|  |  |
| --- | --- |
| CURSO DE CIÊNCIA DA COMPUTAÇÃO – TCC | |
| ( ) PRÉ-PROJETO     (  X ) PROJETO | ANO/SEMESTRE: 2024/1 |

**SYSPOWER ANALYZER: FERRAMENTA DE MEDIÇÃO DO CUSTO ENERGÉTICO EM EXECUÇÕES DE ROTINAS SISTÊMICAS**

Matheus Felipe da Silva Sychocki

Prof. Danton Cavalcanti Franco Junior – Orientador

# Introdução

Hoje em dia, os softwares desempenham um papel cada vez mais central em nossas vidas, integrando-se em nossos smartphones, eletrodomésticos e veículos. De acordo com Matos, Coelho, Carapeto (2014), a evolução humana não se limita ao aspecto biológico; a tecnologia desempenha um papel importante na moldagem de nossos hábitos, pensamentos e ações. A computação, em particular, representa a última grande revolução que alterou profundamente nossa forma de viver. Inicialmente, houve avanços no hardware, seguidos pelo desenvolvimento do software, que criou uma existência virtual comandada e intermediada por entidades de software.

Entretanto, essa crescente presença de softwares na nossa vida não é isenta de preocupações. Segundo o World Economic Forum (2023), espera-se que 70% dos valores econômicos criados na próxima década se baseiam em plataformas empresariais digitais. Isso é uma tendência preocupante, uma vez que o consumo de energia associado aos softwares contribui significativamente para as mudanças climáticas, devido à alta emissão de CO2 na atmosfera. Estima-se que o setor de Tecnologia da Informação e Comunicação (TIC) seja responsável por 2 a 10% das emissões globais de CO2, comparável e, em alguns casos, até superior à emissão de CO2 da aviação.

O consumo energético é uma questão crítica, especialmente considerando que a maioria da energia é gerada a partir de combustíveis fósseis, como petróleo, carvão mineral e gás natural, responsáveis por 64% das emissões globais de gases de efeito estufa, de acordo com Miranda (2014). Diante deste cenário, é essencial abordar o desenvolvimento sustentável no setor de TI, adotando técnicas para otimizar o consumo energético e computacional e, assim, reduzir o impacto ambiental.

Segundo Noureddine *et al.* (2015)o aumento dos custos de energia em computadores e dispositivos móveis exige a otimização e a adaptação dos sistemas de computação. Pesquisas em GreenIT já propõem diversas estratégias para economizar energia em computadores e software. Uma condição essencial para atingir essas economias é monitorar o consumo de energia do sistema. No entanto, muitas abordagens modernas se concentram apenas no hardware ou fornecem feedbacks de energia superficiais para o software.

Nesse contexto, há uma escassez de ferramentas acessíveis para medir o consumo de energia de um software. As ferramentas disponíveis atualmente requerem um alto nível de conhecimento técnico, tornando-as inacessíveis para pessoas com pouca familiaridade com o funcionamento básico de um computador. Assim, este trabalho tem como objetivo desenvolver um método para extrair dados sobre o consumo de energia de um ou mais softwares em um computador e apresentar essas informações de forma clara e simplificada para o usuário final, sem a necessidade de conhecimento técnico prévio.

## OBJETIVOS

O objetivo deste trabalho é construir uma ferramenta que realize a métrica de consumo de energia de software e apresente de forma clara e simplificada para o usuário final.

Os objetivos específicos são:

1. construir uma ferramenta intuitiva para que pessoas com menor conhecimento técnico possa utilizar;
2. quantificar o consumo energético de um software;
3. apresentar informações interpretáveis e valiosas sobre o consumo de energia do software;
4. contribuir para o avanço da computação verde.

