Eficiência Energética em Python: Um Estudo Empírico Comparando o uso de Biblioteca e Python Puro na Leitura de Arquivos CSV.

Caio Henrique dos Santos,Everton Cezar Gonçalves,
João Pedro Guez de Oliveira, Marcio Marcos

November 3, 2025

Abstract

Introduction: Processing tabular CSV data is a common task in software engineering. The Python language offers multiple approaches to this, ranging from native libraries like the csv module to more robust, high-performance data analysis libraries such as Pandas. These approaches, with their different levels of abstraction and optimization, exhibit distinct patterns of resource usage, including energy consumption, an increasingly critical quality metric from the perspective of green software engineering.

Methods: In this study, we evaluate the energy efficiency of two Python implementations for processing a CSV file containing personal data. One application will be implemented using pure Python, and another using the Pandas library. Both applications will perform the same task: read the CSV file, store the data in memory, iterate over the records, and print them to the console. Through empirical experiments, using the PyrAPL tool to measure the energy footprint of a host machine during the execution of Python code, we will measure the energy consumption in Joules of the applications, with the pure implementation vs. library approach as the main independent variable.

Results: Through this research, we aim to quantify the energy efficiency gains achieved by utilizing the library, as well as determine whether a potential energy efficiency gain can be realized, providing the developer with not only cleaner code but also more energy-efficient code. The data analysis compares the energy consumption between code developed in pure Python and code implemented with the help of a library. We hope to show how the use of more effective methods can generate benefits both for the software, in terms of performance, and for the hardware, through reduced energy consumption. The expected results seek to foster debate on the importance of selecting appropriate tools and implementation approaches in the software development process. We propose a reflection on the need to build lighter, more sustainable solutions that are aligned with the principles of computational and environmental efficiency.

Keywords: Energy efficiency, Python, Pandas, Green software, CSV processing.

Resumo

Introdução: O processamento de dados tabulares CSV é uma tarefa comum na engenharia de software. A linguagem Python oferece múltiplas abordagens para tal, desde bibliotecas nativas como o módulo csv até bibliotecas mais robustas de análise de dados de alto desempenho como Pandas. Essas abordagens, com seus diferentes níveis de abstração e otimização, exibem padrões distintos de uso de recursos, incluindo o consumo de energia, uma métrica de qualidade cada vez mais importante e crítica sob a ótica da engenharia de software verde.

Métodos: Neste estudo, avaliamos a eficiência energética de duas implementações Python para o processamento de um arquivo CSV de dados de pessoa física. Serão implementadas uma aplicação utilizando a linguagem Python de forma pura e outra utilizando a biblioteca Pandas, ambas as aplicações executarão a mesma tarefa: ler o CSV, guardar os dados em memória e iterar sobre os registros e imprimir no console. Por meio de experimentos empíricos, utilizando a ferramenta PyrAPL para medir a pegada energética de uma máquina host ao longo da execução de um código Python, iremos medir o consumo de energia em joules das aplicações, tendo a abordagem de implementação pura vs. biblioteca como principal variável independente.

Resultados: Esperamos através desta pesquisa buscar quantificar o ganho de eficiência energética proporcionado pela utilização da biblioteca e se teremos um possível ganho de eficiência energética, proporcionando ao desenvolvedor não somente um código mais limpo, mas também um código mais eficiente energeticamente. A análise de dados compara o consumo energético entre um código desenvolvido em Python puro e outro implementado com o auxílio de biblioteca, esperamos evidenciar como o uso de métodos mais eficazes pode gerar benefícios tanto para o software, em termos de desempenho, quanto para o hardware, por meio da redução do consumo de energia. Os resultados esperados buscam fomentar o debate sobre a importância da seleção adequada de ferramentas e abordagens de implementação no processo de desenvolvimento de software. Propondo uma reflexão sobre a necessidade de construir soluções mais leves, sustentáveis e alinhadas com os princípios de eficiência computacional e ambiental.

Palavras-chave: Eficiência energética, Python, Pandas, Software verde, Processamento de CSV.

1 Introdução

O processamento de dados desempenha um papel fundamental em quase todas as áreas da nossa sociedade. Volumes massivos de informação são comumente utilizados no comércio, na medicina, no transporte incluindo a análise de telemetria e na pesquisa científica. Grande parte desses dados é armazenada e trocada no formato CSV (Comma-Separated Values), que, apesar de sua simplicidade, se tornou uma linguagem universal para dados tabulares. Para manipular esses dados, um ecossistema de ferramentas emergiu, no qual a linguagem Python é uma das mais importantes e dominantes.

