Toward a Sustainable Data Lake: Measuring and Forecasting Energy Usage of CRUD Operations

Thandar Muang Muang[1]*, Chinnapong Angsuchotmetee[1], Phillipe Roose[2], Hernán Humberto Álvarez Valera[3,]
[1]Division of Computational Science, Prince of Songkhla University, Songkhla, Thailand
[2]IUT de Bayonne, Universite de Pau et des Pays de l'Adour, ANGLET, France
[3]Technopole Domolandes, ST-GEOURS-DE-MAREMNE, France
*e-mail: 6710220083@psu.ac.th

Abstract: The rapid growth of data lakes deployment for big data and analytics raises concerns about their sustainability due to high energy demands. While overall system-level consumption has been studied, little is known about the energy footprint of fundamental operations. This work investigates the energy usage of Create, Read, Update, Delete (CRUD), and Idle states in a Spark–Delta Lake environment using the EcoFloc energy usage profiling tool. The first step is therefore to measure the energy cost of these operations before suggesting future optimization to reduce the use of energy. Experiments were repeated five times with single-record operations to ensure consistency, measuring both CPU and RAM consumption. The results show that write-heavy operations require substantially more energy compared to read or insert tasks, while idle consumption is negligible. These findings highlight that optimizing update and delete operations is essential for reducing the long-term energy footprint of data lakes, providing a baseline for optimizing sustainable data lake designs.

Introduction:

The rapid rise of big data has accelerated the adoption of data lakes as a cost-effective, scalable, and schema-on-read framework for storing and processing diverse data types. Unlike traditional data warehouses, data lakes can efficiently support modern analytics and machine learning workloads [1],[2]. However, the sustainability of such infrastructures is an increasing concern, particularly given the significant energy demands of large-scale data processing systems. These demands mainly arise from the massive utilization of CPU, RAM, storage (HDD/SSD), and I/O operations that support continuous data ingestion and analytics at scale. As climate change intensifies, the computing community faces growing pressure to evaluate and minimize the carbon footprint of data-intensive technologies.

Prior studies on cloud platforms, distributed systems, and database operations reveal wide variability in power consumption and highlight inconsistent measurement practices [3],[4],[5]. While energy profiling tools have been applied at the application or cluster level, there remains a lack of detailed analysis of operation-level energy usage in data lakes. Existing research tends to emphasize total system consumption, offering limited insight into the energy cost of Create, Read, Update, and Delete (CRUD) operations [6] since they are fundamental operations of any database and data lakes. Most of the higher-level activities in data lakes like ingestion, transformation, and querying can be refined into these CRUD activities.

This gap is critical for two reasons. First, CRUD operations represent the majority of data lake operations core, and their cumulative execution directly influences overall energy demand. Second, long-term forecasting of data lake utilization requires an understanding of how these operations behave under varying workloads. There is a clear need to develop this line of research because existing studies often measure energy at the system or cluster level, but they do not isolate the cost of the CRUD operations that actually dominate workloads. In the absence of this knowledge, optimization strategies risk is likely to be either too broad or misdirected. The goal of creating a fine-grained perspective of CRUD energy will bring the missing detail required to design greener systems. To address this, we focus on Spark with Delta Lake, which enhances reliability through ACID (Atomicity, Consistency, Isolation, and Durability) transactions. CRUD operations become meaningful in a multi-user, distributed environment only when they are paired with ACID guarantees. When it is enabled by Spark and Delta Lake, making it possible to execute millions of Create or Update operations without corruption, conflicts, or data loss. The integration of CRUD and ACID can therefore be said to be the foundation and the stability of data lake management. [7], [8].

Before proposing energy-saving techniques, the first step is to understand how much energy is consumed by the fundamental operations of a data lake. This involves monitoring the CPU cycles, voltage, frequency, and memory accesses that are done in CRUD actions. The sustainability of CRUD operations is a process that should be measured through close observation of the hardware resources that it consumes. Some of generic monitoring tools report a total system consumption, but do not distinguish between the cost of one process or query. To overcome this, we apply EcoFloc [16] a monitoring system designed to measure CPU and RAM energy per-process energy tracking in Linux, making it suitable for each CRUD operations process profiling. It can provide CPU and RAM-level measurements at process granularity.

