EcoLLM: Energy-Aware Benchmarking of LLMs for Data Processing Workloads

Pratyush Agnihotri
TU Darmstadt and DFKI

Manisha Luthra
Ruhr University Bochum and TU Darmstadt

Carsten Binnig
hessian.AI, TU Darmstadt and DFKI

Abstract

Large language models (LLMs) are increasingly integrated into data management systems, yet their energy consumption remain largely unexplored. In this paper, we present EcoLLM, a workload-centric benchmark for studying energy-aware trade-offs of LLMs in data-processing tasks. EcoLLM models workload families, operator complexity, and data scale as well as supports both local and API-based models, and employs a hybrid energy measurement methodology. Our evaluation reveals several non-obvious findings. First, energy efficiency and task effectiveness are not aligned: highly efficient models can fail completely on pipeline generation tasks, while more energy-intensive models achieve correct results. Second, we observe a consistent trade-off between latency and energy, where lower latency is often achieved at disproportionately higher energy cost, leading to distinct execution regimes across deployment modes. Finally, per-task energy consumption appears small yet it scales to substantial cost at production workloads, making energy a critical factor in system design. These findings highlight the need for energy-aware and workload-aware model selection in LLM-based data systems.

CCS Concepts

• **Computer systems organization → Real-time systems**.

Keywords

Sustainable data systems, Energy-aware benchmarking, LLM-based data systems

ACM Reference Format:
Pratyush Agnihotri, Manisha Luthra, and Carsten Binnig. 2026. EcoLLM: Energy-Aware Benchmarking of LLMs for Data Processing Workloads. In *Nineth International Workshop on Exploiting Artificial Intelligence Techniques for Data Management (aiDM '26), May 31-June 05, 2026, Bengaluru, India.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3814940.3815324

1 Introduction

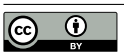
Why Energy-Aware Benchmarking Matters? Energy consumption has become a first-class concern in data-intensive computing. Data centers already account for a significant share of global electricity demand [10, 19], and recent reports show that AI workloads are accelerating this trend [26–28, 41, 64, 67]. Even small per-request energy costs (e.g., 0.001 kWh) scale to megawatt-hours at production workloads, making energy a critical factor in large-scale ML-based data systems. As a result, energy use and its CO_2 emissions directly impact system design [9, 57, 59]. Despite this shift, database research has traditionally focused on optimizing performance metrics such as latency and throughput [20, 30], and more recently accuracy and monetary cost in ML-driven systems. *Energy consumption, however, remains largely unaccounted for.* This creates a critical gap: system design decisions increasingly involve energy trade-offs, yet this dimension is rarely understood.

Energy Remains Largely Unexplored in Data Systems. Recent work integrates Large Language Models (LLMs) directly into data-processing pipelines as semantic operators, enabling tasks such as filtering, extraction, and reasoning over structured and unstructured data. Systems such as Palimpzest [34], Lotus [45], DocETL [56], ThalamusDB [31], and CAESURA [63] exemplify this paradigm. In this setting, LLMs become part of the physical execution layer of query plans, introducing new optimization choices such as model selection, prompting strategies, and deployment modes (local vs. API-based). These choices are typically evaluated in terms of accuracy, latency, and monetary cost, yet *their energy implications remain largely unexplored*, despite potentially large variation across models and workloads.

Energy Consumption Not Reported in Benchmarks. Benchmarking is central to evaluating systems and guiding design decisions, but existing approaches fall short in reporting energy. Traditional database benchmarks focus on performance [2, 7, 13, 40, 47, 69], while LLM benchmarks emphasize task accuracy on NLP workloads [11, 12, 14, 22, 33]. *Energy consumption is largely absent from both.* Thus, current benchmarks provide limited insight into how LLMs behave when embedded in data-processing pipelines, particularly as workload complexity and data scale increase.

Research Question. This gap motivates the research question: *How do accuracy and latency compare to energy consumption when LLMs are used in database workloads, and whether smaller or cheaper models achieve similar accuracy with significantly lower energy use?*

Our Approach: We introduce **EcoLLM**, a workload-centric benchmark for analyzing energy–accuracy–performance trade-offs of LLMs in data-management scenarios. Unlike prior benchmarks, EcoLLM models structured pipelines where LLMs act as operators and exposes three key dimensions: (i) *workload families (Family 1– Family 4)*, capturing different classes of data-processing tasks such as reasoning, anomaly detection, analytics, and code generation, (ii) a *complexity ladder (C1–C4)*, capturing increasing operator composition, and (iii) *table size*, capturing input data scale. The benchmark defines database tasks with deterministic scoring and adopts a hybrid energy measurement methodology, combining local resource-based estimation with provider-side attribution (e.g., EcoLogits [50]). While focusing on *energy consumption and carbon*



aiDM '26, Bengaluru, India

ACM ISBN 979-8-4007-2719-1/2026/05
https://doi.org/10.1145/3814940.3815324

footprint, EcoLLM jointly reports accuracy and latency to characterize trade-offs.

Contributions. EcoLLM[1] is an energy-aware benchmark for LLMs in data-management workloads that captures energy emissions alongside accuracy and latency. The benchmark models LLMs as logical operators within structured pipelines such that LLMs are used either for execution or for pipeline construction enabling systematic analysis across data scale and workload complexity. To report energy consumption, we propose a hybrid energy measurement methodology that combines local resource-based estimation with provider-side attribution for API-based inference.

Using EcoLLM, we empirically analyze how energy consumption varies across models, workloads, and deployment modes. Our results reveal several non-obvious findings. First, lower latency often comes at the cost of higher energy. Second, higher energy does not guarantee better results (e.g., in terms of accuracy). Third, model choice is critical: only some models benefit from increased computation, while others fail regardless of complexity. Finally, deployment mode plays a central role. Offline models trade speed for lower energy and can improve with more complex tasks when capable, whereas API-based models remain fast and consistent but incur significantly higher energy cost. These findings show that selecting the right model depends on the desired balance between speed, energy, and correctness.

Structure of paper. In the following, we first present an overview on the benchmark in Section 2, followed by the benchmark evaluation in Section 3, related work in Section 4 and conclusion in Section 5.

2 EcoLLM Benchmark Overview

EcoLLM is an energy- and workload-centric benchmark designed to study how LLMs behave when used as *semantic operators within query execution pipelines*, analogous to physical operators in data management systems. Its goal is to enable systematic analysis of trade-offs between task quality, latency, monetary cost, energy consumption and CO_2 emission under controlled variations in workload complexity and data scale.

In contrast to existing LLMs benchmarks that primarily focus on natural-language reasoning, EcoLLM models data-processing workloads where LLMs operate over tabular representations that combine structured attributes with textual (unstructured) fields, and produce structured outputs. This reflects how LLMs are increasingly integrated into data systems as execution-time components rather than standalone interfaces. Consequently, EcoLLM targets the following systems-level question: *how do model choice, workload structure, and data scale jointly influence energy consumption and system efficiency?*

Design Goals and Novelty. The design of EcoLLM is guided by four principles: (i) *workload realism*, where tasks reflect data-management operations; (ii) *structured complexity modeling*, where difficulty is tied to operator semantics; (iii) *energy-aware evaluation*, where energy and CO_2 are treated as first-class metrics; and (iv) *reproducibility*, achieved through deterministic scoring and strict output contracts.

