https://doi.org/10.54361/ajmas.269221

Original article

Workload-Aware Energy-Efficient Query Scheduling for Cloud Database Systems: Experimental Study

Salmi Tantoun[1]*, Anwar Alhenshiri[2]

[1]*Department of Software Engineering, Faculty of Information Technology, University of Misurata, Misurata, Libya*
[2]*Department of Computer Science, Faculty of Information Technology, University of Misurata, Misurata, Libya*
Corresponding email. slma.tantoun@it.misuratau.edu.ly

Abstract
Energy efficiency is a growing challenge in cloud database systems, particularly for analytical workloads with intensive CPU and disk I/O demands. Traditional query scheduling strategies, such as First-Come First-Served (FCFS) and Shortest Job First (SJF) focus on performance optimization and do not explicitly consider energy consumption. This paper proposes an Energy-Driven Adaptive Scheduling (EDAS) strategy that prioritizes queries based on estimated CPU and disk I/O costs without modifying the database engine. Experiments were conducted on a cloud-based MySQL system using light, medium, and heavy workloads derived from *Sakila* and *TPC-H* benchmarks. Results showed that energy-aware scheduling is workload-dependent: SJF performed well under light and medium workloads, while EDAS achieved measurable energy savings under heavy workloads and greater resilience under CPU throttling. The study demonstrates the importance of workload-aware query scheduling for improving cloud database energy efficiency.
Keywords. Cloud Database Systems, Query Scheduling, Energy Efficiency, Workload-Aware Scheduling.

Introduction
Cloud computing has become the dominant platform for hosting modern data-intensive applications, and relational database systems remain at the core of these environments. As organizations increasingly migrate analytical and transactional workloads to the cloud, energy consumption has emerged as a critical operational and environmental challenge. Data centers now account for a significant portion of global electricity usage, and database servers are among the major contributors due to their continuous processing and storage demands [1]. Consequently, improving the energy efficiency of database systems has become an important research priority. Energy efficiency in cloud environments has traditionally been addressed at the infrastructure level through techniques such as virtual machine consolidation, dynamic voltage and frequency scaling, and energy-aware resource allocation [2]. While these methods reduce overall power usage, they operate largely outside the database layer and do not consider the specific behavior of database workloads. Recent research emphasizes that application-level and database-level optimizations are necessary to complement infrastructure-centric approaches [3].
Within database systems, energy consumption is closely tied to workload characteristics and query execution behavior. Analytical workloads, in particular, generate heavy CPU utilization and large volumes of disk I/O, both of which have a direct impact on power usage. Experimental studies have shown that different query plans and execution patterns can lead to significant variations in energy consumption, even when processing the same data [4]. These findings suggest that query management decisions such as execution order and scheduling can influence not only performance but also energy efficiency.
Despite this potential, most existing query scheduling strategies remain focused on traditional performance objectives. Common approaches, such as First-Come First-Served (FCFS) and Shortest Job First (SJF) aim to minimize response time or maximize throughput, without explicitly considering energy impact [5]. Energy-aware scheduling has been extensively studied in operating systems and distributed computing, but comparatively little attention has been given to energy-oriented scheduling at the database query level [6]. As a result, database administrators typically rely on performance-driven scheduling policies that may unintentionally increase energy usage.
Another practical challenge arises from the nature of modern cloud platforms. Many widely used instance types, such as the AWS T-series, employ CPU credit mechanisms that can throttle performance under sustained load [7]. Under such conditions, the order in which queries are executed may have a substantial impact on both execution time and energy consumption. However, the interaction between query scheduling and CPU credit throttling remains largely unexplored in the literature. Furthermore, benchmarking frameworks such as TPC-H provide standardized analytical workloads that are widely used to evaluate database performance [8]. While these benchmarks are frequently employed for performance studies, they are less commonly used to investigate energy efficiency or energy-aware scheduling behavior. This highlights a gap between traditional database benchmarking practices and emerging sustainability concerns.
Motivated by these challenges, this paper investigates whether simple, application-level control over query execution order can lead to measurable energy savings in cloud database systems. We propose an Energy-Driven Adaptive Scheduling (EDAS) strategy that ranks queries based on lightweight estimates of CPU usage and disk I/O intensity, without requiring any modifications to the underlying DBMS. By conducting experiments on a cloud-hosted MySQL environment using workloads derived from Sakila and TPC-H, we analyze how scheduling decisions affect energy consumption under different workload intensities. By

