Revista Internacional de Métodos Numéricos
para Cálculo y Diseño en Ingeniería

ARTICLE

A New Green ETL Optimization Technique for Sustainable Data Warehousing Environment: A Scalable Energy Efficient Solution

El Yazid Gueddoudj[1], Abdelouahab Attia[2], Abdulaziz T. Almaktoom[3], Kamal M. Othman[4], Abdulfattah Noorwali[4], Esam Y. O. Zafar[4] and Ali Wagdy Mohamed[5,*]

[1]Institute of Architecture and Urbanism, University of Batna1, 05000 Batna, Algeria
[2]Department of Computer Science, University Mohamed El Bachir El Ibrahimi of Bordj Bou Arreridj, Bordj Bou Arreridj, Algeria
[3]Department of Operations and Supply Chain Management, Effat University, Jeddah, Saudi Arabia
[4]Department of Electrical Engineering, College of Engineering and Architecture, Umm Al-Qura University, Makkah, Saudi Arabia
[5]School of Business, Zewail City of Science and Technology, University of Science and Technology, Giza, 6th of October City, 12588, Egypt
*Corresponding Author: Ali Wagdy Mohamed. Email: aliwagdy@gmail.com

Received: 22 March 2026; Accepted: 13 May 2026

ABSTRACT: Energy consumption is an emerging concern in many fields, including information technology, particularly in data warehousing environments where Extract, Transform, Load (ETL) processes account for a significant portion of operational costs and resource utilization. Despite advances in hardware-level optimization, limited attention has been given to software-level energy optimization within ETL workflows. In reality, software is as important as hardware, and it is equally responsible for a decrease or increase in energy consumption. We argue that for modern applications in which energy efficiency is a priority, ETL processes should be optimally designed. This paper addresses this gap by proposing a Green ETL (GETL) approach designed to reduce energy consumption while maintaining high performance. The proposed method integrates transformation-level reuse through a shared transformation cache and adaptive parallel execution using Apache Spark, enabling efficient resource utilization and elimination of redundant computations. The proposed GETL removes unnecessary calculations and reduces both execution time and energy consumption, without requiring any modifications to the underlying data processing engines. To evaluate the effectiveness of the proposed GETL, experiments were conducted using the Transaction Processing Council Data Integration (TPC-DI) benchmark across multiple scale factors. The results demonstrate that the proposed approach achieves an average energy reduction of approximately 30%, with higher savings observed under large-scale workloads. In addition, GETL improves execution efficiency and reduces resource utilization compared to a traditional Spark-based ETL implementation.

KEYWORDS: Green ETL (GETL); green computing; energy-efficiency; sustainable data warehousing; ETL optimization; distributed computing; energy consumption

1 Introduction

Integrating heterogeneous data sources into analytical platforms depends heavily on ETL workflows in large-scale data analytics. With the ever-increasing data sizes and pipeline complexity, ETL processes are consuming more and more computational resources and energy in current data-centric systems. This has led to growing doubts about the sustainability of data processing infrastructure and the environmental costs associated with large-scale analytics. According to [1], data centers alone will account for nearly 10% of global electricity consumption by 2030. On the other hand, among the other software installed in conventional data centers, database management systems are the most significant users of computational resources, making them a significant energy consume [2]. Current methods predominantly emphasize

hardware-level enhancements, whereas software-level energy efficiency remains underexplored [3]. Developing energy-efficient software is crucial for minimizing energy utilization in complex systems like data warehouses and promoting sustainable information technology (IT) practices [4,5].

Research on "green software" is being fueled by the undeniable impact of software on energy usage. In recent literature, great importance has been associated with improving energy efficiency and performance in data warehouse applications. Poess et al. [6] attempted to incorporate the energy parameter conceptually into data warehouse design. Green algorithms are designed to improve system efficiency while minimizing energy consumption [7]. They play a crucial role in green computing by reducing the energy consumption of computing systems, thereby mitigating the adverse impact of computing on the environment. Researchers have developed various algorithms to significantly decrease the power consumption of computing systems. For instance, the Dynamic Voltage and Frequency Scaling (DVFS) approach is widely used in mobile devices to minimize power consumption [3]. This algorithm is implemented in wireless sensor networks to reduce power consumption by setting sleep intervals for sensors when they are not needed. In other words, energy-conscious task scheduling, cache management, and workload distribution are methods for creating green algorithms. Optimizing software code to enhance energy efficiency allows developers to reduce energy usage and enhance the overall environmental sustainability of software applications [8]. Therefore optimization strategies are essential for enhancing energy efficiency by minimizing computational demands and optimizing resource utilization. This involves eliminating redundant code, optimizing algorithms for performance, and reducing the number of instructions executed during program execution. The primary objective of this research is to develop and validate a new approach for Green ETL (GETL) processes that minimize energy consumption in modern data warehouse applications while maintaining high performance and reliability. Unlike previous studies that focus on energy-efficient hardware or general optimizations, our approach specifically targets energy consumption within ETL processes, a traditionally overlooked area in green computing. This work aims to develop an energy-efficient ETL framework that minimizes computing redundancy in the transformation phase. Our goal is to reduce the amount of energy consumed by ETL processes during the transformation phase. Despite progress in energy-efficient computing, ETL processess continue to experience considerable energy waste owing to redundant transformations across many pipelines. Current methodologies mostly emphasize hardware or query optimization, neglecting redundancy at the transformation level. This constraint inspires the proposed GETL, which minimizes unnecessary calculations and enhances energy efficiency via transformation reutilization. The main contributions of this paper are:

- A Green ETL (GETL) execution method for green data integration: We present GETL, a transformation-aware ETL execution model that cuts down on the energy consumption in a data warehouse system while preserving performance and reliability.
- Energy-efficient optimization of ETL transformations. We propose an optimization approach that identifies reusable transformations and enables their parallel and shared execution, thereby reducing execution time and energy consumption during the transformation phase.
- We propose an energy-efficient and transformation-aware ETL execution architecture that integrates workflow-level optimization and transformation reuse.
- Quantitative analysis of energy inefficiencies in traditional ETL processes, emphasizing computational redundancy and addressing a neglected area in green data warehouse research.