In this study, EcoFloc was selected over alternatives such as Scaphandre [17] because it allows direct monitoring of individual processes, a feature that is highly relevant for CRUD-level profiling in containerized environments. This study establishes a methodological basis for evaluating the sustainability of data lake operations and provides empirical evidence to guide greener system design.

Literature Review:

Data lakes have emerged as a preferred alternative to traditional data warehouses because of their flexibility, scalability, and ability to handle diverse data types. Tools such as Apache Spark have further accelerated adoption by enabling efficient processing across large-scale platforms [1], [2]. Profiling studies have also been conducted at system and application levels, often focusing on cluster behavior or database workloads [3], [4], [5]. CRUD operations have been widely recognized as the foundation of database systems [6], but previous studies have largely examined them from a performance perspective rather than an energy footprint. These studies suggest the hypothesis assumes that the cost of energy of all CRUD operations is not the same. The overheads involved in reading and inserting individual records ought to be light, but those of updating and deleting must be heavier, since they require scanning of the data, rewriting it, and ensuring data consistency. Meanwhile, Idle is expected to serve only as a baseline reference.

Other studies have addressed ingestion management in data lakes to better handle variety and continuous data streams [7] and knowledge graph technologies have been proposed to empower data lakes with semantic insights [8]. Alongside this growth, the Lakehouse concept has combined the strengths of both data lakes and warehouses, supporting hybrid analytics and machine learning integration [9], [10]. While these advances improve performance and usability, they raise growing concerns about energy efficiency. Prior research has introduced green data lake models [11], energy-aware scheduling in Spark [12], and query optimization for energy-efficient databases [13].

Although earlier work highlights the importance of sustainable data management and some attempts at energy profiling exist, no prior study has provided precise monitoring of CRUD operations in a data lake environment under controlled workloads, nor extended the results into long-term (yearly) forecasting. This gap motivates the present research. In order to design improvements, Energy cannot be optimized without first identifying its sources and patterns of use. Understanding the amount of energy consumed, the timing of its demand, and the reasons behind it is a necessary foundation before proposing solutions for greener data lake operations.

EcoFloc: Energy Measuring System Tool for Linux:

To address the lack of precise energy profiling at the operation level in data lakes, this study employs EcoFloc, an open-source tool for monitoring energy consumption in GNU/Linux environments [16]. Developed by the R&D laboratory of Technopôle Domolandes with the support of LIUPPA Research Lab of UPPA Université de Pau et des Pays de l'Adour and Université de Toulouse, EcoFloc tracks energy use across hardware components including CPU, RAM, GPU, storage (HD), and Network Interface Controllers (NICs). Its modular design allows extensibility and process-level granularity, enabling researchers to isolate the energy cost of specific applications—an ability highly relevant for studying CRUD operations in Spark–Delta Lake environments. By capturing precise CPU and RAM usage, EcoFloc provides the basis for forecasting the long-term energy footprint of data lake workloads. It measures energy by sampling hardware performance data such as CPU frequency, voltage, and usage time, then applying a power model using the following equation:

$$W = P\times C\times V^2\times F$$

P (Utilization): the percentage of CPU time allocated to the process. Higher utilization means the CPU spends more time actively executing instructions.
C (Capacitance): a hardware property of the processor that reflects how much electrical charge the circuits store and switch during operation.
V (Voltage): the operating voltage of the CPU cores. Power consumption grows quadratically with voltage (small increases in V cause large increases in W).
F (Frequency): the operating clock speed of the CPU cores. Higher frequencies result in faster processing but also greater energy use.

