UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

NÍCOLAS PARIS

Análise da Emissão de Carbono em Função da Frequência de Operação dos Subsistemas *Core* e *Uncore*

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Engenharia da Computação

Orientador: Prof. Dr. Arthur Francisco Lorenzon

Porto Alegre
2025

“We can only see a short distance ahead,
but we can see plenty there that needs to be done.”
— Alan Turing

AGRADECIMENTOS

Gostaria de agradecer a minha família pelas inúmeras oportunidades por eles a mim oferecidas. Essas oportunidades, começando pela mais antiga que tenho memória - o curso de inglês quando criança -, até a mais recente delas - poder mudar de cidade para frequentar a universidade federal - foram as peças com as quais construí o caminho que sigo hoje. Se sou uma pessoa confiante, independente e responsável, é por todas as palavras de encorajamento que ouvi deles a vida toda.

Também sou grato ao meu orientador, Arthur. Sem sua ajuda, os passos finais da minha graduação indubitavelmente seriam muito mais difíceis.

Por fim, quero agradecer a todos os amigos que trilharam o caminho da graduação ao meu lado. Tanto os momentos de alegrias quanto as frustrações diárias fizeram parte dessa linda caminhada traçada por nós.

RESUMO

É consenso que a ebulição global é causada principalmente pela ação humana e que seus efeitos são sentidos em várias regiões do mundo. Além disso, há um aumento acentuado da quantidade de sistemas de computação de larga escala, que consumem grandes quantidades de energia (muitas vezes proveniente da queima de combustíveis fósseis) para operar. Portanto, reduzir a pegada de carbono de sistemas de computação é uma necessidade urgente. Esta monografia procura divulgar essa urgência, revisar trabalhos publicados para expor alternativas para conquistar essa redução e, por fim, analisar qual é impacto da variação da frequência dos subsistemas *core* e *uncore* na emissão de carbono do sistema de computação. Os resultados dessa análise fornecem informações relevantes sobre quais configurações de frequência escolher dependendo do comportamento da aplicação quando o intuito é minimizar a pegada de carbono, conquistando reduções entre 12,5 e 38 por cento ao comparar a configuração ótima encontrada com a configuração padrão que visa maximizar o desempenho, totalizando uma redução de 22,55% ao somar as reduções de todas as comparações.

Palavras-chave: Emissão de carbono. Pegada de carbono. Frequência de operação. Core. Uncore.

Analysis of Carbon Emissions as a Function of the Operating Frequency of Core and Uncore Subsystems

ABSTRACT

It is widely agreed that global boiling is primarily caused by human actions and that its effects are felt in various regions around the world. Moreover, there has been a sharp increase in the number of large-scale computing systems, which consume vast amounts of energyoften derived from the burning of fossil fuelsto operate. Therefore, reducing the carbon footprint of computing systems is an urgent necessity. This dissertation aims to highlight this urgency, review published studies to present alternatives for achieving this reduction, and ultimately analyze the impact that varying the frequency of the core and uncore subsystems has on the carbon emissions of computing systems. The results of this analysis provide relevant insights into which frequency configurations to choose depending on the application's behavior when the goal is to minimize the carbon footprint, achieving reductions between 12.5 and 38 percent when comparing the optimal configuration found to the standard configuration aimed at maximizing performance, resulting in a total reduction of 22.55% by summing up the reductions from all comparisons.

Keywords: Carbon emission. Carbon footprint. Operating frequency. Core. Uncore.

LISTA DE ABREVIATURAS E SIGLAS

DVFS	Dynamic Voltage and Frequency Scaling
UFS	Uncore Frequency Scaling
CPU	Central Processing Unit
GPU	Graphics Processing Unit
IA	Inteligência Artificial
SO	Sistema Operacional
ULA	Unidade Lógico e Aritmética
UC	Unidade de Controle
ACPI	Advanced Configuration and Power Interface
FFT	Fast Fourier Transform
HPCG	High-Performance Conjugate-Gradient
LULESH	Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics

LISTA DE FIGURAS

LISTA DE LISTAGENS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O mundo está passando por um aumento acelerado da temperatura média global causado pelo estímulo exagerado do efeito estufa e muitos cientistas passaram a chamar esse aumento de ebulição global (AMNUAYLOJAROEN, 2023). O efeito estufa é um fenômeno natural fundamental para a sobrevivência humana, porque é o responsável por manter a temperatura do planeta estável por meio da absorção de parte da energia solar refletida pelos oceanos e pela superfície terrestre (KWEKU et al., 2018). No entanto, desde a revolução industrial, esse fenômeno natural vem sendo acelerado por meio, principalmente, da queima de combustíveis fósseis, que libera majoritariamente dióxido de carbono (CO_2), o mais predominante gás responsável pelo efeito estufa. Este cenário traz consequências negativas para o meio ambiente, como, por exemplo, o aumento da ocorrência de eventos extremos, e precisa ser analisada e freada urgentemente.

Somando a isso, a quantidade de *data centers* e sistemas de computação de larga escala só aumenta e tem impacto significativo na mudança climática. Conforme Lannelongue, Grealey and Inouye (2021), torna-se evidente a necessidade de medidas para reduzir este impacto. A pegada de carbono dos sistemas de computação é proveniente tanto da energia consumida pelo funcionamento das máquinas, quanto da manufatura das peças e da infraestrutura necessária para sua operação (GUPTA et al., 2021). Então, existem diversos escopos possíveis para propor soluções que diminuem esse impacto. Trazendo uma abordagem bastante abrangente, Gupta et al. (2021) analisa todo o ciclo de vida de sistemas de computação de larga escala, desde a fabricação do *hardware*, passando pelo tempo que ele estará operando, até o processamento do lixo eletrônico gerado quando sua operação for encerrada.

Já do ponto de vista específico de estratégias para reduzir a potência consumida por sistemas de computação, algumas focam somente em aspectos de *software*, como a otimização do número de *cores* utilizados por uma aplicação paralela, enquanto outras se concentram em otimização no escopo do *hardware*, como *dynamic voltage and frequency scaling* (DVFS) e *uncore frequency scaling* (UFS), que ajustam dinamicamente a frequência de operação de componentes de *hardware* de um processador dependendo da necessidade de processamento atual do *chip* (MAAS et al., 2024). DVFS agindo no subsistema do *core* (composto por unidade de controle, unidade lógica aritmética e caches L1 e L2) e UFS agindo no subsistema do *uncore* (composto por cache L3, controlador de memória, entre outros componentes compartilhados) são úteis para reduzir o impacto

destes sistemas no consumo de energia e consequente quantidade de carbono emitida.

No entanto, dadas as características heterogêneas das aplicações, não existe uma configuração universal de frequência de *core* e *uncore* para cada sistema computacional capaz de minimizar consistentemente os efeitos de carbono emitidos por sistemas de computação. Cada aplicação possui padrões específicos de acesso à memória, intensidade aritmética e uso de recursos, o que torna inviável aplicar uma abordagem única de ajuste de frequências que atenda a todos os cenários com eficiência. Por isso, um estudo que analise caso a caso as demandas computacionais e energéticas de cada aplicação é necessário.