The remainder of this paper is structured as follows: Section 2 reviews existing research on energy consumption and energy-aware strategies in databases and data warehouses. Section 3 presents the proposed GETL approach. Section 4 describes the implementation details. Section 5 presents the experimental evaluation and results. Section 6 discusses the limitations of the study. Finally, Section 7 concludes the paper.

2 Related Work

Attaran et al. [9] and As Boiko et al. [10] point out, one of the major challenges in integrating green computing into data warehouse projects is the high energy consumption associated with storing and

operating on large amounts of data, as hosting, processing, and analyzing large data sets require significant computing power and, therefore, considerable energy expenditure. In addition, many organizations do not yet have energy-efficient infrastructure, which hinders the adoption of green computing principles. Despite these obstacles, the adoption of green computing in data storage technologies and big data opens the way to considerable benefits, such as reductions in operational costs. Indeed, energy-efficient techniques and technologies can reduce the energy consumption of data centers, thus leading to lower utility costs [11]. Researchers have proposed techniques to reduce energy consumption in data warehouses by optimizing storage and query processing. Several research have concentrated on energy-efficient data warehousing through the optimization of storage, query processing, and system architecture [12,13]. These works emphasize the importance of designing energy-efficient algorithms and hardware for data warehousing. With an emphasis on both the database management system (DBMS) and application eco-design, Bellatreche et al. [14] proposed a framework for incorporating energy efficiency into query optimizers of DBMS for data warehouses. Through experimentation, it evaluates the impact of energy considerations on Oracle DBMS and PostgreSQL. In order to balance energy consumption and query response time and improve overall system efficiency, Behzadnia et al. [15] proposed a dynamic power management model with model predictive control, data allocation optimization, and migration algorithms. It focused on DBMS energy-conscious disk storage management. Several studies have explored energy-efficient data center operations, including workload scheduling in conjunction with renewable energy sources [16,17], data reduction methodologies like compression and deduplication [18], and cooling optimization techniques [19]. Efficient cooling systems play a crucial role in minimizing the overall energy footprint of data warehouses [20] focused on dynamic resource allocation to reduce energy consumption by minimizing the number of active servers and consolidating workloads using virtualization. This approach enhances resource utilization and contributes to lower overall energy usage. Standardized metrics are crucial for assessing and contrasting the energy efficiency of data warehousing systems throughout their lifecycle. Hardware components, like the CPU, memory, and network interfaces, contribute to energy consumption; however, software behavior significantly influences overall energy utilization as well [5]. The application of green computing helps to save energy and decrease the energy consumption level of traditional computing systems [21]. Several recent studies and research efforts have focused on green computing, aiming to enhance the sustainability of the computing sector in terms of both hardware and software. Nazaré et al. [22] conducted a survey on of 74 works on green computing and its various subtopics, including the use of renewable energy sources, energy-efficient hardware design, software optimization, and sustainable behaviors. The study highlights that, green algorithms can help computing systems use less energy and leave a smaller carbon impact. The results emphasize the importance of implementing green computing practices in order to reduce the negative environmental effects of computing, such as waste production, energy consumption, and greenhouse gas emissions. Asad et al. [4] segmented the Big Data enterprise into six plans, which are considered crucial due to their impact on the energy consumption of data centers. They conducted a study on strategies aimed at making these six plans more environmentally friendly. Okewu et al. (2017) [23] classified the optimization of green computing knowledge in Africa particularly in Nigeria as a stochastic optimization problem. They employed a metaheuristic search algorithm to develop an online e-Green computing system aimed at promoting eco-friendly behavior among users. Their work addresses challenges such as limited bandwidth and unreliable power supply in emerging regions, proposing a green computing maturity model to enhance public awareness. However, the study's focus on Nigeria may limit the generalizability of its findings to other African countries with different socioeconomic conditions. To improve energy management, resource allocation, and task scheduling in green cloud computing environments. Authors in [24] employed machine learning and deep learning techniques in their green computing optimization algorithms to enhance resource allocation and reduce energy consumption. Specifically, convolutional neural networks (CNNs) and recurrent neural networks (RNNs) were utilized to improve architectural efficiency and deliver accurate temperature predictions, which support the development of alternative energy-efficient cooling strategies. Authors in [25] discuss AI-based green computing algorithms designed to minimize digital waste, optimize energy consumption, and reduce carbon footprints. These methods promote sustainable computing through efficient resource utilization and energy-

aware management. Recent studies propose task scheduling approaches based on Dynamic Voltage and Frequency Scaling (DVFS) to reduce energy consumption in cloud environments [26]. Furthermore, query optimization techniques have been investigated to improve the energy efficiency of relational and NoSQL databases [27]. However, these approaches do not fully explore all optimization opportunities for reducing energy consumption, providing only a partial view of the potential energy savings in database systems. Materialized View Selection (MVS) in data warehousing plays an important role in improving query performance by reducing access to source data. In [28], a technique is proposed for selecting energy-efficient logical data warehouse schemas using anti-monotonicity constraints to reduce the search space. While this approach demonstrates that logical architectures can significantly impact energy savings, it primarily focuses on logical optimization and does not address scalability or adaptability to diverse data types. Recent studies have further explored energy-efficient data processing and optimization in modern data platforms. For example, Yazidi [29] investigates green cloud and AI-optimized data architectures for enterprise systems, while Ejime et al. [30] analyze cost-performance trade-offs in large-scale ETL workloads in cloud-native environments. Similarly, Lalaoui et al. [31] propose energy-efficient architectures and AI-driven strategies for real-time big data processing. These works highlight the growing importance of energy-aware optimization in large-scale systems. However, existing approaches mainly focus on query optimization, hardware improvements, or workflow scheduling, without considering transformation-level reuse across multiple ETL pipelines within an energy-aware framework. This limitation motivates the proposed GETL approach, which introduces a transformation-aware execution architecture enabling cross-pipeline reuse of intermediate results. A comparison with related work is presented in Table 1.

Table 1: Comparative analysis of related work and the proposed GET.

