Capítulo 1

Computação Sustentável: da Teoria à Prática com Ferramentas para Medida de Consumo de Energia e Pegada de Carbono das Aplicações

Silvana Rossetto (UFRJ), Luiz Paulo Correa da Silva (UFRJ),
Daniel Cordeiro (USP), Alvaro Luiz Fazenda (UNIFESP),
Alessandro Santiago dos Santos (IPT), Emilio Francesquini (UFABC)

Abstract

Sustainable computing *is concerned with the socio-environmental impacts brought about by computing. Its practice focuses on the adoption of techniques that enable the development of computational solutions that are aware of energy consumption and environmentally sustainable. In this chapter, we will present the relevance and current challenges related to the concept and practice of sustainable computing, its elements and scope, associated techniques, tools, and metrics. We hope that this text emphasizes the importance of the topic, provides theoretical and practical foundations for it, and encourages the reader to use the techniques and tools presented.*

Resumo

A área de estudo ***Computação Sustentável*** *preocupa-se com os impactos socioambientais decorrentes da computação. Sua prática volta-se à adoção de técnicas que permitam o desenvolvimento e o monitoramento de soluções computacionais cientes do consumo energético e ambientalmente sustentáveis. Neste capítulo, apresentaremos a relevância e os desafios atuais da prática de computação sustentável, seus elementos e sua abrangência, bem como as técnicas, ferramentas e métricas associadas. Espera-se que este texto chame a atenção para a importância do tema, forneça fundamentação teórica e prática e estimule o leitor a aprofundar o domínio das técnicas e ferramentas apresentadas.*

Este capítulo foi realizado com recursos da FAPESP, processos 23/00811-0 (“EcoSustain - Ciência de Dados e Computação para o Meio Ambiente”) e 24/01115-0 (“CCD - Cidades Carbono Neutro”); e da FAPERJ, processo E-26/210.101/2025 (“Computação Sustentável e Energeticamente Eficiente”).

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

1.1. Introdução

A crescente demanda por aplicações computacionais de alto desempenho e larga escala — que dependem, em particular, de infraestruturas de *data centers* para processamento e armazenamento de dados — tem chamado a atenção para o consumo de energia requerido e suscitado preocupação, tanto pelos custos financeiros quanto pelos impactos socioambientais associados. Dados apontam que *data centers* consumiram entre 1% e 2% da eletricidade global em 2025.[1] Observa-se também o crescente uso das tecnologias de informação e comunicação (TICs) para automatizar atividades em diversas áreas, entre elas, agricultura, indústria, transporte, comércio, comunicação social, e outras — atingindo diretamente tarefas cotidianas da sociedade moderna. [Launikas et al. 2024] chamam a atenção para a automatização das atividades do dia a dia das pessoas, o tempo acumulado de interação dos usuários com essas aplicações e a consequente responsabilidade individual pelo consumo de energia e pelo impacto ambiental das aplicações que desenvolvemos ou escolhemos usar.

Diante desse cenário, a Sociedade Brasileira de Computação aponta, entre seus grandes desafios para a década de 2025 a 2035, a necessidade de "redução do impacto socioambiental no desenvolvimento e uso de aplicações por meio de Computação Sustentável" [Santos and Wagner 2025][2] . Dentro do escopo desse desafio, as seguintes ações são recomendadas: (a) desenvolver algoritmos energeticamente eficientes; (b) incentivar métodos de programação conscientes; (c) criar novas formas de automação inteligente para apoiar o processo de desenvolvimento; (d) desenvolver hardware e software especializados; (e) incorporar sustentabilidade às métricas de desempenho para Computação de Alto Desempenho.

Construir arquiteturas de hardware e software, bem como aplicações computacionais cientes do consumo de energia, é um desafio e uma prática fundamental já conhecidos nos contextos de computação móvel e da Internet das Coisas (IoT). Nesses casos, o foco é minimizar o consumo de energia para reduzir o aquecimento dos dispositivos e maximizar o tempo de carga das pilhas e baterias. No contexto de aplicações que executam em computadores conectados todo o tempo à rede elétrica — desde aplicações cotidianas até aplicações de larga escala e de alto desempenho — a preocupação com o consumo de energia é mais recente. Essa preocupação decorre da necessidade percebida de minimizar os impactos econômicos e socioambientais gerados pelas próprias aplicações computacionais, incluindo a demanda energética tanto para executá-las quanto para manter a infraestrutura de execução necessária. Desse modo, o consumo de energia de um algoritmo ou aplicação computacional torna-se uma métrica relevante, que precisa ser contrabalançada com métricas mais gerais de tempo de execução e de consumo de memória.

A área de estudo **Computação Sustentável** preocupa-se, de forma ampla, com os impactos econômicos e socioambientais negativos decorrentes da própria Computação e volta-se ao estudo de técnicas e novas práticas que nos permitem mitigar esses impactos [Paul et al. 2023]. No seu escopo mais amplo, preocupa-se não apenas com o

[1]Revista Pesquisa FAPESP – Edição 349 – Março de 2025: `https://revistapesquisa.fapesp.br/os-impactos-ambientais-da-computacao/`.

[2]Grandes Desafios da Computação no Brasil 2025-2035: `https://doi.org/10.5753/sbc.17434.6`

consumo de energia das aplicações, mas com todo o ciclo de produção (projeto e desenvolvimento), utilização e descarte; nos contextos de hardware, software, infraestrutura de comunicação e de armazenamento. Sua prática envolve, entre outras ações: otimizar projetos de hardware e software; monitorar e ter ciência do consumo de energia e da pegada de carbono das aplicações; gerenciar o descarte; e usar, de forma combinada, fontes de energia tradicionais e renováveis.

