb.html`

gráfico bidimensional. Como pode ser visto na Figura 1, o eixo X representa a intensidade aritmética (IntA), o eixo Y o desempenho de ponto flutuante (DPF) e as linhas diagonais representam a largura de banda (LB) das respectivas memórias. Cada teto (reta) do gráfico representa o pico teórico do *hardware* para uma determinada característica da aplicação, onde através do Gráfico 1 é possível observar tetos relacionados ao DPF e tetos relacionados à LB, os quais podem ser obtidos por meio de especificações do *hardware* ou de *benchmarks*. De acordo com a necessidade do usuário podem ser definidas técnicas de otimização para a aplicação, tais como: reconhecimento dos níveis de memória (cache e DRAM), vetorização, paralelismo em nível de *thread*, paralelismo em nível de instrução, entre outras. Para avaliar o desempenho da aplicação é necessário obter os parâmetros práticos, compostos por IntA, LB e o DPF, e relacioná-los aos parâmetros teóricos.

O *Intel Advisor* [2] é uma das principais ferramentas que atuam por meio do modelo *Roofline*. A ferramenta além de analisar o desempenho da aplicação em relação ao *hardware*, dá ênfase para análise de eficiência de vetorização e *multithreading*. Com a adição do pico de desempenho das memórias *cache*, métricas como: operações de ponto flutuante, tráfego de dados e LB são percebidos em diferentes níveis, podendo melhorar significativamente as diretrizes para otimização de *softwares*. Para a coleta dos parâmetros o *Advisor* executa a coleta *Survey* e *Trip Counts*. Durante a coleta *Survey* são obtidas as informações sobre o *hardware*, sobre o tempo de execução de cada *loop* e função de maneira individual, sobre problemas de desempenho e de vetorização da aplicação. Durante a coleta *Trip Counts* são obtidas as informações sobre as operações de ponto flutuante e de números inteiros, e sobre o tráfego de memória.

Na Figura 1 é possível observar que cada *loop* e função da aplicação é representada por um círculo, onde a distinção de cores e tamanhos auxiliam na análise do custo computacional de cada função. Os círculos vermelhos são custosos, amarelo custo moderado e verde baixo custo. Essa informação pode ser usada para tomar a decisão de quais funções apresentam possibilidade de otimização. O teto acima do ponto representa o limitador de desempenho daquele ponto em específico, onde cada teto representa seu próprio limite de *hardware*. Para ultrapassar um limite de *hardware*, será necessário realizar a otimização representada pelo próximo teto. Além disso, o desempenho do ponto pode ser afetado pelo teto que esteja abaixo. Um exemplo disso é caso o ponto esteja sutilmente acima de um teto, sendo um sinal de utilização ineficaz dos dois tetos em torno do ponto. A caracterização dos *loops* e funções da aplicação podem ser definidos baseado nas cores de fundo do gráfico. Nos gráficos das Figuras 2 e 3 é possível observar que a parte do gráfico onde a tonalidade é mais clara, é considerada intensiva em memória, tonalidade moderada é considerada um equilíbrio entre memória e processamento, já a tonalidade mais forte representa intensivo em processamento.

2.1. Trabalhos Relacionados

Alguns trabalhos utilizam o modelo *Roofline* para a compreensão dos limites do *hardware* e avaliação do *software*. Em [Kim et al. 2011], o *Roofline* é utilizado para compreender os limites do *hardware* e auxiliar na definição das otimizações necessárias ao algoritmo *Finite-Difference Time-Domain*. Além disso, o modelo foi utilizado para comparar o desempenho do algoritmo entre duas máquinas distintas. O trabalho de

[2] `https://software.intel.com/content/www/us/en/develop/tools/advisor`

[Marques et al. 2017] propõe a integração do *CARM* ao *Intel Advisor*. O *CARM* é implementado com base no modelo *Roofline* e representa os limites de processamento paralelo em processadores *multicore* com hierarquias de memória complexas. Essa integração possibilitou a compreensão do desempenho da aplicação em diferentes níveis de memória. Este estudo também analisa os gargalos mais críticos e possíveis passos de otimização de dez aplicações com a ferramenta *Intel Advisor*.