[1] https://github.com/pratyushagnihotri/EcoLLM

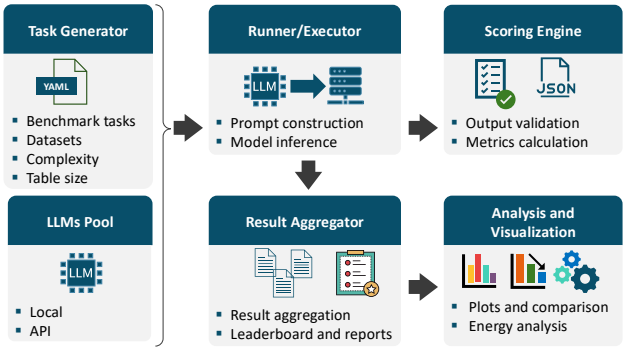


Figure 1: Overview of the EcoLLM benchmark architecture. The system comprises task generation, model execution, deterministic scoring, and aggregation, enabling reproducible evaluation across models, workload families, complexity levels, and data scales, with energy and CO_2 emissions treated as first-class metrics.

Existing LLMs benchmarks [22, 33] abstract away data-management characteristics such as operator composition and table structure. Similarly, prior work on learned components in data systems focuses primarily on performance cost such as latency or throughput [3, 37], without explicitly modeling energy. EcoLLM bridges this gap by combining workload-aware task design with energy-aware evaluation.

In this paper, we focus on *energy consumption* as the primary dimension of analysis, while using accuracy and latency to interpret trade-offs.

2.1 EcoLLM Architecture

EcoLLM follows a modular pipeline consisting of *task generation, model execution, deterministic scoring, and result aggregation* as shown in Figure 1. The architecture can be interpreted as a data-processing pipeline where each task corresponds to a logical operator specification, and the LLM participates either in execution or in pipeline construction, depending on the workload family.

For workload families focused on data transformation (Families 1–3), the LLM acts as a physical operator implementation, directly processing tabular inputs and producing structured outputs. In this setting, the LLM behaves similarly to a user-defined function (UDF) embedded within a query plan, executing semantic operations such as filtering, classification, or join-like reasoning over textual attributes.

In contrast, for pipeline/code generation tasks (Family 4), the LLM operates at a higher level of abstraction. Instead of executing a data transformation, it functions as a pipeline synthesizer, generating an executable program (e.g., a SQL query) that defines a sequence of physical operators. The generated program is then validated and, if required, executed by the evaluation framework. Thus, the LLM contributes to the construction of the execution plan rather than its direct execution.

This distinction highlights a dual role of LLMs in EcoLLM: they act both as *execution-time operators* and as *planning-time* components. This mirrors classical database system architectures, where

query planning and execution are separated, and emphasizes that LLMs blur this boundary by supporting both phases within a unified framework.

Running Example. To illustrate the workflow, consider a task from Family 4 (pipeline/code generation), where the goal is to generate a SQL query computing the top-k states by average energy consumption. The task generator produces a structured input table and a task specification. The runner converts this into a prompt, and the LLM generates a SQL query representing a composed operator pipeline (e.g., aggregation, sorting, and limit). Unlike Families 1–3, where the LLM directly produces the final result, here it produces a program that encodes the computation.

The remaining components of the architecture operate uniformly across all workload families. The task generator defines structured inputs and expected outputs, the runner ensures consistent prompt construction, the scoring engine validates outputs against strict schemas (including execution-based validation for generated programs), and the result aggregator collects metrics across models, workloads, and scaling dimensions. This unified pipeline enables controlled comparison while supporting both execution-oriented and generation-oriented uses of LLMs. In the following let us look into the subset of the dataset for family 4 and the role of each component of EcoLLM.

```
State   Year   Report_Text                      Consumption
TX      2020   "Demand surged due to..."        1200
CA      2020   "Industrial growth usage..."     950
NY      2020   "Stable demand with..."          870
TX      2019   "High summer demand..."          1100
CA      2019   "Moderate increase in..."        920
NY      2019   "Slight decrease..."             860
```

The task is: *"Generate a SQL query to compute the top 2 states by average consumption across all years."*

Task Generator. The generator produces structured tasks in YAML format, parameterized by dataset, workload family, complexity level, and table size. Each task includes: (i) tabular input, (ii) task specification, and (iii) expected output schema. This design ensures consistency across models and enables controlled scaling. In contrast to simple query-answering tasks, *Family 4* tasks require synthesizing executable programs, making them representative of pipeline construction in data systems.

Runner/Executor and Prompt Construction. The runner transforms each task into a structured prompt and executes it on selected models. Unlike generic NLP prompts, EcoLLM uses a template that explicitly encodes data and computation, e.g.;

```
You are given the following table:

State   Year   Report_Text                      Consumption
TX      2020   "Demand surged due to..."        1200
CA      2020   "Industrial growth usage..."     950
NY      2020   "Stable demand with..."          870
TX      2019   "High summer demand..."          1100
CA      2019   "Moderate increase in..."        920
NY      2019   "Slight decrease..."             860

Task: Generate a SQL query to compute the top 2 states
by average consumption across all years.

Output format:
{"query": "<SQL query>"}
```

Each prompt consists of (i) input data, (ii) task specification, and (iii) a strict output contract. This design is critical: without explicit schemas, generated code becomes ambiguous and difficult to validate. By enforcing structured outputs, EcoLLM ensures that LLMs behave as predictable operators rather than free-form generators.

Model Execution. The selected LLM processes the prompt and produces a structured output, e.g.:

```
{"query": "SELECT state, AVG(consumption) AS avg_cons
           FROM energy
           GROUP BY state
           ORDER BY avg_cons DESC
           LIMIT 2;"}
```

This output represents a composed operator pipeline (aggregation + sorting + limit), reflecting how LLMs can synthesize multi-step execution logic.

Scoring Engine. Outputs are validated against strict JSON contracts and evaluated for correctness. For code-generation tasks, scoring involves syntactic validation and semantic checks, e.g., executing the query and comparing results with ground truth. This ensures deterministic and reproducible evaluation.

Result Aggregation and Analysis. Results are aggregated across models, workloads, and scaling dimensions, producing trade-off analyses over latency and energy. This modular design decouples workload definition, model execution, and evaluation, enabling reproducible and extensible benchmarking. More importantly, it allows EcoLLM to treat LLMs within a unified operator-centric abstraction: as execution-time operators for data transformation tasks and as pipeline generators for program synthesis tasks. This makes LLM-based processing comparable to classical query execution while extending it with semantic reasoning capabilities unique to LLMs. The corresponding workload families that capture these roles, along with the model selection strategy, are described in the following sections.

2.2 Workload Families and Complexity Design

In this section, we present the workload families and the corresponding complexity design.

2.2.1 Workload Families. A key design decision in EcoLLM is to define task difficulty in terms of *data-management semantics* rather than linguistic complexity. Unlike NLP benchmarks [35, 58, 62, 65] that vary prompt length or reasoning steps, EcoLLM ties difficulty to *operator structure*, similar to how query plans are defined in database systems. EcoLLM defines four workload families representing core operator patterns.