Methods	Optimization	Energy Awareness	ETL-Specific	Cross-Pipeline Reuse	Workflow-Level Parallelism
Lang et al. [12]	Database cluster design	Yes	No	No	Limited
Harizopoulos et al. [13]	Energy-efficient DB systems	Yes	No	No	No
Bellatreche et al. [14]	Data warehouse design	Yes	Partial	No	No
Roukh et al. [32]	Data warehouse eco-design	Yes	Yes	No	No
Ghabri et al. [28]	Logical schema optimization	Yes	Partial	No	No
Boiko et al. [10]	Data integration architecture	Partial	Yes	No	Limited
Mahajan et al. [27]	Query optimization	Yes	No	No	Limited
GETL	ETL workflow optimization	Yes	Yes	Yes	Yes

3 The Propose GETL

This section presents the GETL framework. The novelty of our research resides not in the application of distributed computing per se, but in the formulation of an execution model that utilizes transformation-level reuse via a communal transformation cache, coupled with an adaptive execution planning approach. This method facilitates the eradication of superfluous computations inside ETL pipelines, enhancing both performance and energy efficiency.

3.1 Methodology

The proposed architecture comprises three primary components: data collection, the GETL engine, and the target data warehouse (Fig. 1).

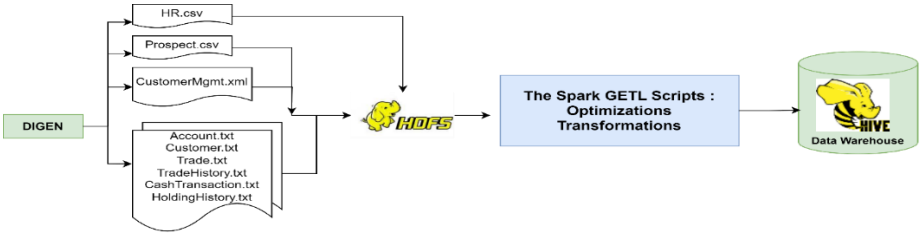


Figure 1: Architecture of the proposed GETL, including data ingestion, storage in Hadoop Distributed File System (HDFS), transformation via Spark GETL scripts, and loading into the data warehouse.

3.2 The Proposed GETL Optimization Algorithm Using Parallelization with APACHE spark and Shared Caching

GETL is based on four principles: the elimination of duplicate transformations, the reduction of disk input/output (I/O) via in-memory processing, the optimization of CPU utilization, and the facilitation of shared caching across pipelines. These principles enhance resource efficiency and overall efficacy.

3.2.1 Shared Transformation Cache (STC)

The STC is inspired by the shortcomings of Apache Spark's own caching mechanisms in particular, "persist ()" and "cache ()", which are limited to the job scope, are not semantically aware of transformations, and operate reactively, rather than proactively applying reuse across workflows. The STC is implemented as a key-value framework, with keys representing transformation signatures and values representing cached intermediate datasets.

On the other hand, STC adds a semantic, transformation-aware caching layer that facilitates the reuse of intermediate ETL results across jobs and pipelines, thereby eliminating redundant computation at a large extent and improving resource efficiency in distributed data processing.

Definition 1: *Let,* D *be an input dataset;* $T = \{t_1, t_2, \dots, t_n\}$ *be a sequence of transformations and* $R = T(D)$ *be the transformed result. Every transformation is linked to a distinct signature characterized as a function of its input dataset, transformation logic, and parameters. These signatures function as keys in the Shared Transformation Cache (STC), facilitating the efficient identification and reutilization of previously computed outcomes. The STC utilizes a Least Recently Used (LRU) eviction strategy alongside memory thresholds to manage memory consumption, ensuring that frequently accessed transformations are retained while outdated entries are eliminated.*

The principle of our optimization method, is to divide an ETL workflow or pipeline into groups of sub-flows. Each of these groups must contain sub-flows that can be executed in parallel.

Our optimization approach, parallelization with Apache Spark and shared caching achieves optimization by implementing new ETL processes for shared caching and parallelization during runtime. A parallel algorithm is made up of several separate job modules, some of which are capable of running concurrently. In contrast to Spark's inherent caching techniques, which depend on execution lineage and are restricted to intra-job reuse, GETL facilitates cross-pipeline reuse grounded in transformation semantics.

The proposed GETL processes optimization approach includes three main steps (Fig. 2), as follows:

A. *Find the execution trees that make use of a cache.* Line-synchronized components that follow one another can be organized into an execution tree. The algorithm operates recursively, starting from the root of the Directed Acyclic Graph (DAG) and continuously examining the next activity. If the activity is line-synchronized, it is added to the execution tree. If not, the current execution tree is completed with the previous activity, and a new tree begins with the next one. By using a single cache for each execution tree, data is not duplicated between different execution trees. The execution trees found are linear flows.
B. *Parallelize an execution tree's operations.* In an execution tree, all activities are executed per row and can be processed in parallel. The maximum parallelization is the number of rows in the input recordset.

The degree of parallelism ranges from one to the number of input rows. An optimization algorithm dynamically determines the optimal degree of parallelism, defined as the number of parallel execution pipelines that minimizes the overall execution time of the execution tree.

C. *Execution within an activity can also be parallelized.* Execution is divided into threads. When all threads terminate, the output is merged using a row-order synchronizer. The row-order synchronizer ensures that the order of the input and output data rows is the same. This is necessary if, for example, a merge action follows.

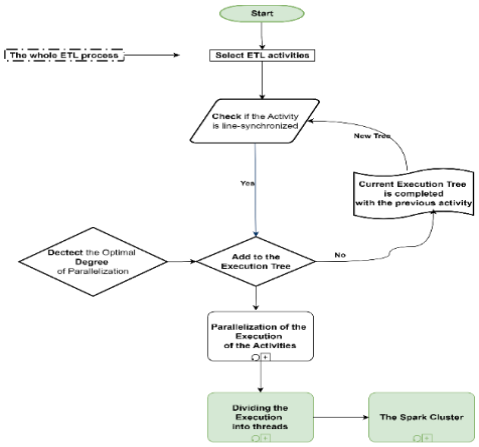


Figure 2: The proposed algorithm.

The pseudo-code in Algorithm 1 describes the construction of the execution tree and the computation of the optimal degree of parallelism at each stage. It also illustrates how ETL transformations are executed in parallel while leveraging a shared transformation cache to overcome the limitations of Spark's LRU-based caching and job-level execution.