Outros trabalhos utilizam o *Roofline* para avaliar o desempenho e auxiliar na caracterização das aplicações em GPUs. Em [Lopes et al. 2017], foi proposto a implementação de um modelo *Roofline* para GPUs, pois inicialmente o modelo foi desenvolvido apenas para CPUs. Para validação foram analisados 23 *benchmarks* executados em oito GPUs distintas. Posteriormente [Yang et al. 2018] propôs o desenvolvimento da ferramenta *Empirical Roofline Tool* que oferece um conjunto mais realista de limites de desempenho para GPU e CPU, incorporando diferentes padrões de acesso à memória e contadores de FLOPs mais próximos ao consumo real. Ainda com foco em GPU, [Ibrahim et al. 2020] utiliza o modelo para analisar as características de desempenho de aplicações do *NAS* implementadas em *OpenACC*.

Apesar de haverem trabalhos com a proposta do uso do modelo *Roofline* para identificar os gargalos e identificar possíveis otimizações em aplicações, não foram encontrados trabalhos que utilizem essa metodologia para identificar padrões entre conjuntos de aplicações, como proposto neste trabalho. Além disso, este trabalho procura estabelecer uma metodologia para interpretar os resultados e caracterizar os requisitos das aplicações.

3. Metodologia de Experimentos

Nesta seção são apresentadas as arquiteturas e o conjunto experimental, utilizados com o objetivo de estudar os principais requisitos computacionais por meio do modelo *Roofline* e da ferramenta *Intel Advisor*. Na Tabela 1 é apresentada uma das arquiteturas utilizadas e suas principais informações de *hardware* que são utilizadas pelo modelo *Roofline*. Além desta arquitetura os experimentos também foram executados em uma arquitetura x86_64 com processador Intel Core i5-9500T 2.20 GHz. Porém, os resultados sobre a caracterização das aplicações e os gargalos computacionais foram similares devido ao fato das duas arquiteturas serem equilibradas entre memória e processamento. Assim, devido à limitação de espaço, somente uma das arquiteturas utilizadas é detalhada e apenas os resultados nesta arquitetura são descritos na Seção 4.

Tabela 1. Especificações da arquitetura utilizada.

Processor (i7-8700)		**Memory**	
CPU clock	3.20 GHz	MEM clock	2.66 GHz
Cores / threads	6 / 12	RAM size	64 GB
SIMD elements	8	Cache size	12.288 MB
Multiplier-Accumulator Units	2	Channels / Byte per channel	2 / 8
Peak GFlops	307.2	Peak Bandwidth	42.56

Foram executadas e analisadas seis aplicações do *NAS* [Frumkin et al. 2009], que representam aplicações usadas em *HPC*, mais especificamente na área de Dinâmica de Fluidos Computacionais (*CFD*), implementadas em *OpenMP*: BT (*Block Tri-diagonal*), LU (*Lower-Upper Gauss-Seidel*), CG (*Conjugate Gradient*), MG (*Multi-Grid*), SP (*Penta-diagonal scalar*) e FT (*Fourier Transform*). Os tamanhos das aplicações do *NAS*

variam conforme os tamanhos dos problemas aumentam. Cada aplicação foi executada 10 vezes para cada tamanho de problema entre A e C[3], e o número de *threads* foi predefinido para ocupar todos os núcleos de processamento disponíveis.

Foram executados e analisados quatro algoritmos de IA representativos dos usuários do supercomputador Santos Dumont [4]: K-Means, CNN (*Convolutional Neural Network*), ANN (*Artificial Neural Network* modelo *Multilayer Perceptron*) e SVR (*Support Vector Regression*). Todos os algoritmos foram treinados em uma base de dados sintética de 10.000 exemplos (sem levar em consideração a precisão do modelo, pois o objetivo foi avaliar o desempenho do algoritmo) executados 2 vezes cada, e o número de *threads* foi predefinido para ocupar todos os núcleos de processamento disponíveis.

4. Resultados

Na Figura 2(a) é apresentado o resultado para a aplicação BT. Baseado na tonalidade de cores de fundo de seus principais pontos (vermelhos e amarelos) podemos a caracterizar como uma aplicação equilibrada na utilização de memória e processamento. Também podemos observar que existem dois principais gargalos de desempenho para essa aplicação: a memória L3 e a adição escalar. Neste caso, por mais que haja uma quantidade significativa de pontos (verdes) limitados pela memória DRAM, são pontos que possuem baixo custo computacional e representariam um ganho de desempenho insignificante caso fossem otimizados. Fazer um melhor uso de memória cache L2 e vetorização das funções poderiam aumentar significativamente o desempenho da aplicação.