Python é considerado o padrão de facto para automação de scripts e ciência de dados. Ele fornece uma camada de abstração que permite a especialistas de diferentes áreas integrar e processar dados de forma eficiente. Além disso, Python compreende um conjunto abrangente de bibliotecas de código aberto. No centro do processamento de dados em Python, existem duas abordagens principais: o uso de bibliotecas nativas, como o módulo csv (incluído na instalação padrão), e o uso de frameworks robustos de análise de dados de alto desempenho, como a biblioteca Pandas. Ambas as abordagens são amplamente utilizadas, com o Pandas sendo baixado dezenas de milhões de vezes por semana, incentivando significativamente o reuso de código e a padronização de pipelines de dados.

A eficiência energética do software tem sido uma preocupação recorrente entre os desenvolvedores de software. Isso é estimulado por fatores que incluem o impacto ambiental dos data centers, custos operacionais especialmente em computação em nuvem e a dependência de dispositivos movidos a bateria. O domínio do processamento de dados em larga escala é diretamente afetado por essa questão. Decisões arquiteturais simples, como a escolha de uma biblioteca, podem ter um impacto na eficiência energética do software, sendo a ferramenta de parsing e manipulação de dados um fator determinante. No caso do Python, desenvolvedores e engenheiros de dados podem escolher entre a abordagem pura e bibliotecas Pandas. Atualmente, essa escolha é frequentemente baseada na conveniência, legibilidade do código ou velocidade de execução, mas deve ser feita com uma compreensão limitada do seu impacto na eficiência energética do sistema.

Neste artigo, conduzimos uma análise sistemática do consumo de energia associado a uma tarefa comum de processamento de dados, implementada em Python puro e com a biblioteca Pandas. O objetivo é preencher a lacuna de conhecimento sobre como essas duas abordagens diferem em termos de demanda energética para uma operação fundamental: ler um arquivo CSV de dados de pessoa física, armazenar esses dados em memória e iterar sobre eles para impressão no console. Para abordar esta questão, exploramos uma variável independente principal: a abordagem de implementação Python puro vs. Pandas. Utilizamos a ferramenta PyrAPL para medir com precisão o consumo de energia em joules, isolando a pegada energética do código executado. Esperamos através desta pesquisa quantificar o ganho de eficiência energética proporcionado pela utilização da biblioteca, verificando se um código considerado mais limpo ou de mais alto nível também pode ser mais eficiente energeticamente.

O público-alvo deste estudo inclui pesquisadores em Engenharia de Software Verde, bem como profissionais e engenheiros de dados envolvidos no desenvolvimento de pipelines de dados em Python. Este trabalho fornece insights valiosos para ajudar a otimizar tarefas de I/O, tomar decisões de design informadas e conduzir experimentos em software energeticamente eficiente. Ele incentiva os profissionais a considerar o consumo de energia como uma métrica de qualidade, além do desempenho e da manutenibilidade.

Este artigo contribui com insights sobre o consumo de energia de duas abordagens de processamento de dados amplamente utilizadas. Pode ser usado como fonte de inspiração para o desenvolvimento de software mais verde, promovendo práticas ambientalmente conscientes no desenvolvimento de software. Ele também fornece um arcabouço metodológico e orientação prática para a condução de experimentos futuros neste campo. Finalmente, fornecemos um pacote de replicação completo e dados experimentais para beneficiar tanto pesquisadores quanto profissionais da indústria.

2 Definição de Pesquisa

Esta seção detalha o desenho experimental do estudo, definindo as hipóteses, as questões de pesquisa que guiam a investigação, as métricas utilizadas para respondê-las e as variáveis controladas e observadas.

2.1 Hipóteses

Com base no objetivo da pesquisa de quantificar o ganho de eficiência energética proporcionado pela utilização da biblioteca, definimos as seguintes hipóteses:

2.1.1 Hipótese Nula (H_0):

Não existe diferença estatisticamente significativa no consumo de energia (em joules) entre a aplicação que utiliza Python puro (com o módulo csv nativo) e a aplicação que utiliza a biblioteca Pandas para a tarefa

definida (ler o CSV, guardar os dados em memória, iterar e imprimir no console).