# trabalhos correlatos

Nesta seção serão apresentados três trabalhos que contribuem para o trabalho proposto. A subseção 2.1 apresenta Monitoring Energy Hotspots in Software elaborado por Noureddine *et al.* (2015)**.** A subseção 2.2 apresenta Monitoring Performance and Power for Application Characterization with the Cache-aware Roofline Model proposto por Antão *et al.* (2014). A subseção 2.3 apresenta SchedMon: A Performance and Energy Monitoring Tool for Modern Multi-cores desenvolvido por Taniça *et al.* (2014)

## Monitoring Energy Hotspots in Software

Noureddine *et al.* (2015) propõem um framework que visa monitorar e modelar a variação de energia consumida por métodos de software com base em seus parâmetros de entrada. Para alcançar esse objetivo, eles utilizam uma abordagem combinada de técnicas de perfilamento detalhado, monitoramento em tempo real e modelagem empírica, visando identificar tendências de consumo de energia, detectar pontos quentes (*hotspots*) de energia e fornecer informações relevantes para otimizar o desempenho energético das aplicações.

Segundo os autores, uma das vantagens significativas do framework é a sua capacidade de realizar um perfilamento detalhado, que permite identificar métodos individuais com alto consumo de energia, proporcionando uma visão aprofundada dos possíveis pontos quentes de energia em uma aplicação. Por exemplo, em um estudo sobre o consumo de energia de um programa de Torres de Hanói, o framework revelou que 88% do consumo de energia foi atribuído à Unidade Central de Processamento (Central Processing Unit - CPU), enquanto 12% foram atribuídos ao disco rígido conforme apresentado na Figura 1. Isso destaca que, mesmo em aplicações simples, podem surgir desafios significativos relacionados ao consumo de energia.

Figura 1 – Consumo de energia da CPU e do disco rígido do algoritmo das Torres de Hanói.A pie chart with a blue triangle

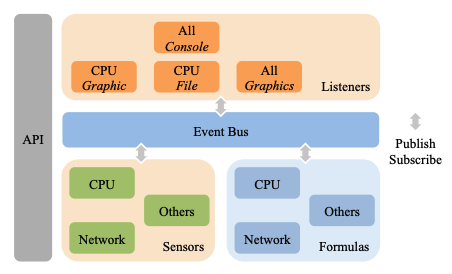
Description automatically generated

Fonte: Noureddine *et al.* (2015).

Além disso, o framework apresenta flexibilidade, adaptando-se para diferentes cenários de hardware e software, monitorando diversos componentes, como CPU, disco rígido e rede, para oferecer uma visão abrangente do consumo energético em uma aplicação. Por exemplo, ele pode ser usado para monitorar dispositivos com diversas configurações de hardware e sistemas operacionais, como Windows ou MacOS. A capacidade de adaptação torna o framework útil em diferentes ambientes de execução, desde servidores até dispositivos móveis.

Segundo Noureddine *et al.* (2015) a arquitetura do PowerAPI é modular, com cada componente representado como um módulo de energia (Figura 2). Utiliza um barramento de eventos com o paradigma *publish/subscribe* para a comunicação entre módulos. Os módulos de sensores coletam informações do sistema operacional e do hardware. Por exemplo, o sensor de CPU monitora o tempo gasto pela CPU em cada frequência do processador e publica esses dados para todos os módulos ouvintes no barramento de eventos.

Figura 2 – Arquitetura de Referência ao framework proposto



Fonte: Noureddine *et al.* (2015).

A abordagem empírica adotada pelo framework é crucial para avaliar o impacto de diferentes valores de parâmetros de entrada nos métodos, fornecendo uma visão realista do consumo de energia. Isso permite aos desenvolvedores entender como mudanças nos parâmetros de entrada podem afetar o consumo de energia e, consequentemente, tomar decisões mais informadas sobre a otimização do código. Por exemplo, a variação de energia baseada em parâmetros de entrada pode ser observada em diferentes cenários de aplicação, mostrando o potencial de otimização.

O framework também utiliza modelos eficientes para calcular o consumo de energia com base nos recursos utilizados, como CPU, disco rígido e rede, durante a execução dos métodos. Essa abordagem baseada em modelos facilita a análise dos dados e ajuda a identificar oportunidades de otimização na Figura 3 é apresentado a modelo matemático para calcular o custo energético de um software. A análise do consumo de energia do programa de Torres de Hanói revelou, por exemplo, que o acesso à memória também pode ter um impacto significativo na eficiência energética.