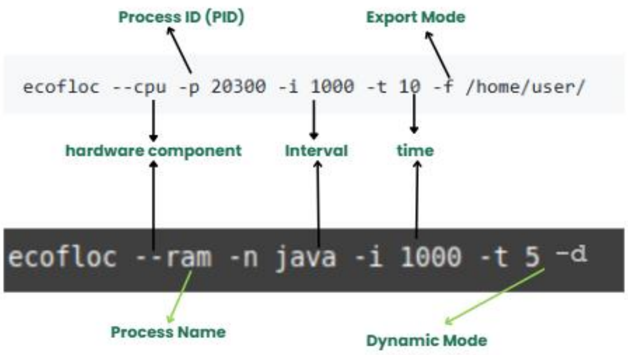
In practice, power increases when the CPU is active for longer periods, operates at higher voltage, or runs at higher frequency. EcoFloc applies this model to estimate CPU power at each sampling interval, reporting both instantaneous power (watts) and accumulated energy (joules). CPU time is defined as the sum of user and system times, which reflects actual compute work and excludes idle overheads.

EcoFloc measures RAM energy by monitoring the number of memory operations performed by the target process through the Linux `perf` tool. There are two event types, memory reads (mem-loads) and memory writes (mem-stores). Each operation is charged with a fixed energy cost depending on hardware characterization 6.6 nJ/load and 8.7 nJ/store, and the following estimation model is obtained:

$$\bar{E}_{RAM=} (\text{store } \times 8.7 \text{ nJ}) + (\text{loads} \times 6.6 \text{ nJ})$$

Through the multiplication of event counts by these coefficients, EcoFloc approximates the total energy consumption of RAM in joules and subsequently calculates the average power in watts by dividing this value by the observation interval. In contrast to the CPU, which provides real execution time through user and system counters, RAM lacks a concept of "active time," as it solely reacts to access requests and perpetually consumes refresh power. Therefore, within the context of this study, the processing time for RAM is delineated as the fixed measurement window (-t) outlined in EcoFloc, during which memory access events are recorded.

EcoFloc reports both **average power (watts)** and **total energy (joules)**, providing a clear picture of how energy is consumed over time and how much is required to complete a task. These metrics can be captured at the process level using identifiers (PIDs), which is especially useful when profiling CRUD operations in a data lake. EcoFloc can be used via a command-line interface (CLI) for automated benchmarking or through a graphical interface (GUI) for easier result visualization.



```
ecofloc --cpu -p 20300 -i 1000 -t 10 -f /home/user/
```

```
ecofloc --ram -n java -i 1000 -t 5 -d
```

Figure 1. EcoFloc command-line usage for energy profiling

For example, EcoFloc can profile the CPU energy usage of a process with a specified process identifier (PID). The following command measures the CPU consumption of the process with PID 203 for 10 seconds at 1000 ms intervals, exporting the results to a CSV file:

```
. /ecofloc --cpu -p 203 -i 1000 -t 10 -f
```

The output from this command is reported as:

```
******************************
     /ECOFLOC_CPU_PID_203
******************************
Average Power (CPU): 0.47 Watts
Total Energy (CPU): 4.72 Joules
******************************
```

This example demonstrates that EcoFloc can also provide the average power in watts and the total energy in joules, which are central measurements to this experiment to assess CRUD and Idle operations within the data lake setting.

Although EcoFloc can monitor CPU, RAM, GPU, storage, and network devices, this study focuses on **CPU and RAM**, as they are the dominant contributors to Spark–Delta Lake workloads. Concentrating on these components provides a clearer baseline for understanding the energy cost of CRUD operations. EcoFloc results were then used to identify the most energy-intensive operations and to forecast long-term consumption at daily, weekly, monthly, and yearly scales. While EcoFloc offers reliable approximations based on kernel-level counters, the results remain estimates, with future extensions expected to improve accuracy and hardware coverage [1].