2.1.2 Hipótese Alternativa (H_1)

A aplicação que utiliza a biblioteca Pandas apresenta um consumo de energia significativamente menor, sendo mais eficiente energeticamente do que a aplicação que utiliza Python puro para a tarefa de processamento de CSV definida.

2.2 Perguntas de Pesquisa (RQ_s)

Para guiar este estudo empírico, formulamos as seguintes perguntas de pesquisa:

2.2.1 RQ_1

Qual é a diferença no consumo de energia (medido em joules) ao processar um arquivo CSV utilizando a biblioteca Pandas em comparação com uma implementação em Python puro?

2.2.2 RQ_2

A abordagem de implementação com a biblioteca Pandas é mais eficiente energeticamente do que a abordagem com Python puro para a tarefa de ler, armazenar dados em memória e iterar sobre os registros?

2.3 Métricas

Para responder às perguntas de pesquisa, coletamos as seguintes métricas:

2.3.1 Métrica Principal:

Consumo Energético: Esta é a métrica central do estudo.

- Definição: A pegada energética da execução do código Python de cada abordagem;
- Unidade de Medida: Joules;
- Ferramenta de Coleta: A medição será realizada utilizando a ferramenta PyrAPL;
- Como responde às RQ_s: Esta métrica responde diretamente à RQ_1, quantificando a diferença no consumo energético, e à RQ_2, ao determinar qual das abordagens é mais eficiente.

2.3.2 Métrica Secundária (Contextual):

Tempo de Execução (Desempenho):

- Definição: O tempo total que cada aplicação (Python puro vs. Pandas) leva para completar a tarefa idêntica. Embora o foco seja a energia, a literatura frequentemente compara a eficiência energética com a velocidade de execução e o desempenho geral;
- Como responde às RQ_s: Esta métrica fornece um contexto crucial para a RQ_2. Ela permite analisar se a abordagem mais eficiente energeticamente também é a mais rápida, ou se existe um trade-off (troca) entre consumo de energia e desempenho.

2.4 Variáveis Dependentes e Independentes

O experimento é desenhado com base nas seguintes variáveis:

2.4.1 Variável Independente:

- Definição: A abordagem de implementação utilizada para o processamento de dados;
- Níveis:
  1. Implementação com Python de forma pura.
  2. Implementação com a biblioteca.

2.4.2 Variável Dependente:

- Definição: O consumo de energia das aplicações;
- Unidade: Joules.

2.5 Fluxo de Execução e Medição

Este diagrama de sequência visualiza o fluxo de interação entre os componentes do experimento:

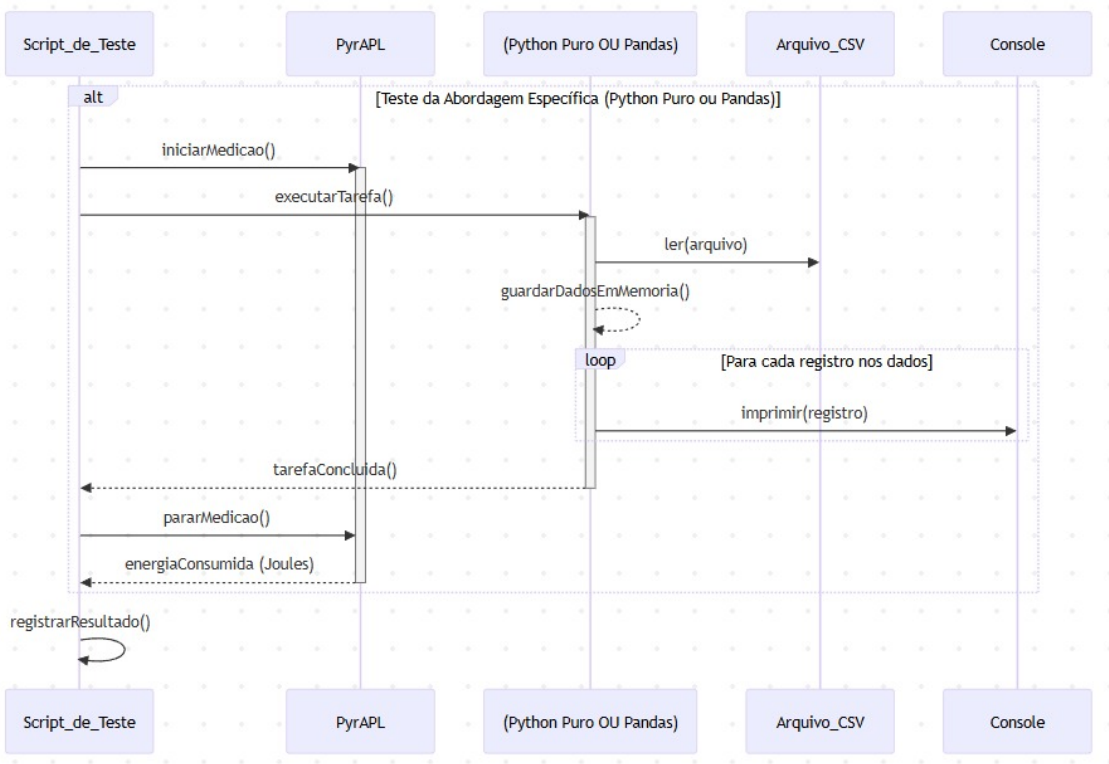

Figure 1: Diagrama do fluxo de execução e medição. Fonte: autoria própria.

1. Etapas:
   - Script_de_Teste: O orquestrador do experimento, responsável por iniciar e parar a medição e executar a tarefa;
   - PyrAPL: A ferramenta de medição , responsável por monitorar e reportar o consumo de energia;
   - Implementação: Representa a variável independente. Ele a versão Python Puro ou a versão Pandas;
   - Arquivo_CSV: A fonte de dados;
   - Console: O destino da saída dos dados.
2. Fluxo:
   - O Script_de_Teste primeiro aciona o PyrAPL para iniciar a medição;
   - Imediatamente após, ele executa a tarefa de processamento (seja ela a versão Pura ou Pandas);
   - A Implementacao executa as três etapas descritas do artigo:
     (a) Lê o Arquivo_CSV;
     (b) Guarda os dados em memória (etapa interna);
     (c) Entra em um loop para iterar sobre os registros e imprimir cada um no Console.
   - Quando a tarefa termina, o Script_de_Teste manda o PyrAPL parar a medição;
   - Finalmente, o PyrAPL retorna o dado da métrica principal: energiaConsumida (Joules).
   - O bloco alt (alternativa) significa que este fluxo completo é executado uma vez para a implementação em Python Puro e outra vez para a implementação em Pandas, permitindo a comparação.

3 Definição do Experimento

Para responder às perguntas de pesquisa e validar as hipóteses, foi estruturado um experimento empírico. O experimento é desenhado para comparar diretamente o consumo de energia de duas abordagens de software distintas (Python puro vs. Pandas) ao executar uma tarefa idêntica.

3.1 Medição das Métricas

As métricas definidas na Seção 2 serão coletadas da seguinte forma:

3.1.1 Consumo Energético (Métrica Principal):

- Ferramenta: A medição será realizada utilizando a ferramenta PyrAPL. Esta ferramenta é projetada para medir a pegada energética da execução de código Python;
- Granularidade: O consumo de energia será medido em joules. A medição será configurada para capturar o consumo total de energia da CPU durante a execução completa da tarefa de processamento de cada aplicação.

3.1.2 Tempo de Execução (Métrica Secundária):

- Ferramenta: Será feito via implementação pegando o tempo de início e o tempo de fim e calculando o tempo de execução na aplicação;
- Granularidade: O tempo será medido em segundos, registrando o tempo total de execução da aplicação.

3.2 Orquestração do Experimento

A orquestração do experimento será gerenciada por um script automatizado para garantir que ambas as aplicações sejam executadas sob condições idênticas. O processo segue os seguintes passos:

1. Execução da Tarefa: Será executado de forma automatizada ambas as implementações respeitando número de 40 execuções. Será compilado os resultados em uma tabela, que será utilizada como base de dados para os cálculos estatísticos;
2. Definição da Tarefa: Ambas as aplicações executarão a mesma tarefa funcional: ler o arquivo CSV de dados de pessoa física, guardar os dados em memória, iterar sobre todos os registros e imprimir os registros no console;
3. Variável Independente: Será feito as execuções de forma automatizada, sendo que primeiro as execuções da implementação 1, e posteriormente as execuções da implementação 2, que constituem nossa variável independente;
    - Implementação 1: A aplicação utilizando Python puro;
    - Implementação 2: A aplicação utilizando a biblioteca Pandas.
4. Monitoramento: A ferramenta PyrAPL será iniciada pelo script imediatamente antes da execução da tarefa e interrompida logo após sua finalização, isolando assim a medição do consumo de energia.