Figura 3 – Fórmula utilizada para calcular o custo energético de um software



Fonte: Noureddine *et al.* (2015).

Os resultados iniciais do framework indicam que é possível identificar métodos e classes que consomem a maior parte da energia em uma aplicação de software. A análise das variações de energia com base em diferentes valores de parâmetros de entrada revelou informações úteis sobre os métodos mais intensivos em energia. Esta capacidade de avaliar a influência de diferentes componentes de hardware, como CPU, disco rígido e rede, forneceu insights importantes sobre o desempenho energético da aplicação.

## Monitoring Performance and Power for Application Characterization with the Cache-aware Roofline Model

Antão *et al.* (2014)proporão dois métodos de monitoramento que combinam as vantagens do modelo Cache-Aware Roofline com facilidades precisas de monitoramento em tempo real, possibilitando aos desenvolvedores de aplicativos relacionarem o comportamento do aplicativo com as características da arquitetura. As duas ferramentas propostas, SpyMon e KerMon, baseiam-se em contadores de medição de desempenho de hardware (Hardware Performance Measurement Counters - HPMCs) e fornecem informações sobre características de desempenho e consumo de energia em tempo real para aplicativos em execução.

Segundo os autores, SpyMon é uma ferramenta de monitoramento leve que pode ser usada no espaço do usuário. Permite monitorar qualquer processo ou *thread* em um modo orientado ao núcleo, independentemente do processo que lançou os *threads* em execução. A ferramenta é configurável em tempo real e pode plotar os eventos coletados no modelo Cache-Aware Roofline para visualizar as limitações arquitetônicas e de desempenho dos aplicativos. Na Figura 4 é apresentado uma percepção de como fica atrelados os monitoramentos os threads de um processado, A leitura eficiente e precisa dos eventos é realizada por meio do MSRDriver, que acessa diretamente os contadores de eventos nos MSRs (Registros Específicos de Modelo).

Figura 4 – Percepção espacial do SpyMon durante o monitoramento de 5 threads de 3 aplicaçõesA diagram of a computer program

Description automatically generated

Fonte: Noureddine et al. (2015).

Segundo os autores, a ferramenta KerMon é projetada para ser usada no espaço do kernel e oferece uma visão mais precisa e detalhada do comportamento do aplicativo em execução. Ela permite a leitura direta dos HPMCs no espaço do kernel, proporcionando maior controle e precisão sobre o monitoramento de eventos de desempenho. KerMon é configurável para medir eventos específicos de interesse, como ciclos de *clock*, instruções aposentadas, falhas de predição de ramos e falhas de cache. A ferramenta pode ser configurada para coletar eventos em intervalos específicos ou com base em eventos específicos, com dados armazenados para posterior análise, na Figura 5 é apresentado um fluxograma de como é executado o algoritmo de escolha de tarefas.

Figura 5 – Algoritmo para escolher uma tarefa a ser executada

A diagram of a schedule

Description automatically generated

Fonte: Noureddine *et al.* (2015).

Os autores concluíram que os resultados experimentais apresentados mostram que tanto KerMon quanto SpyMon obtêm resultados similares de caracterização de desempenho. No entanto, o SpyMon, que realiza a análise orientada a núcleo, pode aumentar o consumo de energia se configurado para monitorar núcleos que não estão sendo utilizados por nenhum dos aplicativos em execução. Por outro lado, como o KerMon exige alterações no escalonador do sistema operacional, é complexo de instalar em um sistema e requer o acesso *root*. Apesar dessas diferenças, em geral, ambos os métodos de monitoramento permitem ao usuário/programador obter uma visão clara do comportamento do aplicativo e de como sua execução é afetada pelas limitações arquitetônicas do processador.

