Databases Are Blind to Energy

TOBIAS RAHN*†, ELCA, Switzerland
ANDRES NAVARRO PEDREGAL*, ETH Zurich, Switzerland
MICHAL FRIEDMAN, ETH Zurich, Switzerland

Databases run in every data center, yet their energy cost is rarely measured. TPC-Energy exists but reports only whole-system power, not per-transaction breakdowns. Without that granularity, system designers cannot tell whether their engine choice or configuration wastes 2× or 90× more energy than the alternative.

We benchmark PostgreSQL, MariaDB, and SQLite on identical hardware under TPC-C and TPC-H, reporting energy per transaction (Wh/Txn). Engine architecture alone creates up to 11.6× energy gaps. Within a single engine, durability guarantees cost 2–90× more energy on direct-to-disk storage, and a missing index costs up to 9.7×, while across engines, optimizer and execution differences account for up to 6.4×. These differences are measurable today with existing infrastructure, but invisible in current benchmarks. Reporting Wh/Txn makes them actionable.

CCS Concepts: • **Information systems → Database management system engines**; • **Social and professional topics → Sustainability**.

Additional Key Words and Phrases: Databases, Energy Benchmarking, Sustainability, Green Computing

1 Introduction

Data centers consumed roughly 1.5% of global electricity in 2024 and are on track to double by 2030 [28]. Databases run behind nearly every application driving that demand [8, 17, 22], yet their energy consumption is rarely measured [42]. State-of-the-art benchmarks (TPC-C, TPC-H) record only throughput and latency, query optimizers ignore energy [33, 66], and cloud dashboards report server-level power without showing which query caused the spike [5, 49].

Prior work has profiled individual engines under a single benchmark [14, 35, 38, 61] or proposed energy-aware DBMS designs [23, 34, 62]. What is missing is a controlled cross-engine comparison that isolates how architecture and configuration choices affect energy under the same workload and hardware. Without such a comparison, system designers cannot know whether switching engines, adding an index, or relaxing durability saves energy, and they cannot justify the tradeoff.

To close this gap, we benchmark PostgreSQL [57], MariaDB [37], and SQLite [13] under TPC-C [59] (OLTP) and TPC-H [60] (OLAP) on identical hardware. These three engines cover two distinct architectures: client-server with group-commit WAL (PostgreSQL, MariaDB) and embedded with serialized commits (SQLite). We choose them because they are among the most widely deployed open-source engines and span these two architectures. By running them under controlled conditions, we can attribute energy differences to specific software mechanisms rather than hardware variation.

Following prior work [42, 48, 61], we normalize by useful work done and report energy per transaction (Wh/Txn).

We report two findings:

(1) **Database architecture determines energy cost, and the most efficient engine depends on the workload.** PostgreSQL uses 11.6× less energy than SQLite on commit-heavy writes, but SQLite uses ∼2× less on read-only queries. The reason is that group-commit WAL amortizes `fsync` across transactions, while serialized commits pay per transaction (§3.1).
(2) **Configuration and plan choices affect energy by orders of magnitude.** Within one engine, disabling synchronous commits saves up to 90× and adding an index up to 9.7×. Across engines on the same query, optimizer and execution differences account for up to 6.4×. In particular, durability has a quantifiable energy cost that ranges from 2× to 90× depending on the engine and the guarantee level (§3.2).

Once per-transaction energy is visible, it becomes an optimization target. In §4, we outline how Wh/Txn enables energy-aware query planning and carbon-aware workload scheduling.

2 Methodology

Our goal is to identify which database components (commit strategy, storage access, query planning) drive energy differences. All experiments run on dedicated machines under controlled conditions, so measured differences come from software, not hardware.

Hardware. We run the experiments on an Intel Xeon E3-1225 v6 (3.3 GHz, 4 cores), 32 GB DDR4-3200, 500 GB 7200 RPM HDD, running Ubuntu 24.04 LTS. We turn off Hyper-Threading and Turbo Boost. A 365 W Dell 80 Plus Gold PSU powers the system. We choose this hardware because Intel exposes RAPL registers for fine-grained energy accounting and because a single-socket desktop gives us full control over the measurement environment, enabling consistent calibration against a wall-power meter. We use an HDD rather than an SSD to amplify the `fsync` stalls that differentiate commit strategies [39]; on faster storage the same mechanisms apply, but the absolute gaps shrink (§4).

To check that our findings hold on server-class hardware and at larger scale, we repeat the key experiments on a Dell PowerEdge R630 (two Intel Xeon E5-2630 v3, with a total of 16 cores and 256 GB DDR4), pinning all benchmark processes to one socket and reading only that socket's RAPL counters, so the measurement reflects the workload itself rather than unrelated background activity. Its PERC H730 RAID controller with six 600 GB 15,000 RPM SAS HDDs (Seagate ST600MP0005) has a 1 GB write-back cache that acknowledges an `fsync` as soon as data reaches the controller, making the per-commit disk stall nearly free. Results are reported from the default setup unless stated otherwise.

*Equal contributions.
†Work done while at ETH Zurich.

Authors' Contact Information: Tobias Rahn, ELCA, Zurich, Switzerland; Andres Navarro Pedregal, anavarro@ethz.ch, ETH Zurich, Zurich, Switzerland; Michal Friedman, michal.friedman@inf.ethz.ch, ETH Zurich, Zurich, Switzerland.

Energy measurement. A Voltcraft SEM5000CH socket meter records whole-machine AC power once per minute and serves as our calibration reference.

We model whole-machine energy as four components: CPU, DRAM, disk, and a fixed baseline. For CPU and DRAM, we run seven RAPL-based tools: CarbonTracker [7], CodeCarbon [12], Experiment Impact Tracker [25], Intel VTune [27], Scaphandre [26], EnergAt [24], and Eco2AI [10]. All read Intel RAPL registers for per-socket CPU and DRAM energy. We validate them against each other and the socket meter; all show similar trends. We report CarbonTracker [7] throughout because it has the lowest variance.