***Received**: 12-12-2025 - **Accepted**: 09-02-2026 - **Published**: 16-02-2026*

focusing on query-level scheduling rather than low-level engine modifications, this study provides a practical approach that can be adopted in real cloud deployments with minimal overhead. The results contribute to the growing body of work on sustainable and energy-efficient database systems. To position this work within the broader research landscape, it is important to review existing approaches to energy efficiency in cloud and database systems.
Work on energy efficiency in cloud and database systems spans several complementary directions: (1) infrastructure and VM-level energy management; (2) energy-aware database system design and query optimization; (3) scheduling and task-level energy management in cloud environments; and (4) empirical studies that profile I/O and CPU behavior of database workloads. Below, we summarize representative work in each area and highlight the gap addressed by this paper.
Much of the early work on reducing datacenter energy consumption focused on resource consolidation, VM placement, and power-aware resource allocation. Beloglazov and Buyya [2] proposed energy-aware heuristics for VM allocation to reduce datacenter power use; Dayarathna, Wen, and Fan [1] surveyed data-center energy models and emphasized the role of infrastructure controls (e.g., DVFS, cooling) in overall energy cost. These infrastructure approaches are effective at the physical and VM layers but do not exploit the database-level workload structure that can be controlled by query ordering or scheduling.
There is growing literature that directly targets energy efficiency within DBMSs. Guo et al [3] provided a comprehensive survey of energy-efficient database techniques, including storage and query-plan optimizations. Tsirogiannis et al [4] performed one of the first detailed measurements showing how query plan choices translate into different energy footprints on a database server. Subsequent work has extended profiling and modeling of per-operator and per-query energy costs, enabling optimizer-level decisions that consider energy as an objective alongside latency or cost. In parallel, the operating systems and cloud scheduling communities developed energy-aware schedulers and task allocation schemes. Meisner, Gold, and Wenisch [9] introduced PowerNap to eliminate idle power at the server level; other works investigate energy-aware workload placement and scheduling heuristics for large clusters and cloud platforms [6]. However, these techniques often operate at task/VM granularity and assume control over placement or hardware settings; they are not directly applicable to DBMS-level query ordering without engine modification. Research specifically on query scheduling within database systems highlights that execution order can affect response time and fairness [10]. More recent studies analyze multi-class and priority-aware query execution to meet QoS guarantees, but energy is typically not the primary objective. Work that examines query ordering or admission control with resource heterogeneity suggests scheduling at the query level can influence system resource contention and performance, which implies potential energy effects that remain underexplored in practice [11]. Detailed workload characterization is essential to understanding when energy-aware approaches will help. [12],[13] provide empirical analyses of I/O behavior and workload heterogeneity in cloud storage and compute environments, demonstrating that analytical queries often produce bursty, high-volume I/O that dominates energy consumption. These profiling studies strengthen the intuition behind targeting scheduling policies to redistribute I/O-intensive queries and mitigate energy peaks.
Standard benchmarks such as TPC-H remain useful for reproducible evaluation of analytical workloads [8]. Complementary measurement frameworks and energy datasets (e.g., Cloud Carbon Footprint) provide coefficients and methodology for translating utilization and I/O statistics into energy estimates when direct power meters are unavailable [14]. Prior experimental DB-energy papers commonly adopt analytical models or per-query estimators when full hardware instrumentation is infeasible [3]. Taken together, the literature shows (a) infrastructure-level controls and VM scheduling reduce energy but do not exploit query semantics, (b) DBMS-level energy-aware optimizations exist but often require internal engine changes or operator-level instrumentation, and (c) task-level schedulers are promising but are not yet translated into practical, engine-agnostic query ordering policies. Our paper targets this gap by proposing EDAS, an application-level ordering heuristic that uses lightweight per-query CPU and I/O estimates to produce energy-aware execution orders without modifying the DBMS. We evaluate EDAS experimentally on cloud MySQL and show that it is particularly effective for heavy, I/O-dominated analytical workloads and under CPU-credit throttling scenarios emphasized by both I/O characterization and cloud scheduling studies.

