Energy Efficient Coding Practices for Sustainable Software Development

R. Manimegalai[1], Srivatsan Sandhanam[2], A. Sunitha Nandhini[3], Priyam Pandia[4]

{drrm@psgitech.ac.in[1*], srivatsan.santhanam@sap.com[2], asn@psgitech.ac.in[3], priyam.pandia@sap.com[4]}

Professor, PSG Institute of Technology and Applied Research[1], *Corresponding Author
VicePresident, SAP Labs India[2],

Assistant Professor (Sl.Gr), PSG Institute of Technology and Applied Research[3],
Principal Architect, SAP Labs India[4]

Abstract. Cloud computing has revolutionized the IT landscape, offering unparalleled scalability and flexibility to both businesses and end-users. But there's growing concern about how much electricity cloud data centers consume. The increasing need for cloud services has made it imperative to conduct research on maximizing energy efficiency in cloud computing. This research study compares two different applications to examine energy-efficient coding methods in cloud computing and provides an overview of current green data center architecture aspects. The goal of this effort is to pinpoint procedures that can drastically lower energy usage in cloud environments. The two main steps in the suggested methodology are as follows. Initially, the energy usage is evaluated for two distinct programs that are operating in a cloud environment. First, the energy consumption is assessed for two different applications running in a cloud environment: Application-A coded in JavaScript, which utilizes conventional coding practices, and Application-B coded in Java, which integrates advanced energy-efficient coding techniques. Both applications are subjected to a series of experiments to measure power consumption and performance metrics, to quantify the energy savings achieved by Application-B which employs energy efficient coding practices. Second, coding practices employed in each application are explored to identify key differences that contribute to their varying energy efficiency. Through code profiling and analysis, specific lines of code, algorithms, and design principles that make Application-B more energy-efficient than its counterpart, are highlighted. It has been observed that green-data-center alone will not contribute to overall sustainable development and it is important to incorporate energy efficient coding practices in software development to support sustainable environment. This research paper contributes to the ongoing green-efforts to reduce the environmental impact of cloud computing by providing practical insights for energy efficiency enhancement, while optimizing resource utilization. The findings of this research work hold immense value for cloud service providers, software developers, and policymakers seeking to make informed decisions regarding energy-efficient coding practices in the cloud computing domain.

Keywords: Cloud Computing, Energy-Efficient Coding Standards, Sustainable Software Development.

ICSETPSD 2023, November 17-18, Coimbatore, India
Copyright © 2024 EAI
DOI 10.4108/eai.17-11-2023.2342635

1. Introduction

In an era where the digital realm is omnipresent, cloud computing stands as a cornerstone of modern information technology, offering unparalleled flexibility and scalability. The cloud's ability to deliver services and applications at an unprecedented scale has transformed industries, streamlined operations, and empowered businesses with newfound capabilities. Yet, beneath the seemingly boundless horizon of cloud computing lies a growing and urgent concern i.e. its environmental footprint. The energy consumption of data centres, which power the cloud, has escalated to unprecedented levels, and this surge in energy usage poses serious challenges to sustainability and resource conservation. This research paper embarks on understanding the as-is-state of green-data-centre design within the context of cloud computing and studies the impact of Cloud computing coding practices with implication for energy savings. As the global drive towards sustainability and environmental responsibility intensifies, the need for energy-efficient solutions is more compelling than ever.

In the quest to uncover energy-efficient solutions for cloud computing, the role of software development practices cannot be overstated. Code, the lifeblood of software applications, is instrumental in determining the energy consumption of cloud-based services. This work delves into the intricate world of coding practices, comparing two distinct applications running in a cloud environment to discern the extent to which energy-efficient coding can reduce energy consumption. Specific coding techniques, algorithms, and design principles that can significantly enhance energy efficiency in cloud-based software arerecommended.

Complementing the quest for energy-efficient coding practices, this work ventures into the realm of data-centre design with a focus on environmental sustainability. Data centres are the powerhouses of cloud computing, and as such, their design and operations play a critical role in determining the overall energy efficiency of the cloud. The main objective of this work is to contribute to the growing body of knowledge surrounding energy-efficient practices in cloud computing, with an emphasis on tangible, real-world solutions. The findings of this work has the potential to guide and inspire cloud service providers, software developers, and policymakers in their endeavours to navigate the intersection of technological innovation and environmental stewardship. In a world where the demand for cloud services continues to surge, the need for sustainable cloud computing solutions is paramount, and this research paper seeks to pave the way for a more energy-efficient and environmentally conscious cloud computing landscape. In nutshell, software development can be made energy efficient and sustainable not just with Green-data-centres design practices but needs to be coupled with energy efficient coding practices, user awareness and responsibility.