Experiments

The objective of this experiment is to measure and forecast the energy consumption of common data lake operations Create, Read, Update, and Delete (CRUD). An additional **Idle** state was included as a baseline for comparison. By isolating these tasks, the study aimed to provide a controlled view of how different workloads contribute to the overall energy footprint of data lakes. To achieve this, we deployed a Spark–Delta Lake environment inside a Docker container, ensuring reproducibility and minimizing side effects from unrelated system processes.

The experiments were conducted in a controlled Spark–Delta Lake environment running inside a Docker container. The underlying hardware was configured as a server, with specifications summarized in Table 1. This setup provided sufficient computational resources while allowing precise tracking of CPU and RAM usage.

Table 1. Experimental Server Specifications

Feature	Specification
System Model	B660M DDR4
Processor	Intel Core i7-12700F (12th Gen)
RAM	64GiB DDR4 @ 2400 MHz
Cache	25MiB L3 Cache
Storage	1TB NVMe
Architecture	x86_64 (64-bit)
Operating Modes	32-bit, 64-bit
Operating System	Linux Mint 21.2
Containerization	Docker with Spark and Delta Lake

Energy measurements were performed using EcoFloc, an open-source profiling tool executed in parallel with query processing. For each operation, EcoFloc was executed with CPU and RAM modules enabled, sampling at 1,000 ms intervals over a 10-second observation window. Because The 10 s observation window was chosen as it was necessary to balance accuracy and stability. Smaller intervals can only get noise, whereas longer intervals can expose more background process risk. Setting the period to 10 seconds allows to make all of each CRUD measurement is representative while minimizing side-effects. CPU energy was estimated from the time the processor spent in user and system modes, representing its active workload by the Linux time utility, while RAM energy was instead measured continuously across the same interval, as memory consumes energy on every access operation and lacks a discrete notion of processing time, regardless of active CPU time.

Each CRUD operation was executed five times, with queries designed to affect only a single row in the Delta Lake table to maintain fine-grained control. Repetition was required in this experiment to improve accuracy. This study performed multiple repetitions of all the operations to make sure that the observed values were steady, representative, and not the result of a single measurement. The Create operation inserted new rows, Read retrieved existing rows with filters, Update modified selected fields, and Delete removed specific rows. The Idle baseline was measured by keeping the Spark–Delta Lake environment active without executing any query. Results from the five repetitions were averaged to obtain reliable daily energy values, which were later extrapolated to weekly, monthly, and yearly forecasts.

During each execution, the following values were recorded:

- Average power consumption (W)
- Total energy usage (J)
- Processing time (s).

CPU and RAM processing times were defined distinctly in this research, as each of the subsystems presents performance data in a variant form. In case of CPU measurements, Linux time utility was implemented with the EcoFloc. EcoFloc shows the real (wall-clock time), user (time in user space), and sys (time in kernel space) in the output results.

As an example:

```
*****************************
/ECOFLOC_CPU_COMM_java
*****************************
Average Power: 18.26 Watts
Total Energy: 146.12 Joules
*****************************

real    0m13.064s
user    0m1.311s
sys     0m2.238s
```

Average daily energy consumption ($\bar{E}_{daily}$) was obtained from five repeated executions for each CRUD and Idle operation; the values were used to forecast longer-term consumption. The forecasting was performed by simple multiplication of the daily average by standard time periods, as shown below:

$$E_{weekly} = \bar{E}_{daily} \times 7$$

$$E_{monthly} = \bar{E}_{daily} \times 30$$
$$E_{yearly} = \bar{E}_{daily} \times 365$$

Experiment Results:

The outcomes of the measurements are summarized in Table 2, which presents the average CPU and RAM energy consumption, along with processing time, for each CRUD operation and the Idle state. These results were obtained by averaging five independent executions per operation to minimize variability and ensure reliability.