Methodology

Experimental Environment

The experiments were conducted on a cloud-based database platform hosted on Amazon Web Services (AWS). A single EC2 *t3. small* instance was used as the experimental platform, configured with two virtual CPUs, 2 GB of RAM, and Ubuntu 22.04 as the operating system. The database engine was MySQL 8.0, deployed with default configuration parameters. All scheduling strategies were implemented externally at the application layer to avoid any modification to the internal query optimizer or execution engine. This design choice ensures that the proposed approach remains practical and deployable in real-world environments without requiring changes to the DBMS source code. To minimize external interference, all experiments were executed on an otherwise idle instance. Prior to each experimental run, operating system caches were cleared to maintain consistency. Network latency effects were avoided by executing query workloads locally on the same instance hosting the database server.

Received*: 12-12-2025 -* ***Accepted****: 09-02-2026 -* ***Published****: 16-02-2026*

Measurement and Monitoring
Accurate measurement of resource utilization is essential for evaluating energy-aware scheduling. During query execution, per-query metrics were collected using the Python *Psutil* library, which provides lightweight access to system-level performance counters. For each query, *CPU utilization percentage, execution time, disk read volume, and disk write volume* were the recorded metrics.
These measurements were collected continuously throughout query execution and stored for subsequent energy estimation. To validate the stability of system behavior, Amazon CloudWatch was used as an external monitoring tool to observe instance-level CPU and disk activity trends during experiments. CloudWatch was used only for validation purposes and not as part of the energy estimation process. All monitoring scripts were executed on the same virtual machine as the database server to eliminate network-related measurement overhead.

Database Workloads
To evaluate scheduling behavior under varying levels of resource intensity, three workload categories were defined: light, medium, and heavy. The workloads were designed to reflect realistic database usage scenarios with increasing computational and I/O complexity. Table 1 summarizes the main characteristics of the light, medium, and heavy workloads used in the experiments.

Table 1. Characteristics of the Experimental Workloads

Workload	Database	Number of Queries	Query Characteristics
Light	Sakila	15	Simple SELECT statements, minimal joins, low CPU and I/O usage
Medium	TPC-H (SF=0.1)	30	Moderate complexity queries with 2–3 table joins and basic aggregations
Heavy	TPC-H (SF=0.1)	30	Complex analytical queries with 4–6 table joins, nested aggregations, GROUP BY, ORDER BY, HAVING

Using the same number of queries for both medium and heavy workloads ensure that observed differences in energy consumption are primarily due to query complexity and resource demand, rather than workload size.

Scheduling Strategies
Three query scheduling strategies were evaluated in this study, which are explained as follows:
1. **First-Come First-Served (FCFS)**
FCFS executes queries in the exact order in which they arrive. This strategy represents the default behavior in many practical systems and serves as the primary baseline for comparison. FCFS does not consider query complexity, execution time, or resource consumption.
2. **Shortest Job First (SJF)**
SJF prioritizes queries based on their estimated execution time, executing shorter queries before longer ones. Execution time estimates were obtained from prior profiling runs under identical system conditions. SJF is widely used as a performance-oriented scheduling policy and provides a strong baseline for comparison against energy-aware approaches.
3. **Energy-Driven Adaptive Scheduling (EDAS)**
The proposed Energy-Driven Adaptive Scheduling (EDAS) strategy prioritizes queries based on their estimated energy cost rather than execution time alone. EDAS is implemented as a static scheduling mechanism that determines the execution order before workload submission.
For each query Q_i, an energy score was computed as:

$$EnergyScore\ (Q_i) = \propto . CPU_{cost}(Qi) + \beta . IO_{cost}\ (Qi) \times T(Qi)$$

Where:
- $CPU_{cost}\ (Q_i)$ represents the estimated CPU intensity of the query,
- $IO_{cost}\ (Q_i)$ represents estimated Disk I/O intensity,
- $T(Q_i)$ is the estimated execution time,
- $\propto = 0.4$ and $\beta = 0.6$ are weighting parameters giving slightly higher importance to disk I/O.