1.1 Objetivos

Considerando a discussão realizada anteriormente, o objetivo principal desta monografia consiste em avaliar os efeitos que a variação da frequência de operação dos domínios de *core* e de *uncore* de um processador tem sobre a emissão de carbono do sistema de computação, com o intuito de entender quais são as configurações que minimizam a pegada de carbono da execução da aplicação. Para isso, os seguintes objetivos específicos foram definidos:

- Analisar a variação da frequência de operação do *core* e *uncore* fazendo uso do conjunto de ferramentas LIKWID[1].
- Definir e executar as doze aplicações escolhidas para cada uma das frequências desejadas coletando métricas de desempenho, consumo de energia e emissões de carbono.
- Compilar e analisar os resultados obtidos visando encontrar padrões de configurações que minimizam as emissões de carbono.

1.2 Estrutura do Texto

O restante desta monografia está organizada em cinco capítulos. Primeiro, a fundamentação teórica aborda uma definição do que são arquiteturas de *Central Processing Unit* (CPU), expõe maneiras de variar a frequência de operação dos subsistemas *core* e *uncore*, com foco principal em ferramentas de *software* no escopo do usuário e discute

[1] https://github.com/RRZE-HPC/likwid/wiki

sobre diferentes maneiras de estimar a pegada de carbono de um sistema de computação. Em seguida, na parte de trabalhos relacionados, são analisadas diferentes estratégias para diminuição da emissão de carbono, algumas seguindo escopos menores (somente no escopo do *software*) e outras mais abrangentes - levando em consideração todo o ciclo de vida de um supercomputador. Na sequência, a metodologia é descrita, tendo como pontos focais a definição das aplicações executadas, especificação da máquina na qual os experimentos foram realizados, exposição das ferramentas - e seus respectivos comandos - utilizadas para variar a frequência de operações do *hardware* de interesse e descrição de como foram feitos os cálculos para estimar a pegada de carbono das execuções. Logo após isso, a análise dos resultados foi feita de maneira comparativa entre as execuções de uma mesma aplicação e também entre aplicações com o intuito de encontrar padrões de comportamento para elas. Por fim, é feita a conclusão.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo é apresentada a fundamentação teórica, dividida em três seções. A primeira busca definir o que é uma arquitetura de CPU e explicar o que são e quais componentes compõem os subsistemas *core* e *uncore* em processadores Intel. A segunda explica o que é a frequência de operação de uma CPU e por que é importante poder variá-la. Além disso, descreve duas estratégias para executar essa variação em sistemas Linux, uma que utiliza *governors* disponibilizados pelo sistema operacional e outra que foca no uso de ferramentas de *software* no escopo de usuário. Encerrando o capítulo, a terceira descreve diversas técnicas usadas para estimar a pegada de carbono de um sistema de computação e estabelece comparações entre elas.

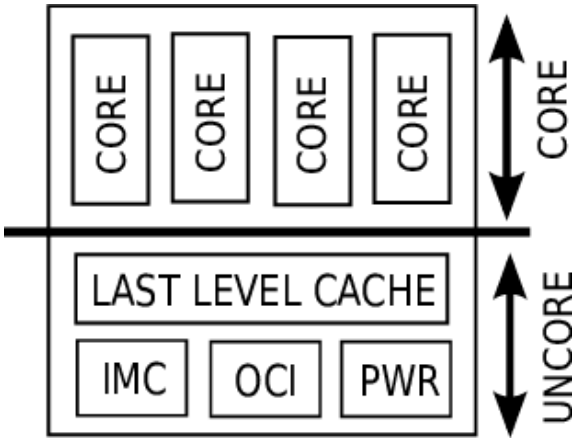
2.1 Arquitetura de CPU

Uma arquitetura de CPU refere-se a todos os atributos que tem impacto direto na lógica de execução de um programa (e.g. conjunto de instruções, mecanismos de I/O e técnicas de endereçamento de memória), ditando quais são as instruções disponíveis, como é feita a gerência dos dados e como a CPU comunica-se com outros componentes do computador (STALLINGS, 2016). A separação explicita em dois blocos, *core* e *uncore*, como mostrado na Figura 2.1, é uma abstração exclusiva da Intel introduzida originalmente na microarquitetura Nehalem (DIXON et al., 2010).

O *core* de um processador é o sistema responsável pela execução das instruções fornecidas pelo sistema operacional e pelos programas (STALLINGS, 2016). (STALLINGS, 2016) também afirma que cada *core* funciona como um processador independente em um *chip* e é responsável por realizar operações fundamentais, como cálculos matemáticos, manipulação de dados e controle do fluxo de execução. Seus principais componentes são:

- Unidade de Processamento: composta pela Unidade Lógico e Aritmética (ULA) que executa os cálculos e pela Unidade de Controle (UC) que gerencia o fluxo de execução das instruções.
- Memória Cache: memória de alto desempenho utilizada para salvar dados usados com frequência pela Unidade de Processamento com o intuito de acelerar a computação. Usualmente composta por dois níveis, L1 e L2.

Figura 2.1: Domínios de *Core* e *uncore* em um processador com quatro núcleos.



Fonte: Gupta et al. (2012)

No que diz respeito ao *uncore*, Gupta et al. (2012) afirma que esse é um conjunto de componentes de um processador que não fazem parte do *core* mas são fundamentais para o funcionamento dele, servindo como um elo entre os múltiplos *cores*, suas interfaces e os sistemas externos à CPU, segundo Dixon et al. (2010). Alguns de seus componentes são:

- Cache de Último Nível (LLC): também conhecida como cache L3, funciona de maneira similar às caches L1 e L2, mas, diferente delas, é compartilhada entre os *cores*.
- Controlador de Memória: é a unidade responsável por gerenciar o acesso da CPU à memória principal (RAM), coordenando as operações de leitura e escrita e assegurando que os dados sejam transferidos de maneira sincronizada entre o processador e a memória.

2.2 Métodos para Variação de Frequência dos Componentes de *Hardware*

A frequência de operação da CPU, ou velocidade do clock, refere-se ao número de ciclos por segundo executados pela CPU, em GHz. Variar essa frequência é importante para otimizar tanto o desempenho, em aplicações que são consideradas CPU intensivas, quanto a eficiência energética, em aplicações que fazem uso intenso da memória[1]. Por meio do subsistema CPUFreq, o *kernel* Linux oferece duas camadas de abstração para escalonar a frequência de operação: *governors* de escalonamento e *drivers* de escalona-

[1] https://wiki.archlinux.org/title/CPU_frequency_scaling

mento[1]. Os *governors* são abstrações disponibilizadas pelo SO que permitem ao usuário escolher qual estratégia de escalonamento de frequência será utilizada (MAAS et al., 2024). Os mais utilizados são:

- *Performance*: Mantém a CPU na máxima frequência para garantir o melhor desempenho possível.
- *Powersave*: Focado em minimizar a potência dissipada, configura a CPU para operar na frequência mínima.
- *Ondemand*: Ajusta dinamicamente a frequência baseado na carga da CPU sendo computada.
- *Userspace*: Permite que o usuário configure a frequência como desejar, por intermédio dos *drivers* (e.g. acpi_cpufreq).

Utilizando os *governors*, DVFS é utilizado para variar a frequência de operação do domínio do *core* (MAAS et al., 2024). De maneira semelhante, o subsistema *uncore* também pode ser gerenciado pelo SO utilizando UFS, que determina automaticamente a frequência apropriada para estes elementos (MAAS et al., 2024). Com isso, Maas et al. (2024) conclui que o processador, com base nas demandas de trabalho, pode fornecer velocidades de *clock* mais altas ou mais baixas, ajustando dinamicamente a frequência do *core* e do *uncore*, melhorando a eficiência energética do sistema.