Table 2. Daily Energy Consumption and Time for (CRUD & IDLE) Operations in Data Lake

Operation Type	CPU			RAM		
	Average Power (W)	Average Total Energy (J)	Average Processing Time (sec)	Average Power (W)	Average Total Energy (J)	Observation Window (sec)
CREATE	0.044	0.45	0.016	0.876	8.77	10
READ	0.006	0.05	0.0164	0.016	0.18	10
UPDATE	0.35	3.53	0.0178	4.61	46.14	10
DELETE	0.318	3.17	0.0168	3.08	30.86	10
IDLE	0.00	0.02	10	0.00	0.02	10

The finding as in Table 2 shows that Delete (3.17 J) and Update (3.53 J) consumed the most CPU energy whereas Create (0.45 J) and Read (0.05 J) required much less. RAM energy followed a similar trend, with Update (46.14 J) and Delete (30.86 J) slightly higher than Create (8.77 J), whereas Read used almost no RAM energy (0.18 J). Idle does not perform any computation, but EcoFloc measures energy within a defined 10-second observation. That is why an average processing time is also present with Idle, although it is just a measure interval.

Overall, these results indicate that write-intensive operations (Update and Delete) require substantially more resources compared to read or insert tasks. Read remains the most power-efficient operation, while Create shows moderate energy demand. Idle consumption is minimal, serving as a baseline. These finding confirm that heavy write operations dominate the energy footprint in data lake workloads.

These findings suggest that energy efficiency can be improved by batching updates and deletes, applying caching or indexing to reduce redundant writes, and scheduling write-heavy tasks during low-load periods to minimize energy spikes.

The following chart, Figure 3, illustrates the projected energy impact of the CPU operations under conditions of CRUD and Idle over time (daily, weekly, monthly and yearly). In this study, the energy footprint of Delete and Update operations are the most energy intensive, while Read is the most efficient as per results. Create falls in between, and Idle remains negligible.

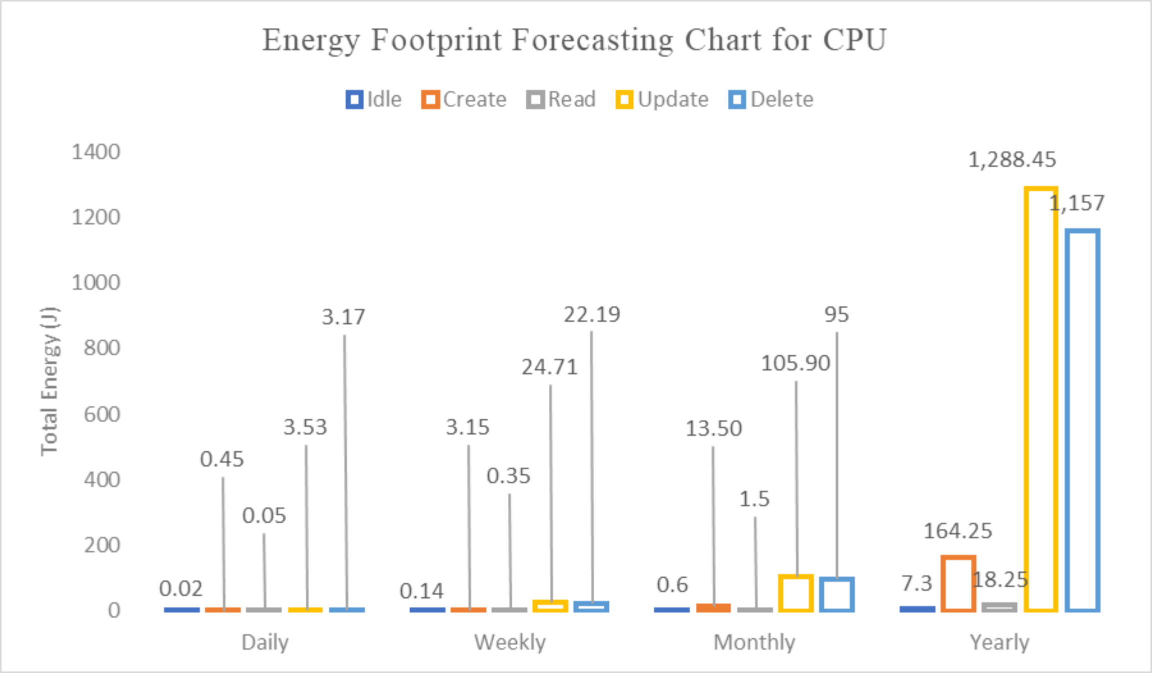


Figure 3. Forecasted CPU Energy Footprint for CRUD and Idle Operations (Daily, Weekly, Monthly, Yearly)