Neste capítulo, nos atemos ao estudo de: (i) técnicas, ferramentas e métodos para medir o consumo de energia e a pegada de carbono das aplicações; e (ii) estratégias que vêm sendo usadas para minimizar essas métricas. Medir o consumo de energia e a pegada de carbono das aplicações não é uma tarefa trivial. Há vários desafios e diferentes alternativas de procedimento. A pegada de carbono (associada às emissões de CO_2) referente à execução de aplicações computacionais deve considerar todo o ciclo de vida das infraestruturas de hardware, software e comunicação necessárias à execução das aplicações, bem como a geração e transmissão de energia elétrica que sustentam essas infraestruturas. Levantar essa métrica requer, portanto, informações adicionais, para além da aplicação em questão. A partir da ciência dessas métricas, busca-se aplicar técnicas de modelagem e desenvolvimento de software para reduzir o consumo de energia e impacto socioambiental negativo das aplicações.

O restante do texto está organizado em mais quatro seções. A Seção 1.2 apresenta conceitos básicos, desafios e relevância atual da área de estudo da Computação Sustentável. A Seção 1.3 introduz as principais métricas de consumo de energia e de impacto ambiental de aplicações computacionais e apresenta exemplos de ferramentas que auxiliam no levantamento dessas métricas, com destaque para a interface RAPL e a ferramenta CodeCarbon. A Seção 1.4 apresenta brevemente abordagens que vêm sendo utilizadas no desenvolvimento e na execução de aplicações computacionais sustentáveis. Por fim, a Seção 1.5 completa o texto com considerações finais, demandas de pesquisa e aprofundamentos na área.

1.2. Computação Sustentável

Em um cenário mundial de evolução tecnológica crescente e de preocupação com questões ambientais e de sustentabilidade, os temas relacionados à visão de Computação Sustentável têm atraído cada vez mais a atenção e suscitado novas questões de pesquisa. [Phung et al. 2018] destacam que a necessidade de balancear requisitos de desempenho, armazenamento e conservação de energia se reflete em trabalhos de pesquisa que desenvolvem algoritmos de escalonamento cientes de energia [Hoffmann 2015], modelos de potência [Möbius et al. 2013, Phung et al. 2018] e técnicas para redução de consumo de energia em *data centers* [Lee and Zomaya 2012].

Construir soluções computacionais cientes do consumo de energia e de seu impacto ambiental e social torna-se cada vez mais necessário. Trata-se de uma área estratégica que deve viabilizar o desenvolvimento de sistemas de computação que lidam com problemas complexos e envolvem grandes volumes de dados, de forma eficiente, robusta e ambiental e socialmente sustentável. [Lottick et al. 2019] reforçam que "a pegada de carbono dos algoritmos deve ser medida e reportada de forma transparente; só assim os cientistas da computação assumirão um papel honesto e ativo na sustentabilidade ambi-

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

ental."

A Computação Sustentável dialoga com diversas áreas do conhecimento, uma vez que os desafios ambientais atuais são de escala global e interdisciplinares. Significativos avanços na tecnologia da informação e comunicação (TIC), ciência de dados e inteligência artificial nas duas últimas décadas trazem oportunidades para utilizar esses conhecimentos e tecnologias em amplo benefício do meio ambiente.

Por outro lado, os avanços da Computação e o crescimento da complexidade dos problemas tratados geram demandas de consumo de energia significativas. Para frear ou reduzir esse consumo de energia, são necessários esforços em diversas direções, desde o desenvolvimento de soluções computacionais (hardware e software) de baixo consumo energético e alto desempenho até as atualizações na formação dos profissionais da área. Torna-se necessário fomentar e criar condições para a cultura de desenvolvimento de hardware e software cientes do consumo de energia e das emissões de CO_2 e, ao mesmo tempo, com suporte adequado para atender às demandas colocadas, sem implicar aumento significativo da carga de trabalho dos programadores nem custo econômico.

1.2.1. Desafios

Entre os desafios enfrentados atualmente para realizar Computação Sustentável, é possível destacar [Lottick et al. 2019, Paul et al. 2023]:

- **falta de padronização**: amplo espectro de dispositivos, configurações, tecnologias e protocolos, dificultando compatibilidade e interoperabilidade das ferramentas;
- **barreiras técnicas**: necessidade de lidar com *trade-offs* impostos entre diferentes métricas de avaliação das aplicações computacionais;
- **dependências externas**: necessidade de combinar informações externas para medir emissões de carbono;
- **recursos limitados**: poucos investimentos e esforços ainda nessa frente.

A falta de padronização implica a ausência de uniformidade nos métodos utilizados para avaliar a efetividade e a eficiência das tecnologias e soluções propostas para a Computação Sustentável, bem como a dificuldade de comparar resultados e tirar conclusões. Outro desafio associado é a ausência de consenso sobre melhores práticas para medir e reportar o consumo de energia dos sistemas de computação, o que dificulta a comparação da acurácia entre as diferentes soluções.

Em relação às barreiras técnicas, é preciso lidar com o *trade-off* entre eficiência energética e outras métricas de desempenho, como capacidade de processamento, vazão de dados, acurácia e disponibilidade, bem como atender às demandas da sociedade e das novas gerações, ao mesmo tempo em que se reduz o impacto socioambiental negativo.

Para reportar a emissão de CO_2 de uma aplicação, é preciso levar em conta o ciclo de vida dos dispositivos, determinar o local onde a aplicação é executada e o *mix* específico de fontes de energia que abastece este local enquanto a aplicação é executada.

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

A composição de fontes de energia que alimentam um sistema de computação é, em muitos casos, o fator mais relevante para determinadas emissões de carbono.

Por fim, a Computação Sustentável requer investimento adicional em hardware energeticamente eficiente e fontes de energia renováveis, além de treinamento e aprendizado adicional para o descarte e a implementação de práticas sustentáveis.

1.3. Como medir consumo de energia de aplicações computacionais

O consumo de energia normalmente é reportado por meio de medidas de energia ou de potência consumida. **Energia** é uma grandeza física escalar que mede a capacidade de um sistema de realizar trabalho ou fornecer calor. Sua unidade de medida no Sistema Internacional (SI) é o joule (J), definido como o trabalho realizado para manter uma corrente de 1 ampere através de uma resistência de 1 ohm durante o intervalo de tempo de exatamente 1 segundo. **Potência** é a taxa de energia consumida (ou gerada) em um dado intervalo de tempo (medida em watts). A medida de *1 watt* corresponde a uma taxa de *1 joule por segundo* (gerada ou consumida), ou seja, a potência refere-se à rapidez de geração ou consumo de energia. Calcula-se a *energia* a partir da *potência* multiplicando-se o valor da potência (gerada ou consumida) pelo intervalo de tempo (de geração ou de consumo). Por exemplo, o consumo de uma lâmpada de 100 watts por 5 segundos corresponde a $100 \times 5 = 500$ joules.