2. Related Work

Green cloud computing has become a prominent subject of research in today's tech-driven world, driven by the increasing need for large-scale data storage and computational capabilities.

IT companies are shifting towards cloud computing to expand their infrastructure while maintaining ecological and cost-effective practices. Cloud computing offers an efficient means of virtualizing servers and data centers, with a strong focus on energy efficiency. Jerome Rocheteau et al. have explained how to estimate Java source code energy usage and what inferences can be made from these measurements [1]. The work done in [1] presents a structured framework for optimal coding practices, employing a semantics grounded in quantitative metrics that directly correlate with the savings in time, memory, and energy achieved through the implementation of these practices. Furthermore, it elucidates the methodology for assessing such codebases, ensuring consistent and reliable measurements by integrating both physical and logical sensors. It encourages the enhancement of both the measurement platform and the measurement protocol. Nayan Agrawal et al. have presented a review on various aspects of green standards and green approaches, strategies, technologies proposed in the literature [9]. Barriers and benefits of green computing are discussed in [9] and it provides a concise overview of green cloud computing, addressing its challenges and global benefits.

Daniele D'Agostino et al. have presented the state of the art of powerful, less power-hungry processors and energy-aware tools and techniques for energy-efficient computing in scientific programming [12]. Cutting-edge solutions encompassing both hardware and software innovations, along with methodological approaches aimed at enhancing the energy efficiency of scientific software are presented in [12] including some intriguing System-on-Chip (SoC)-based solutions. Giuseppe Procaccianti et al. have highlighted that there is a growing recognition of the significance of energy efficiency in software development [3]. Despite numerous empirical studies in this field, there is a lack of empirically validated guidelines for creating energy-efficient software. The primary objective of work in [3] is to assess the impact of best practices derived from prior studies on achieving energy-efficient software. Specifically, the study aims to quantify the energy savings resulting from these practices, identify the resources they affect, and elucidate potential trade-offs related to energy consumption. An empirical experiment is conducted in a controlled environment and two distinct green software practices are applied to two different software applications: query optimization in MySQL Server and the use of the *sleep* instruction in the Apache web server [3].

Muhammad Salam et al. have delved into the realm of green and sustainable software development, a concept that aims to create software solutions that are both eco-friendly and capable of meeting the present and future needs of users while minimizing their adverse environmental and societal impacts [6]. This approach is gaining prominence, particularly in the context of Global Software Engineering (GSE), where multi-sourcing vendor organizations are actively integrating environmentally responsible practices into their software development projects.
Candy and Pena Vinces have presented a overview on conventional accounting systems that currently operate solely from a financial perspective, disregarding essential environmental data such as environmental costs and corporate expenditures related to environmental impact [16]. The proposed methodology in [16] presents a novel framework called the Green Accounting

System (GAS), which incorporates firms' environmental impact into their accounting processes. Rajni Sehgal et al. have presented a study on the industry commonly embraces the adoption of refactoring techniques to rectify code flaws identified as code smells within a given project [15]. This practice is widely recognized as a pivotal means of enhancing software quality. Refactoring is a technique frequently employed by developers as software evolves during maintenance phases. This study seeks to delve into the substantial role that refactoring plays in reducing power consumption. Haitao Steve Zhu et al. have introduced Eco, an innovative programming model tailored for the development of Java-like applications with sustainability-focus [2]. Eco features novel abstractions that enable the shaping of supply and demand, thereby engaging programmers in the realm of sustainability management. It fosters energy-aware and temperature-aware programming encouraging developers to consider energy efficiency and temperature optimization in their application designs [2].

Abdulaziz Alarifi et al. have presented the escalating demand for cloud computing services, driven by the accelerating digital transformation and the inherent adaptability of cloud technology [11]. The pressing need to enhance the electrical energy efficiency of cloud data centers is highlighted in [11] and it introduces an Energy-Efficient Hybrid (EEH) framework aimed at optimizing the utilization of electrical energy in data centers, and its performance is meticulously assessed. Luca Ardito et al. have discussed the models, and tools for the creation and advancement of environmentally-friendly software in [3]. Energy efficiency within the realm of information technology has gained considerable prominence in recent years. It necessitates not only a shift in mindset among software developers and designers but also the creation of models and tools capable of quantifying and mitigating the impact of software on the energy consumption of the underlying hardware [3]. A conceptual framework that offers a comprehensive perspective on the existing strategies, models, and tools, is presented in [3] for the development of more environmentally friendly software.