## SchedMon: A Performance and Energy Monitoring Tool for Modern Multi-cores

Taniça *et al.* (2014) proporão uma ferramenta de monitoramento de desempenho e energia direcionadas a aplicações, o SchedMon que visa criar perfis de aplicações complexas com multi-threading aninhado, abrangendo a execução total ou níveis de funções específicas conforme definido pelo usuário. O SchedMon obtém precisão nas medições ao interagir com o escalonador do sistema operacional, garantindo o isolamento dos threads externos. Para minimizar o overhead nas aplicações monitoradas, o SchedMon opera com dois componentes principais: um módulo de kernel do Linux ou driver, que implementa suas funcionalidades básicas, e uma ferramenta em espaço de usuário (Smon), que traduz as capacidades do driver em uma interface de usuário intuitiva. Onde essa ferramenta só pode ser utilizada em sistemas operacionais Linux

Segundo os autores, a comunicação entre o espaço de usuário e o driver ocorre por meio de mapeamento de memória e chamadas de sistema específicas (mmap e ioctl) para o dispositivo do SchedMon, conforme apresentado na (Figura 6), que representa um arquivo em /dev e aciona funções do driver ao receber solicitações. O SchedMon também utiliza um buffer de anel mapeado na memória para compartilhar informações de monitoramento entre espaços de usuário e de kernel. Amostras de desempenho e consumo de energia são obtidas pelo módulo de kernel do Linux, que interage com as facilidades de hardware.

Figura 6 – Interação e disposição dos componentes do SchedMon nas camadas de privilégio do sistema operacional.

A diagram of a computer system

Description automatically generated

Fonte: Noureddine *et al.* (2015).

Ao combinar as funcionalidades do driver e da ferramenta Smon, o SchedMon possibilita análises de desempenho inovadoras baseadas no Modelo Roofline Consciente de Cache (CARM) e rastreamento de chamadas de função, facilitando a avaliação de desempenho e identificação de gargalos, conforme destacado por Taniça et al. (2014).

Segundo os autores, o Smon é a interface de usuário do SchedMon, que oferece uma maneira simples e intuitiva de interagir com a ferramenta de monitoramento na figura 7 é apresentado o processo de rastreamento de uma chamada de função. Ele facilita a configuração e o início do monitoramento de desempenho e energia/potência para aplicações específicas, seja em sua totalidade ou no nível de funções específicas, além de permitir o monitoramento por thread em ambientes de multi-threading. O Smon oferece suporte ao CARM e ao rastreamento de chamadas de função, permitindo uma avaliação precisa do desempenho da aplicação e identificação de gargalos. Também inclui ferramentas para análise de resultados e visualização de dados, promovendo uma análise detalhada do comportamento da aplicação.

Figura 7 – Processo de instrumentação de rastreamento de chamadas de função.A close-up of a computer code

Description automatically generated

Fonte: Taniça *et al.* (2014).

Para lidar com a complexidade das arquiteturas de processadores modernos foi utilizado (CARM) que aborda a complexidade das arquiteturas de processadores modernos, considerando o impacto de diferentes aplicações executadas simultaneamente em múltiplos núcleos. O CARM oferece uma visão gráfica única das limitações práticas e do desempenho máximo de arquiteturas modernas multi-núcleo. O modelo relaciona o desempenho máximo de ponto flutuante (Flops/s), a intensidade operacional (Flops/byte) e a largura de banda máxima de memória para cada nível de cache na hierarquia de memória (Bytes/s). Assim, o CARM considera o tráfego de dados tanto nos domínios de memória *on-chip* quanto *off-chip*, conforme percebido pelo núcleo.

A ferramenta de monitoramento de desempenho e energia orientada a aplicações SchedMon oferece uma solução eficaz para caracterizar o comportamento de aplicações em tempo real. Combinando monitoramento de desempenho e energia com análises avançadas baseadas em CARM e rastreamento de chamadas de função. O SchedMon permite uma avaliação precisa e detalhada do desempenho de aplicações. Resultados experimentais mostram que a ferramenta é precisa e eficiente, com *overheads* mínimos, capaz de identificar interferências entre várias aplicações em execução simultaneamente. Assim, segundo os autores, o SchedMon é uma ferramenta valiosa para programadores e arquitetos de computadores em busca de otimizações de desempenho e eficiência energética.