Na Figura 2(b) é apresentado o resultado da aplicação MG. Entre as aplicações do *NAS*, a MG é a que apresenta uma maior utilização de memória e menor IntA. Entretanto, fazendo uma interpretação sobre o modelo *Roofline*, ela ainda é caracterizada como uma aplicação equilibrada entre memória e processamento, tendendo mais para memória. Há espaço para otimizações de memória para essa aplicação, desde que se faça um uso eficiente da memória cache compartilhada. Porém, será necessária uma maior IntA para que otimizações de processamento sejam eficazes. Isso significa reestruturar o código caso seja possível, visando reduzir o tráfego de memória. Poderia ser utilizada, por exemplo, uma técnica de compressão e descompressão que aumentam a IntA (chegando a aplicação para a direita no gráfico), permitindo espaço maior de desempenho atingível até a aplicação chegar ao seu teto de desempenho.

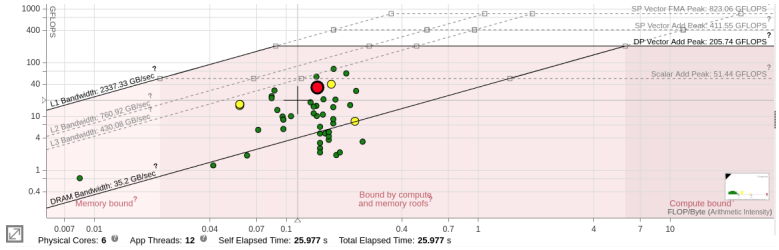
Os resultados apresentados na Figura 2 são para aplicações do *NAS* apenas do tamanho C, porém foram muito similares para os tamanhos A e B. Isso se explica pelo fato de o *Roofline* utilizar o tempo de execução da aplicação como um divisor para as operações de ponto flutuante, construindo no final de sua execução uma média do DPF por segundo. Portanto, os resultados geralmente são independentes do tamanho do problema.

Assim, pela limitação de espaço e pela similaridade entre os resultados, os tamanhos A e B não serão apresentados graficamente, bem como para as demais aplicações (LU, CG, SP e FT). Porém, os resultados indicam que todas as aplicações se caracterizaram como equilibradas entre memória e processamento, com algumas particularidades em relação aos seus gargalos. Por exemplo, as aplicações CG e FT são limitadas principal-

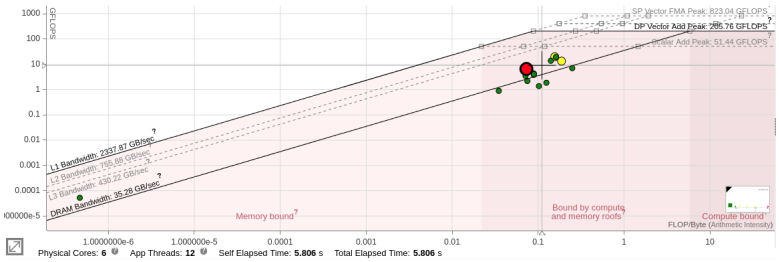
[3]As letras atribuídas aos pontos do Gráfico na Figura 1 são apenas para fins de representação, não possuem ligação com essa nomenclatura de tamanhos do *NAS*.

[4]https://sdumont.lncc.br/

mente pelo teto de adição escalar, tendo como prioridade para otimização a vetorização de seus principais pontos. A aplicação LU é limitada principalmente pela memória compartilhada, além de possuir também um gargalo de adição escalar. Já a aplicação SP apresenta um gargalo que difere das demais aplicações do *NAS* com um gargalo de memória DRAM.



(a) BT.