Algorithm 1: GETL Execution Planning

Input:

A = {a1, a2, …, an} //Set of ETL activities, each activity is a set of operations [33].
R //Available cluster resources
C //Shared Transformation Cache (STC)

Output:

Executed ETL pipeline with optimized parallelism

```
1: Initialize empty Execution_Tree T
2: Set current_Node ← root (T)
3: for each activity ai ∈ A do
4:       if is_Line_Synchronized (ai) then
5:             finalize_Subtree(T, current_Node)
6:             current_Node ← add_Sequential_Node (T, ai)
7:       else
8:             current_Node ← add_Parallel_Node (T, ai)
9:       end if
10: end for
11: for each level Lj in T do
12:       DoPj ← compute_Optimal_Parallelism (Lj, R)
13: end for
14: for each node ni in T do
15:       if C.contains (semantic_Hash(ni)) then
16:             reuse_Cached_Result (ni)
17:       else
18:             execute_Transformation (ni)
```

```
19:             C.store (semantic_Hash(ni), result (ni))
20:         end if
21: end for
22: for each level Lj in T do
23:         spark_Activities ← divide_Into_Threads (Lj, DoPj (5))
24:         submit_To_Spark_Cluster (spark_Activities)
25:         wait For Completion (Lj)
26: end for
27: return execution_Success
```

The degree of The an ETL $activity_i$, formally defined as:

$$DoP_i = \min(|L_i|, \frac{CPU_{available}}{CPU_{perOp}}, \frac{Memory_{available}}{Memory_{perOp}}) \quad (1)$$

DoP_i, denotes the greatest quantity of concurrent operations that can be performed without surpassing available resources. It is restricted by algorithmic limitations and system resources (CPU and memory). This architecture allows GETL to dynamically ascertain an optimal and secure level of parallelism during execution, preventing resource overutilization and performance decline.

4 Implementation

We use Apache Spark with parallelization and shared caching to develop energy-efficient ETL procedures, and we use the TPC-DI benchmark, a standard for data integration, to evaluate our GETL methodology. By maximizing resource utilization (CPU, memory) and performance (execution time) via distributed computing, our innovative optimization algorithm reduces energy consumption. Fig. 3 shows the entire GETL process. The simulation is conducted in the following four stages:

1. *Input Data Generation.* Synthetic data are generated by TPC-DI benchmark using the Digen tool, generating heterogenous data sources.
2. *ETL Pipeline Execution.* Output files are ingested into ETL pipelines running on HDFS, each executing a fixed series of extract, transform, and load steps that simulate a real data warehouse ingestion process.
3. *GETL-Enabled Spark Processing.* ETL transformations are performed using Spark with GETL optimizations, minimizing redundant computations.
4. *Data Warehouse Loading Appache Hive.*
5. *Monitoring and Metric Collection.* Execution time and power consumption are logged during Spark execution, and energy is calculated from these measurements.

This end-to-end process guarantees that GETL is tested with actual ETL workloads and isolates the effect of transformation reuse on energy consumption at system level. The cache lookup procedure utilizes hash-based indexing, achieving an average time complexity of $O(1)$, hence rendering it considerably more efficient than recalculating transforms.

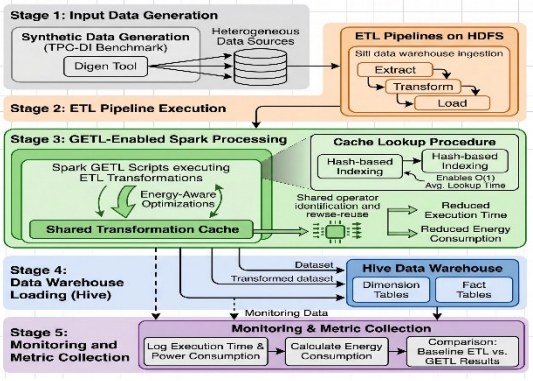


Figure 3: GETL framework overview.

5 Experiments and Results

This section presents the experimental evaluation of the proposed GETL approach. We describe the experimental setup, including the TPC-DI benchmark and the hardware/software environment, as well as the performance and energy-efficiency metrics, as well as the performance and energy-efficiency metrics. To further validate the effectiveness of the proposed approach, we compare GETL with a standard Spark-based ETL implementation, representing a widely adopted baseline in existing data processing systems. Experimental results demonstrate that GETL consistently outperforms the baseline in terms of execution time, resource utilization, and energy consumption across all workloads. These findings confirm the superiority of GETL, particularly in large-scale scenarios, where transformation reuse and optimized execution significantly reduce redundant computations and improve overall efficiency.

5.1 Experimental Setup

To evaluate the effectiveness of the proposed Green ETL (GETL) approach, we conducted a series of experiments on a local machine with high-performance specifications. The hardware environment consisted of an Intel(R) Core(TM) i7-8565U CPU @ 1.80 GHz 1.99 GHz, 16 GB RAM, and a 64-bit Windows 11 operating system. The software tools used included Oracle 12 for database management, Apache Spark 2.4.4 for parallel data processing, Scala 2.11 for ETL scripting, and Eclipse IDE for development activities. All ETL processes were developed using Scala programming language and executed on a local Spark setup to leverage parallelization effectively. A Python-based monitoring application was created to continuously record CPU utilization, memory utilization, and energy consumption during ETL execution. The baseline refers to a standard Spark-based ETL implementation devoid of any optimization strategies.

The experiments were performed in a single-node setting to ensure contolled measurement of performance and energy consumption. This configuration does not account for the implications of distributed systems, such as network latency and inter-node communication, but it facilitates the isolation of the impact of transformation reuse. Power usage was assessed via a software-based monitoring method. A Python script was employed to gather CPU and memory utilization during execution, while power consumption was estimated using a utilization-based methodology. This estimation method is frequently employed in energy-aware computing research to provide relative comparisons among various execution methods under uniform experimental settings. Consequently, the provided energy levels are to be regarded as comparative measures of efficiency rather than precise physical measurements. To ensure reproducibility, system configurations remained unchanged, and each experiment was repeated ten times under uniform settings for each configuration, encompassing both the traditional ETL and the proposed GETL across various scale factors.