# proposta de ferramenta

Esta seção abordará a proposta de ferramenta, justificando sua criação, definindo os requisitos funcionais e não funcionais a serem aplicados, e a metodologia e cronograma a serem seguidos.

## JUSTIFICATIVA

No Quadro 1 é apresentado um comparativo entre os trabalhos correlatos. As linhas representam características e as colunas dos trabalhos

Quadro 1 - Comparativo dos trabalhos correlatos

|  |  |  |  |
| --- | --- | --- | --- |
| Trabalhos Correlatos  Características | Noureddine *et al.* (2015) | Antão *et al.* (2014) | Taniça et al. (2014) |
| Escopo de Monitoramento | Variação de energia consumida por métodos de software | Monitoramento do comportamento do aplicativo com base no modelo Cache-Aware Roofline. | Monitoramento de desempenho e energia em tempo real em aplicações complexas com multi-threading aninhado. |
| Sistemas Operacionais compatíveis | Windows, MacOs, Linux | Windows, MacOs, Linux | Linux |
| Ferramentas | PowerAPI | SpyMon e KerMon | Smon e Intel Ralp. |
| Metodologias | Perfilamento detalhado, monitoramento em tempo real, modelagem empírica. | Medição de desempenho de hardware (HPMCs) e visualização no Cache-Aware Roofline. | Modelo Roofline Consciente de Cache (CARM), rastreamento de chamadas de função. |
| Utiliza mais de um software | Não | Sim | Sim |

Fonte: elaborado pelo autor.

Conforme pode ser analisado no Quadro 1, os trabalhos apresentados por Noureddine *et al.* (2015) e Antão *et al.* (2014) possui o mesmo objetivo, apresentar o consumo energético de um software ou processo sistêmico, que por sua vez pode gerar maior aprimoramento no uso de software ou até mesmo na implementação de rotinas sistêmicas. Entretanto os autores utilizam de técnicas distintas para realizar a análise das informações obtidas. Noureddine *et al.* (2015) faz uso de uma análise mais empírica utilizando cálculos de templo x consumo dos ciclos de máquina para aferir o gasto energético. Já Antão *et al.* (2014) faz uso de um modelo (CARM), que gráfica o consumo e possui uma maior assertividade no contexto de um processador mais atual, onde é possível medir o consumo em processadores *multi-threading*. As três abordagens apresentadas acima não possuem uma integração direta com a extração das informações até a apresentação dos dados finais para o usuário e necessitam que o usuário tenha um conhecimento prévio de hardware e funcionamento do Sistema Operacional (SO) para que consiga replicar os testes apresentados.

A partir das características expostas, este trabalho se mostra relevante, tendo como foco trazer uma forma mais simplificada de realizar a métrica de consumo de energia de um software, onde não será necessário realizar a extração das informações com um software e análise dos dados obtidos em outro ou até mesmo análise das informações manualmente. Além disso, a implementação de uma solução mais eficiente para medir o consumo de energia pode contribuir significativamente para o meio ambiente, ao promover a otimização do uso de recursos energéticos e reduzir o desperdício, favorecendo práticas mais sustentáveis na área de desenvolvimento de software.

## REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

A ferramenta a ser desenvolvida deverá contemplar os seguintes Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF):

1. permitir que o usuário visualize o consumo de energia em software (RF);
2. permitir interação do usuário com o gráfico de consumo (RF);
3. permitir que o usuário visualize o tempo que foi feito o monitoramento (RF);
4. permitir que o usuário visualize os aplicativos que foram medidos (RF);
5. permitir o usuário visualizar o consumo do sistema operacional (RF);
6. salvar os dados obtidos em um arquivo local (RNF);
7. ser desenvolvido para plataforma MacOS (RNF);
8. ser desenvolvido na versão NET 9 (RNF);
9. ser utilizado na arquitetura ARM (RNF);
10. utilizar o PowerApi para extração das informações (RNF).