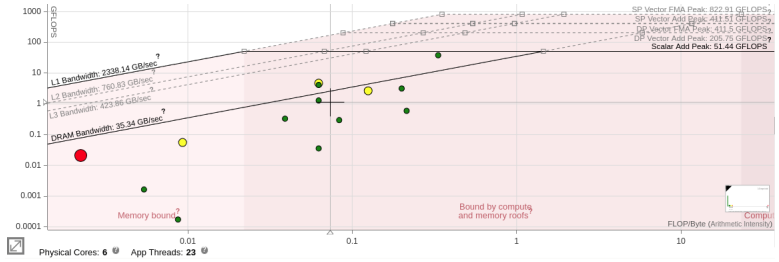
(b) MG.

Figura 2. Resultados do Roofline (Advisor) para as aplicações do *NAS*.

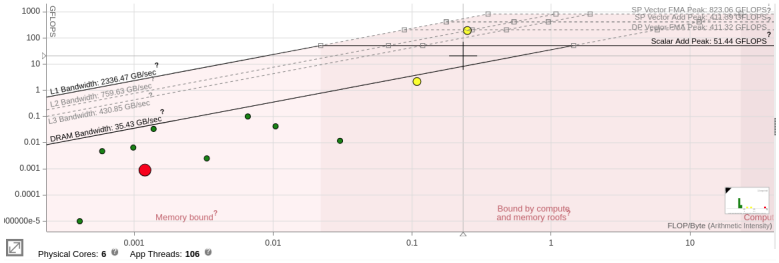
Na Figura 3 são apresentados os resultados para duas aplicações de IA. O resultado do K-Means Figura 3(a) se caracteriza como intensiva em memória, baseado nos principais pontos do gráfico, e que possui alguns requisitos de processamento (dois pontos de custo moderado à direita). Seu principal gargalo, a memória DRAM, é explícito, sendo necessário um melhor uso de memória cache compartilhada para atingir o máximo de desempenho da aplicação. Otimizações baseadas em processamento apresentariam um baixo ganho de desempenho, pois apenas se aplicarão em pontos de custo baixo e moderado. Assim, uma máquina com uma alta disponibilidade de memória e processamento baixo seriam o suficiente para a execução do algoritmo em seu estado atual.

A Figura 3(b) apresenta o resultado do algoritmo CNN, que assim como o K-Means, é um algoritmo caracterizado como intensivo em memória, considerando o principal ponto do gráfico. Possui baixo requisito de processamento em dois pontos de custo moderado à direita, com um destes pontos atingindo um alto DPF fazendo bom uso de vetorização e memória cache, sendo limitado principalmente pela memória L2. Assim como o K-Means, seu ponto principal possui um gargalo de memória DRAM, entretanto, o DPF, IntA e a eficiência do uso de memória desse algoritmo CNN é inferior ao K-Means.

Também foram analisados e caracterizados os algoritmos ANN e SVR. A ANN foi caracterizada como intensiva em memória e possui como principal gargalo a memória DRAM, com nenhuma de suas funções ultrapassando o teto do *hardware* de memória DRAM. A SVR, diferindo dos demais algoritmos de IA analisados, é caracterizada como equilibrada entre memória e processamento, isso vem do fato da SVR apresentar uma IntA maior que os demais algoritmos de IA. Entretanto, suas principais características e seu principal gargalo, assim como os demais algoritmos de IA, é a memória DRAM.



(a) K-Means.



(b) CNN.

Figura 3. Resultados do Roofline (Advisor) para as aplicações de IA.

Apesar de cada algoritmo apresentar suas particularidades de implementação e algumas características distintas, é possível observar um padrão de comportamento entre os conjuntos de aplicações. Enquanto aplicações do *NAS* requerem mais processamento e oferecem mais otimizações voltadas a processamento, os algoritmos de IA analisados possuem mais requisitos de memória e requerem muito pouco processamento. Além disso, observa-se uma discrepância entre os gargalos, enquanto a maioria dos algoritmos do *NAS* possuem gargalos voltados à memória cache compartilhada e vetorização, os algoritmos de IA apresentam um gargalo de memória DRAM.

Esses resultados e a metodologia utilizada para a interpretação dos gráficos demonstram como o modelo *Roofline* é útil para a compreensão dos gargalos e dos requisitos computacionais das aplicações. Essa compreensão são de grande importância para otimizações e tomadas de decisões voltadas para o desempenho. Ter o entendimento sobre os gargalos das aplicações auxilia em otimizações no *software*, melhorando seu desempenho. Já as características de um conjunto de aplicações auxiliam na aquisição e no desenvolvimento de novos *hardwares*. Além disso, conhecer as características da aplicação auxilia também na escolha de um *hardware* que melhor atende às necessidades de desempenho da aplicação, evitando aquisições além do necessário.