Family 1: Tabular reasoning (relational queries over unstructured data like *text*). This setting is particularly relevant as unstructured data is becoming a central focus in database research, and LLMs are well-suited for interpreting and reasoning over such textual information. This workload includes filtering, aggregation,

and ranking operations. Unlike classical SQL, filtering depend on interpreting textual attributes which can be performed by an LLM. *Example:* find states with highest consumption where reports indicate increased demand. Here, the filtering condition depends on interpreting text rather than only numeric values.

Family 2: Data-quality reasoning (anomaly detection / validation). This setting reflects common data-quality and validation tasks in data pipelines, where identifying anomalies or inconsistencies is essential, and LLMs can capture patterns that are difficult to express using fixed rules. This includes detection and classification of anomalies or inconsistencies in structured data, commonly used in ETL and monitoring pipelines. Unlike Family 1, the output is a categorical label (e.g., anomaly type, valid/invalid, spike/drift) rather than a deterministic value. This family models *prediction tasks*, where the system assigns a label to each input instance based on learned patterns, and correctness is evaluated against ground-truth labels rather than exact numeric outputs. Here the *logical operator* for LLM is feature extraction or classification (ML/UDF operator). For example the task can be: `classify_anomaly` each state-year pair into anomaly_type $\in$ {normal, spike, drift} based on historical patterns. Here, `classify_anomaly` represents a learned operator (e.g., ML model or LLM) that assigns labels based on derived features.

Family 3: Multi-step analytics (join-based queries). This setting captures increasingly complex analytical workloads that combine multiple data sources, where semantic alignment across structured and unstructured attributes is required, a task where LLMs can provide additional flexibility. This includes analytical tasks that combine information across multiple relations or conditions, often requiring semantic alignment based on textual descriptions. For example, the task can be to compare consumption with external event reports linked via text. Here, the join condition involves semantic filtering over text rather than purely relational keys.

Family 4: Pipeline code generation (query/program synthesis). This setting reflects emerging data-system scenarios where LLMs are used to automate query and pipeline generation, enabling users to express complex analytical tasks without manually specifying execution logic. This includes generation of executable SQL or data-processing pipelines that combine multiple operators. Such queries are typically generated by LLMs in data-assistant or pipeline automation systems.

2.2.2 Complexity Design. The workload families capture *operator type*, but not execution difficulty. For example, a simple aggregation and a multi-step join both belong to tabular reasoning but differ significantly in execution complexity. Therefore, EcoLLM introduces a four-level complexity (C1–C4), capturing increasing operator depth and composition. This distinction is necessary because prompt length does not reflect computational effort: two tasks with similar input size may require fundamentally different execution patterns (e.g., aggregation vs. join). By grounding complexity in operator semantics, EcoLLM isolates the true drivers of computation and energy consumption.

Running Example for Complexity Levels. We illustrate the complexity level using a single tabular reasoning task over the SEDS dataset, showing how operator structure evolves. Here, we describe the main task, what is executed by LLM and execution complexity.

C1: Single-step operation. The task requires a simple aggregation. The model is asked to compute a single aggregate value from the table, conditioned on a simple textual filter. The task requires minimal reasoning, involving straightforward filtering and aggregation, resulting in a short and deterministic output.

C2: Multi-operator query. The task combines aggregation and ranking. The model groups data, compute aggregates, and rank results under a textual condition. The task involves multiple reasoning steps, including filtering, grouping, aggregation, and sorting, producing a structured multi-row output.

C3: Conditional reasoning across subsets. The task compares values across time. The model must compare values against a derived statistic computed over a semantically filtered subset. The task requires intermediate reasoning, where the model first derives a reference value and then applies it to filter results, increasing reasoning depth and dependency between steps.

C4: Composed analytical pipeline. The task requires stateful, multi-step computation. The model is required to construct a full analytical pipeline combining filtering, aggregation, and ranking under semantic conditions. The task involves composing multiple operators into a coherent pipeline, requiring sustained reasoning and producing longer, structured outputs.

Across all levels, the system processes tasks using the same pipeline: the task generator constructs structured inputs, the runner converts them into prompts, the LLM executes the task, and the scoring engine validates structured outputs. The complexity level therefore isolates how *operator composition within the same execution flow* affects energy consumption and system behavior.

Conceptually, workload families capture *what type of computation* is performed, while complexity levels capture *how that computation is structured*. This distinction enables isolating whether observed system behavior is driven by task type or by execution complexity. Alternative approaches based on prompt length or token count conflate representation size with computation. By grounding complexity in operator semantics (e.g., joins, aggregations, window operations), EcoLLM enables analysis aligned with real query-plan structure in data systems. In the next section, we describe the model selection strategy for EcoLLM.

2.3 Model Selection Strategy

The model set reflects realistic deployment choices in data systems. Instead of evaluating a single model or family, EcoLLM includes both local and API-based models to capture two execution regimes: *local inference*, where energy is tied to hardware utilization, and *remote inference*, where energy must be attributed indirectly.

Models are selected to cover (i) diverse architectures (e.g., Qwen, Llama, Mistral, Gemma), (ii) varying parameter sizes, and (iii) specialized capabilities (e.g., code generation). This enables analysis of *how model characteristics influence energy consumption and whether smaller models can achieve comparable effectiveness at lower cost.* In the following we present how these models are evaluated over the workload families and what are the trade-offs in efficiency and energy.

Category	Details
Models	Offline: Qwen2.5-3B, Qwen2.5-7B, Mistral-7B, Llama3, Mistral, Gemma, DeepSeek-Coder, CodeLlama, Llama2, Qwen Online: GPT-4.1, GPT-4.1-mini, o3-mini
Workloads	F1: tabular reasoning; F2: anomaly reasoning; F3: multi-step analytics; F4: code generation
Complexity	C1: single-step; C2: top-k/aggregation; C3: joins/deltas; C4: window/ranking
Scaling	Table size: 20–1000 rows
Dataset	SEDS (U.S. energy statistics)
Hardware	CloudLab m510, Ubuntu 22.04, Intel Xeon D, 48 GB RAM, 256 GB disk, 2.0 GHz
Metrics	Quality: pass rate; Performance: avg latency; Efficiency: energy (kWh)
Energy Method	Offline: runtime + CPU utilization; Online: provider-side attribution

Table 1: Experimental setup of EcoLLM.

3 Benchmark Evaluation

In this section, we present a benchmark evaluation of EcoLLM to analyze how energy consumption varies across different LLMs, workload structures, and data scales. Rather than focusing only on accuracy or latency, our goal is to expose the *energy-aware trade-offs* that arise when LLMs are used as operators in data-management pipelines. We first describe the experimental setup and evaluation questions, and then report results on the effects of task effectiveness and execution latency.

3.1 Experimental Setup and Questions

We evaluate EcoLLM to understand how energy consumption behaves across different models, workloads, and data scales in LLM-based data-management tasks. Our setup is designed to reflect realistic deployment scenarios while enabling controlled analysis of key system factors, including model size, operator complexity, and input data scale. Table 1 summarizes the experimental setup.

Models. We include both locally deployed (offline) and API-based (online) LLMs spanning multiple families (Qwen, Llama, Mistral, Gemma), parameter scales (3B–7B+), and specialized variants for code generation. This diversity enables analysis of how model architecture and size influence energy consumption and system behavior.

Workloads and Scaling. We generate benchmark tasks across four workload families (Family 1– Family 4) and four complexity levels (C1–C4), capturing common database operations such as filtering, aggregation, joins, anomaly reasoning, and code generation. Input table sizes vary from 20 to 1000 rows to study how energy scales with data size under fixed task semantics. By varying table size while keeping task semantics fixed, we isolate whether energy scales primarily with input size or is instead dominated by model and workload complexity.

