Northeastern University

(Computer Science)

Team Indigenous Tech Circle

Green Software Engineering

Author: **Liyao Zhang**
ID: 002837359
E-mail:
zhang.liya@northeastern.edu

Author: **Wei Zhao**
ID: 001969892
E-mail:
zhao.wei2@northeastern.edu

Author: **Jun Liu**
ID: 001611944
E-mail: *liu.jun5@northeastern.edu*

Author: **Ruijin Zhang**
ID: 002622440
E-mail:
zhang.ruijin@northeastern.edu

Tutor: **Yvonne Coady**
Department: Computer Science
E-mail: *m.coady@northeastern.edu*

October 2, 2024

Contents

1 Problem Statement

The rapid expansion of the Information and Communication Technology (ICT) sector is significantly increasing global greenhouse gas emissions, accounting for approximately 2-3% of total emissions. As a core component of ICT systems, software indirectly drives substantial energy consumption through its use of hardware resources. However, there is currently no unified framework to measure the resource and energy efficiency of software, making it difficult for developers and researchers to compare results, replicate findings, or systematically improve energy usage. Despite growing awareness of the environmental impact of software, the lack of standardized tools and methods has led to fragmented and inconsistent efforts in evaluating and optimizing software efficiency, hindering progress in reducing its environmental footprint. The issue of **green software** is becoming increasingly urgent. As global attention to sustainability rises, the carbon footprint of software has emerged as a critical challenge. Traditional software development often overlooks energy efficiency and carbon emissions, leading to resource waste. While green technologies are advancing in many industries, the software sector lags behind. Without a standardized framework for green software measurement, unchecked software-driven energy consumption will continue to burden the ICT industry and slow global progress toward carbon neutrality and sustainability goals.

2 Proposed solution

Our solution focuses on designing a data processing and analysis system that prioritizes energy efficiency, making it greener. This is achieved through several key design choices at different stages of the system architecture, including data collection, storage, and front-end optimizations. Related work on green software engineering either focus on best practices but without quantifying their impact on a holistic basis or propose an evaluation methodology without concrete examples from real-world applications. By drawing from the lessons learned in the literature, our solution incorporates green software principles while providing a practical example of implementation in a real-world context. We compare and contrast two distinct system design architectures to explore and discuss the following: first, the options to consider for creating a greener system design in real-world scenarios; second, methods for quantifying the impact of greener implementations using established evaluation techniques; and finally, the specific components that may significantly influence energy consumption.

2.1 Literature Review

Previous research indicates that while practitioners are aware of and care about energy efficiency in software engineering, they often face significant barriers due to a lack of comprehensive information and tools available. Manotas et al. conducted an empirical study that highlights this challenge, revealing that developers need support to incorporate such practices into their workflows [1]. Mourão et al. conducted a systematic mapping study that categorizes and synthesizes various researches concerning sustainable software engineering, emphasizing the need for more validation studies focusing on the impacts of sustainability in the future [2]. Brunnert proposed specific green software metrics, resource demand, as the basis to help developers allocate the energy consumption of

the entire server to individual software components, yet no substantial prototype has been implemented to finalize the carbon emission calculation [3]. As web technologies continue to thrive and to be adopted, various resources are available to the community, aiming to bring awareness regarding sustainable software engineering [4]. Meanwhile, best practices identified in mobile application development adapting energy constraints on mobile devices can be effectively translate to web development. Rani et al. explored 20 mobile energy patterns that could be adapted for web and interviewed several expert web developers to challenge them. Finally, two of the ported patterns were evaluated against their antipatterns in terms of energy consumption [5]. A more detailed analysis focusing on architectural design was also within the domain of mobile development, where Singh proposed a novel RMVRVM paradigm to remove the heavylifting to the back-end to reduce unnecessary data transferred to the client devices, resulting in significant decrease in battery consumption and improvement in response time. [6]