The values of $\propto$ and β were chosen based on the observation that analytical database workloads are typically more sensitive to disk activity than CPU utilization. Queries are sorted in ascending order of their energy score, and execution follows this order. EDAS does not alter query execution plans; it only changes the order in which queries are submitted to the DBMS, making it lightweight and easy to deploy.

Energy Consumption Model
Direct measurement of hardware power consumption was not feasible due to a lack of physical instrumentation. Therefore, an established analytical energy estimation model was adopted to compute per-query and workload-level energy usage.

Server Energy Estimation
Server energy consumption is estimated using a linear CPU power model:

$$E_{server} = (P_{idle} + (P_{max} - P_{idle}) \times U_{cpu}) \times T$$

Where:
- U_{cpu} is the average CPU utilization during query execution,
- T is execution time,
- $P_{idle} = 1.21W$ and $P_{max} = 9.96W$ are power coefficients obtained from the cloud carbon footprint dataset.

Disk Energy Estimation
Disk I/O energy is estimated based on the volume of data transferred:

$$E_{disk} = (D_{read} + D_{write}) \times E_{coeff}$$

where E_{coeff} =0.1J/MB is a commonly used empirical coefficient for storage systems.

Total Energy
Total query energy is calculated as:

$$E_{total} = E_{server} + E_{disk}$$

Workload-level energy consumption is obtained by summing the energy of all queries within a workload. This model provides consistent relative comparison across scheduling strategies, which is the primary objective of this study.

Evaluation Metrics
Scheduling strategies were evaluated using two primary metrics:
1. Total Energy Consumption: The aggregated energy consumed by all queries in a workload, computed using the model above.
2. Energy Savings (%): The relative reduction in energy consumption compared to the FCFS baseline:

$$EnergySavings = \frac{E_{FCFS} - E_{Strategy}}{E_{FCFS}} \times 100$$

These metrics allow direct comparison of how different scheduling strategies influence overall energy efficiency. This methodology enables systematic evaluation of query scheduling strategies under controlled and reproducible conditions. By combining realistic workloads, lightweight monitoring, and an established energy estimation model, the study provides practical insights into when and how energy-aware scheduling can benefit cloud database systems.
Because energy consumption was estimated using a deterministic analytical model and all experiments were conducted on an otherwise idle system under identical conditions, result variability was minimal. For this reason, formal statistical significance testing was not applied. Instead, consistency of trends across workloads and scheduling strategies was used as the primary basis for comparison. Figure 1 illustrate the overall methodology workflow for this experimental study.

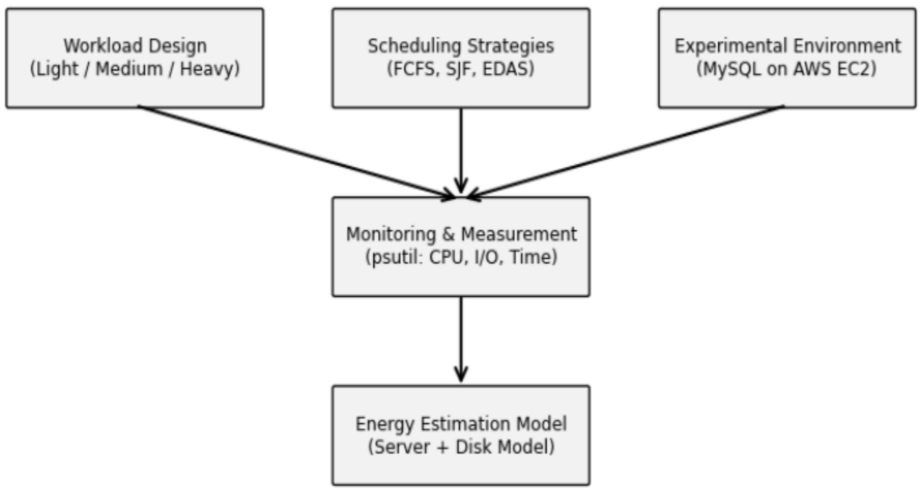
Workload Design (Light / Medium / Heavy)
Scheduling Strategies (FCFS, SJF, EDAS)
Experimental Environment (MySQL on AWS EC2)
Monitoring & Measurement (psutil: CPU, I/O, Time)
Energy Estimation Model (Server + Disk Model)

Figure 1. Methodology Workflow.