Medidas de energia elétrica são normalmente reportadas em quilowatt-hora (kWh) ou megawatt-hora (MWh). Um quilowatt-hora representa energia consumida a uma taxa de 1.000 watts por hora, o que corresponde a $1.000 \times 3.600 = 3.600.000$ joules. Medidas ambientais associadas à energia consumida são normalmente expressas como emissões por unidade de energia. Por exemplo, *quilogramas de CO_2* emitidos por *quilowatt-hora*.

Há diferentes métodos, com diferentes níveis de granularidade, para medir o consumo de energia. [Jay et al. 2023] propõem a seguinte categorização:

- **Dispositivos externos**: medidores externos posicionados entre o ponto de energia e a fonte de alimentação do computador. Fornecem dados sobre o consumo total de energia do computador.
- **Dispositivos internos**: medidores instalados dentro dos computadores. Podem medir o consumo de energia de componentes individuais.
- **Modelos de energia**: modelos matemáticos e computacionais que aproximam o consumo real.
- **Sensores de hardware e interfaces de software**: sensores digitais e circuitos integrados em componentes de hardware que medem consumo de energia em intervalos de tempo definidos.

O uso de medidores externos requer a aquisição e a implantação desses medidores separadamente e não oferece granularidade e acurácia adequadas para análises detalhadas. Estudos indicaram que a acurácia de soluções baseadas em medidores internos também não é adequada [Kavanagh et al. 2016]. Modelos matemáticos e computacionais para

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

estimar a energia consumida a partir de contadores de desempenho são impactados por elevadas flutuações na carga de trabalho e pela complexidade e variabilidade do hardware e do software [McCullough et al. 2011].

Neste capítulo, focamos nos métodos e ferramentas de medição do consumo de energia que utilizam **sensores de hardware e interfaces de software**. Em particular, nos centramos em soluções construídas com base na interface RAPL (*Running Average Power Limit*) [Weaver et al. 2012]. RAPL é hoje a interface de referência para estimar o consumo de energia dos processadores [Phung et al. 2018, Thamm and Leser 2025, Diamond and Stoico 2026].

1.3.1. Interface RAPL

A interface RAPL (*Running Average Power Limit*) foi introduzida na arquitetura *Sandy Bridge* da Intel em 2011 [Weaver et al. 2012] e evoluiu em arquiteturas posteriores, como *Haswell* e *Skylake*. Existem hoje interfaces semelhantes em arquiteturas de outros fabricantes, como a da AMD. Permite medir, monitorar e definir limites para a energia consumida pela CPU e pela DRAM associada. Ferramentas de monitoramento do consumo de energia, como CodeCarbon[3] e Scaphandre[4], utilizam a interface RAPL.

[Khan et al. 2018] detalham o funcionamento do RAPL e apresentam uma avaliação cuidadosa das experiências de uso da interface.

Foram utilizados tanto *microbenchmarks* personalizados quanto *benchmarks* de nível de aplicação bem conhecidos, com o objetivo de investigar a acurácia, a granularidade e o desempenho das medidas retornadas pela interface RAPL. Os resultados mais relevantes apontados pelos autores foram: (i) boa acurácia, podendo ser afetada por diferentes arquiteturas, configurações de BIOS e outras configurações de desempenho; e (ii) baixa sobrecarga, podendo variar de uma arquitetura para outra. Além disso, observou-se uma correlação entre o consumo de energia e a temperatura. Na arquitetura Haswell, a energia cresceu de 10–12% entre 37 °C e 74 °C, e, em Skylake, de 8–10% entre 23 °C e 32 °C, indicando a necessidade de "pré-aquecer" a CPU antes das medições.

A interface RAPL é executada automaticamente quando o processador é iniciado, então não há sobrecarga de inicialização na tomada de medidas. Além da possibilidade de medir e monitorar o consumo de energia (foco deste texto), é importante destacar que a interface RAPL também permite definir limites para a potência média consumida pelos componentes internos do processador, o que possibilita o controle do aquecimento desses componentes.

A Figura 1.1 destaca os domínios físicos de gestão de energia abarcados pelo RAPL (que podem variar de uma arquitetura para outra). Para cada domínio, é possível informar o consumo de energia, limitar a potência média consumida dentro de um intervalo de tempo definido, monitorar o impacto da limitação de potência e fornecer outras informações, como unidades de medida e potências mínimas e máximas admitidas pelo domínio. Os domínios destacados incluem:

[3]CodeCarbon: `https://docs.codecarbon.io/`

[4]Scaphandre: `https://hubblo-org.github.io/scaphandre-documentation/`

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

- **Package**: energia consumida em um *socket* (inclui o consumo de todos os *cores*, da GPU integrada, do cache e do controlador de memória).
- **Power Plane 0**: energia consumida apenas pelos *cores* de um *socket*.
- **Power Plane 1**: energia consumida apenas pela GPU embutida no *socket*.
- **DRAM**: energia consumida apenas pela RAM associada ao controlador de memória do *socket*.
- **PSys**: introduzido na arquitetura Skylake (outros que não CPU e GPU embutidas).

O único domínio disponível em todas as arquiteturas Intel é o domínio *Package*. Em sistemas com mais de um *socket*, o RAPL reporta valores distintos para cada um deles.