Giuseppe Procaccianti et al. have evaluated the energy-saving impact of best practices derived from prior research with the aim of identifying the resources influenced by these practices and potential trade-offs concerning energy consumption. An empirical experiment is conducted in a controlled environment, implementing two distinct Green Software practices within two software applications: query optimization in MySQL Server and the use of the *sleep* instruction in the Apache web server. R. Pereira et al . have conducted an extensive examination of the energy usage patterns within various Java Collection Framework (JFC) implementations in [5]. The proposed JFC framework systematically analyzes the energy consumption associated with each method across implementations while considering different data sizes. It introduces an energy optimization strategy for Java programs, by identifying the most energy-efficient methods within each implementation. M. Kumar et al have investigated the energy efficiency of Java, a widely used language in ICT systems [7]. The research delves into the energy consumption characteristics of Java, examining data types, operators, control statements, exceptions, and objects at a granular level. The study leverages Intel Running Average Power Limit (RAPL) technology to measure the relative power consumption of small code segments.

Several significant insights and observations are uncovered through this investigation, which may serve as a foundation for standardizing Java's energy consumption attributes.

Biswajit Saha has discussed the current trends in efficient energy consumption, e-waste recycling and management, IT products eco-labeling and longevity, data centers optimization, virtualization, etc. in [8]. The concept of Green Computing is more critical as we move towards smart cities. Shokhista et al. have identified and assessed a total of 169 metrics in terms of their suitability for analyzing energy consumption within the context of invasive software development processes in [10]. This comprehensive study highlights the significance of the questions raised in this field while also revealing its current state of development. There is a notable absence of evolutionary research and the absence of a method for assessing energy consumption at various stages of software product development. These findings underscore the importance of technical work in this domain and emphasize the necessity for its further advancement. Z. Ournani et al have conducted a comprehensive study with two primary objectives in [13]. First, it seeks to evaluate the energy consumption associated with various well-known I/O library methods, examining whether different read/write methods exhibit varying energy consumption patterns. Specifically, using micro-benchmarks, the study assesses the energy consumption of 27 I/O methods across various file sizes to discern the most and least energy-efficient methods. Artem Kruglov et al. have reviewed the responsibility of minimizing energy consumption within a system has predominantly fallen on the shoulders of hardware developers in [14]. This emphasis stems from the prevailing belief that hardware constitutes the primary energy consumer.

3. Proposed Framework for Collecting Energy Footprint and Metrics

This work has proposed two architectures, namely, the design time architecture and run time architecture, which can be used for collecting energy efficiency metrics that shed light on coding practices, code patterns and key data points for areas of optimization with energy efficiency in focus. The architecture defined is in two different dimensions and for consumers of the same to become aware of the energy consumption done during the application development and while the application is getting executed in the data center during its usage by the customers. Classification of these two scenarios has been done as design time, i.e. during application development and run time, i.e. when the application is used. This section presents details of the proposed design time and run time architecture that can be employed for green coding practices.

3.1. Design Time Architecture for Green Coding

It is imperative that the developers have a local way to derive the energy footprint that their code is going to produce during the Application-Build and run. Hence, the proposed architecture offers an IDE plugin, which can be installed in the development machine by the developers and used for measuring the energy emissions. This plugin can suitably work with all types of applications be it front or backend applications and can be used in IDEs such as Jet Brains

IntelliJ, Virtual Studio Code, Atom, etc. This would give an insight to the application developers to see how and where energy emission is happening the most and how it can be reduced. The dashboard to visualize the same can be hosted locally. The dashboard fetches the data from the plugin and show the overall energy emission from the application and the details of the code which has been modified by the developers. Fig. 1. Illustrates the design architecture along with Energy Metric Measuring Plugin and Metrics Dashboard with Energy Statistics.

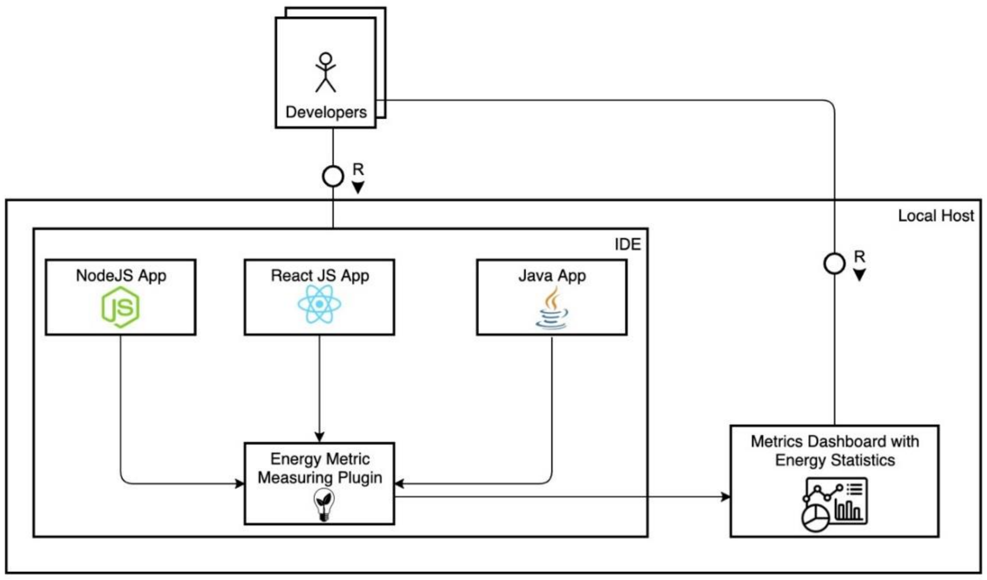
Developers
R
Local Host
IDE
NodeJS App
React JS App
Java App
R
Energy Metric Measuring Plugin
Metrics Dashboard with Energy Statistics