Esse escalonamento dinâmico passou a ser possível com a implementação da especificação *Advanced Configuration and Power Interface* (ACPI). Introduzida em meados da década de 1990, a ACPI é uma especificação aberta que possibilita a configuração de dispositivos e o gerenciamento de energia pelo sistema operacional. Ela estabelece interfaces independentes de plataforma para descoberta de *hardware*, configuração, gerenciamento de energia e monitoramento. Seu objetivo era substituir padrões anteriores, como Advanced Power Management, MultiProcessor Specification e BIOS Plug and Play. O padrão transferiu o controle do gerenciamento de energia, anteriormente centralizado no *Basic Input/Output System* (BIOS), para o sistema operacional, eliminando a dependência de *firmware* específico da plataforma para definir políticas de energia e configuração. O objetivo principal da ACPI é consolidar e aprimorar os padrões de configuração e de gerenciamento de energia, existentes para dispositivos de *hardware*[2].

[2]https://uefi.org/htmlspecs/ACPI_Spec_6_4_html/Frontmatter/Overview/Overview.html

2.2.1 Ferramentas de *Software Userspace*

Usualmente os *governors* suprem adequadamente as necessidades do usuário, mas é eventualmente necessário aplicar configurações customizadas diretamente na CPU, como no caso desta monografia. Isso pode ser realizado por meio de ferramentas de *software* no escopo de usuário (*userspace*). Para isso, elas utilizam *drivers*, como o acpi_cpufreq. Uma opção disponível é a LIKWID, um conjunto de ferramentas projetadas para monitorar, medir e otimizar o desempenho de sistemas *multicore*[3]. Algumas dessas ferramentas são:

- *likwid-setFrequencies*: implementa um *daemon* e um *script* de controle para manipular os *governors* do SO e fixar a frequência de operação do *core* e do *uncore* para o valor desejado. Não é compatível com o *driver* intel_pstate, sendo necessário desabilitá-lo para usar o acpi_cpufreq.
- *likwid-topology*: exibe a topologia das *threads* e *caches* de sistemas *multicore*
- *likwid-perfctr*: ferramenta que usa os *hardware performance counters* de processadores Intel e AMD recentes para fazer o *profiling* de aplicações sem a necessidade de modificar o código delas.

Outra opção para os desenvolvedores de *software* é a *cpupower*, uma extensão para *Gnome Shell* que funciona como gerenciador de potência de CPU. Tem funcionalidades bastante similares com a LIKWID, mas, diferente dela, apresenta suporte para utilização tanto do *driver* acpi_cpufreq quanto do intel_pstate, o que é uma vantagem. Apesar disso, a LIKWID foi escolhida como ferramenta para manipular a frequência seguindo a sugestão do orientador, que já havia familiaridade com esta.

2.3 Cálculo da Pegada de Carbono

Gupta et al. (2021) afirma que a maioria da emissão de carbono, no contexto de computação, é gerada por dois fatores principais: produção da energia consumida pela operação do *hardware* e manufatura de componentes e de infraestrutura. Gupta et al. (2021) também mostra que o segundo fator é mais impactante na pegada de carbono que o primeiro. Apesar disso, por ser mais simples de quantificar e também mais fácil de otimizar para reduzir as emissões, esta monografia e muitos outros trabalhos focam

[3] https://github.com/RRZE-HPC/likwid/wiki

somente no primeiro fator, como (LANNELONGUE; GREALEY; INOUYE, 2021), que, para manter uma maior abrangência nas tarefas de computação cobertas pelo modelo proposto, deixa de analisar as emissões causadas por fatores de manufatura.

Lannelongue, Grealey and Inouye (2021) descreve uma metodologia utilizada para estimar a pegada de carbono de qualquer tarefa de computação e implementa uma calculadora disponibilizada gratuitamente para que o público geral possa calcular sua própria pegada de carbono. A metodologia de cálculo empregada é composta por dois passos. No primeiro, é calculada uma estimativa para a energia consumida pelo processo computacional, levando em consideração o *hardware* utilizado - CPU, GPU e memória -, tempo de execução e a eficiência energética da plataforma na qual está sendo realizada a computação. O segundo passo converte o resultado do primeiro para a pegada de carbono estimada, utilizando fatores de emissão específicos da região geográfica onde foi produzida e distribuída essa energia. A calculadora aplica essa metodologia e os parâmetros informados pelo usuário para estimar a pegada de carbono da tarefa de computação dele. Além dos parâmetros descritos anteriormente, pode-se utilizar um parâmetro adicional, *Pragmatic Scaling Factor*, para estimar o custo de execuções repetidas de uma mesma tarefa. Uma das principais vantagens dessa ferramenta é a praticidade, já que não é necessário modificar nenhum código existente para aplicá-la. Analogamente, mas focado em aprendizado profundo para modelos de inteligência artificial (IA), Anthony, Kanding and Selvan (2020) propõe a *carbontracker*, que é uma ferramenta para monitorar e prever o consumo de energia e as emissões de carbono durante o treinamento de modelos de aprendizado profundo. A metologia de cálculo é bastante semelhante a de Lannelongue, Grealey and Inouye (2021), diferindo principalmente na fonte de onde se busca a informação da quantidade de carbono emitida por unidade de energia produzida, conhecida como intensidade de carbono, usada para estimar a emissão total.

A eficiência energética é um dos pontos-chave usados por Lannelongue, Grealey and Inouye (2021) para calcular a pegada de carbono de uma tarefa de computação, então otimizá-la é fundamental para redução do impacto ambiental dos sistemas de computação. Para fazer a otimização dos parâmetros de execução com objetivo de maximizar a eficiência energética, a maioria dos trabalhos existentes tem como foco a análise estática da tarefa de computação ou predições em tempo de execução. Wang et al. (2016) traz uma abordagem para o problema que utiliza um modelo misto entre as duas técnicas anteriormente citadas. Por meio de um processo de duas etapas que utiliza concorrência no número de *threads* e DVFS, é possível obter uma configuração ótima para a eficiência

energética da carga de trabalho. A modelagem empregada por Wang et al. (2016) para calcular a eficiência energética considera principalmente três fatores, todos sendo inversamente proporcionais a essa eficiência: concorrência no número de *threads*, potência dissipada e tempo de execução. Com isso e com a informação de que em diversas aplicações paralelas, a parte serial corresponde a até 87% da aplicação, Wang et al. (2016) conclui que, muitas vezes, os benefícios de desempenho alcançados por meio do aumento no número de *threads* são ofuscados pelo acréscimo no consumo de potência causado por esse aumento.

3 TRABALHOS RELACIONADOS

Neste capítulo são apresentados os trabalhos relacionados. Primeiro discute-se sobre a importância da diminuição da emissão de carbono de sistemas de computação, expondo algumas consequências negativas da ebulição global e também qual é o papel dos sistemas de computação nessa ebulição. Em seguida são descritas algumas estratégias para alcançar essa diminuição. Por fim é feita a contextualização relacionando este capítulo com o restante da monografia.