Dataset Selection. We use the SEDS dataset[2], which provides structured, multi-dimensional energy data supporting aggregation, ranking, temporal analysis, and anomaly detection. The dataset choice is critical: while synthetic or TPC-style benchmarks offer controlled environments, they do not capture the semantic reasoning required in LLM-based workflows. SEDS enables multiple workload families within a single dataset, allowing consistent evaluation across task types while preserving realistic structure. Although EcoLLM is dataset-agnostic, we focus on SEDS to establish a controlled baseline; extending to additional domains such as electricity market data (ENTSO-E [18]), building energy consumption datasets, financial or sensor data is future work.

Hardware and Execution. Offline experiments are conducted on a CloudLab cluster [17] design using *r*7525 nodes running Ubuntu 22.04, equipped with dual AMD EPYC 7542 processors, 512 GB RAM, 2 TB disk, and 2.9 GHz clock speed. Additionally, each node is provisioned with two NVIDIA Tesla V100S (32 GB) GPUs.

Metrics and Evaluation Dimensions. EcoLLM evaluates models across three dimensions: quality [11, 25], performance [16, 48], and efficiency dimensions [6, 46], forming a multi-objective space analogous to classical query optimization, extended with an energy dimension. Although EcoLLM supports additional metrics such as CO_2 emissions, cost, and numeric error, this initial evaluation focuses on *latency*, *accuracy* and *energy consumption*.

Performance is measured using average latency and accuracy, and efficiency is captured using energy consumption (kWh), which quantifies computational effort. Task accuracy is measured using a workload-specific metric *pass rate*, defined as the fraction of tasks for which the model produces a fully correct, schema-compliant output. Pass rate lies in $[0, 1]$ and captures end-to-end correctness of structured computation, where partial outputs are treated as failures. We use pass rate as the primary accuracy metric, as it captures end-to-end correctness under strict execution semantics. In data-processing pipelines, outputs must be syntactically valid and semantically executable; even minor deviations lead to failure. Therefore, a binary metric is appropriate to reflect whether an LLM can reliably function as an operator within a system.

Energy Measurement. Energy measurement [60] depends on the execution environment and poses a fundamental challenge in LLM-based systems. For locally executed models, energy can be estimated from resource utilization, whereas for API-based models, computation occurs on remote infrastructure and cannot be directly observed. As a result, a single measurement approach does not suffice for both settings.

In case of *local (offline) models*, hardware-level approaches (e.g., RAPL [32]) directly measure power consumption and provide high accuracy, but require controlled environments and specialized access. In contrast, software-based approaches such as CodeCarbon [15] estimate energy consumption by monitoring resource utilization (CPU, GPU, memory) and combining usage time with hardware-specific power models. These approaches approximate energy as:

$$E \approx \sum_{r \in \{\text{CPU, GPU}\}} P_r \cdot t_r,$$

where P_r denotes the estimated power draw of resource r and t_r its utilization time. While less precise than hardware measurements,

[2] https://www.eia.gov/state/seds/CDF/Complete_SEDS.csv

they are practical and reproducible for local experiments, though sensitive to hardware configuration and utilization assumptions.

For *API-based (online) models*, direct measurement is not feasible because inference is executed on remote infrastructure. Client-side measurements capture only idle or network-related energy and therefore significantly underestimate actual consumption. To address this, provider-side attribution approaches such as EcoLogits [50] estimate energy per request by mapping input/output token counts and model characteristics to pre-calibrated efficiency models of large-scale inference systems.

EcoLLM adopts a hybrid approach that aligns energy estimation with the execution mode. For locally executed models, we use resource-based estimation derived from runtime and CPU utilization as detailed above. For API-based models, we use EcoLogits-style provider-side attribution to estimate energy per request. This design is essential for fair comparison as using local estimation for API models would underestimate energy, while using provider-side attribution for local models would be inapplicable. By matching the measurement method to the execution setting, EcoLLM enables consistent and comparable energy analysis across heterogeneous deployments, which is critical for studying energy-aware trade-offs in LLM-based data systems.

Our initial evaluation focuses on understanding energy-aware performance trade-offs in LLM-based data-processing tasks. In particular, we analyze how *energy consumption* relates to execution latency and accuracy and how this relationship varies across deployment modes and workload complexity.

EQ1 (Energy vs Performance Trade-offs). How does energy consumption relate to execution latency across models and workload families? What trade-offs emerge in the Pareto frontier between low-latency and energy-efficient execution?

EQ2 (Impact of Deployment Mode and Complexity). How does the trade-offs differ between locally deployed (offline) and API-based (online) models under increasing workload complexity for complex pipeline tasks (Family 4) at larger input sizes?

3.2 EQ1: Energy vs Latency Trade-offs

We first focus on the energy–latency trade-off, as latency is the primary performance metric optimized by database systems, while energy represents an emerging but critical efficiency objective that most database benchmarks neglect. Analyzing their interaction is essential for understanding the trade-offs faced by modern data systems integrating LLMs. Therefore, we analyze this across workload families under the highest complexity setting (C4). We use Pareto plots, a standard tool in multi-objective optimization, to characterize the frontier of optimal trade-offs between latency and energy and to identify non-dominated models as shown in Figure 2. Here, each point represents a model with metrics averaged across multiple task instances and input table sizes.

Across all workload families (*Family 1*– *Family 4*), the models do not form a smooth trade-off curve. Instead, they split into two clear groups: one with low latency but high energy, and another with higher latency but lower energy. Lightweight models such as `o3-mini` consistently appear in the lower-left region, achieving both low latency and low energy, indicating minimal computational effort. In contrast, API-based models such as GPT-4.1 and GPT-4.1-mini occupy the low-latency but high-energy region, e.g., ∼5–7k ms latency in *Family 1: tabular reasoning* and *Family 3: multi-step analysis* at 10^{-4}–10^{-3} kWh (cf. Figures 2a and 2c), while several local models cluster in the high-latency but low-energy region (e.g., 10k–20k ms at 10^{-5}–10^{-4} kWh). This creates a characteristic L-shaped Pareto structure, where improvements in latency are achieved at a disproportionate increase in energy cost.

Although the absolute energy per task is small, the relative differences across models are significant and amplify at scale. More importantly, these differences lead to fundamentally different trade-offs between latency and effectiveness, showing that energy must be considered as a first-class objective in LLM-based data systems. For example, a per-request energy cost of 0.001 kWh (GPT-4.1 across all workloads) corresponds to 1 MWh for one million requests per day, which scales to 365 MWh annually. Given that models can differ by an order of magnitude in energy consumption, this translates into hundreds of megawatt-hours difference per year, making energy a critical factor in large-scale deployments.