On a yearly basis, Update reaches (1,288.45 J) and Delete (1,157 J), far exceeding Create (164.25 J) and Read (18.25 J). Idle remains minimal at only 7.3 J per year. These results prove that, over time, write-heavy operations dominate CPU energy usage, while Read continues to be the most sustainable operation.

The following chart, Figure 4, shows the measured daily energy consumption of RAM and forecasted RAM energy footprint under CRUD and Idle operations. Daily energy consumption of Update recorded the highest RAM energy consumption at (46.14 J), followed by Delete (30.86 J) and Create (8.77 J), while Read used only 0.18 J. Idle consumption remained negligible at 0.02 J. These results highlight Spark’s memory-intensive behavior, particularly for write operations.

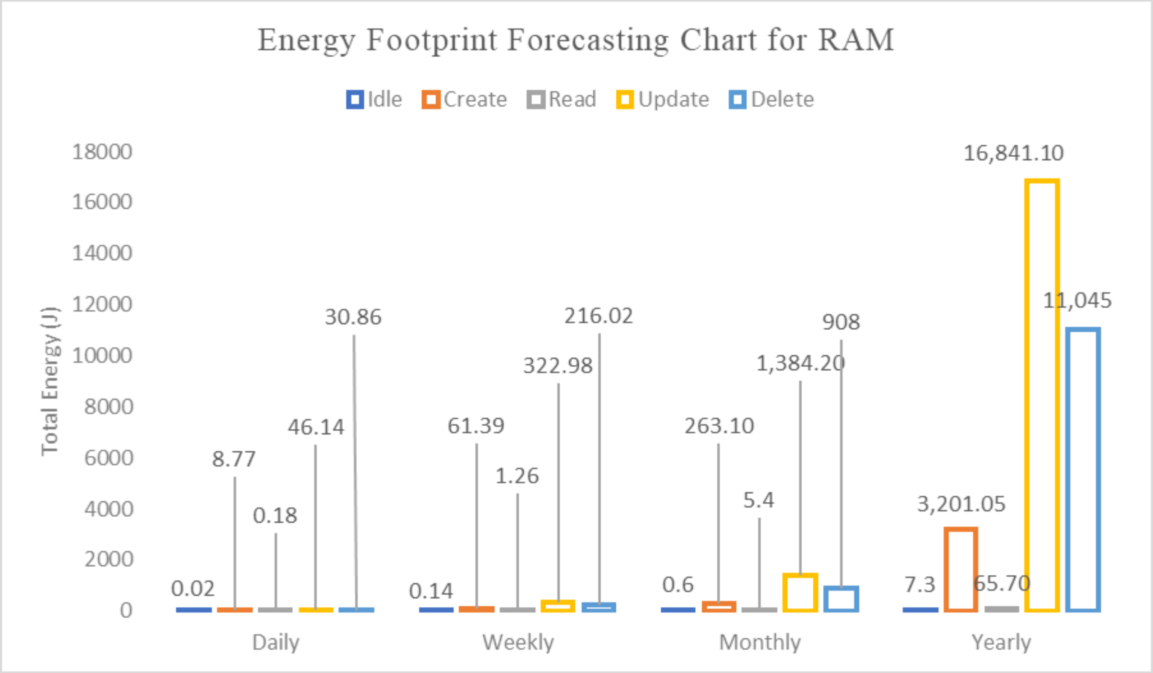


Figure 4. Forecasted RAM Energy Footprint for CRUD and Idle Operations (Daily, Weekly, Monthly, Yearly)

Overall, breaking down the yearly basis, Update consumes (16,841 J), Delete (11,045) and Create (3201.05 J). Read is relatively low, at (65.70 J) and Idle is nearly zero, at 7.3 J/year. These findings indicate that memory activity is the largest contributor to Spark's long run energy footprint with write-intensive operations dominating most of the RAM usage.