RAPL does not capture PSU losses, peripherals, or disk. For disk, we use the standard idle-plus-active model [22, 61]: our HDD datasheet gives 3.7 W idle and 9.7 W peak, so $P_{\text{disk}} = 3.7 + 6u$ W, where $u \in [0, 1]$ is disk utilization from `dstat`. A fixed 5 W baseline, calculated as the mean difference between the socket meter and RAPL+disk during idle, covers the remaining gap (motherboard, fans, PSU losses). Total idle draw at the socket is ~20 W (~11 W CPU/DRAM via RAPL, 3.7 W disk, 5 W baseline). This disk model and idle breakdown are specific to the default setup. On the second setup, we model the disk consumption with 5.3 W idle and 8.7 W peak from the disk datasheet. We multiply the values by six since there are six disks. We omit the 5 W baseline on the second setup as we could not measure it against the socket meter.

Metric. Our primary metric is energy per transaction [17, 47, 48]:

$$\text{Wh/Txn} = \frac{\text{total energy}}{\text{transactions completed}} = \frac{\text{CPU + DRAM + disk}}{\text{transactions completed}}$$

Unlike total energy, which mixes static power with useful computation, Wh/Txn penalizes engines that draw power without producing proportional work. We exclude the fixed baseline so the metric reflects only software-driven components (CPU, DRAM, disk).

Databases. We benchmark three widely deployed open-source SQL engines that represent two distinct architectures. PostgreSQL v17.10 [57] and MariaDB v11.4.4 [37] are client-server engines that use MVCC and a write-ahead log (WAL) with group commit, amortizing disk syncs across concurrent transactions. SQLite v3.37.2 [13] is an embedded engine that serializes commits through a rollback journal, issuing one `fsync` per transaction. All run with default settings unless noted.

Benchmarks. We use BenchBase [15] to run TPC-C [59] (OLTP) and TPC-H [60] (OLAP) at small scale: one warehouse and one client terminal (~127 MB) for TPC-C and a Scaling Factor (SF) of 0.1 (~200 MB) for TPC-H. Our goal is to identify which software mechanisms drive energy differences, not to predict absolute consumption at production scale. At this scale the dataset fits in RAM, so measured differences reflect the commit path, query plan, and index strategy rather than disk-cache effects. To validate that the observed trends hold at large scales, we repeat representative experiments with larger deployments (up to 64 warehouses) for TPC-C and an increased SF (up to 1) for TPC-H.

We deliberately test at low concurrency, where group commit batches the fewest transactions per flush. Since group commit's advantage grows with concurrency [58], our energy gaps are lower bounds [14, 61]. We also run experiments with a higher thread count to test concurrency. TPC-C runs use the default transaction mix[1] in 3-minute windows. TPC-H runs each of the 22 queries once per repetition. We report the mean and standard deviation, as error bars, over three repetitions.

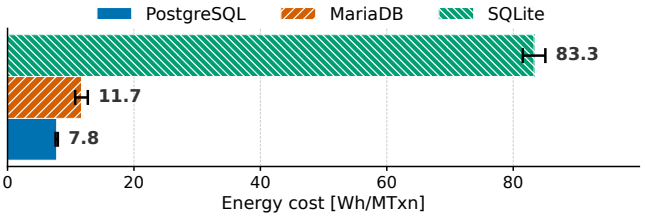

Fig. 1. Energy per million business transactions (Wh/MTxn) under the default TPC-C mix (New-Order 45%, Payment 43%, Order-Status / Delivery / Stock-Level 4% each) on the default setup. Lower is better.

3 Evaluation

We structure the evaluation around two questions. First, does database architecture affect energy, and if so, which engine is most efficient (§3.1)? Second, how much can configuration choices within a single engine affect energy consumption (§3.2)?

3.1 Architecture Effects on Energy

We compare the three engines on TPC-C to isolate how database architecture affects per-transaction energy. Under the default transaction mix (Figure 1), SQLite is 10.7× more expensive per transaction than PostgreSQL. Almost all of that gap comes from two write-heavy transactions, New-Order and Payment, which together are 88% of the mix.

To understand why, we run each transaction type in isolation (Figure 2). On New-Order (Figure 2a), SQLite uses 11.6× more energy per transaction than PostgreSQL, while on Payment (Figure 2b) it is ~6× more. The reason is the commit `fsync`. PostgreSQL and MariaDB use group commit [58], flushing many concurrent transactions with one disk sync, while SQLite serializes each commit [29, 54, 63] and pays the cost per transaction.

The pattern flips when commits are rare. On Order-Status (Figure 2c, read-only) and Delivery (Figure 2d, which batches ten orders into one commit), SQLite uses up to 3× less energy than MariaDB and ~2× less than PostgreSQL. Without `fsync` overhead, SQLite's embedded design becomes an advantage: no client-server IPC, no shared buffer pool, no background writer; all CPU cycles go to the query. On Stock-Level (Figure 2e), the three engines land within ~20% of each other; without commits to differentiate them, architectural differences become irrelevant.

MariaDB shares PostgreSQL's group-commit WAL, so on write-heavy workloads it performs similarly to PostgreSQL (~1.2× on New-Order, Figure 2a) and avoids SQLite's sync penalty. On the read-heavy Order-Status workload (Figure 2c), MariaDB is the most energy-expensive of the three: 1.75× worse than PostgreSQL and 3.1× worse than SQLite. The reason is throughput, not power: both engines draw essentially the same energy over the run and sit at ~0%

[1] New-Order 45%, Payment 43%, Order-Status / Delivery / Stock-Level 4% each.