Reproducibility. To ensure reproducibility, the implementation of the proposed GETL approach is publicly available at: *https://github.com/gueddoudjelyazid8/GETL-Execution-Planning*.

5.2 Dataset

The dataset utilized in the experiments were generated using the DIGen tool [34] based on the TPC-DI benchmark, with scale factors ranging from 1 to 28 (1, 5, 8, 12, 15, 18, 21, 25, 28). These scale factors will determine the size of the generated data. After generation, the dataset will be loaded into the Hadoop Distributed File System (HDFS), ensuring it is accessible across the distributed nodes. The data will then undergo transformation and cleaning using GETL process scripts, which are written in Scala.

5.3 Evaluation Metrics

The evaluation of the GETL framework was conducted using a tripartite metric system focused on performance and sustainability. The execution time T, defined as the total duration of each ETL pipeline's execution, served as the primary performance indicator. Energy consumption E is estimated as $E = P \times T$, where P represents the average power obtained from system ressource utilization. This estimation

serves to compare the relative energy efficiency of different approaches rather than to provide absolute power measurements.

5.4 Statistical Analysis

Statistical indicators such as mean, standard deviation, variance, and confidence intervals were calculated (Table 2) to guarantee the robustness and trustworthiness of the results. Mean: $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x_i}$; Standard Deviation: $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x_i} - \boldsymbol{\mu})^2$ and Confidence Interval: $\boldsymbol{CI} = \boldsymbol{\mu} \pm \mathbf{1.96} \cdot \frac{\sigma}{\sqrt{n}}$. These metrics assess central tendency, variability, and confidence in the observed outcomes. To assess the scalability of the proposed GETL, tests were performed across various scale factors (SF), encompassing both small and big workloads. Fig. 4 illustrates the execution time of both GETL and the baseline at varying scale factors, emphasizing the influence of workload size on system performance.

Table 2: Execution time statistical analysis.

Method	Mean (s)	Std Dev (s)	Variance	95% CI
Traditional ETL	120.0	±1.9	3.61	±1.2
GETL	85.3	±1.6	2.56	±1.0

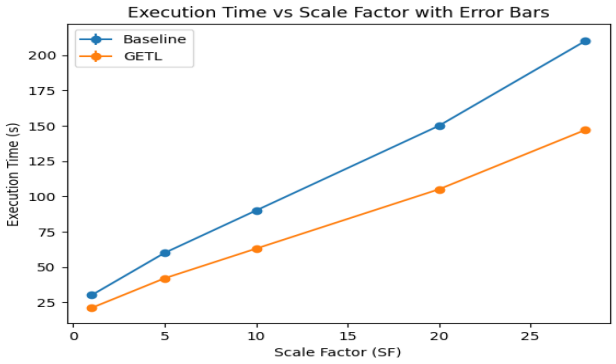


Figure 4: Execution time as a function of the scale factor for both GETL and the traditional ETL. Error bars are the standard deviation from 10 repeated runs.

Fig. 4 shows that execution time increases with the scale factor for both methods, indicating a higher workload. Nonetheless, GETL consistently outperforms the traditional ETL method at all scales. The widening gap between the curves underscores its scaling advantage, while the minimal error bars signify low variability and consistent performance.

5.5 Experimental Results and Analysis

5.5.1 CPU and Memory Usage Comparison

This experiment examines CPU and memory utilization under different data workloads for both the traditional ETL process and the GETL approach. The results are presented in Table 3a,b, respectively. Table 3a shows that CPU utilization in the traditional ETL process increases significantly with workload size, attaining 91% at a scaling factor of 28, while memory utilization reaches 92%. In contrast, Table 3b demonstrates that GETL exhibits much reduced resource utilization, with CPU Utilization limited to 50% and memory utilization at 41% under identical workload conditions. Fig. 5a,b illustrates these results.

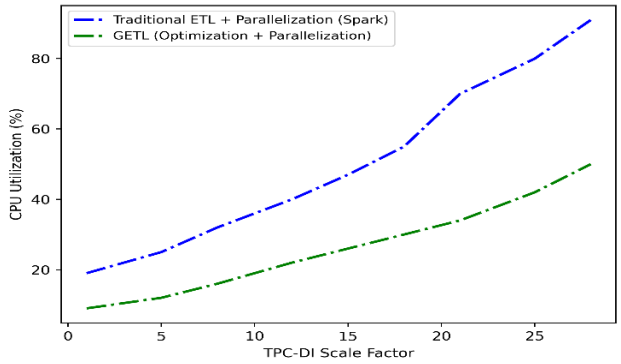
Table 3: (**a**). Traditional ETL + Parallelization (Spark); (**b**). GETL (Optimization + Parallelization).

(**a**)

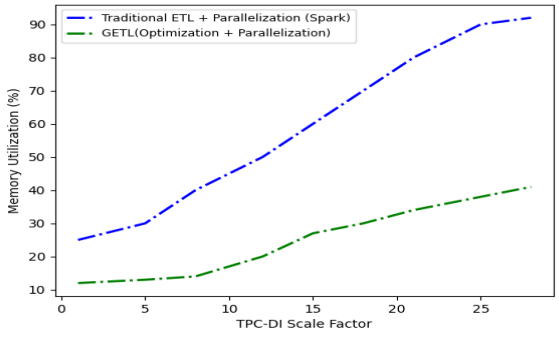
Various ETL Workloads	CPU Utilization (%)	Memory Utilization (%)
1	19	25
5	25	30
8	32	40
12	40	50
15	47	60
18	55	70
21	70	80
25	80	90
28	91	92

(**b**)

Various ETL Workloads	CPU Utilization (%)	Memory Utilization (%)
1	08	12
5	12	13
8	16	14
12	22	20
15	26	27
18	30	30
21	34	34
25	42	38
28	50	41



(**a**)



(**b**)

Figure 5: (**a**). CPU utilization (%) across TPC-DI scaling factors: Traditional ETL (Spark-based) vs. GETL; (**b**). Memory utilization (%) across TPC-DI scaling factors: Traditional ETL (Spark-based) vs. GETL.