2.2 Architecture Design

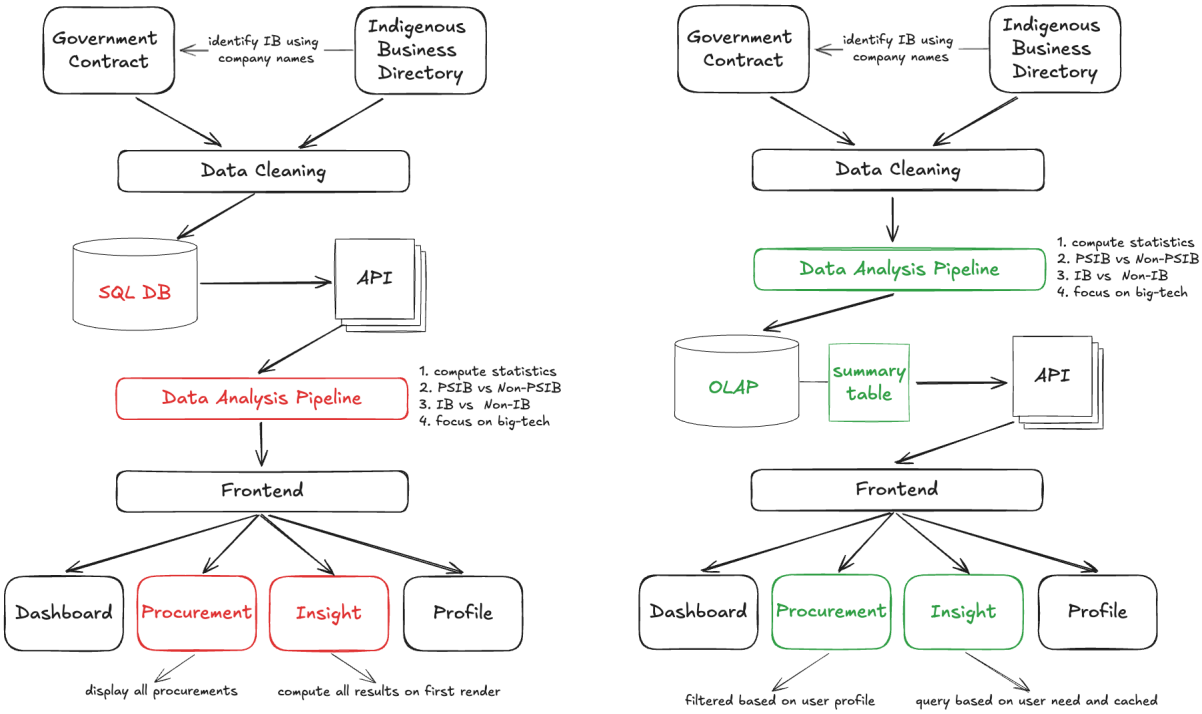

Figure 1: Two possible system architecture diagrams

2.2.1 Data Collection

Instead of relying on real-time updates, our system will primarily conduct historical retrospective analysis. This shift is driven by the fact that the procurement data we are handling is not updated frequently. Real-time data collection can be resource-intensive, especially in terms of energy consumption and network usage.

2.2.2 Data Storage

The cleaned data will go through a data analysis pipeline and will be stored in a database. However, the type of database and the order of the process may vary. An OLAP data warehouse is preferred over a traditional relational schema because the warehouse promotes summary tables that store computed results which can be efficiently retrieved to generate charts and diagrams. Moreover, all the new data is supposed to go through the pipeline only once.

2.2.3 Front-end Optimization

While there are overwhelming amount of tips and best practices regarding sustainable web development, we decided to focus on one major aspect: allocate resources based on user needs and reduce unnecessary data flow resulted. For example, users will see procurement contracts filtered based on their company profile and they will be able to further edit it on the fly. As for the insights, data from different sections will be queried upon user's requests and rendered accordingly.

References

[1] I. Manotas et al. "An Empirical Study of Practitioners' Perspectives on Green Software Engineering". In: *Proceedings of the 38th International Conference on Software Engineering (ICSE '16)*. New York, NY, USA: Association for Computing Machinery, May 2016, pp. 237–248. DOI: 10.1145/2884781.2884810.

[2] B. C. Mourão, L. Karita, and I. do Carmo Machado. "Green and Sustainable Software Engineering - A Systematic Mapping Study". In: *Proceedings of the XVII Brazilian Symposium on Software Quality (SBQS '18)*. New York, NY, USA: Association for Computing Machinery, Oct. 2018, pp. 121–130. DOI: 10.1145/3275245.3275258.

[3] A. Brunnert. "Green Software Metrics". In: *Companion of the 15th ACM/SPEC International Conference on Performance Engineering (ICPE '24 Companion)*. New York, NY, USA: Association for Computing Machinery, May 2024, pp. 287–288. DOI: 10.1145/3629527.3652883.

[4] R. D. Caballar. "We Need to Decarbonize Software". In: *IEEE Spectrum* (Mar. 2024). Retrieved September 15, 2024. URL: https://spectrum.ieee.org/green-software.

[5] P. Rani et al. "Energy Patterns for Web: An Exploratory Study". In: *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Society*. Lisbon, Portugal: ACM, Apr. 2024, pp. 12–22. DOI: 10.1145/3639475.3640110.

[6] L. Singh. "RMVRVM – A Paradigm for Creating Energy Efficient User Applications Connected to Cloud through REST API". In: *Proceedings of the 15th Innovations in Software Engineering Conference (ISEC '22)*. New York, NY, USA: Association for Computing Machinery, Feb. 2022, pp. 1–11. DOI: 10.1145/3511430.3511434.