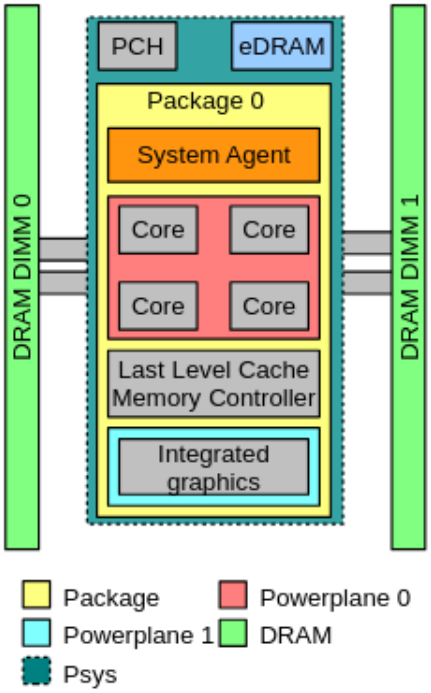

Figura 1.1. Domínios de potência do RAPL (extraído de [Khan et al. 2018])

A forma mais básica de acessar os contadores da interface RAPL é por meio dos MSRs (*Model-Specific Registers*). Trata-se de registradores de 32 ou 64 bits que são atualizados aproximadamente a cada 1 ms e indicam a energia consumida desde o instante em que o sistema foi ligado. Cada arquitetura adota uma unidade de medida básica, e o valor da energia consumida deve ser multiplicado por essa unidade (na arquitetura Sandy Bridge, por exemplo, a unidade básica é de 15,3 microjoules; já nas arquiteturas Haswell e Skylake, a unidade básica é de 61 microjoules). O Código 1.1 mostra um exemplo de acesso à interface RAPL por meio de registrador MSR[5]

[5]Um exemplo mais completo pode ser encontrado aqui: `https://web.eece.maine.edu/~vweaver/projects/rapl/rapl-read.c`, escrito por Vince Weaver.

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

Código 1.1. Exemplo de código para acesso à interface RAPL (adaptado de [Khan et al. 2018]).

```
uint64_t msr_value;
/* Haswell: units of 61 microjoules  */
/* MSR_PKG_ENERGY_STATUS is at address 0x611 */
/* Package domain */
double energy_units = pow (0.5, 14);
int fd = open ("/dev/cpu/0/msr", O_RDONLY);
if (fd > 0) {
   if (pread (fd, &msr_value, 8, 0x611) > 0) {
      double energy = msr_value * energy_units;
      printf ("%f\n", energy);
}}
```

No exemplo, o modelo da CPU (Haswell) e sua unidade de medida (61 microjoules) foram detectados previamente. O código acessa o dispositivo `/dev/cpu/0/msr` utilizando o endereço do registrador `MSR_PKG_ENERGY_STATUS` (0x611) que informa o consumo de energia do domínio *Package*.

Entre os desafios e limitações de uso do RAPL, [Khan et al. 2018] destacaram a necessidade de acesso com privilégios de administrador do sistema (*root*) aos registradores MSR e o fato de a interface não permitir (na versão avaliada) medir a energia consumida por *core* (mesmo tendo registradores MSR distintos, todos registram o mesmo valor). Além disso, há a possibilidade de estouro dos contadores (limitados a 32 ou 64 bits). Quando eles atingem o limite máximo, retornam a zero. Ao calcular a diferença entre duas medições consecutivas ($E2 - E1$) nesse intervalo, será retornado um valor negativo, pois o primeiro valor será menor que o segundo. Nesses casos, é necessário aplicar correções a esses valores. Por fim, observaram-se também atrasos nas atualizações dos contadores, o que indica que tais atualizações não são atômicas.

Como se observa, o acesso direto aos registradores MSR não é trivial, pois eles fornecem dados brutos que precisam ser tratados. As leituras dos registradores RAPL também podem ser feitas por meio da interface *sysfs powercap* (versões do Linux a partir de 3.13) e com auxílio do `perf_event_open` (versões do Linux a partir de 3.14). [Diamond and Stoico 2026] chamam a atenção para a sobrecarga que pode ser causada pela leitura de energia e investigam estratégias de mitigação. [Kennes 2023] listam diversos trabalhos que reportam o uso do RAPL para a medida do consumo de energia. Há várias ferramentas de medição de energia que usam o RAPL como base. Uma lista extensa está disponível no GitHub[6].

Powercap Trata-se de um *framework* do sistema operacional Linux que provê uma interface de acesso no espaço do usuário, via `sysfs`, na forma de uma *árvore de objetos de controle*. Esses objetos permitem que ferramentas possam monitorar e limitar o consumo de energia de dispositivos de hardware de forma padronizada. Um objeto é sempre

[6] `https://github.com/Green-Software-Foundation/awesome-green-software`

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

carregado na memória e, para facilitar a visualização do usuário, aparece virtualmente como um diretório no gerenciador de arquivos. No contexto do powercap, o objeto no topo (raiz) da árvore representa os tipos de controle. Dentro de cada objeto de tipo de controle, temos acesso aos domínios de potência, que contêm atributos de monitoramento e de restrição (fornecendo opções para limitar o consumo de energia de um componente).

Para exemplificar, o Código 1.2 realiza a leitura do consumo de energia via *powercap* em um computador com processador Intel(R) Core(TM) i5-7300HQ CPU 2.50GHz e arquitetura Kaby Lake, utilizando linguagem Python. O domínio de energia consultado foi o *Package*.