Fig. 1. The Proposed Design Time Architecture for Green Coding

3.2. Run Time Architecture for Green Coding

In the proposed architecture for run time scenario, the applications are containerized and are deployed on AWS Hyperscaler. For integration between front-end and back-end application, AWS API gateway is used. The database used is AWS native DB and it's a choice for SQL or No SQL database, that can be made. A lightweight agent is implemented and deployed along with the applications for monitoring the energy consumption and emission. The data from the agent is also logged in the elastic logging service and aggregated. The logs from AWS native applications are read from the hyper-scalar logging service such as AWS cloud watch etc. Data from both the logging sources are read and different dashboards are derived from it. Sustainability report is a meaningful dashboard for senior leaders and customers who consider the net carbon emissions from the company and the services consumed. Business metrics along with energy emissions data are portrayed in the dashboard for product managers.

Technical and infrastructure metrics, to observe the operating system, performance, network latency, and additionally the energy emissions while the applications are running is particularly useful for technical leads and developers. Fig. 2. Illustrates the run-time architecture with sustainable report with Dashboard. The runtime architecture of cloud-based services that incorporate energy metrics measurement using elastic logging involves several key aspects to efficiently monitor, collect, and analyze energy-related data. Some of the important components and considerations in run-time architecture are presented in Table 1. A strong framework that permits real-time monitoring, historical analysis, and anomaly detection of energy consumption trends is provided by the runtime architecture suggested for cloud-based applications that incorporate Elastic logging for energy metric measurement. IT organizations can optimize resource usage, reduce operational costs, and minimize their environmental impact in the cloud environment, by effectively collecting, storing, and analyzing energy metrics.

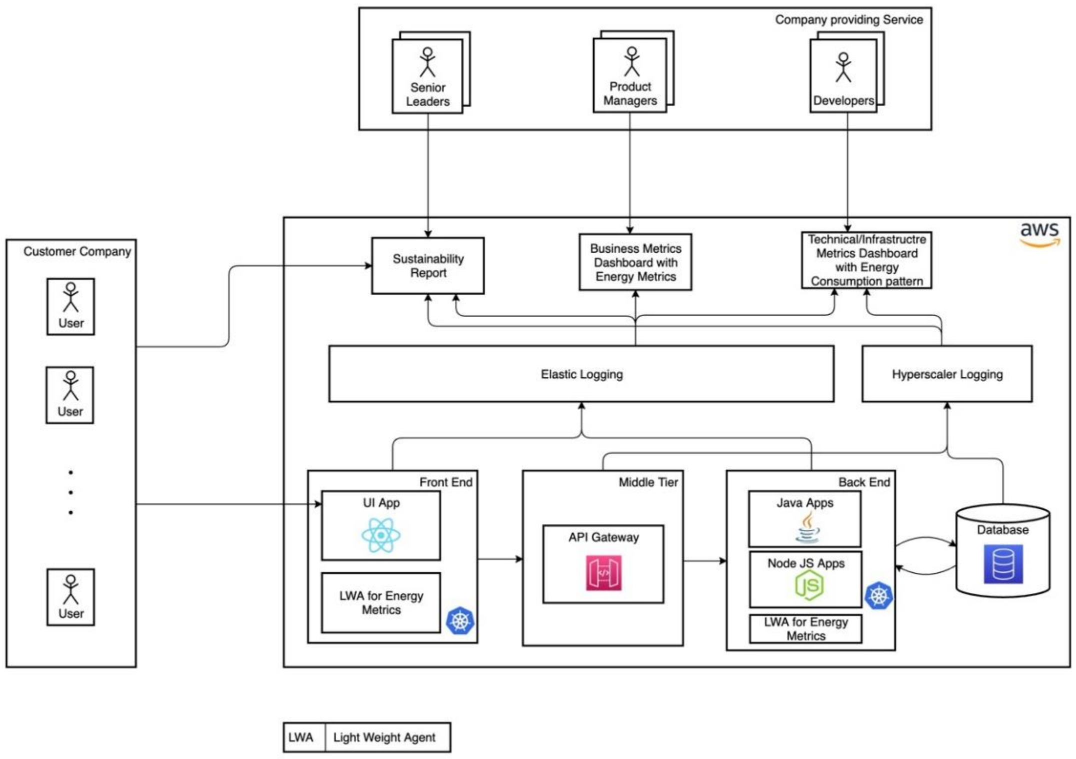

Fig. 2. The Proposed Run Time Architecture for Green Coding

Table 1. Essential Components in Runtime Architecture

Type of Component	Description
Service Components	It is important to design cloud-based service components such as Virtual Machines (VMs), containers, micro-services, and networking elements to collect data related to CPU usage, memory