Results

This section presents the experimental results obtained from evaluating the three scheduling strategies, FCFS, SJF, and EDAS, under light, medium, and heavy database workloads. The objective was to analyze how different query execution orders influenced overall energy consumption and resource utilization in a cloud-based database environment. All experiments were conducted under identical system configurations to ensure fair comparison.

Results for Light Workload

The light workload consisted of 15 simple queries executed on the Sakila database, primarily involving basic SELECT operations with minimal joins. This workload generated very low CPU utilization and negligible disk I/O activity across all scheduling strategies. Under these conditions, total energy consumption remained small, and differences between strategies were minimal. SJF achieved a slight reduction in energy consumption compared to FCFS due to its prioritization of shorter queries. In contrast, EDAS resulted in marginally higher energy usage. This behavior can be attributed to the limited opportunity for optimization when queries are already lightweight and short-lived, as the overhead of energy-based ordering did not translate into measurable gains. These observations indicate that when workloads exhibit very low resource intensity, scheduling strategy has limited influence on energy efficiency, and traditional performance-oriented approaches remain sufficient.

Results for Medium Workload

The medium workload consisted of 30 analytical queries derived from the TPC-H benchmark with scale factor 0.1. These queries involved moderate joins and aggregation operations, resulting in sustained yet balanced CPU and disk utilization. Under this workload, scheduling decisions began to have a measurable impact on total energy consumption. Both SJF and EDAS achieved lower energy usage compared to the FCFS baseline. Among the evaluated strategies, EDAS produced the lowest energy consumption, followed closely by SJF. The improvement achieved by EDAS can be attributed to its ability to distribute resource-intensive queries more evenly over time, thereby avoiding periods of concentrated I/O activity. Although average CPU utilization remained similar across strategies, differences in execution order influenced the temporal distribution of disk accesses, leading to variations in estimated energy consumption. Figure 2 illustrates the comparative total energy consumption for the medium and heavy workloads. As shown in the figure, EDAS achieved the lowest energy usage in both cases, with the advantage becoming more pronounced under the heavy workload.

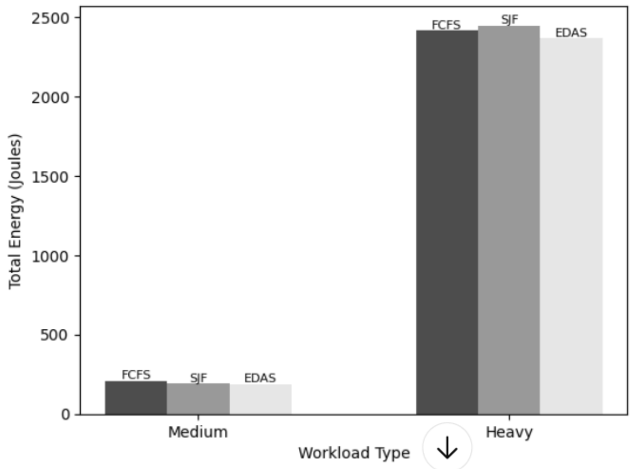

Figure 2. Zoomed Energy Comparison for Medium and Heavy Workloads.

These findings suggest that for workloads of moderate complexity, energy-aware scheduling provides tangible benefits, although the advantage over established heuristics such as SJF remains moderate.

Results for Heavy Workload

The heavy workload represented the most demanding scenario and consisted of 30 complex TPC-H queries with multiple joins, large intermediate result sets, and extensive aggregation operations. This workload generated sustained pressure on both CPU and disk subsystems. In this scenario, the impact of the scheduling strategy became significantly more pronounced. EDAS achieved the lowest total energy consumption among all evaluated strategies. Compared to FCFS, EDAS produced measurable energy savings, whereas SJF resulted in the highest overall energy usage. The inferior performance of SJF under heavy workloads indicates that minimizing execution time does not necessarily minimize energy consumption. By prioritizing shorter queries, SJF postponed long and highly I/O-intensive queries, which subsequently executed consecutively and created extended periods of high resource usage. In contrast, EDAS distributed such queries more evenly, reducing energy peaks and improving overall efficiency. These results demonstrated that energy-aware scheduling was most effective when workloads contained complex and heterogeneous CPU- and I/O-intensive queries.