Código 1.2. Exemplo de código usando a interface powercap.

```
STRESS_FUNC = ["stress-ng", "--matrix", "0", "-t", "1m", "-v"]
TAXA_AMOSTRAGEM = 1.0

def leitor_powercap():
    '''Lê o contador de energia e o momento da leitura.'''
    caminho_contador = "/sys/class/powercap/intel-rapl/subsystem
       /intel-rapl:0/energy_uj"
    with open(caminho_contador, "r") as powercap:
        valor = powercap.readline().strip()
        tempo_medicao = time.perf_counter()
        return valor, tempo_medicao

def loop_leitor_powercap(duracao, resultados):
    '''Realiza leituras contínuas durante o período de tempo
       dado.'''
    tempo_inicio = time.perf_counter()
    tempo = tempo_inicio
    while (tempo - tempo_inicio <= duracao):
        leitura = [0] * 2
        leitura[0], leitura[1] = leitor_powercap()
        time.sleep(TAXA_AMOSTRAGEM)
        tempo = time.perf_counter()
        resultados.append([leitura[0], leitura[1]])

resultados = []
loop_leitor_powercap(10, resultados)
#inicia stress-ng
stress = subprocess.Popen(STRESS_FUNC)
while(stress.poll() == None):  #enquanto o processo não terminar
    leitura = [0] * 2
    leitura [0], leitura[1] = leitor_powercap()
    resultados.append([leitura[0], leitura[1]])
    time.sleep(TAXA_AMOSTRAGEM)

#10 segundos de execução sem stress-ng
loop_leitor_powercap(10, resultados)
#salva leituras realizadas (...)
```

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

Para estressar a CPU, utilizou-se a função `stress-ng -matrix 0` [7] por 1 minuto. Essa função utiliza operações de ponto flutuante e de memória que exigem alto uso da CPU. Note que, para reduzir a sobrecarga da própria medição, o Código 1.2 apenas realiza a leitura dos contadores e faz a persistência dos dados. Para obter os valores de energia consumida, é necessária a execução de outro código que realiza o tratamento e processamento dos dados do RAPL.

Como saída, obtemos um arquivo com diversas linhas e duas colunas (mostradas abaixo). Cada linha representa uma medição, sendo a primeira coluna o valor do contador de energia do RAPL (em microjoules) e a segunda, o valor do contador de tempo (em segundos).

```
37366890973 1108197.640825755
37370550815 1108198.641376323
37374133508 1108199.642308569
37377637650 1108200.643097468
...
```

Para obtermos a potência média consumida, calculamos a diferença entre duas medidas consecutivas (Δe na coluna de energia e Δt na coluna do tempo). A potência média em watts será dada por:

$$p = \Delta e \times 10^{-6} / \Delta t$$

Usando um código auxiliar, processamos os dados coletados para calcular a potência média consumida em cada intervalo de tempo. Como saída, obtemos um arquivo de texto (mostrado abaixo) contendo os valores de potência média (em watts) e o tempo (em segundos) que a medição foi realizada.

```
3.657828116509902 1.0005505681037903
3.579356159605825 2.0014828140847385
3.50137976533455 3.0022717129904777
3.507390855194336 4.0031550319399685
...
```

Com base nesses dados de saída, plotamos o gráfico da Figura 1.2. Podemos observar que a potência média consumida varia claramente ao longo do intervalo de tempo em que o código de estresse foi executado.

1.3.2. Ferramenta Scaphandre

O Scaphandre[8] é um agente de monitoramento escrito na linguagem Rust que permite obter o consumo de energia por processo, por máquina virtual ou por contêiner em execução em uma CPU. A ferramenta utiliza a interface RAPL e é compatível com os sistemas operacionais Windows e GNU/Linux, com algumas funcionalidades restritas a cada sistema operacional.

[7] `https://wiki.ubuntu.com/Kernel/Reference/stress-ng`
[8] `https://github.com/hubblo-org/scaphandre`

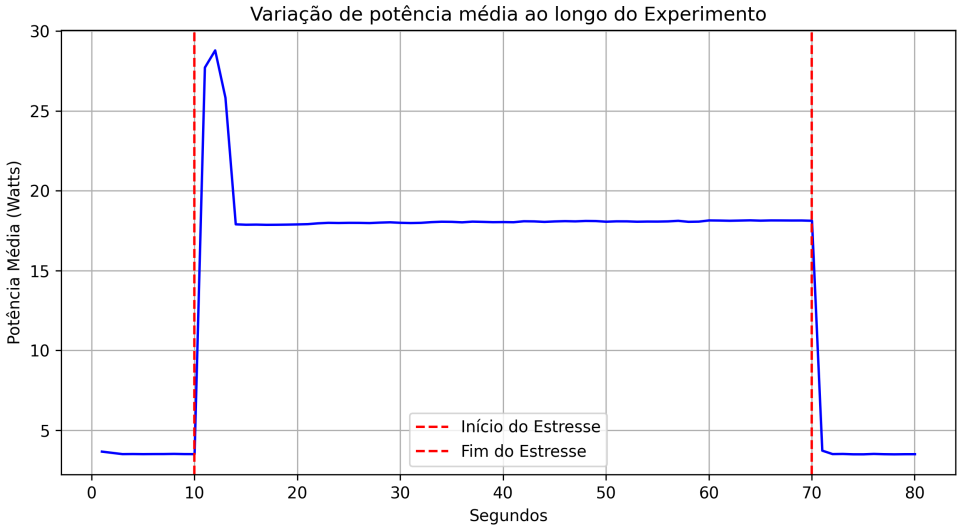

Figura 1.2. Resultado do exemplo usando Powercap.

Para calcular o consumo de energia das aplicações, o Scaphandre coleta continuamente dados dos sensores de hardware e os associa ao tempo de CPU dessas aplicações. A interface de hardware utilizada depende do ambiente no qual a ferramenta será executada. A cada coleta, são lidos os valores do tempo de uso da CPU dos processos ativos no intervalo de medição. Em seguida, é feito o cálculo da estimativa para o consumo de energia por meio da relação entre o consumo de energia total do processador no intervalo medido e a fatia de tempo de CPU que o processo alvo utilizou. Quando programas são executados em processos diferentes ao mesmo tempo, é necessário agregar o consumo de energia de cada processo para obter o gasto total da aplicação.

Em ambiente Linux, o Scaphandre coleta os dados do RAPL via interface *sysfs powercap*, priorizando o domínio PSYS por contemplar a maioria dos componentes. Se esse domínio não estiver disponível na arquitetura-alvo, os dados dos domínios Package e DRAM são somados. Em ambiente Windows, a leitura é feita diretamente dos registradores MSR, via instruções RDMSR.

O Scaphandre foi projetado para ser extensível, limitando-se às tarefas de coleta e pré-processamento das métricas de consumo de energia para, em seguida, exportá-las para outras ferramentas que permitam, por exemplo, visualizar os dados levantados.