5. Considerações Finais e Trabalhos Futuros

Neste trabalho foi detalhado o modelo *Roofline* e como interpretar seus resultados. Também foi utilizada a ferramenta *Intel Advisor*, que implementa o modelo básico, além de acrescentar informações tais como, o mapeamento de *hardware* por meio das 3 tonalidades de cores do fundo do gráfico, técnicas de otimização voltadas para vetorização e *multithreading* e informação se o *loop* é otimizável ou não. Apesar de o *Roofline* não ser uma proposta nova, sua utilização e interpretação ainda são desafiadoras devido à complexidade inerente ao modelo, à falta de documentação e às divergências na literatura sobre os parâmetros básicos que permitem sua implementação (o mesmo acontece com

o *Intel Advisor*) e interpretação. Foram executados experimentos com um conjunto de aplicações do *NAS* que caracterizam aplicações de *CFD* e um conjunto de aplicações de IA, ambas utilizadas tipicamente em ambientes *HPC*. O objetivo foi avaliar o desempenho desses conjuntos diferentes de aplicações e os limitadores de desempenho (gargalos) frente ao *hardware*, além de demonstrar a utilidade do modelo *Roofline*. Além disso, foi apresentada uma metodologia para interpretação dos resultados do modelo que permitem caracterizar os principais requisitos computacionais das aplicações. Os resultados e a metodologia utilizada para a interpretação dos gráficos demonstram como o modelo *Roofline* foi útil para a compreensão dos gargalos e dos requisitos computacionais das aplicações. Conhecer os gargalos das aplicações auxiliam nas otimizações e melhorias de desempenho, enquanto identificar os requisitos computacionais das aplicações auxiliam na aquisição e no desenvolvimento de novos *hardwares* e na escolha de um *hardware* que melhor atende às necessidades de desempenho da aplicação.

Para trabalhos futuros, já está em desenvolvimento um aplicativo *web* que permita a geração do modelo *Roofline*. Além disso, serão estudados e implementados formas de ampliar o modelo para avaliar além do desempenho, a eficiência de consumo de energia das aplicações e suas técnicas de otimizações.

Agradecimentos

Os autores agradecem o apoio financeiro do LNCC, CNPq e FAPERJ.

Referências

[Frumkin et al. 2009] Frumkin, M., Jin, H., and Yan, J. (2009). Implementation of nas parallel benchmarks in high performance fortran.

[Ibrahim et al. 2020] Ibrahim, K., Williams, S., and Oliker, L. (2020). *Performance Analysis of GPU Programming Models Using the Roofline Scaling Trajectories*, pages 3–19.

[Kim et al. 2011] Kim, K.-H., Kim, K.-H., and Park, Q.-H. (2011). Performance analysis and optimization of three-dimensional fdtd on gpu using roofline model. *Computer Physics Communications*, 182:1201–1207.

[Lopes et al. 2017] Lopes, A., Pratas, F., Sousa, L., and Ilic, A. (2017). Exploring gpu performance, power and energy-efficiency bounds with cache-aware roofline modeling. In *2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 259–268.

[Marques et al. 2017] Marques, D., Duarte, H., Ilic, A., Sousa, L., Belenov, R., Thierry, P., and Matveev, Z. A. (2017). Performance analysis with cache-aware roofline model in intel advisor. In *2017 Int. Conference on HPC Simulation (HPCS)*, pages 898–907.

[Williams et al. 2009] Williams, S., Waterman, A., and Patterson, D. (2009). Roofline: an insightful visual performance model for multicore architectures. *Commun. ACM*, 52(4):65–76.

[Yang et al. 2018] Yang, C., Gayatri, R., Kurth, T., Basu, P., Ronaghi, Z., Adetokunbo, A., Friesen, B., Cook, B., Doerfler, D., Oliker, L., Deslippe, J., and Williams, S. (2018). An empirical roofline methodology for quantitatively assessing performance portability. In *2018 IEEE/ACM Int. Workshop on Performance, Portability and Productivity in HPC (P3HPC)*, pages 14–23.