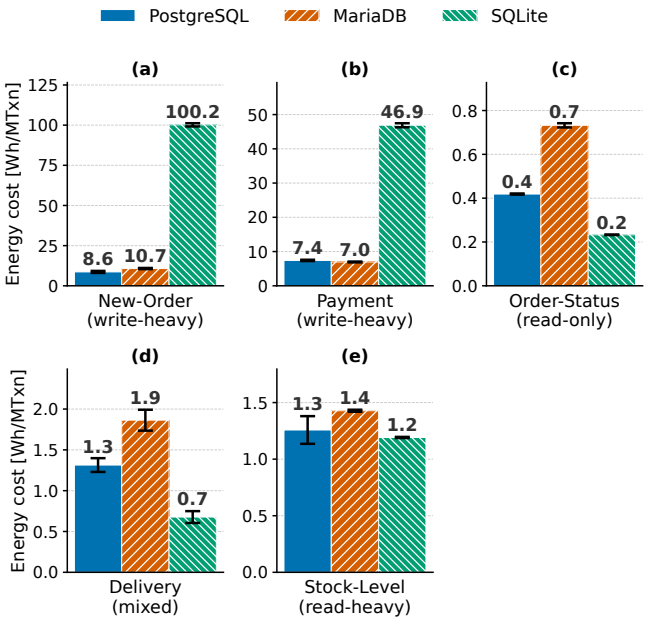

Fig. 2. Per-transaction energy cost (Wh/MTxn) for each TPC-C transaction type in isolation on the default setup. Lower is better.

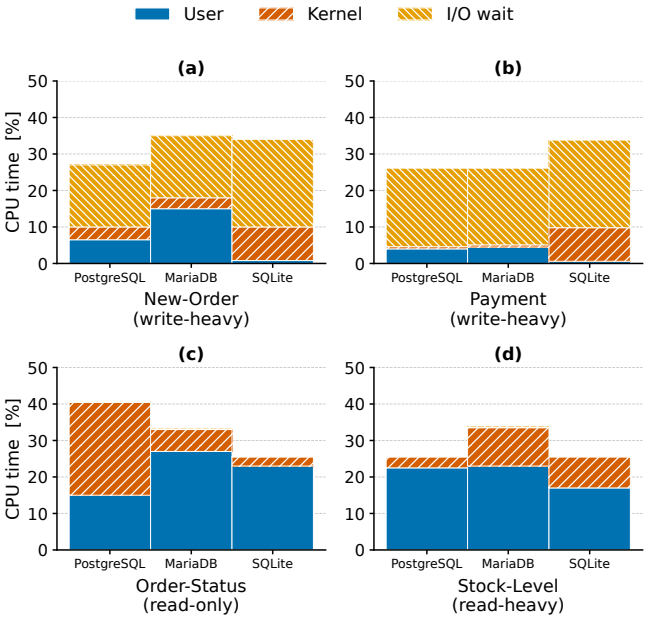

Fig. 3. CPU time breakdown (user, kernel, I/O wait) per TPC-C transaction type on the default setup. Delivery is omitted because its target table empties during the run at our scale, causing high variance and inconsistent results. The remainder to 100% is idle.

I/O wait, but MariaDB executes ~1.7× more CPU instructions per Order-Status transaction, so it completes fewer queries per second and the per-transaction energy rises accordingly.

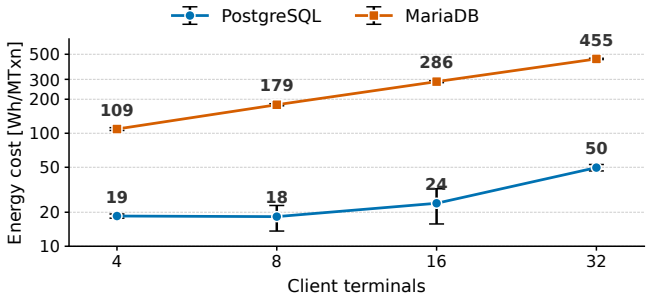

Fig. 4. Energy per million transactions (Wh/MTxn, log scale) for a TPC-C concurrency sweep (64 warehouses, 4–32 terminals) on the second setup, PostgreSQL vs. MariaDB. Lower is better.

Figure 3 confirms that the per-transaction `fsync` in SQLite wastes energy by keeping the CPU powered during I/O wait rather than doing useful work. On write-heavy New-Order and Payment (Figure 3a,b), SQLite spends up to 24% of the time in I/O wait with the CPU drawing power while waiting for `fsync` to complete, with less than 1% of useful computation. PostgreSQL and MariaDB show lower I/O wait because group commit reduces the number of stalls. On read-heavy queries (Figure 3c,d), I/O wait disappears and user-space CPU dominates for all three engines.

The gap persists under concurrency. Our default runs use a single client terminal. To test whether the client-server engines diverge under load, on the second setup, we sweep the TPC-C client terminals from 4 to 32 on a larger 64-warehouse database (~8 GB); we omit SQLite, whose single writer serializes commits and gains nothing from extra terminals. MariaDB's per-transaction energy stays well above PostgreSQL's across the sweep (Figure 4), from 5.9× at 4 terminals to 9.2× at 32, as MariaDB's query-processing overhead scales linearly while PostgreSQL's group commit keeps per-transaction energy roughly flat up to 32 clients. The architectural gap thus grows, rather than closes, with concurrency.

> **Takeaway:** *On TPC-C, SQLite costs 11.6× more energy than PostgreSQL on commit-heavy writes but up to 3× less on queries that rarely commit.* ***No SQL engine is most efficient across workloads. The per-transaction `fsync` dominates write energy, while embedded vs. client-server architecture dominates read energy.***

3.2 Configuration Impact

§3.1 showed that engine architecture matters. Here we hold the engine fixed and vary three configuration knobs, namely commit mode, indices, and query plan, to measure their individual energy impact.

A. Asynchronous commits. We first isolate the cost of durability by toggling synchronous commits on and off, measuring how much energy the disk-sync guarantee alone adds to write-heavy transactions on TPC-C.

By default, both SQLite and PostgreSQL hold commits until the transaction is durable on disk: PostgreSQL waits for the WAL `fsync`, while SQLite waits for the rollback journal and database-file syncs.