Impact of CPU Credit Throttling

To evaluate behavior under realistic cloud constraints, an additional experiment was conducted under CPU credit throttling conditions, which commonly occur on AWS T-series instances during sustained high utilization. Under throttling, the behavior of the scheduling strategies changed noticeably. Both FCFS and SJF experienced substantial increases in execution time and total energy consumption. Although SJF achieved a slightly shorter makespan than FCFS, its energy advantage remained limited. EDAS demonstrated superior performance under throttling conditions. By executing more energy-intensive queries earlier and avoiding long sequences of heavy queries toward the end of execution, EDAS reduced the duration of throttled operation. This resulted in both lower makespan and significantly reduced total energy consumption compared to FCFS and SJF. Figure 3 compares the total energy consumption of the three strategies under CPU credit throttling conditions and shows that EDAS consistently achieved the lowest energy usage.

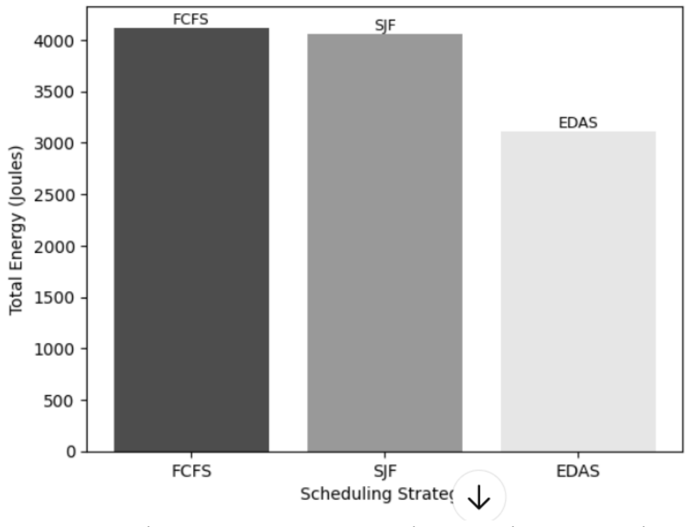

Figure 3. Total Energy Consumption under CPU Throttling

These findings highlight an important practical advantage of EDAS: in cloud environments where performance may degrade dynamically due to platform-level constraints, energy-aware scheduling can also contribute to improved robustness and predictability.

Discussion

The experimental results indicated that the impact of query scheduling on energy consumption was strongly dependent on workload characteristics. Under light workloads, where queries generated minimal CPU and disk activity, differences between scheduling strategies remained limited due to the dominance of baseline server power consumption. In such scenarios, the choice of execution order had little practical influence on total energy usage, which is consistent with prior studies emphasizing that scheduling effects become less visible when resource demand is low.As workload complexity increased, the influence of scheduling became more evident. For medium workloads, EDAS achieved moderate energy savings by distributing resource-intensive queries more evenly over time. By reducing concentrated bursts of disk activity, EDAS lowered energy peaks and improved overall efficiency compared to FCFS. These observations align with existing research highlighting the role of workload heterogeneity and I/O behavior in shaping database energy consumption. The benefits of EDAS were most pronounced under heavy analytical workloads. In these scenarios, queries exhibited substantial variability in CPU and I/O intensity, and execution order significantly influenced system behavior. Performance-oriented strategies such as SJF tended to postpone longer and more complex queries, resulting in consecutive execution of highly intensive operations and prolonged periods of elevated resource usage. In contrast, EDAS explicitly accounted for both CPU and disk costs, leading to a more balanced execution pattern and lower total energy consumption. This outcome reinforces the broader observation that minimizing execution time does not necessarily minimize energy usage. The experiment conducted under CPU credit throttling conditions further demonstrated the practical relevance of energy-aware scheduling in cloud environments. When sustained utilization triggered performance constraints, FCFS and SJF experienced increased execution times and higher energy consumption. EDAS reduced the duration of throttled operation by prioritizing more energy-intensive queries earlier in the execution sequence, thereby improving both energy efficiency and system stability under constrained conditions. Overall, the findings confirm that query ordering represents a practical and