Haines et al. (2006) discute os impactos na saúde humana decorrentes da mudança climática causada pela queima de combustíveis fósseis. Dentre os impactos discutidos estão o aumento na ocorrência de eventos extremos, como inundações e secas, aumento do alcance de doenças transmitidas por vetores, como a dengue, e ondas de calor e de frio, provavelmente afetando mais duramente populações de baixa renda. Os efeitos negativos para saúde causados pela mudança climática são inversamente proporcionais ao desenvolvimento socioeconômico local e a efetividade com que medidas de adaptação são implementadas, além disso, são experienciados juntamente com outros fatores que dependem mesmas características regionais, como superpopulação e falta de saneamento básico, o que pode ocasionar a magnificação dos impactos sentidos (HAINES et al., 2006).

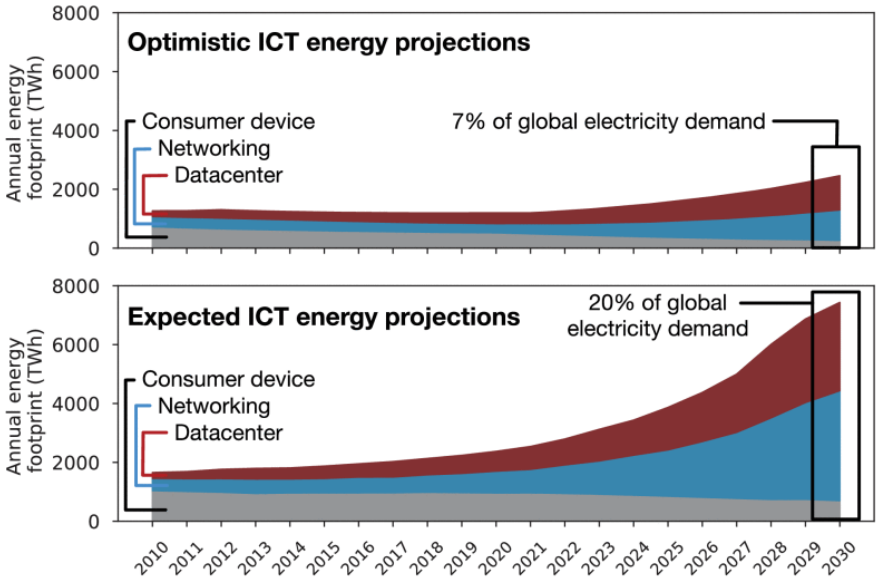
Tendo em vista as inúmeras consequências negativas causadas pela ebulição global, como as discutidas no parágrafo acima, e que a porcentagem de energia consumida por sistemas de computação pode alcançar aproximadamente 20% da demanda global total por energia, segundo o projetado pelos gráficos da Figura 3.1, é de extrema importância adotar medidas para diminuir a emissão de carbono dos sistemas de computação.

3.1 Estratégias para Diminuição da Emissão de Carbono

Analisando a emissão causada pela operação de *clusters* de *High Performance Computing* (HPC) e usando como exemplo o treinamento de modelos de IA, Lannelongue, Grealey and Inouye (2021) sugere limitar as execuções de uma mesma tarefa ao mínimo possível. Isso incluí realizar ajuste fino de parâmetros somente quando necessário e construir exemplos em menor escala para *debugging*. Somando a isso, Gupta et al. (2021) recomenda treinar os modelos com *data-sets* que contenham somente a quantidade necessária de dados que garanta a precisão desejada.

Do ponto de vista da emissão causada pela manufatura do *hardware* e da infra-

Figura 3.1: Projeção do consumo de energia global por sistemas de computação.



Fonte: Gupta et al. (2021)

estrutura utilizada, Gupta et al. (2021) indica que os projetistas de sistemas de maior escala, como *data centers* e redes móveis, planejem com cuidado o *hardware* utilizado para garantir o desempenho desejado, com o intuito de evitar exagero na escala do sistema montado e para sistemas que já existem é recomendado diminuir gradativamente a escala de operação, se possível. Outra estratégia relevante proposta por Gupta et al. (2021) para reduzir a pegada de carbono relacionada com a fabricação de *hardware* é investir em melhorias na confiabilidade e resistência dos dispositivos, estendendo a vida útil deles.

A partir da constatação - presente em Radovanovi et al. (2023) - que a intensidade de carbono varia conforme o horário e a localização geográfica, muitos trabalhos exploram essa característica apontada para cumprir seus objetivos. Explorando o aspecto geográfico da intensidade de carbono, Zhou et al. (2016) demonstra que a otimização em tempo real, usando técnicas como balanceamento geográfico de carga e escalonamento da velocidade dos servidores, pode reduzir significativamente as emissões ao redirecionar tarefas para regiões com menor intensidade de carbono. Seguindo uma linha semelhante, Radovanovi et al. (2023) introduz um sistema usado pela Google para gestão de computação carbono-inteligente, que além de realizar balanceamento geográfico das cargas, utiliza previsões de intensidade de carbono para atrasar trabalhos flexíveis no aspecto temporal, deixando para executá-los em horários de menor emissão. Processamento de vídeos, simulações, pipelines de aprendizado de máquina e compactação de dados são alguns exemplos de aplicações que se encaixam nessa otimização.

Diferentemente das anteriores, o objetivo principal de Dou et al. (2017) é minimizar o custo financeiro, especialmente de *data centers* que possuem fontes de energia renovável locais, como placas solares e turbinas eólicas. Dou et al. (2017) diz que parte do problema de otimização que eles pretendem solucionar é causado por oscilações na quantidade de energia gerada pelas fontes de energia sustentáveis *in loco*, sendo necessário consumir energia de fontes externas nos períodos de menor geração de energia, o que aumenta o custo. A solução proposta por eles é um algoritmo de gerenciamento de cargas que enfileira cargas de trabalho que são tolerantes a atrasos para serem executadas em horários com maior disponibilidade de energia renovável. Como consequência de executar essas cargas utilizando energia de fontes renováveis, a emissão também é reduzida.

3.2 Contextualização

Os trabalhos citados nas seções anteriores deste capítulo são relevantes para esta monografia principalmente porque ressaltam a necessidade de reduzir a emissão de carbono e como alcançar esse objetivo - especialmente aqueles que propõe soluções no escopo de *software*, já que esse é o escopo escolhido por esta monografia. Portanto, esta monografia avança no estado da arte ao estudar o impacto que a frequência de operação dos componentes de *core* e *uncore* têm na pegada de carbono dos sistemas computacionais.