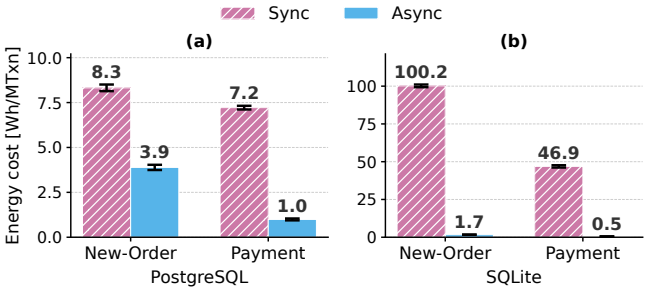

Fig. 5. Energy cost per million transactions (Wh/MTxn) for synchronous vs. asynchronous commits under TPC-C New-Order and Payment for PostgreSQL and SQLite, on the default setup. Lower is better.

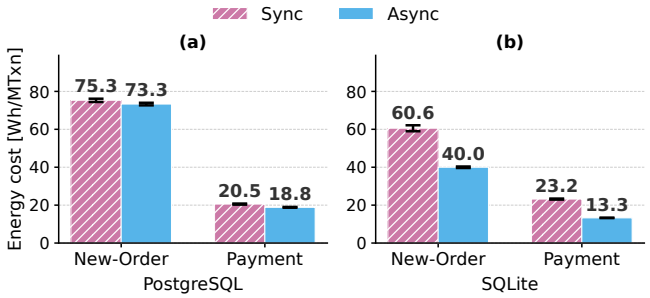

Fig. 6. Energy cost per million transactions (Wh/MTxn) for synchronous vs. asynchronous commits under TPC-C New-Order and Payment for PostgreSQL and SQLite, on the second setup with a RAID write-back cache. Lower is better.

Every `fsync` is a physical-I/O stall: the CPU draws idle power while waiting for the disk to confirm the write [22, 61].

We turn durability off[2] and accept a weaker guarantee: on a crash, PostgreSQL may lose the last few committed transactions, but stays consistent [58], while SQLite risks database corruption [54].

Figure 5 shows the effect on New-Order and Payment, the two most commit-heavy transactions. We omit Order-Status and Stock-Level (read-only, no commit sync) and Delivery (batches ten orders into one commit, so the per-transaction sync cost is already low). Async commits save up to 90× on SQLite Payment and 59× on New-Order (Figure 5b). PostgreSQL gains 2× on New-Order and 7× on Payment (Figure 5a).

SQLite synchronizes on every commit by default, so removing the sync removes most of the cost. Async SQLite effectively becomes an in-memory engine for writes, which explains the large ratios. PostgreSQL already amortizes the WAL `fsync` across concurrent backends via group commit, so there is less left to save. We omit MariaDB because its WAL-based commit behaves similarly to PostgreSQL's.

These results show that durability and consistency [19] have a quantifiable energy cost, and that cost depends as much on how an engine commits as on the guarantee itself: turning synchronous durability off saves 59–90× on SQLite but only 2–7× on PostgreSQL, whose group commit already amortizes the sync. The guarantee itself spans three levels. At one extreme, disabling all crash guarantees (SQLite async) is the cheapest option but risks database corruption, suitable only for disposable data. In between, PostgreSQL async relaxes durability while preserving consistency; recent commits may be lost on a crash, but the database never corrupts. At the other extreme, full synchronous durability (the default for both engines) is the most expensive. Group commit (PostgreSQL) and per-transaction `fsync` (SQLite) are two ways to provide this guarantee; under both, a committed transaction always survives a crash. They differ only in energy, because group commit lets many concurrent transactions share one disk sync while SQLite pays one sync per commit. Many workloads, such as event logging, caches, analytics pipelines, or session stores, do not require full durability, yet databases default to it. Making this cost visible lets designers choose the guarantee level their application actually needs.

[2] `PRAGMA synchronous = OFF` in SQLite, `synchronous_commit = off` in PostgreSQL.

On the second setup, whose RAID controller absorbs synchronous writes in a write-back cache, these savings collapse (Figure 6). SQLite gains only 1.5–1.7× and PostgreSQL 1.03–1.09×, because group commit already amortizes the WAL sync and the cache hides what remains. The saving is essentially the throughput gain: with durability off the engine completes more transactions for the same CPU+DRAM energy and no extra energy is spent on disk (apart from idle). The contrast with the HDD result (Figure 5, up to 90×) is the key result. An `fsync` that stalls on a physical disk burns idle CPU power on every commit, while a cached `fsync` is nearly free. Durability's energy cost is therefore set by the storage stack, not by the guarantee itself.

Durability is a correctness choice, not an energy one: if an application needs durability, it must enable it; if it does not, it can disable it for more speed. Figure 3 shows why. A synchronous commit performs almost no extra useful work; instead, the CPU spends up to 24% of the run in I/O wait (with under 1% useful computation), drawing power while it waits for the disk to confirm the write. Most of the energy is thus spent on stalls rather than useful computation.

Because of this, the same guarantee can cost far less energy when it is implemented differently. The question is how durability is implemented, not whether it is on.

When durability can be relaxed, the tradeoff is simple. A session store or an event log that can lose a few writes on a crash can run PostgreSQL with asynchronous commit, saving up to 7× energy while staying consistent. On the other hand, a financial ledger cannot, and there, the energy is simply the cost of both correctness and storage implementation.

B. Indices. Next, we measure how much energy a missing index costs by comparing indexed and unindexed executions of the same analytical queries on TPC-H.

An index lets the engine find matching rows without scanning the whole table. For latency, this is well understood; for energy, the mechanism is the same (fewer bytes read, fewer instructions, fewer joules), but it is rarely quantified. BenchBase creates the TPC-H indices for MariaDB and PostgreSQL, but not for SQLite. We use this to compare SQLite against itself, once with the default (unindexed) schema and once with the same indices added by hand.