5.5.2 Energy Consumption Evaluation

The second experiment extendes the analysis by evaluating the energy consumption associated with both ETL approaches. The relationship between resource utilization (CPU and memory) and the

corresponding energy consumption is critical for assessing the sustainability of large-scale data processing operations. As shown in Table 4a,b, the results clearly indicate that GETL significantly reduces both resource utilization and energy consumption compared to the traditional ETL approach. For example, at a scaling factor of 28, GETL consumes only 21 W, whereas the traditional process consumes 67 W. This representing an energy saving of approximately 68%, demonstrating the substantial environmental impact that can be achieved through intelligent software optimization. Fig. 6a,b clearly illustrates that the energy consumption curve for traditional ETL increases at a much faster rate than for GETL, emphasizing the scalability and sustainability of the proposed method. The findings suggest that software-level optimizations introduced by GETL can significantly complement hardware-level energy-saving strategies, paving the way for greener and more efficient data processing architectures. While average energy savings are around 30% across workloads, higher reductions can be observed in specific scenarios depending on resource utilization patterns.

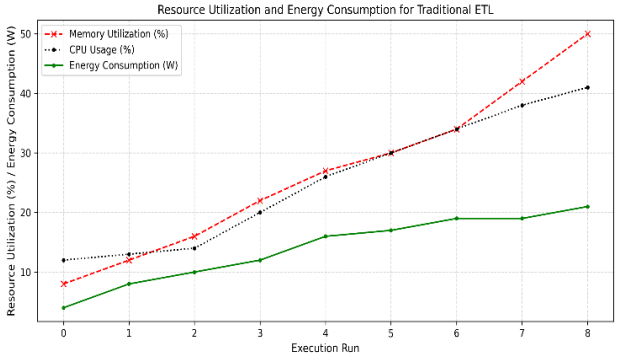
Table 4: (**a**). Resource utilization and energy consumption for the proposed GETL across TPC-DI scaling factors; (**b**). Resource utilization and energy consumption for the traditional ETL approach across TPC-DI scaling factors.

(**a**)

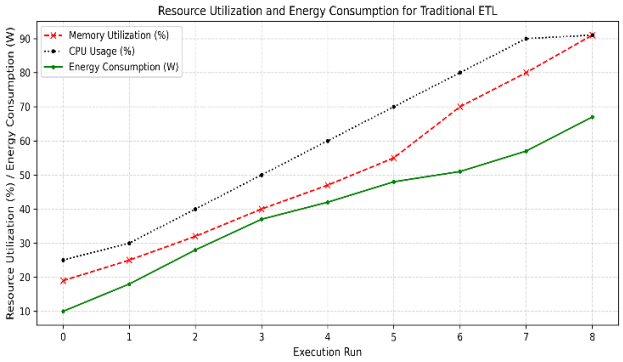
Various ETL Workloads	CPU Utilization (%)	Memory Utilization (%)	Energy consumption (w)
1	08	12	4
5	12	13	8
8	16	14	10
12	22	20	12
15	26	27	16
18	30	30	17
21	34	34	19
25	42	38	19
28	50	41	21

(**b**)

Various ETL Workloads	CPU Utilization (%)	Memory Utilization (%)	Energy consumption (w)
1	19	25	10
5	25	30	18
8	32	40	28
12	40	50	37
15	47	60	42
18	55	70	48
21	70	80	51
25	80	90	57
28	91	92	67



(**a**)



(**b**)

Figure 6: (**a**). Evolution of resource utilization (%) and energy consumption (W) for the traditional ETL approach across execution runs; (**b**). Evolution of resource utilization (%) and energy consumption (W) for the GETL approach across execution steps.

5.5.3 Energy Consumption Reduction

To quantify the energy consumption of ETL processes, we adopt a resource-utilization-based power model inspired by [35,36]. Table 5 and Fig. 7 compare the energy consumption of Traditional vs. Optimized ETL processes under different workloads (5, 8, and 12 GB). In all cases, the optimized ETL achieves consistent energy savings of approximately 27% to 30%, confirming that the approach scales efficiently with increasing data volumes. These results confirm the effectiveness of transformation reuse and optimized execution in reducing energy consumption across varying workloads.

Table 5: Estimated energy consumption of Traditional ETL and GETL across different workloads.

Workload (GB)	Process	Energy (Wh)	Reduction (Wh)	Reduction (%)
5	Traditional ETL	30	–	–
	GETL	21.8	8.2	27.3%
8	Traditional ETL	48	–	–
	GETL	34.1	13.9	28.9%
12	Traditional ETL	72	–	–
	GETL	50.7	21.3	29.6%

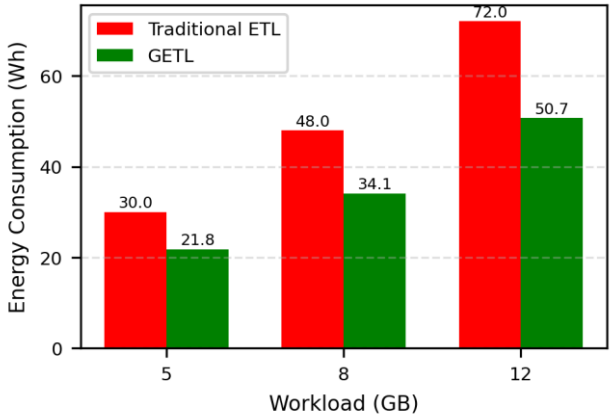


Figure 7: Comparison of energy consumption traditional ETL vs. GETL across different workloads.

5.6 Summary of Findings

The experimental results collectively confirm that the proposed GETL approach significantly improves CPU and memory efficiency, reduces energy consumption, and shortens execution time when compared to traditional ETL processes. These benefits become more pronounced as the size of the dataset increases, indicating strong scalability. The integration of GETL not only enhances system performance but also aligns with the broader objectives of environmental sustainability and energy efficiency in data management practices. Organizations adopting GETL can achieve substantial reductions in their operational costs while simultaneously reducing their carbon footprint, offering a dual advantage in environmentally conscious contexts. Performance improvements vary depending on the scale factor, with more significant benefits noted in large-scale workloads, achieving up to 68% under optimal conditions.