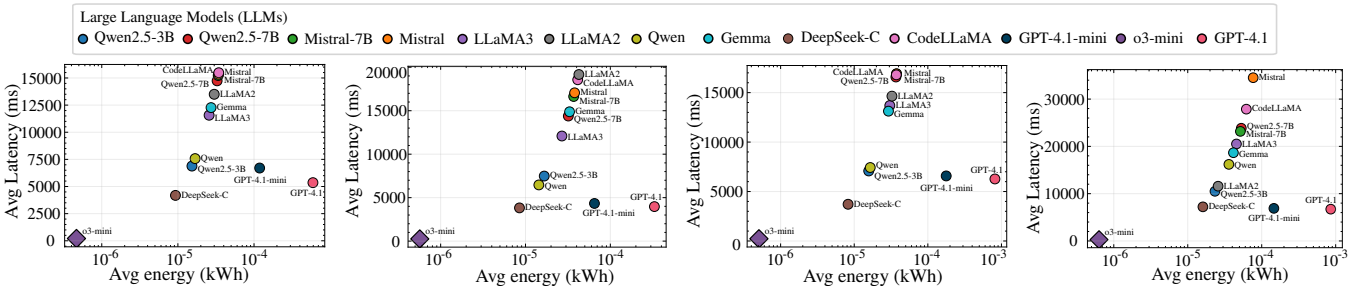
The Pareto frontier is therefore not defined by a single dominant model, but by a set of trade-offs between execution regimes. Models that minimize latency are systematically dominated in energy by smaller local models, while energy-efficient models incur significantly higher latency. This is particularly visible in *Family 4: code generation* (cf. Figure 2d) , where latency spreads widely (up to ∼35k ms) without a corresponding change in energy ordering, and in *Family 2: anomaly reasoning* (cf. Figure 2b), where even moderate latency reductions require noticeable increases in energy. This indicates a fundamental trade-off: improving latency comes at the cost of higher energy, while improving energy efficiency leads to higher latency.

A key observation from the plots is that energy is not proportional to latency. For example, GPT-4.1 achieves lower latency than most local models while consuming substantially more energy, indicating that energy is dominated by backend execution characteristics (e.g., parallel hardware utilization, model scale) rather than wall-clock time. Conversely, models with similar latency (e.g., several mid-sized local models in workload *Family 1* and *Family 3*) can differ noticeably in energy, suggesting that architectural differences and implementation efficiency also influence the trade-off surface.

Overall, the most significant factor shaping the Pareto trade-off is the deployment and inference regime rather than workload complexity or runtime alone. Latency is optimized through aggressive parallelism and large-scale infrastructure, whereas energy efficiency emerges from smaller models and constrained execution environments. This results in a persistent and predictable trade-off surface across all workloads, where optimizing for latency inherently shifts execution toward higher energy regions, and optimizing for energy shifts execution toward higher latency.

3.3 EQ2: Impact of Deployment Mode and Complexity

To further understand the interplay between performance, effectiveness, and efficiency, we provide a deeper analysis of accuracy, latency, and energy by focusing on pipeline/code generation tasks



(a) Family1: Energy vs latency **(b) Family2: Energy vs latency** **(c) Family3: Energy vs latency** **(d) Family4: Energy vs latency**

Figure 2: Energy and average latency trade-offs for all workload families under the highest complexity level (C4). Each point corresponds to one model. Models located closer to the lower-left corner are more desirable, as they combine lower environmental cost with lower latency. Across all families, the plots reveal that low-latency models are not necessarily energy-efficient, indicating a fundamental tension between performance-oriented and energy-aware execution.

(Family 4), which represent the most complex setting and require LLMs to synthesize executable multi-step pipelines. We report offline and API-based models separately in this analysis to highlight their distinct behavior under increasing complexity. While Figure 2 presents a joint view of all models to illustrate global trade-offs, separating them here helps explain how these trade-offs arise from fundamentally different execution settings.

Figure 3a-3d show how latency and energy change as task complexity increases (from C1 to C4). For offline models (Figure 3a), latency increases steadily with complexity, which is expected since the tasks require more steps and more reasoning. In contrast, API-based models (Figure 3b) keep latency relatively low and stable across all complexity levels. However, this does not mean they are more efficient. As shown in Figure 3c and 3d, API-based models consistently use much more energy, while offline models increase energy only moderately. This shows that faster execution in API-based models comes from using more resources, not from doing the work more efficiently.

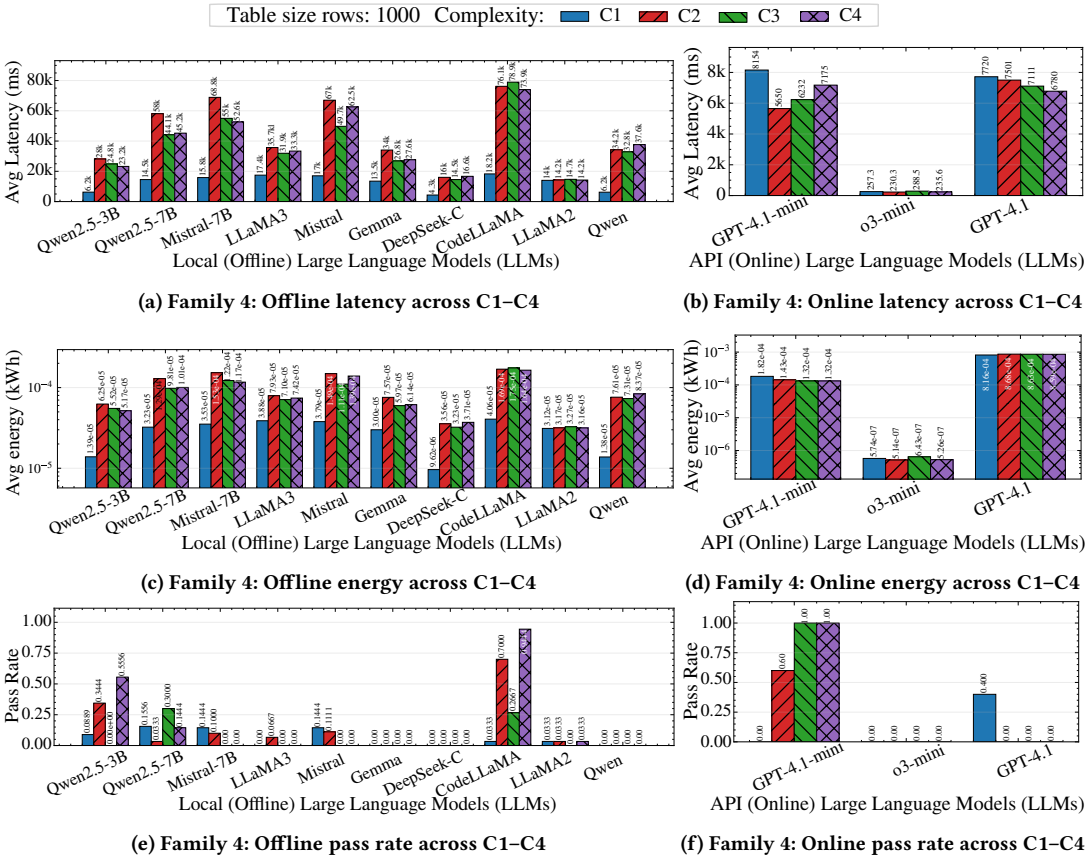
When we look at effectiveness, a different pattern appears. In Figure 3e, improvements with increasing complexity are primarily driven by specific models rather than a general trend. Models such as Qwen and CodeLLaMA achieve the highest pass rates across complexity levels. While CodeLLaMA is specifically designed for code generation, Qwen, as a general-purpose model with strong structured generation capabilities, also performs well on these tasks. In contrast, several other models, including Gemma and Mistral, produce almost no correct outputs regardless of complexity. Interestingly, DeepSeek-Coder, despite being designed for coding tasks, also performs poorly in this setting. This suggests that success on pipeline/code generation tasks depends not only on general coding ability, but also on the model's ability to generate structured, executable queries under strict output constraints. For API-based models (Figure 3f), pass rates do not consistently improve with increasing complexity and vary significantly across models. While GPT-4.1 maintains high effectiveness, models such as GPT-4.1- and o3-mini exhibit low or zero pass rates, showing that performance is not uniformly strong but highly dependent on model capability.