3.3 Repetição e Validação

Para garantir a confiabilidade estatística dos resultados e mitigar o impacto de anomalias do sistema operacional ou "ruídos" de execução, o experimento seguirá um protocolo de repetição.

- Número de Execuções: Cada uma das duas implementações (Python Puro e Pandas) será executada de forma independente pelo menos 40 vezes;
- Coleta de Dados: O consumo de energia (em Joules) e o tempo de execução (em segundos) serão registrados para cada uma dessas execuções;
- Análise: Os dados serão utilizados para a análise estatística, permitindo uma comparação robusta para validar ou refutar as hipóteses de pesquisa.

3.4 Calibração do Ferramental e Execução do Piloto

Para garantir a acurácia dos resultados, será conduzida uma execução piloto (ciclo de testes inicial). O objetivo é duplo:

- Validar a Metodologia: Assegurar que os scripts de orquestração e a captura de dados (via pyRAPL) estão operando corretamente;
- Estabelecer o Baseline: Quantificar o OVERHEAD energético introduzido pelo próprio script de monitoramento. Este valor de referência é crucial para isolar o consumo real das implementações (Python Puro vs. Pandas).

Os dados gerados nesta fase piloto (exemplo abaixo) são, portanto, ilustrativos e servem para validar o formato da saída. Eles serão substituídos pelos dados finais obtidos na execução em volumetria (N=X).

```
|Implementacao|Consumo Medio de Energia (J)|Tempo Medio de Execucao (s)|
| :--- | :--- | :--- |
| Python Puro | 192.45 J | 51.33 s |
| Pandas | 148.12 J | 39.78 s |
```

3.5 Análise (Exemplo)

Os dados preliminares indicam que a Implementação 2 que se refere ao Pandas foi mais eficiente em termos de consumo de energia (148.12 J contra 192.45 J) e também mais rápida (39.78 s contra 51.33 s) para a tarefa definida.

3.6 Artefatos de Execução

Os logs de dados (CSV) e as visualizações (PNG) geradas via Bibliotecas (Matplotlib e pyRAP) serão versionados como artefatos de cada ciclo de execução para garantir a reprodutibilidade.

References

[1] HINDLE, Abram. **Green Software Engineering**. Springer, 2022.

[2] PINTO, Gustavo; CASTOR, Fernando. **Energy Efficiency: A New Concern for Application Software Developers**. Communications of the ACM, v. 60, n. 12, p. 68-75, 2017.

[3] COUTO, Marco et al. **The Green Lab: Experimentation on Software Energy Efficiency**. IEEE Software, v. 35, n. 1, p. 87-92, 2018.

[4] PEREIRA, Raul et al. **Energy Efficiency Across Programming Languages**. Sustainable Computing: Informatics and Systems, v. 30, p. 100512, 2021.

[5] RAMALHO, Luciano. **Fluent Python**. 2. ed. O'Reilly Media, 2022.

[6] MCKINNEY, Wes. **Python for Data Analysis: Data Wrangling with Pandas, NumPy, and Jupyter**. O'Reilly Media, 2022.

[7] Python Software Foundation. **Python 3 Documentation**. Disponível em: https://docs.python.org/3/. Acesso em: 01 nov. 2025.

[8] Pandas Development Team. **Pandas Documentation**. Disponível em: https://pandas.pydata.org/docs/. Acesso em: 01 nov. 2025.

[9] USC-Information Sciences Institute. **RFC 4180: Common Format and MIME Type for Comma-Separated Values (CSV) Files**. 2005. Disponível em: https://www.rfc-editor.org/rfc/rfc4180. Acesso em: 02 nov. 2025.

[10] PowerAPI Team. **pyRAPL: Python RAPL Measurement Toolkit**. Disponível em: https://github.com/powerapi-ng/pyRAPL. Acesso em: 02 nov. 2025.

[11] MICROSOFT. **Principles of Sustainable Software Engineering**. Disponível em: https://learn.microsoft.com/en-us/training/modules/sustainable-software-engineering-overview/. Acesso em: 02 nov. 2025.