6 Limitations

This section discusses the primary constraints and assumptions of the proposed GETL method. Despite the encouraging outcomes achieved with the proposed GETL, there are limitations must be recognized. The initial experimental evaluation was performed in a single-node environment. This configuration facilitates the precise assessment of execution time, resource use, and energy consumption; but, it fails to consider network overhead, inter-node communication, or cluster-level scheduling dynamics. Future work will focus on extending the evaluation to multi-node distributed environments to assess the scalability and robustness of GETL under real-world workloads. Energy measurements are based on software-level estimation models instead of hardware power meters, thereby compromising absolute accuracy but still being suitable for relative comparisons.

7 Conclusion and Future Work

In this paper, we investigated the applicability of green computing in large-scale ETL processes in data warehouse context. We proposed a Green ETL (GETL) framework, a novel system-level approach designed to reduce energy consumption by enabling transformation-level reuse across multiple ETL pipelines and

by coordinating their execution through adaptative parallelism. The experimental results based on the TPC-DI benchmark demonstrated that GETL achieves significant improvements in both energy consumption and execution time compared to traditional ETL approaches, thus validating the effectiveness of transformation reuse and workflow-level optimization. The study also reveals that energy waste in contemporary data warehouses is dominated by repeated computation of the same intermediate results during the transformation phase, and such inefficiency can be alleviated by intelligent caching and scheduling techniques such as those provided by GETL. Future work will focus on extending GETL to distributed and cloud-based environments to investigate how transformation sharing can be exploited across nodes and workloads at even larger scale. Other research directions include using learning-based scheduling algorithms to adapt caching and execution choices dynamically, and incorporating carbon-aware scheduling to match ETL execution with clean energy availability. These extensions will further demonstrate the potential of GETL for sustainable and efficient big data processing. Furthermore, we plan to explore dynamic and workload-aware optimization techniques that adjust execution plans during runtime. A further intriguing avenue entails the incorporation of machine learning methods to facilitate self-adaptive ETL optimization and astute resource management.

Acknowledgement: Not applicable.

Funding Statement: The author(s) received no specific funding for this study.

Author Contributions: El Yazid Gueddoudj, Abdelouahab Attia and Ali Wagdy Mohamed have executed conceptualization, investigation, analysis, visualization, project management, resources. El Yazid Gueddoudj, Ali Wagdy Mohamed, Esam Y. O. Zafar, Kamal M. Othman, Abdulfattah Noorwali and Abdulaziz T. Almaktoom contributes to the software, method, data analysis, simulation, writing original draft, reviewing and editing. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data used to support the findings of this study are available from the corresponding authors upon request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Verdecchia R, Lago P, Ebert C, de Vries C. Green IT and green software. IEEE Softw. 2021;38(6):7–15. doi:10.1109/MS.2021.3102254.
2. Poess M, Nambiar RO. Energy cost, the key challenge of today's data centers: A power consumption analysis of TPC-C results. Proc VLDB Endow. 2008;1(2):1229–40. doi:10.14778/1454159.1454162.
3. Ba H, Heinzelman W, Janssen CA, Shi J. Mobile computing—A green computing resource. In: 2013 IEEE Wireless Communications and Networking Conference (WCNC); 2013 Apr 7–10; Shanghai, China. p. 4451–6. doi:10.1109/WCNC.2013.6555295.
4. Asad Z, Rehman Chaudhry MA. A two-way street: Green big data processing for a greener smart grid. IEEE Syst J. 2017;11(2):784–95. doi:10.1109/JSYST.2015.2498639.
5. Sabharwal M, Agrawal A, Metri G. Enabling green IT through energy-aware software. IT Prof. 2013;15(1):19–27. doi:10.1109/mitp.2012.104.
6. Poess M, Nambiar RO. Tuning servers, storage and database for energy efficient data warehouses. In: Proceedings of the 2010 IEEE 26th International Conference on Data Engineering (ICDE 2010); 2010 Mar 1–6; Long Beach, CA, USA. p. 1006–17. doi:10.1109/ICDE.2010.5447806.
7. Paul SG, Saha A, Arefin MS, Bhuiyan T, Biswas AA, Reza AW, et al. A comprehensive review of green computing: Past, present, and future research. IEEE Access. 2023;11:87445–94. doi:10.1109/ACCESS.2023.3304332.
8. Saleem M, Shakir M, Usman M, Bajwa M, Shabbir N, Shams Ghahfarokhi P, et al. Integrating smart energy management system with Internet of Things and cloud computing for efficient demand side management in smart grids. Energies. 2023;16(12):4835. doi:10.3390/en16124835.