1.3.3. Ferramenta CodeCarbon

CodeCarbon[9] é uma biblioteca em Python que permite estimar as emissões de carbono de uma aplicação. Para realizar as medições de consumo de energia, utiliza a interface RAPL.

O CodeCarbon faz estimativas do consumo de energia dos principais componentes do hardware que podem ser monitorados ou estimados diretamente: CPU, GPU e RAM. Ele não modela separadamente E/S de disco, transferências de rede, monitores,

[9] `https://codecarbon.io`

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

resfriamento nem outros periféricos.

Para estimar as emissões de CO_2 associadas à execução de uma aplicação, o CodeCarbon considera que as emissões, expressas em quilogramas de CO_2-equivalente (CO_2eq), são o produto de dois fatores principais:

- C = a intensidade de carbono da eletricidade utilizada para a computação; quantificada em g de CO_2 emitida por quilowatt-hora de eletricidade;
- E = energia consumida pela infraestrutura computacional: quantificada em quilowatt-hora.

As emissões de dióxido de carbono (CO_2eq) são calculadas como o produto $C \times E$.

Intensidade de carbono: A intensidade de carbono da eletricidade consumida (C) é calculada como a média ponderada das emissões de diferentes fontes de energia utilizadas para gerar eletricidade, incluindo combustíveis fósseis e renováveis. No CodeCarbon, os combustíveis fósseis, como carvão, petróleo e gás natural, são associados a intensidades de carbono específicas: uma quantidade conhecida de dióxido de carbono é emitida por quilowatt-hora de eletricidade gerada. As fontes renováveis ou de baixo carbono incluem energia solar, hidroelétrica, biomassa e geotérmica, entre outras. A rede elétrica local contém uma mistura de combustíveis fósseis e de fontes de energia de baixo carbono, chamada Matriz Energética. Com base na proporção de fontes de energia na rede local, a intensidade de carbono da eletricidade consumida pode ser calculada.

Quando disponível, o CodeCarbon usa a intensidade de carbono da eletricidade por país (com dados do site Our World in Data[10]) ou por provedor de computação em nuvem. Se esses dados não estiverem disponíveis, mas detalhes sobre a matriz energética forem conhecidos, o CodeCarbon é capaz de estimar as emissões totais com base na proporção de cada tipo de fonte de energia (carvão, petróleo, gás natural etc.) utilizada para produzir a eletricidade. Em último caso, se nenhuma dessas fontes estiver disponível para o CodeCarbon, utiliza-se a média global de intensidade de carbono: 475 gCO_2eq/kWh.

Consumo de energia: O consumo de energia do hardware subjacente é monitorado em intervalos regulares. Esse comportamento é controlado por um parâmetro configurável (`measure_power_secs`), cujo valor padrão é 15 segundos e pode ser definido no momento da criação do rastreador de emissões (*emissions tracker*).

O CodeCarbon oferece dois modos distintos para determinar o consumo de energia atribuído ao seu trabalho. No **Machine Mode** (`tracking_mode="machine"`), o CodeCarbon mede a energia total consumida por toda a pilha de hardware (todas as CPUs, GPUs e RAM). No **Process Mode** (`tracking_mode="process"`), o CodeCarbon estima a energia do seu código Python (e de seus subprocessos) por meio da amostragem do tempo de CPU em relação à capacidade total da CPU. Esta é uma aproximação baseada em software — ela não lê os contadores de hardware diretamente — e é preferível em ambientes compartilhados onde outras tarefas são executadas em paralelo.

[10] `https://ourworldindata.org/grapher/carbon-intensity-electricity`

A energia consumida por cada tipo de hardware é medida por meio de uma estratégia diferente. A energia consumida pelas GPUs é medida pela biblioteca `nvidia-ml-py`. A energia consumida pela RAM é estimada em 5 W por cada *slot* de RAM utilizado. Já a energia consumida pela CPU depende do sistema operacional e do hardware utilizado.

Atualmente, o CodeCarbon permite medir o consumo de energia de CPUs em ambientes Linux, Windows e macOS. A técnica utilizada para interagir com o hardware e com os domínios de energia consultados depende do ambiente em que se trabalha. No ambiente Linux, os dados de consumo de energia são obtidos via interface RAPL, com o uso do *sysfs powercap*. No ambiente Windows, utiliza a ferramenta Intel Power Gadget[11] (descontinuada pela Intel em 2023). No ambiente macOS, a ferramenta *powermetrics*, nativa desse ambiente, é utilizada. Caso não seja possível obter as medidas de consumo de energia da CPU, o Codecarbon realiza aproximações com base no tipo da CPU-alvo.

O Código 1.3 apresenta um exemplo simples de como o CodeCarbon pode ser integrado a um código em Python para rastrear as emissões. O Código 1.4 mostra como os resultados do rastreamento podem ser acessados programaticamente.

Código 1.3. Exemplo de rastreamento de emissões usando o CodeCarbon.

```python
from codecarbon import EmissionsTracker

# Inicializa o rastreador de emissões
with EmissionsTracker(project_name="meu-primeiro-rastreamento")
    as tracker:
    # Simula alguma computação
    total = 0
    for i in range(10_000_000):
        total += i

    # Imprime o resultado do cálculo
    print(f"Resultado da computação: {total}")
```

1.3.4. Boas práticas para medir consumo de energia

A partir dos relatos feitos por diversos autores, é possível elencar uma lista de recomendações e boas práticas a seguir ao medir o consumo de energia das aplicações[12]:

- **Taxa de amostragem**:[13] A frequência de coleta das medidas de energia deve ser de pelo menos a metade da duração do menor evento que se deseja capturar. Isso garante a precisão necessária para não perder picos de consumo rápidos durante a execução do experimento.
- **Controle da temperatura** [Diamond and Stoico 2026, Khan et al. 2018]: A temperatura de um processador influencia a medição de energia. Sugere-se esperar 180

[11] `https://www.intel.com/content/www/us/en/developer/archive/tools/power-gadget.html`
[12] `https://docs.green-coding.io/docs/measuring/best-practices/`
[13] `https://docs.codecarbon.io/`