	utilization, disk I/O, and network activity, in order to measure energy metrics effectively
Data Collection Agents	These agents are collect energy-related metrics from various service components.
Elastic Logging Stack	Elasticearch is used to store the energy metrics data efficiently and Logstash is employed to collect, transform, and enrich data from various sources, including data collection agents. Kibana is considered for its user-friendly interface to visualize and analyze the collected metrics.
Energy Metrics Indexing	Energy metrics data should be appropriately indexed within the Elastic Stack for effective storage and retrieval and faster query execution.
Real-time Monitoring	DevOps teams and administrators can use Kibana dashboards to visualize energy metrics as they are collected, enabling immediate insights into resource-intensive operations or unexpected spikes in energy consumption.
Historical Analysis	Historical data can be valuable for capacity planning, performance optimization, and identifying long-term patterns
Integration with Cloud Management Services	Runtime architecture should integrate cloud management services such as AWS CloudWatch, Azure Monitor, or Google Cloud Monitoring to ensure effective control and management of energy consumption to provide optimal cost and resource allocation.
Security and Compliance	Access controls, encryption, and authentication mechanisms should be implemented to protect sensitive energy metrics data. Additionally, compliance requirements, such as GDPR or HIPAA, must be adhered to when handling energy-related data
Anomaly Detection and Machine Learning	Machine learning models can be integrated into the Elastic Stack to detect anomalies in energy consumption. Algorithms can identify deviations from expected energy usage patterns and trigger alerts or automated actions to address these issues promptly.

3.3. Sample Code Snippet in Java for Energy Efficient Coding

This Section discusses general principles for writing energy-efficient code in Java, along with simple examples. Table 2. presents the general principles and guidelines used to write energy-efficient code in Java.

Table 2. Principles for Writing Energy Efficient Coding in Java

Coding Principle	Examples
Minimize CPU Usage	Use of efficient algorithms and data structures and avoiding busy waiting and excessive polling
Optimize Memory Usage	Use of data structures that minimize memory consumption and avoiding unnecessary object creation

Reduce I/O Operations	Do batch I/O operations wherever possible and use of asynchronous I/O for non-blocking operations
Manage Threads Wisely	Use thread pools to limit the number of concurrent threads and consider using lightweight threads instead of heavyweight threads

Observation Case 1: **Use of Threads to Manage Concurrency**

```
Import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.TimeUnit;
 public class EnergyEfficientExample {
    public static void main(String[] args) {
        // Use a thread pool to manage concurrent tasks
efficiently.
        int numThreads =
Runtime.getRuntime().availableProcessors();
        ExecutorServiceexecutorService =
             Executors.newFixedThreadPool(numThreads);

        // Perform a CPU-intensive task using multiple
threads.
        int tasksToRun = 10;
        for (inti = 0; i<tasksToRun; i++) {
            executorService.submit(() -> {
            long result = fibonacci(40);
            // Calculate Fibonacci(40) as an example.
                System.out.println("Result: " + result);
            });
        }
        // Shut down the executor service gracefully.
        executorService.shutdown();
        try {
            executorService.awaitTermination(1,
TimeUnit.MINUTES);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
    private static long CalcFunc(int n) {
        if (n <= 1) {
            return n;
        }
        ReturnCalcFunc(n - 1) + CalcFunc(n - 2);
    }
}
```

Observation Case 2: Loop Optimizations Leading to Better Resource Usage

Energy Inefficient Way: In this code, a simple for loop is used to iterate through the array to sum its elements. While this code works correctly, it may not be the most energy-efficient approach because it involves multiple iterations, especially for larger arrays.

```
public class EnergyInefficientSum {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4, 5};
        int sum = 0;

        for (int i = 0; i<numbers.length; i++) {
            sum += numbers[i];
        }
       System.out.println("Sum of elements: " + sum);
    }
}
```