9. Attaran M, Stark J, Stotler D. Opportunities and challenges for big data analytics in US higher education: A conceptual model for implementation. Ind High Educ. 2018;32(3):169–82. doi:10.1177/0950422218770937.
10. Boiko O, Shendryk V, Malekian R, Komin A, Davidsson P. Towards data integration for hybrid energy system decision-making processes: Challenges and architecture. In: Proceedings of the 19th International conference on Information and Software Technologies (ICIST); 2023 Oct 12–14; Kaunas, Lithuania. Cham, Switzerland: Springer; 2024. p. 172–84. doi:10.1007/978-3-031-48981-5_14.
11. Ahmadi S. A comprehensive study on integration of big data and AI in financial industry and its effect on present and future opportunities. Int J Curr Sci Res Rev. 2024;7(1):66–74. doi:10.47191/ijcsrr/v7-i1-07.
12. Lang W, Harizopoulos S, Patel JM, Shah MA, Tsirogiannis D. Towards energy-efficient database cluster design. Proc VLDB Endow. 2012;5(11):1684–95. doi:10.14778/2350229.2350280.
13. Harizopoulos S, Shah MA, Meza J, Ranganathan P. Energy efficiency: The new holy grail of data management systems research. In: Proceedings of the 4th Biennial Conference on Innovative Data Systems Research (CIDR); 2009 Jan 4–7; Asilomar, CA, USA. p. 1–8.
14. Bellatreche L, Roukh A, Bouarar S. Step by step towards energy-aware data warehouse design. In: Marcel P, Zimányi E, editors. Proceedings of the European Big Data Management and Analytics Summer School (eBISS); 2016 Jul 3–8; Tours, France. p. 105–38. doi:10.1007/978-3-319-61164-8_5.
15. Behzadnia P, Tu YC, Zeng B, Yuan W. Energy-aware disk storage management: Online approach with application in DBMS. Int J Database Manag Syst. 2017;9(1):1–22. doi:10.5121/ijdms.2017.9101.
16. Jin X, Zhang F, Vasilakos AV, Liu Z. Green data centers: A survey, perspectives, and future directions. arXiv:1608.00687. 2016.
17. Goiri Í, Le K, Haque ME, Beauchea R, Nguyen TD, Guitart J, et al. GreenSlot: Scheduling energy consumption in green datacenters. In: Proceedings of the 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'11); 2011 Nov 12–18; Seattle, WA, USA. New York, NY, USA: ACM; 2011. p. 1–11. doi:10.1145/2063384.2063411.
18. Kipps K, Jones AK. Data management in the cloud. Collect Manag Cloud. 2025;32:37–54. doi:10.5040/9798216406143.ch-3.
19. Moore J, Chase J, Ranganathan P, Sharma R. Making scheduling "cool": Temperature-aware workload placement in data centers. In: Proceedings of the USENIX Annual Technical Conference (USENIX ATC'05); 2005 Apr 10–15; Anaheim, CA, USA. Berkeley, CA, USA: USENIX Association; 2005. p. 61–74. doi:10.5040/9798216406143.ch-3.
20. Beloglazov A, Buyya R. Energy efficient resource management in virtualized cloud data centers. In: Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid 2010); 2010 May 17–20; Melbourne, Australia. Piscataway, NJ, USA: IEEE; 2010. p. 826–31. doi:10.1109/CCGRID.2010.46.
21. Li Q, Zhou M. The survey and future evolution of green computing. In: Proceedings of the 2011 IEEE/ACM International Conference on Green Computing and Communications (GreenCom 2011); 2011 Aug 4–5; Chengdu, China. Piscataway, NJ, USA: IEEE; 2011. p. 230–3. doi:10.1109/GreenCom.2011.47.
22. Nazaré T, Gadelha J, Nepomuceno E, Lozi R. Green computing for energy transition: A survey. IEEE Lat Am Trans. 2023;21(9):937–48. doi:10.1109/TLA.2023.10251799.
23. Okewu E, Misra S, Maskeliūnas R, Damaševičius R, Fernandez-Sanz L. Optimizing green computing awareness for environmental sustainability and economic security as a stochastic optimization problem. Sustainability. 2017;9(10):1857. doi:10.3390/su9101857.
24. Ranjana S, Meenakshi A. Machine learning and deep learning algorithms for breast cancer prediction. In: Proceedings of the International Conference on Deep Sciences for Computing and Communications; 2023 Apr 20–22; Chennai, India. p. 109–17. doi:10.1007/978-3-031-68905-5_11.
25. Dahmani S. Computational intelligence for green cloud computing and digital waste management. In: Artificial intelligence and IoT-based technologies for sustainable computing. Hershey, PA, USA: IGI Global Scientific Publishing; 2024. p. 248–66. doi:10.4018/979-8-3693-1552-1.ch013.
26. Khullar R, Hossain G. A new algorithm for energy efficient task scheduling towards optimal green cloud computing. In: Proceedings of the 2022 IEEE/ACS 19th International Conference on Computer Systems and Applications (AICCSA 2022); 2022 Dec 5–8; Abu Dhabi, United Arab Emirates. p. 1–6. doi:10.1109/AICCSA56895.2022.10017609.
27. Mahajan D, Blakeney C, Zong Z. Improving the energy efficiency of relational and NoSQL databases via query optimizations. Sustain Comput Inform Syst. 2019;22:120–33. doi:10.1016/j.suscom.2019.01.017.
28. Ghabri I, Bellatreche L, Ben Yahia S. Selection of a green logical data warehouse *Schema* by anti-monotonicity constraint. In: Proceedings of the 46th International Conference on Current Trends in Theory and Practice of Informatics (SOFSEM 2020); 2020 Jan 20–24; Limassol, Cyprus. Cham, Switzerland: Springer; 2020. p. 350–61. doi:10.1007/978-3-030-38919-2_29.

29. Yazidi A. Energy efficient green cloud and AI optimized data lake architectures for enterprise digital transformation. Int J Adv Eng Sci Inf Technol. 2026;9:42.
30. Ejime O, Eleje E, Kosemani A, Epke M. Cost-performance trade-offs in large-scale ETL workloads: Evidence from cloud-native data platforms. 2025. doi:10.2139/ssrn.6170626.
31. Lalaoui IL, El Haji E, Kounaidi M. Energy-efficient architectures and AI-driven strategies for real-time big data processing. In: Energy-efficient algorithms and systems in computing. Cham, Switzerland: Springer; 2025. p. 123–37. doi:10.1007/978-3-032-04114-2_8.
32. Roukh A, Bellatreche L, Boukorca A, Bouarar S. Eco-DMW: Eco-design methodology for data warehouses. In: Proceedings of the ACM 18th International Workshop on Data Warehousing and OLAP (DOLAP 2015); 2015 Oct 23; Melbourne, Australia. New York, NY, USA: ACM; 2015. p. 1–10. doi:10.1145/2811222.2811230.
33. Gueddoudj EY, Chikh A, Attia A. Os-ETL: A high-efficiency, open-scala solution for integrating heterogeneous data in large-scale data warehousing. Ing Syst Inf. 2023;28(3):557–65. doi:10.18280/isi.280303.
34. Poess M, Rabl T, Jacobsen HA, Caufield B. TPC-DI: The first industry benchmark for data integration. Proc VLDB Endow. 2014;7(13):1367–78. doi:10.14778/2733004.2733009.
35. Kansal A, Zhao F, Liu J, Kothari N, Bhattacharya AA. Virtual machine power metering and provisioning. In: Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC'10); 2010 Jun 10–11; Indianapolis, IN, USA. New York, NY, USA: ACM; 2010. p. 39–50 doi:10.1145/1807128.1807136.
36. Berl A, Gelenbe E, Di Girolamo M, Giuliani G, De Meer H, Dang MQ, et al. Energy-efficient cloud computing. Comput J. 2010;53(7):1045–51. doi:10.1093/comjnl/bxp080.