An important takeaway is that energy and accuracy are not aligned. Some models that use little energy perform very poorly. For example, Gemma, Mistral, and DeepSeek-Coder have low energy usage but near-zero pass rates. The most energy-efficient model, o3-mini, also fails completely on this task. This shows that optimizing only for energy can lead to models that are fast and cheap but do not produce useful results. Overall, the results highlight three key points. First, lower latency often comes with higher energy cost. Second, higher energy does not guarantee better results. Third, model choice matters: only some models benefit from increased computation, while others fail regardless of complexity. In the end, the main difference comes from how the models are deployed. Offline models trade speed for lower energy and can improve with more complex tasks (if they are capable enough). API-based models stay fast and consistent but use much more energy. This shows that choosing the right model depends on what matters most: speed, energy, or correctness.

4 Related Work

LLMs in Data Management. Recent systems increasingly integrate LLMs into data-processing and query-execution pipelines as semantic operators for extraction, transformation, filtering, and reasoning. Examples include Palimpzest [34], LOTUS [45], DocETL [56], ThalamusDB [31], Caesura [63] and Samsara [54]. These works show that LLMs can act as execution-time components in data systems and explore trade-offs such as accuracy, latency, and monetary cost. However, they do not systematically study how energy consumption and CO_2 emissions vary across workload structure, operator complexity, and data scale.

Performance Modeling and Benchmarking. A large body of work in databases and stream processing focuses on learned cost models [1, 3, 21, 23, 37], performance prediction [24, 38, 61], and system benchmarking [2, 7, 13, 40, 47, 69], with the primary goal of understanding or optimizing runtime behavior. These learned performance models and systems benchmarks for classical database emphasize on correctness, latency, and throughput. Outside data management, popular LLM benchmarks [14, 53, 66] focus on reasoning accuracy and capability evaluation. While these approaches are valuable for performance or quality assessment, they largely



(a) Family 4: Offline latency across C1–C4

(b) Family 4: Online latency across C1–C4

(c) Family 4: Offline energy across C1–C4

(d) Family 4: Online energy across C1–C4

(e) Family 4: Offline pass rate across C1–C4

(f) Family 4: Online pass rate across C1–C4

Figure 3: Impact of deployment mode and workload complexity on latency, energy, and task effectiveness for Family 4 (pipeline/code generation) at larger input sizes. The top row shows latency, the middle row shows energy consumption, and the bottom row shows pass rate across complexity levels (C1–C4) for offline and API-based models. Offline models exhibit steadily increasing latency and moderate energy growth with complexity, accompanied by improvements in pass rate, indicating that additional computation contributes to more effective execution. In contrast, API-based models maintain relatively low latency and stable effectiveness across complexity levels, but incur substantially higher energy consumption. Overall, the plots highlight a clear divergence between deployment modes: local execution trades latency for energy efficiency while scaling effectiveness with complexity, whereas remote execution preserves speed and effectiveness at a significantly higher energy cost.

abstract away workload properties that are central in data systems, such as operator composition, table scale, and structured output correctness, and they rarely treat energy as a first-class evaluation objective.

Energy Measurement for ML and LLM Systems. Energy-aware computing has a long history in systems research, including benchmarks such as JouleSort [51]. For ML workloads, tools [4, 8, 15, 43, 50, 68] provide either software-based estimation [15] of local energy use, or supports provider-side attribution [50] for API-based LLM inference. These tools provide important measurement primitives for estimating energy and carbon footprint, but they do not define workload-centric evaluation frameworks for LLMs used inside data-processing pipelines.

Gap and Positioning. Taken together, prior work [5, 11, 12, 14, 22, 22, 29, 33, 36, 39, 42, 44, 46, 49, 52, 55] addresses only parts of the problem. Existing LLM-based data systems demonstrate how LLMs can be embedded into query pipelines, but do not evaluate them from an energy-aware systems perspective. Performance modeling and benchmarking work studies runtime and quality, but not the interaction between workload structure and environmental cost. Energy-measurement tools provide attribution mechanisms, but not workload-aware benchmarks. EcoLLM bridges these lines of work by combining workload-centric task design, controlled complexity and scale, deterministic scoring, and hybrid energy attribution to study energy–accuracy–performance trade-offs in LLM-based data-management workloads.

5 Conclusion

We presented EcoLLM, a benchmark for analyzing energy consumption in LLM-driven data pipelines. Our evaluation shows that higher energy does not consistently improve task effectiveness, with only a subset of models achieving high accuracy at widely different energy

costs. We further observe a consistent energy–latency trade-off, where low-latency API-based models incur substantially higher energy compared to more efficient local models. These findings challenge traditional data-centric assumptions, highlighting that energy behavior is primarily shaped by model characteristics and inference regimes rather than data scale alone. As a future work, we plan to extend the empirical analysis to broader workloads and datasets, and to develop energy-aware cost models and adaptive strategies for efficient LLM-based data systems.

Acknowledgement

This work was supported by the LOEWE Spitzenprofessur programme (III 5-519/05.00.003-(0005)), the etaGPT project (03EN4107), the Athene Young Investigator Programme, and the DFG under Germany's Excellence Strategy (EXC-3057/1 "Reasonable Artificial Intelligence", Project No. 533677015). We also thank DFKI and hessian.AI.

References

[1] Pratyush Agnihotri, Carsten Binnig, and Manisha Luthra. 2025. Learning What Matters: Automated Feature Selection for Learned Performance Models in Parallel Stream Processing. In *Proceedings of the VLDB 2025 Workshop: Applied AI for Database Systems and Applications (AIDB)*.

[2] Pratyush Agnihotri, Boris Koldehofe, Roman Heinrich, Carsten Binnig, and Manisha Luthra. 2025. PDSP-Bench: A Benchmarking System for Parallel and Distributed Stream Processing. In *Performance Evaluation and Benchmarking: Traditional to Big Data to Internet of Things*, Raghunath Nambiar and Meikel Poess (Eds.). Springer International Publishing, 1–23. https://arxiv.org/abs/2504.10704

[3] Pratyush Agnihotri, Boris Koldehofe, Paul Stiegele, Roman Heinrich, Carsten Binnig, and Manisha Luthra. 2024. ZeroTune: Learned Zero-Shot Cost Models for Parallelism Tuning in Stream Processing.. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. 2040–2053.

[4] Lasse F. Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. 2020. Carbontracker: Tracking and Predicting the Carbon Footprint of Training Deep Learning Models. ICML Workshop on Challenges in Deploying and monitoring Machine Learning Systems. arXiv:2007.03051.

[5] Humza Ashraf, Syed Muhammad Danish, Aris Leivadeas, Yazan Otoum, and Zeeshan Sattar. 2025. Energy-Aware Code Generation with LLMs: Benchmarking Small vs. Large Language Models for Sustainable AI Programming. *arXiv preprint arXiv:2508.08332* (2025).