## METODOLOGIA

O trabalho será desenvolvido observando as seguintes etapas:

1. levantamento bibliográfico: pesquisar e estudar sobre modelos matemáticos, coleta de dados do hardware e sistema operacional, e trabalhos correlatos;
2. elicitação de requisitos: baseado no levantamento bibliográficos e nos objetivos do trabalho, reavaliar e/ou agregar novos requisitos;
3. estabelecimento de conexão com o PowerAPI: Este passo envolve o desenvolvimento de uma comunicação direta com a biblioteca PowerAPI;
4. limpeza de dados: após a coleta dos dados, é importante desenvolver uma rotina para realizar a limpeza e conversão dos dados coletados para um formato utilizável. Isso inclui a filtragem de dados redundantes ou irrelevantes, normalização de valores e conversão de formatos de dados para facilitar a análise;
5. processamento de dados: neste passo, uma rotina deve ser criada para processar os dados coletados, identificando padrões, tendências e anomalias nos dados. Isso pode incluir análise estatística, cálculo de métricas e geração de relatórios sobre o desempenho e consumo de energia do sistema monitorado;
6. armazenamento de dados: uma rotina para armazenar os dados coletados deve ser desenvolvida para gravar os dados em um formato estruturado em arquivos locais;
7. validação de métricas: esta etapa envolve a comparação das métricas de consumo de energia obtidas com o custo real na tomada elétrica. Isso ajuda a verificar a precisão dos dados coletados e validar as medições de consumo de energia;
8. interface com o usuário: este passo refere-se à criação de uma interface de usuário amigável para interagir com as ferramentas de monitoramento. A interface deve permitir ao usuário configurar as opções de monitoramento, visualizar os dados coletados e receber relatórios e alertas sobre o desempenho e consumo de energia do sistema.
9. testes com usuário final: realizar testes com pessoas que não possui um grau de conhecimento técnico em computação muito elevado, para apresentar a efetividade da ferramenta.

As etapas serão realizadas nos períodos relacionados no Quadro 2.

Quadro 2 - Cronograma

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 2024 | | | | | | | | | |
|  | jul. | | ago. | | set. | | out. | | nov. | |
| etapas / quinzenas | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 |
| levantamento bibliográfico |  |  |  |  |  |  |  |  |  |  |
| elicitação de requisitos |  |  |  |  |  |  |  |  |  |  |
| estabelecimento de conexão com o PowerAPI |  |  |  |  |  |  |  |  |  |  |
| limpeza de dados |  |  |  |  |  |  |  |  |  |  |
| processamento de dados |  |  |  |  |  |  |  |  |  |  |
| armazenamento de dados |  |  |  |  |  |  |  |  |  |  |
| validação de métricas |  |  |  |  |  |  |  |  |  |  |
| interface com o usuário |  |  |  |  |  |  |  |  |  |  |
| testes com usuário final |  |  |  |  |  |  |  |  |  |  |

Fonte: elaborado pelo autor.

# REVISÃO BIBLIOGRÁFICA

Esta seção discorre sobre os assuntos que fundamentarão o estudo a ser realizado, no caso Monitoramento Energético (ME), Modelos de Energia (ME) e PowerAPI.

## Monitoramento energético

O Monitoramento de Energia (ME) é essencial para compreender e otimizar o consumo de energia em sistemas computacionais. Segundo Silva, Brito e Filho (2013), o ME utiliza instrumentos que, por meio de amostragem, gráficos e dados, fornecem valores numéricos das grandezas monitoradas, permitindo um controle preciso do consumo energético.

A implementação do Monitoramento de Energia não apenas quantifica o consumo, mas também identifica oportunidades para melhorias na eficiência energética. Os dados obtidos por meio do ME são cruciais para avaliar o desempenho ao longo do tempo e ajustar estratégias de otimização, conforme destacado por Silva, Brito e Filho (2013), contribuindo para práticas mais sustentáveis no ambiente computacional.