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

Código 1.4. Exemplo de inspeção do resultado do rastreamento do CodeCarbon.

```
import pandas as pd

# Carrega o relatório de emissões gerado pelo CodeCarbon
df = pd.read_csv("emissions.csv")

# Seleciona e exibe colunas específicas do relatório para
    análise
df[["project_name", "duration", "emissions", "emissions_rate", "
    cpu_power", "ram_power", "energy_consumed"]]

#####
# Você também pode acessar os dados de emissões do objeto
    tracker

print(f"Emissões totais: {tracker.final_emissions * 1000:.4f} g
    CO2eq")
print(f"Duração: {tracker.final_emissions_data.duration:.2f}
    segundos")
print(f"Energia consumida: {tracker.final_emissions_data.
    energy_consumed:.6f} kWh")
```

segundos entre as medições para que os componentes esfriem. Em processadores com mais de 30 cores, esse tempo deve ser maior. Se um sistema sobreaquecer durante a medição, isso deve ser considerado.

- **Minimização do ruído de fundo**:[14] Durante a execução do experimento, o gasto energético total da máquina será capturado por meio de interfaces de hardware, como o RAPL. Para minimizar o possível "ruído" do sistema operacional, é importante diminuir a possibilidade de interrupções na CPU. Recomenda-se tomar as seguintes precauções: (a) desativar conexões de rede (Wi-Fi e Internet), a menos que sejam objeto de teste; (b) evitar qualquer interação com periféricos (mouse e teclado) durante a execução; (c) desligar o escurecimento automático da tela, proteção de tela e modos de suspensão; (d) encerrar processos em segundo plano que não sejam essenciais; (e) desativar tarefas agendadas, atualizações automáticas do sistema operacional e rotinas de manutenção.

- **Escrita de resultados na memória**: Para armazenar os valores das medições, é recomendado realizar a escrita em um arquivo que esteja localizado na memória RAM. A escrita em disco é uma operação que gasta bastante energia e, caso seja feita com frequência, pode alterar o experimento a depender das métricas utilizadas.

Para ferramentas que utilizam *Modelos de Divisão de Potência* para estimar o consumo de energia por aplicação isolada, como é o caso do *Scaphandre*, algumas recomendações adicionais incluem manter a frequência de clock fixa, desativar *Hyperthreading* e

[14] https://www.kernel.org/doc/html/next/power/powercap/powercap.html

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

Turbo Boost [Cadorel and Saingre 2024]. Além disso, salienta-se que a execução de instruções como AVX (*Advanced Vector Extensions*) pode distorcer o fatiamento de energia, uma vez que o tempo de uso de CPU deixa de ter uma relação clara com o consumo de energia (e, consequentemente, com a potência atribuída a cada processo).

1.4. Abordagens para Computação Sustentável

Algumas abordagens e técnicas têm sido propostas e experimentadas com o objetivo de aplicar os conceitos relacionados à Computação Sustentável, incluindo soluções cientes do consumo de energia, no desenvolvimento e na execução de aplicações computacionais [Cordeiro et al. 2023, Lee and Zomaya 2012, Jones et al. 2018, Almeida et al. 2024].

O conceito de *carbon-responsive computing* engloba estratégias que são cientes da intensidade de CO_2 gerada e usam essa informação para tomar decisões [Nafus et al. 2021]. *Follow-the-renewables* é um exemplo de abordagem nessa direção, que visa incorporar informações sobre a disponibilidade de energia renovável nas decisões de escalonamento de tarefas [Vasconcelos et al. 2023].

Técnicas de otimização já conhecidas e aplicadas em contextos como aprendizado de máquina ou em dispositivos com recursos limitados podem ser usadas para melhorar a eficiência energética, incluindo *pruning* (remover partes desnecessárias de modelos durante o treinamento) e *quantização* (converter dados contínuos ou de alta precisão em conjuntos menores, discretos e finitos).

Outras abordagens envolvem *ciência do hardware* [Carro and Nazar 2023], visando tirar proveito das características específicas do hardware subjacente, e técnicas de *computação aproximada*, que fazem uso de aritmética simplificada ou aproximada para reduzir a complexidade das aplicações [Hoffmann 2015, Rocha et al. 2022].

1.5. Considerações finais

Há vários desafios em aberto a serem enfrentados no contexto de Computação Sustentável [Kennes 2023, Lottick et al. 2019]. Para além de medir o consumo de energia de uma aplicação, há outras demandas a explorar.

Com relação ao *mix* de provisão de energia instantânea, com fontes de energia tradicionais e renováveis, um desafio é a dificuldade de comparar os mesmos intervalos de tempo entre a energia gerada e a energia consumida e a discrepância de granularidade entre a produção medida em horas e o consumo medido em instantes com RAPL. [Kennes 2023] argumenta que a redução do consumo de energia deveria ser realizada para reduzir a produção de energia, facilitando a operação do sistema e evitando emissões decorrentes de requisitos adicionais de energia acima do topo do *baseline* dado.

Como argumenta [Lottick et al. 2019], o cálculo da emissão de carbono é um campo científico próprio. Envolve atribuir o carbono e outros poluentes emitidos na geração, transporte, manutenção, disponibilização e descarte dos recursos, tornando o usuário responsável por essas emissões. Nesse sentido, há várias questões. (a) Como calcular o tempo durante o qual o recurso foi usado por determinada aplicação e sua proporção no tempo total de vida do recurso? (b) Como medir a emissão de poluentes considerando que uma aplicação pode usar milhares de componentes ou recursos? (c) Como tratar

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

o compartilhamento de recursos entre aplicações? (d) Como garantir a transparência e a auditabilidade da emissão de poluentes por aplicações? (e) Quem é responsável pela emissão de poluentes quando o recurso está em sua fase útil, mas não está em uso?