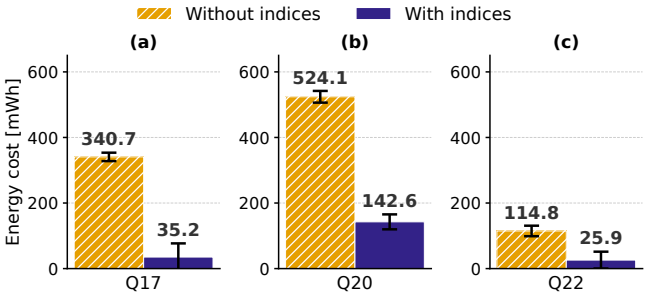

Fig. 7. Energy cost (mWh) for SQLite on the default setup with vs. without TPC-H indices on Q17, Q20, and Q22. Lower is better.

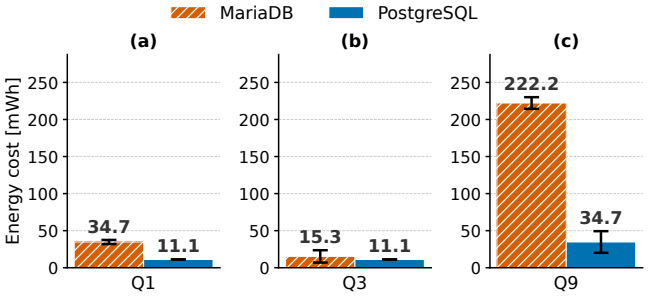

Fig. 8. Energy cost (mWh) for MariaDB and PostgreSQL on the default setup at SF 0.1, on TPC-H queries Q1, Q3, and Q9. Lower is better.

Figure 7 shows the three queries where the effect is the largest: Q17, Q20, Q22. Adding indices cuts SQLite's energy by 9.7× on Q17 (Figure 7a), 3.7× on Q20 (Figure 7b), and 4.4× on Q22 (Figure 7c). Q17 is a highly selective semi-join where the indexed plan reads only a small fraction of the rows; Q20 and Q22 still need large base scans that indices can only partly bypass. The rest of the queries are dominated by large sequential scans or aggregations where indices make no meaningful difference, and we omit them.

Without a suitable index, SQLite falls back to a full table scan, or for joins, to a transient automatic index [55] built and discarded at query time. Indices are not just a performance optimization but an energy optimization. Without an index, cost grows linearly with table size; with an index, it stays nearly constant. This effect persists on our second setup, where the energy difference reaches up to ~2×.

We also study the energy it takes to build the index. Constructing all SQLite TPC-H indices costs 0.07 Wh, about 20% of the energy of a single unindexed Q17 (0.34 Wh). Because the index is built once and reused, its energy cost is negligible for any repeatedly executed workload, so the per-query savings above hold even when the build cost is taken into account.

C. Query plan choice. Finally, we compare the energy impact of different query plans by running the same TPC-H queries on MariaDB and PostgreSQL, as they have different optimizers.

The query optimizer turns SQL into a physical plan, i.e., which operators to use, in what order, and with how much parallelism.

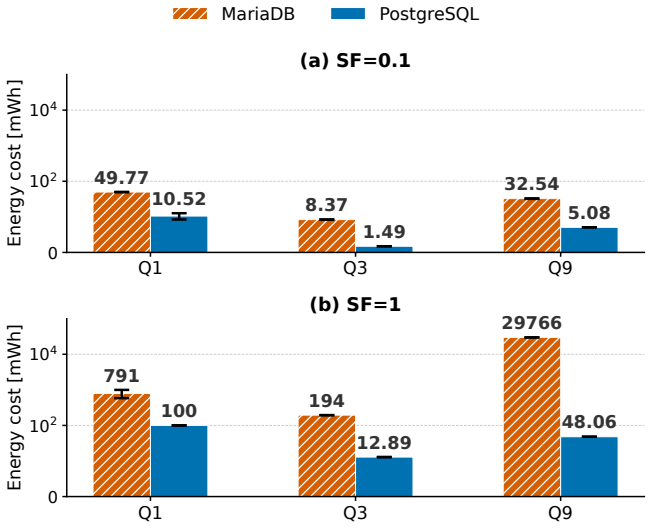

Fig. 9. Energy cost (mWh, log scale) for MariaDB and PostgreSQL on the second setup, on TPC-H queries Q1, Q3, and Q9, at (a) SF 0.1 and (b) SF 1. Lower is better.

Different optimizers on the same query, data, and hardware produce different plans and different energy costs. Prior work has built energy-aware cost models [17, 33, 66], but, to the best of our knowledge, the actual energy differences between plans produced by production engines have not been measured.

Figure 8 shows the three TPC-H queries where MariaDB's plan is worse than PostgreSQL's: Q1, Q3, and Q9. PostgreSQL uses 3.1×, 1.4×, and 6.4× less energy respectively. On the other queries we measured, the two engines are within 50% of each other, but we focus on the cases where the gap is the largest.

On Q1 and Q3, PostgreSQL uses a parallel sequential scan [44] while MariaDB uses a single-threaded one. The parallel plan splits the table across multiple cores, each scanning a portion simultaneously. The total CPU work is similar, but the execution time drops, so the system (CPU, DRAM, disk, fans) draws less total energy even though the instantaneous active power increases. On Q9 (a six-way join), MariaDB picks a join order that materializes more intermediate rows, increasing computation time and energy.

Given the same query, same data, and same hardware, differences in optimizer and execution strategy account for a 1.4–6.4× energy difference between engines. This gap is invisible to users today.

These gaps grow with scale (Figure 9). On the second setup, raising TPC-H from SF 0.1 (~200 MB) to SF 1 (~2 GB) widens the gap between MariaDB and PostgreSQL to orders of magnitude: Q1 from 4.7× to 7.9×, Q3 from 5.6× to 15×, and Q9 from 6.4× to ~620×. Thus, the plan gaps we report are expected to grow.