Energy Efficient Way: In this energy-efficient code, enhanced for loop i.e. for-each loop is used to iterate through the array. This approach is generally more energy-efficient because it can potentially optimize the loop execution at a lower level when compared to the traditional for loop. Energy-efficient coding often involves minimizing unnecessary operations and leveraging language features and optimizations to reduce resource consumption. In this example, using the enhanced for loop is a more efficient and cleaner way to calculate the sum of array elements.

```
Public class EnergyEfficientSum {
    Public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4, 5};
        int sum = 0;

        for (int number : numbers) {
            sum += number;
        }

        System.out.println("Sum of elements: " + sum);
  }
}
```

Energy efficiency at Data Centre level is calculated using Equation (1).

pwrIdx = execTime * (cpuUsgPerCore + memUsg) * DCSustainabilityIndex … (1)

4. Experimental Results

In our quest for more sustainable and energy-efficient cloud applications, we conducted a comparative analysis between two prominent cloud applications developed in Java and JavaScript. Our goal was to evaluate their energy consumption patterns to identify potential optimizations and make informed decisions about the choice of technology stack for future cloud-based projects.

The first application A, developed in Java, exhibited a distinct energy consumption profile. Java applications are known for their robustness and performance. Our analysis revealed that this Java-based application consistently required optimal computational resources.The Java Virtual Machine (JVM), offering benefits such as memory management, reduced the overhead that increased the overall efficiency of the application. Consequently, the Java application consumed less energy during peak usage and idle periods, making it more energy-efficient. On the other hand, the cloud applicationdeveloped in JavaScript (Application-B) demonstrated a different energy consumption pattern. JavaScript, commonly used for web applications, is executed directly by web browsers, which can lead to higher resource requirements compared to Java. As a result, the JavaScript application exhibited a more energy-consuming profile. During peak usage, it consumed slightly more computational resources as compared to Java Application-But this being an interpreted language, based application required more memory over time which resulted in higher energy consumption. Additionally, JavaScript's asynchronous nature allowed the application to better adapt to varying workloads.

However, it's important to note that the choice between Java and JavaScript should not be based solely on energy consumption. Other factors, such as application requirements, development speed, and developer expertise, must also be considered. While JavaScript may offer concurrency and less verbosity advantages in some scenarios, Java remains a strong choice for applications requiring high performance and scalability and is energy efficient as well. Metrics are measured on the proposed architecture using the monitoring tools such as New Relic and Kibana. Infrastructure Metrics are derived using tools such as Sysdig. Using this metrics, power consumption is derived for two major factors such as CPU utilization and memory consumption. This monitoring is performed across all the components in the architecture and metrics are collected and compared over a period of 10-days to validate the results. The components in the architecture are subjected to the same load from the UI to API gateway to back-end services to monitor how each service manages load and what is the pattern of power consumption under the same stress.

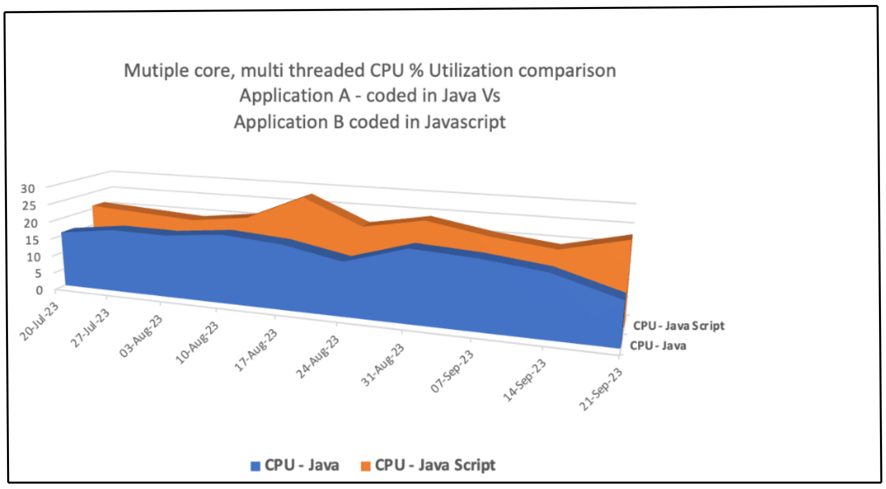

Fig. 3. CPU Consumption for Applications using the Proposed Run-time Architecture with Constant Load