4 METODOLOGIA

Conforme destacado no Capítulo 1, essa monografia visa avaliar a emissão de carbono em função da variação da frequência de operação do *core* e *uncore* da CPU. Para isso, foram definidas e executadas doze aplicações variando a frequência desses subsistemas, todas utilizando o número máximo de *threads* disponível, que, na máquina utilizada, é 88. Elas foram escolhidas por serem aplicações bastante conhecidas e amplamente utilizadas como *benchmark* de sistemas de computação (MAAS et al., 2024). As aplicações escolhidas foram as seguintes:

- *Fast Fourier Transform* (FFT): A Transformada Rápida de Fourier é um algoritmo para calcular a Transformada de Fourier Discreta e sua inversa. Ela é bastante útil como *benchmark* pois executa cálculos intensivos, como operações de ponto flutuante e padrões complexos de acesso à memória (FRIGO; JOHNSON, 1998).
- *High-Performance Conjugate-Gradient* (HPCG): É um *benchmark* projetado para avaliar sistemas típicos de aplicações científicas reais, como a resolução de sistemas lineares esparsos. Ele mede não apenas a capacidade de cálculo, mas também o desempenho da memória e a eficiência em comunicação e computação, considerando otimizações em hierarquias de memória e técnicas de paralelismo. Por isso, é um indicador fiel do desempenho em aplicações reais. (DONGARRA; HEROUX; LUSZCZEK, 2015).
- Método de Jacobi: É um método iterativo utilizado para resolver sistemas de equações lineares da forma $Ax = b$, onde A é uma matriz de coeficientes, b é o vetor de termos constantes, e x é o vetor de incógnitas[1]. Esse método é útil como *benchmark* porque sua implementação permite avaliar o desempenho de sistemas de computação em operações matriciais e em problemas de álgebra linear, que são computacionalmente intensivos. Além disso, ele é representativo de cálculos numéricos em áreas como simulações físicas e processamento de sinais.
- *Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics* (LULESH): É um aplicativo *benchmark* desenvolvido no Laboratório Nacional de Lawrence Livermore (LLNL) que simula equações de dinâmica de fluidos utilizando o método de elementos finitos em malhas não estruturadas. Foi projetado para testar modelos de programação paralela e desempenho em diferentes plataformas, explorando a eficiência de memória e comunicação. Ele ajuda os desenvolvedores a identificar

[1] https://www.ufrgs.br/reamat/CalculoNumerico/livro-py/sdsl-metodos_iterativos_para_sistemas_lineares.html

gargalos de desempenho e otimizar sistemas computacionais. O *benchmark* é amplamente utilizado para estudar as interações entre o *software* e o *hardware*, tendo versões disponíveis que utilizam modelos como OpenMP, CUDA, e MPI (KARLIN, 2012).

- Equação de Poisson: É uma equação diferencial parcial, descrita como $\nabla^2\phi = \rho$, onde ϕ representa o potencial escalar e ρ é a densidade da fonte. Ela é amplamente utilizada para modelar a distribuição de potenciais em campos eletrostáticos, gravitacionais e de outros tipos de sistemas estáticos. Sua aplicabilidade abrange desde o design de semicondutores até a análise de mecânica de fluidos e fenômenos ópticos. Como *benchmark*, a equação de Poisson é particularmente útil devido a sua complexidade matemática e necessidade de soluções numéricas intensivas, especialmente para grandes grades e problemas tridimensionais (NAGEL, 2014).
- *NAS Parallel Benchmarks* (NPB): Desenvolvido pela divisão NASA *Advanced Supercomputing* (NAS), é um conjunto de aplicações idealizadas para ajudar na avaliação do desempenho de computação paralela em supercomputadores. Os *benchmarks* são derivados de aplicações que computam problemas de dinâmica de fluidos [2]. Essa monografia utiliza os *kernels* CG, FT e MG, as pseudo aplicações BT, LU e SP e o *benchmark* de computação não estruturada UA.

Os experimentos utilizaram recursos da infraestrutura PCAD, no INF/UFRGS[3], mais especificamente a partição *blaise* foi considerada, que conta com as especificações destacadas na Tabela 4.1. Para variar a frequência dos componentes do *core* e *uncore* foi utilizada a *likwid-setFrequencies*, que faz parte do *tool set* LIKWID. Primeiro o ambiente foi configurado usando os comandos da Listagem 4.1. Em seguida, a frequência de operação do *core* foi variada entre $1.2Ghz$ e $2.8Ghz$ em incrementos de $0.2Ghz$ usando o comando da Listagem 4.2 e, para cada valor da frequência do *core*, a frequência do *uncore* foi variada com o mesmo intervalo por meio do comando da Listagem 4.3, totalizando 81 execuções por aplicação.

[2] https://www.nas.nasa.gov/software/npb.html
[3] http://gppd-hpc.inf.ufrgs.br/

Listagem 4.1: Configuração do ambiente

```
# Alterar governor para userspace
likwid-setFrequencies -g userspace
# Desligar modo turbo da CPU
likwid-setFrequencies -t 0
```

Listagem 4.2: Configuração da frequência do *core*

```
# CURRENT_FRENQUENCY é o valor escolhido para frequência
likwid-setFrequencies -f ${CURRENT_FREQUENCY}
```

Listagem 4.3: Configuração da frequência do *uncore*

```
# CURRENT_FRENQUENCY é o valor escolhido para frequência
likwid-setFrequencies --umin --umax ${CURRENT_FREQUENCY}
```

Já para estimar a pegada de carbono, foi utilizada a Equação 4.1, descrita por Anthony, Kanding and Selvan (2020). O primeiro fator da equação, consumo de energia, foi medido utilizando o Intel RAPL ao executar as aplicações. Já o segundo, intensidade de emissão de carbono, foi obtido por meio da *carbontracker*. Existe uma limitação prática para a acurácia do segundo fator, pois não existe um banco de dados centralizando contendo todas as informações de intensidade de carbono da malha energética de cada região (ANTHONY; KANDING; SELVAN, 2020). Por isso, a ferramenta busca em diversas fontes para obter a intensidade do local mais próximo da máquina que está executando os experimentos, que nesse caso foi $98.35\ \frac{gCO_2}{kWh}$ para a região de São Paulo, capital.

$$Carbon\ Footprint = Energy\ Consumption \cdot Carbon\ Intensity. \tag{4.1}$$

Tabela 4.1: Especificações da partição blaise

CPU	*RAM*	*Disco*	*Placa-mãe*
2 x Intel(R) Xeon(R) E5-2699 v4, 2.20 GHz, 88 *threads*, 44 *cores*	256 GB DDR4	1.8 TB SSD, 1.8 TB HDD	Supermicro X10DGQ

Fonte: O Autor

5 RESULTADOS

Seguindo os passos descritos na metodologia, foram obtidos os valores da pegada de carbono de cada execução das aplicações-alvo. Esses valores representam uma estimativa de quanto CO_2 foi emitido em decorrência de cada execução. Para auxiliar na análise dos resultados, foram gerados mapas de calor para cada uma das aplicações. Esses mapas têm como eixo *x* os valores de frequência de *uncore*, como eixo *y* os de *core* e mostram a emissão de carbono, medida em gCO_2, gerada pela execução da aplicação para cada par de valores (*x,y*). Eles ajudam a identificar padrões de emissão das aplicações conforme os parâmetros são variados por meio das cores, sendo possível visualizar as regiões mais frias como as de menor emissão e as mais quentes como de maior emissão.