> **Takeaway:** *Commit mode and indices move per-transaction energy by up to* 90× *and* 9.7× *respectively within one engine. Across engines, optimizer and execution differences account for up to* 6.4×. ***Durability guarantees alone cost up to*** 90× ***more energy on direct-to-disk storage.***

4 Discussion

We show that architecture and configuration choices create order-of-magnitude energy differences. We discuss the implications.

Wh/Txn belongs in benchmarks and dashboards. TPC-C and TPC-H have shaped how we benchmark databases, yet neither reports energy. The TPC-Energy extension exists but is rarely used and reports only a benchmark-level aggregate [42], not per-transaction or per-configuration breakdowns. We propose that TPC-C, TPC-H, and their open-source derivatives report Wh/Txn next to throughput and latency at per-query granularity, so a result reads not just "12 000 tpmC at 5 ms P99" but also "0.8 Wh per million transactions." The infrastructure already exists: Kepler [4] exposes per-container energy on Kubernetes, RAPL gives per-socket joules out of the box, and every database already counts completed transactions.

Energy-aware query planning. Our results (Figure 8) show that optimizer and execution differences affect energy by up to 6.4× on the same query and hardware. The optimizer is an implicit energy actor. It decides how much work the system does, but without an energy term in its cost model. Tools like EnergAt [24] already provide per-operator energy attribution, and Xu et al. [66] show that adding power to the cost model yields 11–22% savings. Wh/Txn makes this feedback loop concrete, measurable, and optimizable.

Carbon-aware scheduling. Current carbon-aware schedulers [6, 21, 46] defer whole jobs to low-carbon hours. Per-query Wh/Txn lets them act at finer granularity, deferring or rerouting individual queries rather than entire workloads. Since energy and carbon do not always align [20, 65], per-query energy is what lets a planner weigh them explicitly.

Limitations and future work. Our two setups differ in how storage handles each commit. The default setup uses a commodity HDD that amplifies the per-commit `fsync` stall, while the second setup's RAID write-back cache nearly hides it. How much energy durability costs therefore depends on the storage, ranging from up to 90× on the HDD down to 1.5–1.7× with the cache. Our architecture, index, and plan findings, by contrast, do not depend on the storage, since they are set by CPU and memory work on RAM-resident data rather than by disk caching, and they hold on both setups. These findings are also conservative, because we run at low concurrency and small scale; group commit helps more as concurrency rises (§3.1), and plan and scan costs grow with data size, reaching ~620× at SF 1 (§3.2), so in production these gaps only widen and the values we report are lower bounds.

Three directions address these limitations. (1) *Direct-attached fast storage:* our two setups demonstrate a slow HDD and a cached RAID controller; the untested middle is direct-attached SSD/NVMe, where `fsync` is faster than on the HDD but not hidden by a controller cache, so commit-path gaps should fall between the two extremes we measured. (2) *Carbon:* connect BenchBase to a live grid-intensity signal so Wh/Txn becomes g CO_2e per query. (3) *Disaggregated memory:* CXL-attached memory [2] adds remote-access stalls similar to the disk stalls we measured; memory-heavy plans (hash joins, sorts) would incur more energy on such hardware, potentially changing which plan is cheapest.

5 Related Work

Energy in computing and databases. Significant work targets ML training energy [7, 11, 25, 30, 36, 50, 56], general-purpose computing [1, 9, 18, 68], cooling [3, 32, 41, 67], and carbon-aware scheduling [6, 16, 45]. Recent LLM inference work shows that execution time does not reliably predict energy consumption [5, 51]. Databases suffer a similar gap but lack the measurements to confirm it.

For DBMS energy, Harizopoulos et al. [22] first called for energy as a first-class database metric. Tsirogiannis et al. [61] measured per-query energy on a SQL Server and showed that CPU power varies up to 60% across operators at the same utilization. Subsequent studies profiled PostgreSQL under OLTP [40], compared ARM vs. x86 for transaction processing [52], and tracked MySQL energy across releases [38]. Two recent studies are closest to ours: den Hartog [14] compared PostgreSQL and MySQL on TPC-C with per-transaction energy (finding PostgreSQL is 6× more efficient), and Lella et al. [35] compared four databases on custom create, read, update, and delete workloads (finding 7–38% disparities). We extend this line with three architecturally distinct engines under both TPC-C and TPC-H, and add within-engine configuration experiments that reveal order-of-magnitude gaps. Bachras and Jacobsen [8] called for rearchitecting databases for sustainability. We show that existing knobs already produce large differences, which can help steer the design of new sustainable systems. Kannan et al. [31] explored the durability–energy tradeoff for NVM; we show it holds for disk-based OLTP with different database engines.

Energy-aware design and tooling. A second line of research redesigns DBMS internals for energy: energy-efficient replication [48], cluster power management [34], self-tuning for energy [62], and proactive storage power-state control [23]. Xu et al. [66] added power to a query optimizer's cost model, achieving 11–22% savings. OtterTune [64] tunes DBMS knobs for latency and throughput but does not consider energy. On the measurement side, EnergAt [24], Scaphandre [26], CodeCarbon [12], Kepler [4], and Ecovisor [53] provide increasingly fine-grained software energy attribution. TPC-Energy [42, 43] is the canonical DB-energy benchmark but covers only whole-system power.

6 Conclusion

We measured the energy cost of three SQL engines under TPC-C and TPC-H, and found that database architecture and configuration choices create order-of-magnitude energy differences, up to 11.6× across engines and up to 90× within a single engine from one configuration change. On direct-to-disk storage, the largest factor is durability, ranging from 2× to 90× depending on the engine and guarantee level (a write-back cache shrinks it to 1.5–1.7×).