***Received**: 12-12-2025 - **Accepted**: 09-02-2026 - **Published**: 16-02-2026*

lightweight mechanism for improving the energy efficiency of cloud database systems. While the benefits remain limited for lightweight workloads, the proposed approach proves particularly effective for complex, I/O-dominated analytical scenarios. Although the study relied on an analytical energy estimation model and a single cloud configuration, the consistent trends observed across workloads provide strong evidence that application-level scheduling can meaningfully contribute to more sustainable cloud database operation.

Conclusion

This study examined whether query execution order can be used to improve energy efficiency in cloud database systems. An Energy-Driven Adaptive Scheduling (EDAS) strategy was proposed and compared with traditional FCFS and SJF scheduling approaches. Experimental results showed that the effectiveness of energy-aware scheduling is strongly workload-dependent. For light workloads, scheduling strategy has minimal impact on energy consumption. For medium workloads, EDAS provides moderate energy savings, while for heavy analytical workloads, EDAS consistently achieved the lowest energy usage by distributing CPU and I/O-intensive queries more effectively. The analysis under CPU credit throttling further demonstrates that EDAS offers greater robustness in real cloud environments, reducing both execution time and energy consumption when performance constraints occur. These findings confirm that lightweight, application-level query scheduling can serve as a practical mechanism for improving the energy efficiency of cloud databases without modifying the DBMS. Future work will focus on validating the approach using direct power measurements, evaluating additional cloud platforms, and developing adaptive scheduling techniques for dynamic workloads.

Conflicts of Interest

The authors declare no conflicts of interest.

References

1. Dayarathna M, Wen Y, Fan R. Data center energy consumption modeling: A survey. IEEE Commun Surv Tutorials. 2015;18(1):732-794.
2. Beloglazov A, Abawajy J, Buyya R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Gener Comput Syst. 2012;28(5):755-768.
3. Guo B, et al. Energy-efficient database systems: A systematic survey. ACM Comput Surv. 2022;55(6):1-53.
4. Tsirogiannis D, Harizopoulos S, Shah MA. Analyzing the energy efficiency of a database server. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data. 2010.
5. Xu Z, Tu YC, Wang X. Online energy estimation of relational operations in database systems. IEEE Trans Comput. 2015;64(11):3223-3236.
6. Chen G, et al. Energy-aware server provisioning and load dispatching for connection-intensive internet services. In: NSDI. 2008.
7. Amazon Web Services. Key concepts and definitions for burstable performance instances (CPU credits). 2024. Available from: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/burstable-credits-baseline-concepts.html (docs.aws.amazon.com in Bing) [accessed 2025 Dec 21].
8. Transaction Processing Performance Council. TPC Benchmark H (TPC-H) standard specification. 2023. Available from: http://www.tpc.org/tpch [accessed 2025 Dec 21].
9. Meisner D, Gold BT, Wenisch TF. Powernap: eliminating server idle power. ACM SIGARCH Comput Archit News. 2009;37(1):205-216.
10. Pang H, Carey MJ, Livny M. Multiclass query scheduling in real-time database systems. IEEE Trans Knowl Data Eng. 1995;7(4):533-551.
11. González-Rodríguez M, et al. Study and evaluation of CPU scheduling algorithms. Heliyon. 2024;10(9).
12. Zou Q, et al. Characterization of I/O behaviors in cloud storage workloads. IEEE Trans Comput. 2023;72(10):2726-2739.
13. Ilager S, et al. A data-driven analysis of a cloud data center: statistical characterization of workload, energy and temperature. In: Proceedings of the IEEE/ACM 16th International Conference on Utility and Cloud Computing. 2023.
14. Cloud Carbon Footprint. Cloud Energy Coefficients Dataset. Available from: https://github.com/cloud-carbon-footprint/cloud-carbon-footprint [accessed 2025 Dec 22].

Received*: 12-12-2025 -* ***Accepted****: 09-02-2026 -* ***Published****: 16-02-2026*