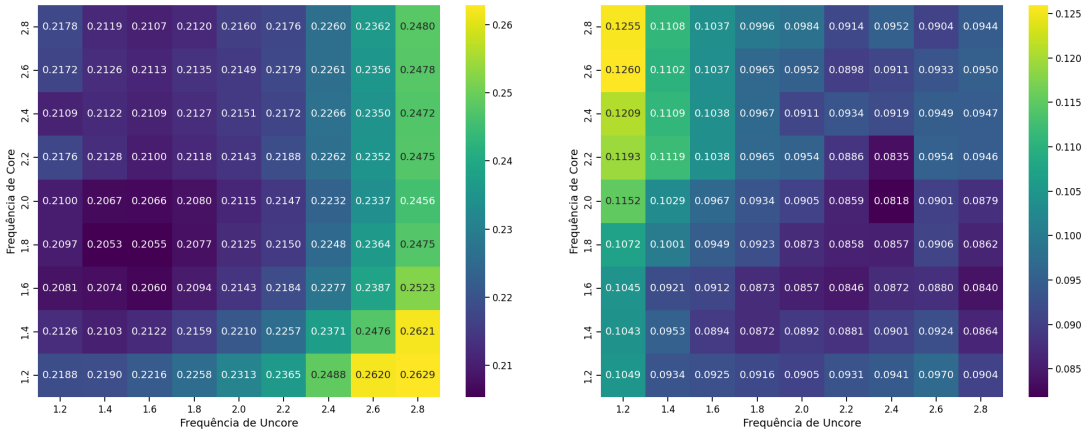
Nesse contexto, a configuração ótima é aquela que minimiza a emissão, em contraponto ao padrão mais frequente dos sistemas operacionais, que foca em maximizar o desempenho, utilizando sempre que possível as maiores frequências disponíveis. É importante também considerar que aumentar a frequência de operação do subsistema *core* melhora o desempenho de tarefas que realizam intensa computação e aumentar a do *uncore* aprimora o desempenho de operação de comunicação entre *threads* de diferentes *cores* ou de acessos à memória. Analisando os gráficos e usando como critério de agrupamento o comportamento do uso de recursos das aplicações, elas foram agrupadas em três categorias: *CPU-bound*, *memory-bound* e balanceadas.

5.1 Aplicações *CPU-Bound*

É caracterizada como *CPU-bound* a aplicação cuja execução é altamente dependente da CPU, geralmente realizando pouca comunicação entre *threads* e poucas operações de I/O. Fazem parte deste grupo as aplicações BT e LULESH. Ao observar quais são as piores configurações presentes na Figura 5.1a, pode-se concluir que a aplicação BT se encaixa como *CPU-bound*, pois aumentar a frequência do *uncore* mantendo a frequência do *core* baixa ocasiona acentuada piora no valor da emissão. Portanto, é recomendado para essa aplicação uma combinação entre valores mais altos no *core* e mais baixos no *uncore*.

Em contrapartida, a aplicação LULESH, mesmo sendo considerada *CPU-bound*, exige comunicação frequente entre *threads* que computam diferentes partes da estrutura de dados utilizadas por ela, apresentando um comportamento mais balanceado que a an-

Figura 5.1: Mapas de calor da emissão em gCO_2 das aplicações *CPU-bound*



(a) BT (b) LULESH

Fonte: O Autor

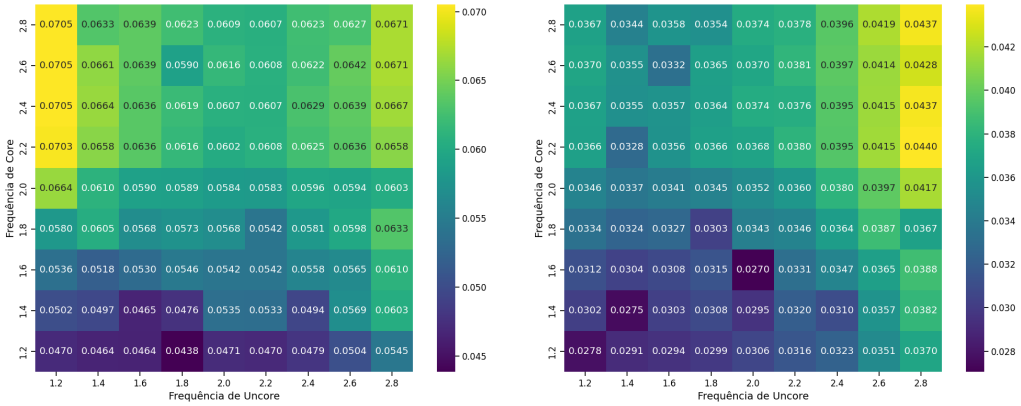
terior. Por isso e pela observação das regiões frias da Figura 5.1b, pode-se concluir que essa aplicação se beneficia de frequências de *uncore* maiores que as recomendadas para a outra.

5.2 Aplicações *Memory-Bound*

Grupo no qual a maioria das aplicações escolhidas se encaixam, é caracterizado pela predominante ocorrência de operações de I/O e comunicação entre *threads*, em detrimento de computações realizadas na CPU. Nele estão presentes as aplicações HPCG, Jacobi, CG, FT, MG e SA.

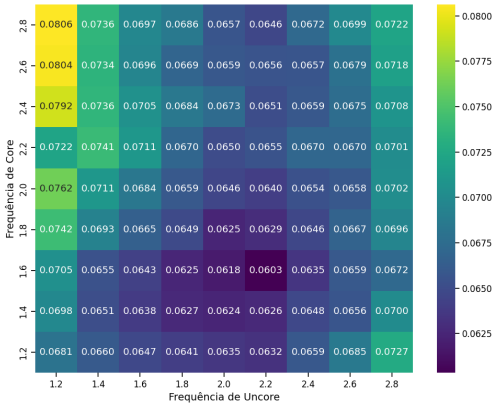
A análise dos mapas de calor das aplicações, presentes na Figura 5.2, evidencia que, para elas, menores frequências do *core* resultam em reduções significativas na emissão de carbono, o que é esperado devido à elevada quantidade de acessos à memória principal e intenso uso do barramento de comunicação *off-chip* pelas aplicações, caracterizando um comportamento *memory-intensive*. Além disso, a comunicação frequente entre *threads* via cache L3 beneficia-se de frequências mais elevadas no *uncore*, contribuindo ainda mais para a diminuição das emissões. Portanto, a combinação ideal para otimizar a pegada de carbono desse grupo ocorre em configurações com frequências reduzidas no *core* e frequências intermediárias ou elevadas no *uncore*.