All of this is measurable today with existing hardware. The missing piece is not infrastructure but visibility: software-level choices, from durability to index design and plan selection, are the dominant energy factors, yet they remain invisible in current benchmarks. Reporting Wh/Txn alongside throughput and latency makes these differences actionable for query planners, database administrators, and carbon-aware schedulers.

References

[1] Bilge Acun, Benjamin Lee, Fiodar Kazhamiaka, Kiwan Maeng, Udit Gupta, Manoj Chakkaravarthy, David Brooks, and Carole-Jean Wu. 2023. Carbon Explorer: A Holistic Framework for Designing Carbon Aware Datacenters. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2* (Vancouver, BC, Canada) *(ASPLOS 2023)*. Association for Computing Machinery, New York, NY, USA, 118–132. doi:10.1145/3575693.3575754

[2] Minseon Ahn, Thomas Willhalm, Norman May, Donghun Lee, Suprasad Mutalik Desai, Daniel Booss, Jungmin Kim, Navneet Singh, Daniel Ritter, and Oliver Rebholz. 2024. An Examination of CXL Memory Use Cases for In-Memory Database Management Systems Using SAP HANA. *Proc. VLDB Endow.* 17, 12 (Aug. 2024), 3827–3840. doi:10.14778/3685800.3685809

[3] Ahmed A. Alkrush, Mohamed S. Salem, O. Abdelrehim, and A.A. Hegazi. 2024. Data centers cooling: A critical review of techniques, challenges, and energy saving solutions. *International Journal of Refrigeration* 160 (2024), 246–262. doi:10.1016/j.ijrefrig.2024.02.007

[4] Marcelo Amaral, Huamin Chen, Tatsuhiro Chiba, Rina Nakazawa, Sunyanan Choochotkaew, Eun Kyung Lee, and Tamar Eilam. 2023. Kepler: A Framework to Calculate the Energy Consumption of Containerized Applications. In *2023 IEEE 16th International Conference on Cloud Computing (CLOUD)*. 69–71. doi:10.1109/CLOUD60044.2023.00017

[5] Vaastav Anand, Zhiqiang Xie, Matheus Stolet, Roberta De Viti, Thomas Davidson, Reyhaneh Karimipour, Safya Alzayat, and Jonathan Mace. 2023. The Odd One Out: Energy is Not Like Other Metrics. *SIGENERGY Energy Inform. Rev.* 3, 3 (Oct. 2023), 71–77. doi:10.1145/3630614.3630627

[6] Thomas Anderson, Adam Belay, Mosharaf Chowdhury, Asaf Cidon, and Irene Zhang. 2023. Treehouse: A Case For Carbon-Aware Datacenter Software. *SIGENERGY Energy Inform. Rev.* 3, 3 (Oct. 2023), 64–70. doi:10.1145/3630614.3630626

[7] Lasse F. Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. 2020. Carbontracker: Tracking and Predicting the Carbon Footprint of Training Deep Learning Models. arXiv:2007.03051 [cs.CY] https://arxiv.org/abs/2007.03051

[8] Michail Bachras and Hans-Arno Jacobsen. 2025. Environmental Footprints of Query Processing: A Vision for Sustainable Database Architectures. *Proc. VLDB Endow.* 18, 11 (July 2025), 4064–4072. doi:10.14778/3749646.3749676

[9] Erik Brunvand, Donald Kline, and Alex K. Jones. 2018. Dark Silicon Considered Harmful: A Case for Truly Green Computing. In *2018 Ninth International Green and Sustainable Computing Conference (IGSC)*. 1–8. doi:10.1109/IGCC.2018.8752110

[10] 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. 2022. eco2AI: carbon emissions tracking of machine learning models as the first step towards sustainable AI. *Doklady Mathematics* 106, Suppl. 1 (2022), S118–S128. doi:10.1134/S1064562422060230

[11] Andrew A Chien, Liuzixuan Lin, Hai Nguyen, Varsha Rao, Tristan Sharma, and Rajini Wijayawardana. 2023. Reducing the Carbon Impact of Generative AI Inference (today and in 2035). In *Proceedings of the 2nd Workshop on Sustainable Computer Systems* (Boston, MA, USA) *(HotCarbon '23)*. Association for Computing Machinery, New York, NY, USA, Article 11, 7 pages. doi:10.1145/3604930.3605705

[12] 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

[13] D. Richard Hipp. 2026. SQLite Home Page. https://sqlite.org/. Accessed: 2026-03-25.

[14] Anne den Hartog. 2024. Database energy benchmarks: an evaluation. https://www.cs.ru.nl/bachelors-theses/2024/Anne_den_Hartog___1044796___Database_energy_benchmarks_-_an_evaluation.pdf

[15] Djellel Eddine Difallah, Andrew Pavlo, Carlo Curino, and Philippe Cudre-Mauroux. 2013. OLTP-Bench: an extensible testbed for benchmarking relational databases. *Proc. VLDB Endow.* 7, 4 (Dec. 2013), 277–288. doi:10.14778/2732240.2732246

[16] Han Dong, Parul Singh, Yara Awad, Felix George, Krishnasuri Narayanam, Sanjay Arora, and Jonathan Appavoo. 2025. Towards Performance and Energy Aware Kubernetes Scheduler. *SIGENERGY Energy Inform. Rev.* 5, 2 (Aug. 2025), 69–75. doi:10.1145/3757892.3757902

[17] Binglei Guo, Jiong Yu, Dexian Yang, Hongyong Leng, and Bin Liao. 2022. Energy-Efficient Database Systems: A Systematic Survey. *ACM Comput. Surv.* 55, 6, Article 111 (Dec. 2022), 53 pages. doi:10.1145/3538225

[18] Udit Gupta, Mariam Elgamal, Gage Hills, Gu-Yeon Wei, Hsien-Hsin S. Lee, David Brooks, and Carole-Jean Wu. 2022. ACT: designing sustainable computer systems with an architectural carbon modeling tool. In *Proceedings of the 49th Annual International Symposium on Computer Architecture* (New York, New York) *(ISCA '22)*. Association for Computing Machinery, New York, NY, USA, 784–799. doi:10.1145/3470496.3527408