[6] Marc Baeuerle, Thomas Bodner, Martin Boissier, Tilmann Rabl, Ricardo Salazar Díaz, Florian Schmeller, Nils Strassenburg, Ilin Tolovski, Marcel Weisgut, and Wang Yue. 2025. TCO2: Analyzing the Carbon Footprint of Database Server Replacements. *Proceedings of the VLDB Endowment* 18, 12 (2025), 5223–5226.

[7] Peter Boncz, Thomas Neumann, and Orri Erling. 2013. TPC-H analyzed: Hidden messages and lessons learned from an influential benchmark. In *Technology Conference on Performance Evaluation and Benchmarking*. 61–76.

[8] S. A. Budennyy, V. D. Lazarev, N. N. Zakharenko, A. N. Korovin, O. A. Plosskaya, D. V. Dimitrov, V. S. Akhripkin, I. V. Pavlov, I. V. Oseledets, I. S. Barsola, I. V. Egorov, A. A. Kosterina, and L. E. Zhukov. 2023. eco2AI: Carbon Emissions Tracking of Machine Learning Models as the First Step Towards Sustainable AI. *Doklady Mathematics* (Jan. 2023). doi:10.1134/S1064562422060230

[9] California State University, Northridge. 2025. AI and Sustainability. https://libguides.csun.edu/ai/ai-and-sustainability. Accessed: 2026-03-27.

[10] Carbon Brief. 2025. AI: Five Charts That Put Data Centre Energy Use and Emissions into Context. https://www.carbonbrief.org/ai-five-charts-that-put-data-centre-energy-use-and-emissions-into-context/. Accessed: 2026-03-27.

[11] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology* 15, 3 (2024), 1–45.

[12] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374* (2021).

[13] Shimin Chen, Anastasia Ailamaki, Manos Athanassoulis, Phillip B Gibbons, Ryan Johnson, Ippokratis Pandis, and Radu Stoica. 2011. TPC-E vs. TPC-C: Characterizing the new TPC-E benchmark via an I/O comparison study. *ACM Sigmod Record* 39, 3 (2011), 5–10.

[14] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168* (2021).

[15] Benoit Courty, Victor Schmidt, Sasha Luccioni, Goyal-Kamal, MarionCoutarel, Boris Feld, Jérémy Lecourt, LiamConnell, Amine Saboni, Inimaz, supatomic, Mathilde Léval, Luis Blanche, Alexis Cruveiller, ouminasara, Franklin Zhao, Aditya Joshi, Alexis Bogroff, Hugues de Lavoreille, Niko Laskaris, Edoardo Abati, Douglas Blank, Ziyao Wang, Armin Catovic, Marc Alencon, Michał Stęchły, Christian Bauer, Lucas Otávio N. de Araújo, JPW, and MinervaBooks. 2024. *mlco2/codecarbon: v2.4.1*. doi:10.5281/zenodo.11171501

[16] Jeffrey Dean and Luiz André Barroso. 2013. The tail at scale. *Commun. ACM* 56, 2 (2013), 74–80.

[17] Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, Gary Wong, Jonathon Duerig, Eric Eide, Leigh Stoller, Mike Hibler, David Johnson, Kirk Webb, Aditya Akella, Kuangching Wang, Glenn Ricart, Larry Landweber, Chip Elliott, Michael Zink, Emmanuel Cecchet, Snigdhaswin Kar, and Prabodh Mishra. 2019. The Design and Operation of CloudLab. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*. USENIX Association, Renton, WA, 1–14.

[18] ENTSO-E. 2026. Power Statistics. https://www.entsoe.eu/data/power-stats/ Accessed: 2026-04-06.

[19] European Commission. 2025. Focus on Data Centres: Energy-Hungry Challenge. https://energy.ec.europa.eu/news/focus-data-centres-energy-hungry-challenge-2025-11-17_en. Accessed: 2026-03-27.

[20] Stavros Harizopoulos, Mehul Shah, Justin Meza, and Parthasarathy Ranganathan. 2009. Energy efficiency: The new holy grail of data management systems research. *arXiv preprint arXiv:0909.1784* (2009).

[21] Roman Heinrich, Carsten Binnig, Harald Kornmayer, and Manisha Luthra. 2024. Costream: Learned cost models for operator placement in edge-cloud environments. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 96–109.

[22] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300* (2020).

[23] Benjamin Hilprecht and Carsten Binnig. 2022. Zero-Shot Cost Models for out-of-the-Box Learned Cost Prediction. *Proc. VLDB Endow.* 15, 11 (2022), 2361–2374.

[24] Benjamin Hilprecht and Carsten Binnig. 2022. Zero-Shot Cost Models for Out-of-the-box Learned Cost Prediction. *arXiv preprint* (2022).

[25] Rob J Hyndman and Anne B Koehler. 2006. Another look at measures of forecast accuracy. *International journal of forecasting* 22, 4 (2006), 679–688.

[26] International Energy Agency (IEA). 2025. Artificial Intelligence and Energy. https://www.iea.org/topics/artificial-intelligence. Accessed: 2026-03-27.

[27] International Energy Agency (IEA). 2025. Energy Demand from AI. https://www.iea.org/reports/energy-and-ai/energy-demand-from-ai. Accessed: 2026-03-27.

[28] International Energy Agency (IEA). 2025. Overcoming Energy Constraints is Key to Delivering on Europe's Data Centre Goals. https://www.iea.org/commentaries/overcoming-energy-constraints-is-key-to-delivering-on-europe-s-data-centre-goals. Accessed: 2026-03-27.

[29] Dragoș Ionescu, Søren Kejser Jensen, and Bent Thomsen. 2025. EnergyBench: Holistic Benchmark for Correct and Energy-Efficient LLM-Generated Code. In *2025 3rd International Conference on Federated Learning Technologies and Applications (FLTA)*. IEEE, 571–579.

[30] Nidhal Jegham, Marwan Abdelatti, Chan Young Koh, Lassad Elmoubarki, and Abdeltawab Hendawi. 2025. How hungry is ai? benchmarking energy, water, and carbon footprint of llm inference. *arXiv preprint arXiv:2505.09598* (2025).

[31] Saehan Jo and Immanuel Trummer. 2024. Thalamusdb: Approximate query processing on multi-modal data. *Proceedings of the ACM on Management of Data* 2, 3 (2024), 1–26.

[32] Kashif Nizam Khan, Mikael Hirki, Tapio Niemi, Jukka K Nurminen, and Zhonghong Ou. 2018. Rapl in action: Experiences in using rapl for power measurements. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)* 3, 2 (2018), 1–26.

[33] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110* (2022).

[34] Chunwei Liu, Matthew Russo, Michael Cafarella, Lei Cao, Peter Baile Chen, Zui Chen, Michael Franklin, Tim Kraska, Samuel Madden, Rana Shahout, et al. 2025. Palimpzest: Optimizing ai-powered analytics with declarative query processing. In *Proceedings of the Conference on Innovative Database Research (CIDR)*. 2.

[35] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM computing surveys* 55, 9 (2023), 1–35.

[36] Sasha Luccioni, Yacine Jernite, and Emma Strubell. 2024. Power hungry processing: Watts driving the cost of AI deployment?. In *Proceedings of the 2024 ACM conference on fairness, accountability, and transparency*. 85–99.