Essas e outras questões estão abertas à pesquisa e à exploração. Esperamos que este texto introdutório desperte o interesse e chame a atenção dos profissionais e pesquisadores da área para as demandas colocadas. Existem ferramentas disponíveis e em desenvolvimento que apoiam o desenvolvimento de aplicações cientes do consumo de energia e da pegada de carbono. Incentivamos o leitor a aprofundar o conhecimento das técnicas e ferramentas apresentadas e a contribuir para a sua evolução e adoção.

Referências

[Almeida et al. 2024] Almeida, G., Torres Do Ó, M., Francesquini, E., and Cordeiro, D. (2024). Reducing carbon emissions of distributed systems: a multi-objective approach. In *Proceedings of the 20th Brazilian Symposium on Information Systems*, pages 1–10.

[Cadorel and Saingre 2024] Cadorel, E. and Saingre, D. (2024). A protocol to assess the accuracy of process-level power models. In *2024 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 74–84. IEEE.

[Carro and Nazar 2023] Carro, L. and Nazar, G. L. (2023). Desafios para a computação energeticamente eficiente. *Sociedade Brasileira de Computação*.

[Cordeiro et al. 2023] Cordeiro, D., Francesquini, E., Amarís, M., Castro, M., Baldassin, A., and Lima, J. (2023). Green cloud computing: Challenges and opportunities. In *Anais Estendidos do XIX Simpósio Brasileiro de Sistemas de Informação*, pages 129–131, Porto Alegre, RS, Brasil. SBC.

[Diamond and Stoico 2026] Diamond, J. and Stoico, V. (2026). What is the cost of energy monitoring? an empirical study on the overhead of rapl-based tools. *arXiv preprint arXiv:2604.26815*.

[Hoffmann 2015] Hoffmann, H. (2015). Jouleguard: Energy guarantees for approximate applications. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 198–214.

[Jay et al. 2023] Jay, M., Ostapenco, V., Lefèvre, L., Trystram, D., Orgerie, A.-C., and Fichel, B. (2023). An experimental comparison of software-based power meters: focus on CPU and GPU. In *2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 106–118. IEEE.

[Jones et al. 2018] Jones, N. et al. (2018). How to stop data centres from gobbling up the worlds electricity. *Nature*, 561(7722):163–166.

[Kavanagh et al. 2016] Kavanagh, R., Armstrong, D., and Djemame, K. (2016). Accuracy of energy model calibration with IPMI. In *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, pages 648–655. IEEE.

[Kennes 2023] Kennes, T. (2023). Measuring IT carbon footprint: What is the current status actually? *arXiv preprint arXiv:2306.10049*.

[Khan et al. 2018] Khan, K. N., Hirki, M., Niemi, T., Nurminen, J. K., and Ou, Z. (2018). RAPL in action: Experiences in using rapl for power measurements. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, 3(2):1–26.

[Launikas et al. 2024] Launikas, A. E., Nakano, F., Coutinho, F. L., Cordeiro, D., et al. (2024). Evolução histórica do desempenho energético de tarefas cotidianas em uma distribuição Linux. In *Simpósio em Sistemas Computacionais de Alto Desempenho (SSCAD)*, pages 81–88. SBC.

[Lee and Zomaya 2012] Lee, Y. C. and Zomaya, A. Y. (2012). Energy efficient utilization of resources in cloud computing systems. *The Journal of Supercomputing*, 60(2):268–280.

[Lottick et al. 2019] Lottick, K., Susai, S., Friedler, S. A., and Wilson, J. P. (2019). Energy usage reports: Environmental awareness as part of algorithmic accountability. *arXiv preprint arXiv:1911.08354*.

[McCullough et al. 2011] McCullough, J. C., Agarwal, Y., Chandrashekar, J., Kuppuswamy, S., Snoeren, A. C., Gupta, R. K., et al. (2011). Evaluating the effectiveness of model-based power characterization. In *USENIX Annual Technical Conf*, volume 20, pages 19–20.

[Möbius et al. 2013] Möbius, C., Dargie, W., and Schill, A. (2013). Power consumption estimation models for processors, virtual machines, and servers. *IEEE Transactions on Parallel and Distributed Systems*, 25(6):1600–1614.

[Nafus et al. 2021] Nafus, D., Schooler, E. M., and Burch, K. A. (2021). Carbon-responsive computing: Changing the nexus between energy and computing. *Energies*, 14(21):6917.

[Paul et al. 2023] Paul, S. G., Saha, A., Arefin, M. S., Bhuiyan, T., Biswas, A. A., Reza, A. W., Alotaibi, N. M., Alyami, S. A., and Moni, M. A. (2023). A comprehensive review of green computing: Past, present, and future research. *IEEE access*, 11:87445–87494.

[Phung et al. 2018] Phung, J., Lee, Y. C., and Zomaya, A. Y. (2018). Modeling system-level power consumption profiles using RAPL. In *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, pages 1–4. IEEE.

[Rocha et al. 2022] Rocha, F. W., Fukuda, J. C., Francesquini, E., and Cordeiro, D. (2022). Accelerating smart city simulations. In Gitler, I., Barrios Hernández, C. J., and Meneses, E., editors, *High Performance Computing*, pages 148–162, Cham. Springer International Publishing.

[Santos and Wagner 2025] Santos, A. L. d. M. and Wagner, F. R., editors (2025). *Grandes Desafios da Computação no Brasil 2025-2035*. Sociedade Brasileira de Computação (SBC), Porto Alegre.

[Thamm and Leser 2025] Thamm, P. and Leser, U. (2025). Strategies to measure energy consumption using RAPL during workflow execution on commodity clusters. *arXiv preprint arXiv:2505.09375*.

[Vasconcelos et al. 2023] Vasconcelos, M., Cordeiro, D., Da Costa, G., Dufossé, F., Nicod, J.-M., and Rehn-Sonigo, V. (2023). Optimal sizing of a globally distributed low carbon cloud federation. In *2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 203–215. IEEE.

[Weaver et al. 2012] Weaver, V. M., Johnson, M., Kasichayanula, K., Ralph, J., Luszczek, P., Terpstra, D., and Moore, S. (2012). Measuring energy and power with PAPI. In *2012 41st international conference on parallel processing workshops*, pages 262–268. IEEE.