## Modelos de energia

Segundo Noureddine et al. (2015), modelos de energia são representações matemáticas ou computacionais que estimam o consumo de energia em sistemas eletrônicos. Eles podem atuar em alto ou baixo nível de software e têm bastante aplicabilidade para identificar áreas de alto consumo de energia, otimizar recursos e desenvolver soluções mais eficientes do ponto de vista energético.

Noureddine et al. (2015) afirma que os modelos para entender como CPUs consomem energia em diferentes condições de carga e configurações de hardware, considerando (Dynamic Voltage and Frequency Scaling - DVFS), voltagem e frequência da CPU. Esses modelos oferecem insights precisos sobre consumo energético, essenciais para a gestão térmica, eficiência de refrigeração e estratégias de economia de energia em ambientes computacionais variados.

## Powerapi

Noureddine et al. (2015) apresentam o PowerAPI como uma biblioteca de sistema que oferece uma interface de programação (API) para monitorar o consumo de energia de software em tempo real, com detalhamento por processo no sistema operacional. Essa ferramenta permite uma estimativa precisa do consumo de energia, oferecendo uma alternativa eficiente aos medidores de energia de hardware, com maior granularidade e facilidade de integração em ambientes de desenvolvimento modernos.

Ao fornecer visibilidade detalhada do consumo de energia por processo, o PowerAPI capacita os desenvolvedores a identificar e mitigar ineficiências energéticas, tornando-se uma ferramenta valiosa para a otimização da eficiência energética em sistemas computacionais. Essa abordagem não apenas simplifica a análise de consumo de energia, mas também promove práticas sustentáveis e econômicas no desenvolvimento de software.

Referências

ANTÃO Diogo, TANICA Luís, ILIC Aleksandar, PRATAS Frederico, TOMÁS Pedro, and SOUSA Leonel. **Monitoring Performance and Power for Application Characterization with the Cache-aware Roofline Model**. 2014. Disponível em: https://www.inesc-id.pt/ficheiros/publicacoes/9239.pdf. Acesso em: 15 de setembro de 2023.

MIRANDA, Mariana Maia de. **Fator de emissão de gases de efeito estufa da geração de energia elétrica no Brasil: implicações da aplicação de Avaliação do Ciclo de Vida.** Dissertação (Mestrado em Ciências da Engenharia Ambiental), p. 19 - 22, 14 de março de 2014. Disponível em: https://www.teses.usp.br/teses/disponiveis/18/18139/tde-22012013-112737/publico/DissertacaoMarianaMaiaDeMiranda.pdf. Acesso em: 15 de setembro de 2023.

MATOS, Paulo; COELHO, José; CARAPETO, Cristina. **Testes de Software na Redução do Consumo Energético dos Sistemas de Informação**. Revista de Ciências da Computação, v. 9, p. 63-77, janeiro 2014. Disponível em: https://repositorioaberto.uab.pt/bitstream/10400.2/3870/1/RCC\_2014.pdf. Acesso em 03 de setembro de 2023.

NOUREDDINE Adel, ROUVOY Romain, SEINTURIER Lionel. **Monitoring Energy Hotspots in Software**. 2015. Disponível em: https://inria.hal.science/hal-01069142/document. Acesso em: 23 de abril de 2024.

NOUREDDINE Adel, ROUVOY Romain, SEINTURIER Lionel. SchedMon: **A Performance and Energy Monitoring Tool for Modern Multi-cores**. 2014. Disponível em: https://www.inesc-id.pt/ficheiros/publicacoes/10247.pdf. Acesso em: 23 de abril de 2024.

SILVA, Jarbele C. da; BRITO, Alisson V.; FILHO, Gilberto F. de S. **Um estudo de caso sobre a avaliação energética de software como uma contribuição à Computação Verde**. Artigo, p. 12, 22 de maio de 2013. Disponível em: https://sol.sbc.org.br/index.php/sbsi/article/view/5723. Acesso em: 22 de setembro de 2023.

World Economic Forum. **World Economic Forum**. [2023]. Disponível em: https://intelligence.weforum.org/topics/a1Gb0000001SH21EAG. Acesso em: 02 out. 2023.