Discussion of Results and Analysis:

The experimental results show a clear distinction between read-oriented and write-oriented operations in terms of energy demand. These findings are consistent with established principles of database systems and transactional data lake architectures. The higher energy cost of Update and Delete operations is an expected outcome, and the root cause lies in Delta Lake's "copy-on-write" mechanism. When performing an Update or Delete, Spark must read the existing data files, apply the changes in memory, write out entirely new data files, and then update the transaction log to invalidate the old files. This process of reading, rewriting, and ensuring transactional consistency is inherently more demanding on both CPU and RAM compared to a simple Create (which appends data) or Read (which streams data).

Therefore, the primary contribution of this work is not the discovery that write-heavy operations are more expensive, but rather the empirical quantification of this cost on a modern Spark–Delta Lake stack. By providing a precise energy baseline for each CRUD operation using EcoFloc, this study establishes the necessary foundation for developing and evaluating future energy-aware optimization strategies tailored to these specific environments.

On the daily energy consumption of CPU, Update consumed an average of 3.53 J, and Delete consumed 3.17 J, compared to only 0.45 J for Create and 0.05 J for Read. This means that Update and Delete required more than seven times the energy of Create and over sixty times that of Read. Similarly, RAM results confirm the imbalance that Update consumed 46.14 J and Delete 30.86 J, while Create used 8.77 J and Read almost none at 0.18 J. Idle remained negligible on both CPU (0.02 J) and RAM (0.02 J), validating its role as a baseline reference.

When extended to long-term forecasts, the same trend is magnified. For CPU energy, Update reached 1,288 J per year, and Delete 1,157 J, compared to Create at 164 J and Read at 18 J. RAM forecasting further emphasizes the memory-intensive nature of Spark workloads, with Update climbed to 16,841 J per year, Delete to 11,045 J, and Create to 3,201 J, while Read was limited to 65.70 J and Idle to 7 J.

These results provide strong evidence supporting the hypothesis drawn from earlier studies [12], [13], [15] that write-heavy operations are the most energy-demanding. The high costs of Update and Delete can be attributed to data rewriting and consistency enforcement, while the low costs of Read confirm its efficiency. The moderate values of Create indicate that insert operations require more resources than reads but remain less demanding than updates or deletions. The evidence therefore shows that the energy footprint of data lakes is dominated by write-intensive operations and has the largest impact on long-term sustainability.

Conclusions and Future Work:

In this study, the energy consumption and processing time of Create, Read, Update, Delete, and Idle operations in a Spark–Delta Lake environment were measured using EcoFloc. Each operation was executed with a single record, repeated five times, and the averages were taken for analysis. The results clearly indicate that Update and Delete are the most energy-intensive, while Read is the most efficient and Create falls in between. Idle contributes almost no measurable energy, serving as a baseline reference.

The forecasting further emphasizes that write-heavy operations dominate long-term energy costs. On the CPU, yearly energy usage reaches 1,288 J for Delete and 1,157 J for Update, compared to only 164 J for Create and 18 J for Read. For RAM, yearly energy climbs to 16,841 J for Update, 11,045 J for Delete, and 3,201 J for Create, while Read remains at just 66 J and Idle at 7 J. These RAM values are derived from a fixed 10-second observation window for each operation, ensuring comparability across CRUD and Idle.

Overall, the results present a strong foundation to assess the sustainability of the data lake operations. It verifies that most long-term energy consumption is attributed to write-intensive operations hence the need to prioritize optimization of such activities in order to achieve a greener system design. In Future research, this study will be plan to approach bigger data sets, batch workloads, and multi-node clusters to capture more realistic usage scenarios. Additional profiling of GPU and storage components will also be explored, as well as the integration of energy-aware scheduling policies, to provide a more comprehensive assessment of sustainable data lake management.