Figura 5.2: Mapas de calor da emissão em gCO_2 das aplicações *memory-bound*

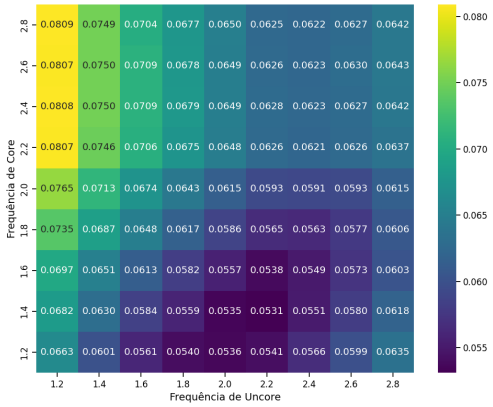


(a) HPCG

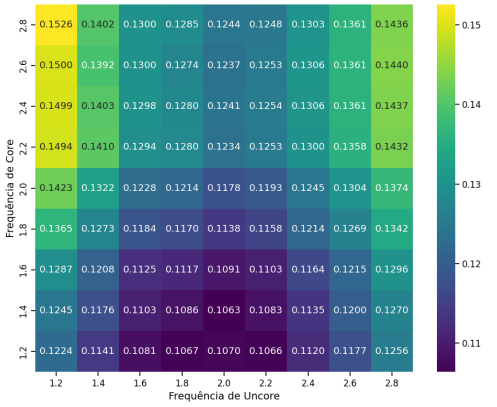
(b) Jacobi



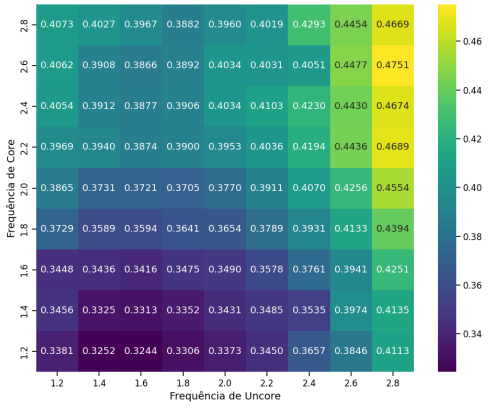
(c) CG



(d) FT



(e) MG



(f) SP

Fonte: O Autor

5.3 Aplicações com Perfil Balanceado

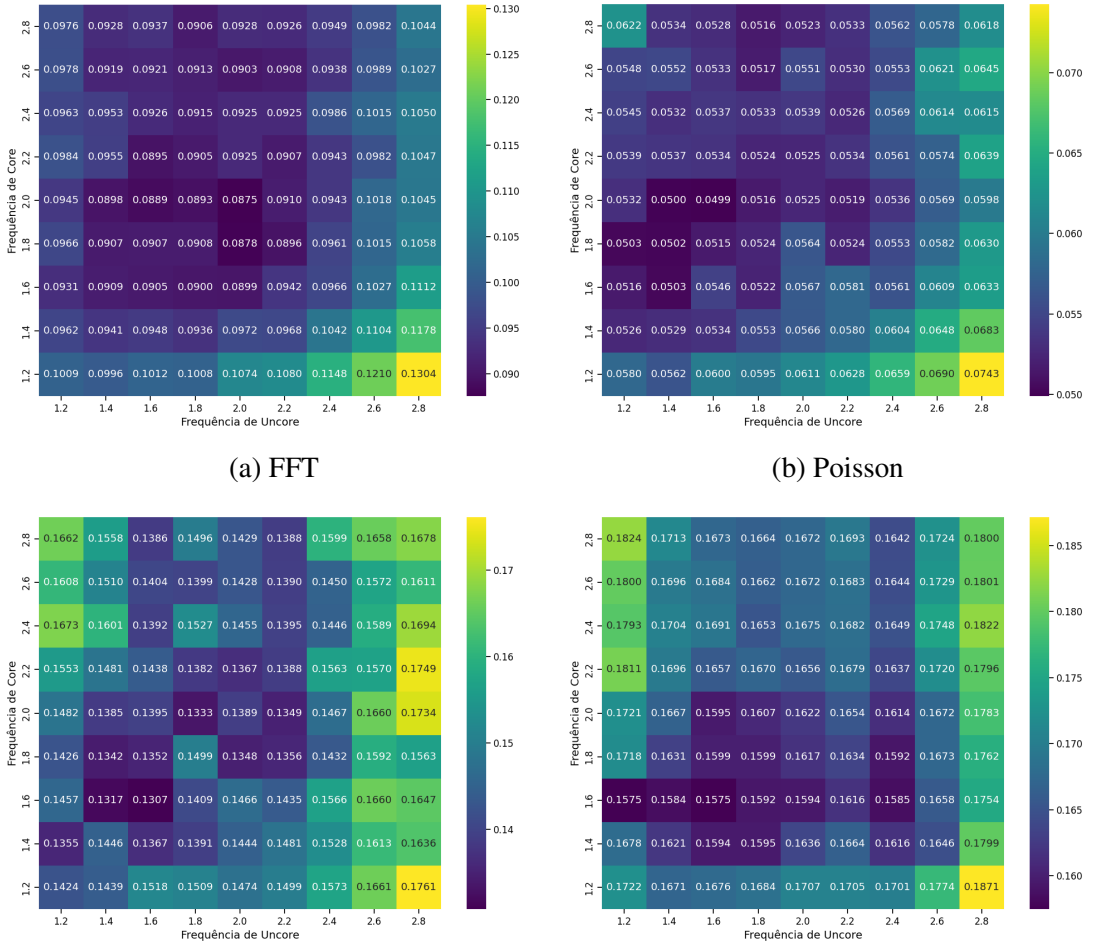
Esse grupo é caracterizado pelo equilíbrio entre computação e acessos à memória e é composto pelas aplicações FFT, LU, Poisson e UA. Seus mapas de calor estão presentes na Figura 5.3.

A aplicação FFT se enquadra neste grupo devido à transição gradual entre fases *CPU-intensive*, predominantes quando a estrutura de dados é pequena, e fases *memory-intensive*, que surgem à medida que o tamanho da estrutura de dados aumenta a cada iteração. Já as aplicações LU e UA podem ser encaixadas neste grupo, pois alternam entre períodos de intensa computação e períodos de comunicação (entre *threads* no caso da primeira e com a memória no caso da segunda).

Diferentemente das anteriores, a aplicação Poisson necessita de pouca comunicação entre *threads* de diferentes *cores* e faz poucos acessos à memória principal, o que justifica o uso de frequências para o *uncore* menores que nas outras do grupo. Além disso, embora a intensidade de computação seja maior que o uso de memória, não é alta o suficiente para torná-la *CPU-bound*, então pode ser considerada balanceada.

Com base nos perfis de emissão das aplicações, é recomendado o uso de frequências intermediárias para os subsistemas *core* e *uncore* ao executar aplicações deste grupo.

Figura 5.3: Mapas de calor da emissão em gCO_2 das aplicações de uso balanceadas



(a) FFT (b) Poisson

(c) LU (d) UA

Fonte: O Autor

5.4 Discussão Geral

Ao analisar os dados da Tabela 5.1, que apresenta uma comparação entre a pegada de carbono da configuração ótima com daquela configuração padrão que visa maximizar o desempenho, conclui-se que o grupo que apresentou as maiores reduções foi o das aplicações *memory-bound*. Isso se deve ao fato de que essas aplicações fazem muito acessos à memória, causando frequentes períodos de ócio da CPU. Utilizar frequências altas (principalmente de *core*) nesses períodos, aumenta a potência dissipada sem que haja uma diminuição no tempo de execução que contrabalanceie esse aumento, causando aumento no consumo de energia, e, por consequência, na pegada de carbono. Corroborando essa conclusão, é possível observar que o ponto correspondente a configuração padrão *(2,8; 2,8)* faz parte da zona quente em todos os gráficos da Figura 5.2 e também que o foco da zona fria deles, que representa a configuração ótima, está distante daquele ponto.

Em complemento a análise dos grupos de aplicações, obtêm-se conclusões interessantes ao observar o conjunto completo. Também utilizando a Tabela 5.1, é possível observar grandes reduções nas emissões - entre 12,5 e 38 por cento -, totalizando uma redução de 22,55% ao somar todas as execuções, o que salienta a importância de escolher corretamente a configuração para cada aplicação.

Tabela 5.1: Comparação da pegada de carbono entre a configuração ótima e a padrão.

Aplicação	Menor pegada de carbono em gCO_2	Pegada de carbono em gCO_2 do par de frequências (2.8, 2.8)	Decréscimo
BT	0.205329	0.248029	17.22%
LULESH	0.081795	0.094433	13.38%
CG	0.060328	0.072191	16.43%
FT	0.053097	0.064238	17.34%
HPPCCG	0.043837	0.067086	34.66%
Jacobi	0.027041	0.043658	38.06%
MG	0.106343	0.143606	25.95%
SP	0.324361	0.466918	30.53%
FFT	0.087512	0.104382	16.16%
Poisson	0.049872	0.061759	19.25%
LU	0.130712	0.167830	22.12%
UA	0.157457	0.180038	12.54%
Total	1.327684	1.714167	22.55%

Fonte: O Autor.