[37] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Chi Zhang, Mohammad Alizadeh, Tim Kraska, Olga Papaemmanouil, and Nesime Tatbul. 2019. Neo: A learned

query optimizer. *arXiv preprint arXiv:1904.03711* (2019).
[38] Ryan Marcus and Olga Papaemmanouil. 2019. Plan-structured deep neural network models for query performance prediction. *arXiv preprint arXiv:1902.00132* (2019).
[39] Mohammadjavad Mehditabar, Saurabhsingh Rajput, Antonio Mastropaolo, and Tushar Sharma. 2025. Smart but Costly? Benchmarking LLMs on Functional Accuracy and Energy Efficiency. *arXiv preprint arXiv:2511.07698* (2025).
[40] Raghunath Othayoth Nambiar and Meikel Poess. 2006. The Making of TPC-DS.. In *Proceedings of the 32nd International Conference on Very Large Data Bases*. 1049–1058.
[41] Nature. 2025. AI is Set to Drive Surge in Data Centre Energy Demand. https://www.nature.com/articles/d41586-025-01113-z. Accessed: 2026-03-27.
[42] Chenxu Niu, Wei Zhang, Yongjian Zhao, and Yong Chen. 2025. Energy efficient or exhaustive? benchmarking power consumption of llm inference engines. *ACM SIGENERGY Energy Informatics Review* 5, 2 (2025), 56–62.
[43] Adel Noureddine. 2022. PowerJoular and JoularJX: Multi-Platform Software Power Monitoring Tools. In *18th International Conference on Intelligent Environments (IE2022)*. Biarritz, France.
[44] Ana Paula Oliveira, Tânia Carraquico, and Clara Martinez-Perez. 2026. Beyond Efficiency: A Systematic Review of Energy Consumption and Carbon Footprint Across the AI Lifecycle. *Sustainability* 18, 3 (2026), 1359.
[45] Liana Patel, Siddharth Jha, Melissa Pan, Harshit Gupta, Parth Asawa, Carlos Guestrin, and Matei Zaharia. 2024. Semantic operators: a declarative model for rich, ai-based data processing. *arXiv preprint arXiv:2407.11418* (2024).
[46] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350* (2021).
[47] Meikel Poess, Raghunath Nambiar, Karthik Kulkarni, Chinmayi Narasimhadevara, Tilmann Rabl, and Hans-Arno Jacobsen. 2018. Analysis of tpcx-iot: The first industry standard benchmark for iot gateway systems. In *Proceedings of the IEEE 34th International Conference on Data Engineering (ICDE)*. 1519–1530.
[48] David MW Powers. 2020. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061* (2020).
[49] K Pronk and Q Zhao. 2025. Benchmarking Energy Efficiency of Large Language Models Using vLLM. *arXiv preprint arXiv:2509.08867* (2025).
[50] Samuel Rincé and Adrien Banse. 2025. Ecologits: Evaluating the environmental impacts of generative AI. *Journal of Open Source Software* 10, 111 (2025), 7471.
[51] Suzanne Rivoire, Mehul A Shah, Parthasarathy Ranganathan, and Christos Kozyrakis. 2007. JouleSort: a balanced energy-efficiency benchmark. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. 365–376.
[52] Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. 2023. From words to watts: Benchmarking the energy costs of large language model inference. In *2023 IEEE high performance extreme computing conference (HPEC)*. IEEE, 1–9.
[53] Gabriele Sanmartino, Matthias Urban, Paolo Papotti, and Carsten Binnig. 2026. The Stretto Execution Engine for LLM-Augmented Data Systems. *arXiv preprint arXiv:2602.04430* (2026).
[54] Uélison Santos, Alessandro Ferri, Szilard Nistor, Riccardo Tommasini, Carsten Binnig, and Manisha Luthra. 2026. Towards Multimodal Stream Processing Systems. In *Proceedings 29th International Conference on Extending Database Technology, EDBT 2026, Tampere, Finland, March 24-27, 2026. International Conference on Extending Database Technology (EDBT)*, Wolfgang Lehner, Vanessa Braganholo, Kostas Stefanidis, Zheying Zhang, Alexander Krause, and Jo ao Felipe Nicolaci Pimentel (Eds.). OpenProceedings.org, 627–633.
[55] Omar Shaikh, Jon Saad-Falcon, Austin P Wright, Nilaksh Das, Scott Freitas, Omar Asensio, and Duen Horng Chau. 2021. EnergyVis: Interactively tracking and exploring energy consumption for ML models. In *Extended abstracts of the 2021 chi conference on human factors in computing systems*. 1–7.
[56] Shreya Shankar, Tristan Chambers, Tarak Shah, Aditya G Parameswaran, and Eugene Wu. 2024. Docetl: Agentic query rewriting and evaluation for complex document processing. *arXiv preprint arXiv:2410.12189* (2024).
[57] Society of Petroleum Engineers (SPE). 2025. Stanford AI Index 2025: Rapid AI Growth, Investment Surge, and Global Challenges. https://jpt.spe.org/twa/stanford-ai-index-2025-rapid-ai-growth-investment-surge-and-global-challenges. Accessed: 2026-03-27.
[58] Linxin Song, Jieyu Zhang, Lechao Cheng, Pengyuan Zhou, Tianyi Zhou, and Irene Li. 2023. Nlpbench: Evaluating large language models on solving nlp problems. *arXiv preprint arXiv:2309.15630* (2023).
[59] Stanford Institute for Human-Centered Artificial Intelligence (HAI). 2025. AI Index Report 2025. https://hai.stanford.edu/ai-index/2025-ai-index-report. Accessed: 2026-03-27.
[60] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th annual meeting of the association for computational linguistics*. 3645–3650.
[61] Ji Sun and Guoliang Li. 2019. An end-to-end learning-based cost estimator. *arXiv preprint arXiv:1906.02560* (2019).
[62] Qingyu Tan, Hwee Tou Ng, and Lidong Bing. 2023. Towards benchmarking and improving the temporal reasoning capability of large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 14820–14835.
[63] Matthias Urban and Carsten Binnig. 2023. CAESURA: language models as multi-modal query planners. *arXiv preprint arXiv:2308.03424* (2023).
[64] U.S. Department of Energy. 2024. DOE Releases New Report Evaluating Increase in Electricity Demand from Data Centers. https://www.energy.gov/articles/doe-releases-new-report-evaluating-increase-electricity-demand-data-centers. Accessed: 2026-03-27.
[65] Shubham Vatsal and Harsh Dubey. 2024. A survey of prompt engineering methods in large language models for different nlp tasks. *arXiv preprint arXiv:2407.12994* (2024).
[66] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems* 37 (2024), 95266–95290.
[67] World Economic Forum. 2025. Data Centres and Energy Demand. https://www.weforum.org/stories/2025/12/data-centres-and-energy-demand/. Accessed: 2026-03-27.
[68] Jie You, Jae-Won Chung, and Mosharaf Chowdhury. 2023. Zeus: Understanding and Optimizing GPU Energy Consumption of DNN Training. In *USENIX NSDI*.
[69] Wang Yue, Martin Boissier, and Tilmann Rabl. 2026. A Survey of Stream Processing System Benchmarks. In *Performance Evaluation and Benchmarking*, Raghunath Nambiar and Meikel Poess (Eds.). Springer Nature Switzerland, 24–43.