References:

1. C.-T. Yang, T.-Y. Chen, E. Kristiani, and S. F. Wu, “The implementation of data storage and analytics platform for big data lake of electricity usage with spark,” *The Journal of Supercomputing*, Nov. 2020.

2. J. L. Hernández, S. Martín, P. Kapsalis, Kyriakos Katsigarakis, Elissaios Sarmas, and V. Marinakis, “Building a Data Lake for Smart Building Data: Architecture for Data Quality and Interoperability,” pp. 1–8, Jul. 2023.
3. M. U. Khan, S. Abbas, S. U.-J. Lee, and A. Abbas, “Measuring power consumption in mobile devices for energy sustainable app development: A comparative study and challenges,” *Sustainable Computing: Informatics and Systems*, vol. 31, p. 100589, Sep. 2021.
4. O. Ullah, M. Hanan, and M. A. Ghafoor, “Energy Efficiency Issues in Android Application: A Literature Review,” pp. 1–6, Oct. 2022.
5. G. Procaccianti, A. Vetro’, L. Ardito, and M. Morisio, “Profiling Power Consumption on Desktop Computer Systems,” *Lecture Notes in Computer Science*, pp. 110–123, 2011.
6. C.-O. Truica, F. Radulescu, A. Boicea, and I. Bucur, “Performance Evaluation for CRUD Operations in Asynchronously Replicated Document Oriented Database,” *2015 20th International Conference on Control Systems and Computer Science*, May 2015.
7. M. Armbrust, A. Ghodsi, R. Xin, and M. Zaharia, “Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics,” 2021.
8. V. Marinakis, “Big Data for Energy Management and Energy-Efficient Buildings,” *Energies*, vol. 13, no. 7, p. 1555, Mar. 2020.
9. Y. Zhao, I. Megdiche, and F. Ravat, “Data Lake Ingestion Management,” *arXiv.org*, 2021.
10. A. Helal, “Data Lakes Empowered by Knowledge Graph Technologies,” *Proceedings of the 2022 International Conference on Management of Data*, pp. 2884–2886, Jun. 2021.
11. Marzieh Derakhshannia, J. Grange, and Nihal Pekergin, “Toward Green Data Lake Management and Analysis Through a CTMC Model,” *Lecture notes in computer science*, pp. 47–61, Jan. 2025.
12. H. Li, H. Wang, S. Fang, Y. Zou, and W. Tian, “An energy-aware scheduling algorithm for big data applications in Spark,” *Cluster computing*, vol. 23, no. 2, pp. 593–609, Jun. 2019.
13. B. Guo, J. Yu, D. Yang, H. Leng, and B. Liao, “Energy-Efficient Database Systems: A Systematic Survey,” *ACM Computing Surveys*, vol. 55, no. 6, pp. 1–53, Dec. 2022.
14. J. Kachaoui and A. Belangour, “Enhanced Data Lake Clustering Design based on K-means Algorithm,” *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 4, 2020.
15. N. Somu, G. Raman M R, and K. Ramamritham, “A deep learning framework for building energy consumption forecast,” *Renewable and Sustainable Energy Reviews*, vol. 137, p. 110591, Mar. 2021.
16. Humberto Valera, Franck Ravat, Philippe Roose, Jiefu Song, Nathalie Vallès-Parlangeau – *ECOFLOC: A Multi-component Energy Measurement Tool* – E2SDS (International Colloquium on Energy-Efficient and Sustainable Distributed Systems) – November 26-27, 2024, Natal, Brazil
17. Shankaranarayanan, Nk & Li, Zhuohuan & Seskar, Ivan & Maddala, Prasanthi & Puthenpura, Sarat & Stancu, Alexandru & Agarwal, Anurag. (2024). POET: A Platform for O-RAN Energy Efficiency Testing. 1-5. 10.1109/VTC2024-Fall63153.2024.10757537.