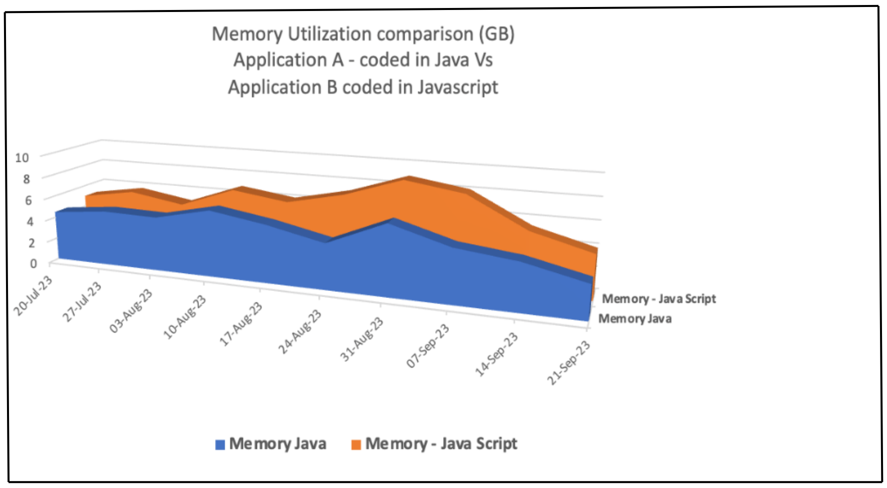

Fig. 4. Memory Utilization for Applications using the Proposed Run-time Architecture with constant Load

Further analyzing and comparing the power consumption patterns for back-end applications revealed that Java Applications are more efficient and less carbon emitting when compared to the Java Script application for the same load. There is always 2 to 4 KW difference in the power consumption of the Java Script application and the Java application considering that the both the services are being deployed on the similar configuration of the containers. This energy profile was further optimized with energy efficient code patterns listed earlier leading to further power savings up to 17% without any noticeable impact to user performance during peak loads.

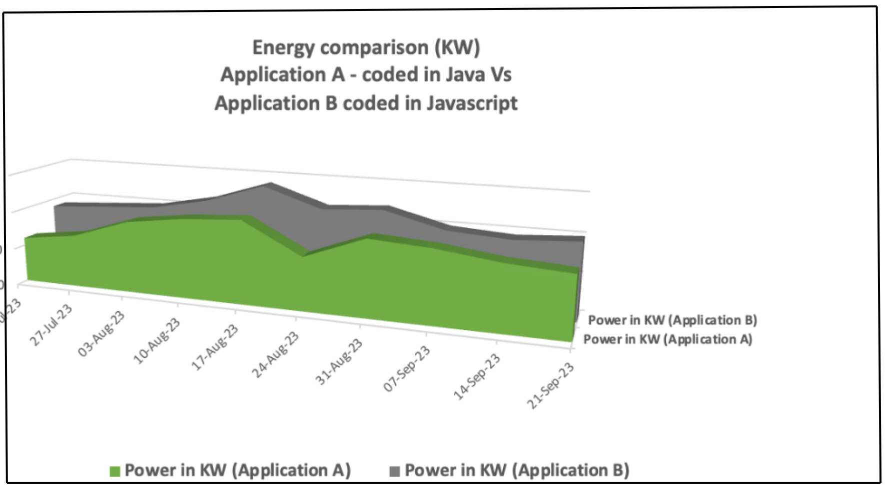

Fig. 5. Comparison of Power Consumption Pattern for Application-A and Application-B

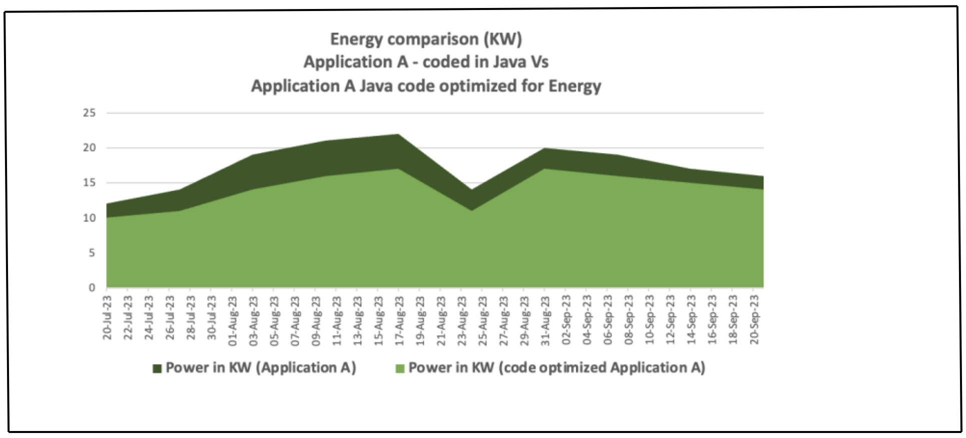

Fig. 6. Comparison of Power Consumption Pattern for Application-A and Application-B

Resource Utilization: The Java application exhibited similar CPU utilization when compared to the JavaScript application under similar workloads. JavaScript applications tended to consume less memory when compared to their Java counterparts.

Energy Consumption: The JavaScript application consistently consumed more energy than the Java application in all tested scenarios. The energy consumption of the JavaScript application was on average 15% higher than that of the Java application.