Tabela 5.2: Pares de configurações ótimas das aplicações e suas respectivas pegadas de carbono.

Aplicação	Par de frequências em GHz (*uncore*, *core*)	Pegada de carbono em gCO_2
BT	(1.4, 1.8)	0.205329
LULESH	(2.4, 2.0)	0.081795
CG	(2.2, 1.6)	0.060328
FT	(2.2, 1.4)	0.053097
HPPCCG	(1.8, 1.2)	0.043837
Jacobi	(2.0, 1.6)	0.027041
MG	(2.0,1.4)	0.106343
SP	(1.6, 1.2)	0.324361
FFT	(2.0, 2.0)	0.087512
Poisson	(1.6, 2.0)	0.049872
LU	(1.6, 1.6)	0.130712
UA	(1.6, 1.6)	0.157457

Fonte: O Autor

Focando mais na escolha da configuração ótima, ao analisar a Tabela 5.2, que lista quais são os pares de configurações de frequência de *core* e *uncore* responsáveis pelos melhores resultados, conclui-se que a configuração ótima varia de aplicação para aplicação, o que é esperado devido às especificidades de cada uma delas.

Outro aspecto importante que pode ser concluído analisando os mapas de calor é que algumas aplicações são mais sensíveis à variação da frequência de operação que outras, observando o tamanho da zona fria dos gráficos. Por exemplo, a Figura 5.2a apresenta uma minúscula zona fria, concentrada ao redor do ponto *(1,8; 1,2)*, explicitando que existem poucas configurações de frequência que minimizam a pegada de carbono da aplicação HPCG. Em comparação, a Figura 5.1a mostra uma zona fria bastante ampla que se estende por toda a lateral esquerda do gráfico, provando que para a aplicação BT, desde que se mantenha a frequência do *uncore* intermediária ou menor, pode-se utilizar qualquer configuração e obter resultados satisfatórios.

6 CONCLUSÃO E TRABALHOS FUTUROS

Ao analisar o impacto que a variação da frequência de operação dos subsistemas *core* e *uncore* tem na emissão de carbono de sistemas computacionais, foi possível evidenciar que a escolha das configurações de frequência tem um papel central na pegada de carbono ao executar as aplicações, alcançando reduções entre 12,5 e 38 por cento ao comparar a configuração ótima encontrada com a configuração padrão que utiliza as maiores frequências possíveis. Além disso, pôde-se identificar que diferentes perfis de uso *CPU-bound*, *memory-bound* e balanceadas apresentaram padrões distintos de configuração ótima, demonstrando que estratégias específicas para cada caso são essenciais para minimizar o impacto causado pelos sistemas de computação. Também observou-se que o impacto da variação das frequências é mais severo em algumas aplicações do que em outras, ou seja, algumas aplicações apresentaram bastantes configurações com emissões semelhantes à ótima e outras não.

Trabalhos futuros poderiam enriquecer os resultados encontrados por esta monografia de diversas maneiras. Ampliar o número de aplicações executadas, escolhendo principalmente as que apresentem perfil *CPU-bound* (grupo com menos integrantes dentre as escolhidas), pode adicionar mais robustez aos resultados. O tempo de execução das aplicações é um dos fatores para calcular a energia gasta pela tarefa de computação, então esse parâmetro não foi ignorado, mas, outra abordagem interessante seria analisar de maneira ponderada tanto a emissão de carbono quanto o tempo de execução da aplicação, tornando a configuração ótima àquela que apresente baixa pegada de carbono aliada a tempo de execução satisfatório.

REFERÊNCIAS

AMNUAYLOJAROEN, T. Perspective on the era of global boiling: A future beyond global warming. **Advances in Meteorology**, Wiley Online Library, v. 2023, n. 1, p. 5580606, 2023.

ANTHONY, L. F. W.; KANDING, B.; SELVAN, R. Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. **CoRR**, abs/2007.03051, 2020. Available from Internet: <https://arxiv.org/abs/2007.03051>.

DIXON, M. et al. **Intel Technology Journal, Volume 14, Issue 3**. 2010. Acessado em: 19 dez. 2024. Available from Internet: <https://www.intel.com/content/dam/www/public/us/en/documents/research/2010-vol14-iss-3-intel-technology-journal.pdf>.

DONGARRA, J.; HEROUX, M. A.; LUSZCZEK, P. High-performance conjugate-gradient benchmark: A new metric for ranking high-performance computing systems. **The International Journal of High Performance Computing Applications**, 2015.

DOU, H. et al. Carbon-aware electricity cost minimization for sustainable data centers. **IEEE Transactions on Sustainable Computing**, v. 2, n. 2, p. 211–223, 2017.

FRIGO, M.; JOHNSON, S. Fftw: an adaptive software architecture for the fft. In: **Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181)**. [S.l.: s.n.], 1998. v. 3, p. 1381–1384 vol.3.

GUPTA, U. et al. Chasing carbon: The elusive environmental footprint of computing. In: **2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)**. [S.l.: s.n.], 2021. p. 854–867.

GUPTA, V. et al. The forgotten 'uncore': On the energy-efficiency of heterogeneous cores. In: . [S.l.: s.n.], 2012. p. 34–34.

HAINES, A. et al. Climate change and human health: impacts, vulnerability, and mitigation. **The Lancet**, Elsevier, v. 367, n. 9528, p. 2101–2109, 2006.

KARLIN, I. Lulesh programming model and performance ports overview. 12 2012. Available from Internet: <https://www.osti.gov/biblio/1059462>.

KWEKU, D. W. et al. Greenhouse effect: greenhouse gases and their impact on global warming. **Journal of Scientific research and reports**, v. 17, n. 6, p. 1–9, 2018.

LANNELONGUE, L.; GREALEY, J.; INOUYE, M. Green algorithms: Quantifying the carbon footprint of computation. **Advanced Science**, v. 8, n. 12, p. 2100707, 2021. Available from Internet: <https://onlinelibrary.wiley.com/doi/abs/10.1002/advs.202100707>.

MAAS, W. et al. An ann-guided multi-objective framework for power-performance balancing in hpc systems. In: **Proceedings of the 21st ACM International Conference on Computing Frontiers**. [S.l.: s.n.], 2024. p. 138–146.

NAGEL, J. R. Numerical solutions to poisson equations using the finite-difference method [education column]. **IEEE Antennas and Propagation Magazine**, v. 56, n. 4, p. 209–224, 2014.

RADOVANOVI, A. et al. Carbon-aware computing for datacenters. **IEEE Transactions on Power Systems**, v. 38, n. 2, p. 1270–1280, 2023.

STALLINGS, W. **Computer Organization and Architecture: Designing for Performance**. 10. ed. Boston: Pearson, 2016.

WANG, S. et al. Application configuration selection for energy-efficient execution on multicore systems. **Journal of Parallel and Distributed Computing**, v. 87, p. 43–54, 2016. ISSN 0743-7315. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0743731515001689>.

ZHOU, Z. et al. Carbon-aware online control of geo-distributed cloud services. **IEEE Transactions on Parallel and Distributed Systems**, v. 27, n. 9, p. 2506–2519, 2016.