Scalability: Both applications scaled horizontally to accommodate increased user loads, but the JavaScript application showed more predictable and linear scaling in terms of energy consumption.

Energy Efficient Code Optimization Patterns: Adapting energy efficient coding practices can further reduce power consumption by at least ~17% without any noticeable impact on response times.

5. Conclusions

The choice of programming language can have a measurable impact on the energy consumption of cloud applications. In this work, it is observed that, the Java-based cloud application demonstrated better energy efficiency than the JavaScript-based application. This is further optimized with energy efficient code patterns. However, it is important to note that energy efficiency is just one of the many factors to consider when selecting a programming language for cloud development, and other factors such as development speed, scalability, and maintainability should also be taken into account. The specific energy efficiency gains observed in this comparison vary depending on the nature of the application, the hardware used, and the optimization efforts applied during development. Nevertheless, developers and organizations should be mindful of the energy implications of their technology choices, especially as sustainability and environmental concerns become increasingly important.

References

[1] Jerome Rocheteau, Virginie Gaillard and Lamya Belhaj.: How Green Are Java Best Coding Practices? 3rdInternationalConferenceonSmartGridsandGreenIT Systems, pp.235-246,2014.

[2] Haitao Steve Zhu Chaoren Lin Yu David Liu: A Programming Model for Sustainable Software, IEEE/ACM 37th IEEE International Conference on Software Engineering, 2015.

[3] Luca Ardito; Giuseppe Procaccianti; Marco Torchiano; Antonio Vetrò: Understanding Green Software Development: A Conceptual Framework, IEEE, Volume: 17,Issue: 1,pp. 44-50,2015.

[4] Giuseppe Procaccianti, Hector Fernandez, Patricia Lago.: Empirical Evaluation of Two Best Practices for Energy-Efficient Software Development. Journal of Systems and Software,2016

[5] R. Pereira, M. Couto, J. Cunha, J. P. Fernandes and J. Saraiva,:The Influence of the Java Collection Framework on Overall Energy Consumption, IEEE/ACM 5th International Workshop on Green and Sustainable Software, pp. 15-21,2016.

[6] M. Salam and S. U. Khan.: Developing green and sustainable software: Success factors for vendors.7th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, pp. 1059-1062,2016.

[7] M. Kumar, Y. Li and W. Shi.:Energy consumption in Java: An early experience. Eighth International Green and Sustainable Computing Conference, pp. 1-8, 2017.

[8] Biswajit Saha.: Green Computing: Current Research Trends. International Journal of Computer Sciences and Engineering,pp.467-469,2018.

[9] Nayan Agrawal, Ms. Jasneet Kaur Saini, Prof. Pallavi Wankhede.: Review on Green Cloud Computing: A Step Towards Saving Global Environment. International Journal of Engineering Research & Technology, Volume 8, Issue 05, 2020.

[10] Shokhista Ergasheva, Dragos Strugar, Artem Kruglov, Giancarlo Succi.: Energy Efficient Software Development Process Evaluation for MacOS Devices. IFIP International Conference on Open Source Systems, 2020.

[11] AbdulazizAlarifi, Kalka Dubey, Mohammed Amoon1, TorkiAltameem ,Fathi E. Abd El-Samie, Ayman Altameem, S. C. Sharma, Aida A. Nasr: Energy-Efficient Hybrid Framework for Green Cloud Computing, IEEE Access, **ISSN:** 2169-3536,pp**:** 115356 – 115369,2020.

[12] Daniele D’Agostino, Ivan Merelli, Marco Aldinucci, Daniele Cesini.: Hardware and Software Solutions for Energy-Efficient Computing in Scientific Programming. Scientific Programming, 2021.

[13] Z. Ournani, R. Rouvoy, P. Rust and J. Penhoat.: Evaluating The Energy Consumption of Java I/O APIs. IEEE International Conference on Software Maintenance and Evolution (ICSME), Luxembourg, pp. 1-11, 2021.

[14] Artem Kruglov, Giancarlo Succi & Gcinizwe Dlamini.: System Energy Consumption Measurement. Developing Sustainable and Energy-Efficient Software Systems, pp:27–38, 2022.

[15] Rajni Sehgal , Deepti Mehrotra, Renuka Nagpal, Ramanuj Sharma, “Green Software: Refactoring Approach”, Journal of King Saud University, Computer and Information Sciences, 34, pp 4635-4643, 2022.

[16] Candy Chamorro Gonzalez, Jesús Peña-Vinces, “A Framework for a Green Accounting System-Exploratory Study in a Developing Country Context, Colombia”, Environment, Development and Sustainability, pp:9517–9541, 2023.

[17] Green software: Best practices for a sustainable future, Tata Consultancy Services. Online Technical Report.