[19] Theo Haerder and Andreas Reuter. 1983. Principles of transaction-oriented database recovery. *ACM Comput. Surv.* 15, 4 (Dec. 1983), 287–317. doi:10.1145/289.291

[20] Walid A. Hanafy, Roozbeh Bostandoost, Noman Bashir, David Irwin, Mohammad Hajiesmaili, and Prashant Shenoy. 2023. The War of the Efficiencies: Understanding the Tension between Carbon and Energy Optimization. In *Proceedings of the 2nd Workshop on Sustainable Computer Systems* (Boston, MA, USA) *(HotCarbon '23)*. Association for Computing Machinery, New York, NY, USA, Article 19, 7 pages. doi:10.1145/3604930.3605709

[21] Walid A. Hanafy, Qianlin Liang, Noman Bashir, David Irwin, and Prashant Shenoy. 2023. CarbonScaler: Leveraging Cloud Workload Elasticity for Optimizing Carbon-Efficiency. *Proc. ACM Meas. Anal. Comput. Syst.* 7, 3, Article 57 (Dec. 2023), 28 pages. doi:10.1145/3626788

[22] Stavros Harizopoulos, Mehul Shah, Justin Meza, and Parthasarathy Ranganathan. 2009. Energy Efficiency: The New Holy Grail of Data Management Systems Research. arXiv:0909.1784 [cs.DB] https://arxiv.org/abs/0909.1784

[23] Yuto Hayamizu, Masaru Kitsuregawa, and Kazuo Goda. 2025. Proactive Energy Management in Database Systems. *SIGENERGY Energy Inform. Rev.* 4, 5 (April 2025), 154–159. doi:10.1145/3727200.3727223

[24] Hongyu Hè, Michal Friedman, and Theodoros Rekatsinas. 2024. EnergAt: Fine-Grained Energy Attribution for Multi-Tenancy. *SIGENERGY Energy Inform. Rev.* 4, 3 (Sept. 2024), 18–25. doi:10.1145/3698365.3698369

[25] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. 2020. Towards the systematic reporting of the energy and carbon footprints of machine learning. *J. Mach. Learn. Res.* 21, 1, Article 248 (Jan. 2020), 43 pages.

[26] Hubblo Org. 2026. Scaphandre. https://github.com/hubblo-org/scaphandre. Energy consumption metrology agent; accessed 2026-03-29.

[27] Intel Corporation. 2026. Intel VTune Profiler. https://www.intel.com/content/www/us/en/developer/tools/oneapi/vtune-profiler.html. Accessed 2026-03-29.

[28] International Energy Agency. 2025. *Energy and AI.* Technical Report. IEA. https://www.iea.org/reports/energy-and-ai

[29] Sooman Jeong, Kisung Lee, Seongjin Lee, Seoungbum Son, and Youjip Won. 2013. I/O stack optimization for smartphones. In *Proceedings of the 2013 USENIX Conference on Annual Technical Conference* (San Jose, CA) *(USENIX ATC'13)*. USENIX Association, USA, 309–320.

[30] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. 2017. In-Datacenter Performance Analysis of a Tensor Processing Unit. arXiv:1704.04760 [cs.AR] https://arxiv.org/abs/1704.04760

[31] Sudarsun Kannan, Moinuddin Qureshi, Ada Gavrilovska, and Karsten Schwan. 2016. Energy aware persistence: Reducing energy overheads of memory-based persistence in NVMs. In *2016 International Conference on Parallel Architecture and Compilation Techniques (PACT)*. 165–177. doi:10.1145/2967938.2967953

[32] I W Kuncoro, N A Pambudi, M K Biddinika, I Widiastuti, M Hijriawan, and K M Wibowo. 2019. Immersion cooling as the next technology for data center cooling: A review. *Journal of Physics: Conference Series* 1402, 4 (dec 2019), 044057. doi:10.1088/1742-6596/1402/4/044057

[33] Willis Lang, Ramakrishnan Kandhan, and Jignesh Patel. 2011. Rethinking Query Processing for Energy Efficiency: Slowing Down to Win the Race. *IEEE Data Eng. Bull.* 34 (01 2011), 12–23.

[34] Willis Lang and Jignesh M. Patel. 2010. Energy management for MapReduce clusters. *Proc. VLDB Endow.* 3, 1–2 (Sept. 2010), 129–139. doi:10.14778/1920841.1920862

[35] Hemasri Sai Lella, Kurra Manasa, Rajrupa Chattaraj, and Sridhar Chimalakonda. 2023. DBJoules: An Energy Measurement Tool for Database Management Systems. arXiv:2311.08961 [cs.SE] https://arxiv.org/abs/2311.08961

[36] Rui Lu and Dan Wang. 2025. A Thermal-Aware Workload Scheduler for High-Performance LLM Inference in Cooling-Regulated Datacenters. *SIGENERGY Energy Inform. Rev.* 5, 2 (Aug. 2025), 98–104. doi:10.1145/3757892.3757906

[37] MariaDB Foundation. 2026. MariaDB Foundation. https://mariadb.org/. Accessed: 2026-03-25.

[38] A. V. Miranskyy, Z. Al-zanbouri, D. Godwin, and A. B. Bener. 2017. Database engines: Evolution of greenness. *Journal of Software: Evolution and Process* 30, 4 (Nov. 2017). doi:10.1002/smr.1915

[39] Lam-Duy Nguyen, Adnan Alhomssi, Tobias Ziegler, and Viktor Leis. 2025. Moving on From Group Commit: Autonomous Commit Enables High Throughput and Low Latency on NVMe SSDs. *Proc. ACM Manag. Data* 3, 3, Article 191 (June 2025), 24 pages. doi:10.1145/3725328

[40] Raik Niemann, Nikolaos Korfiatis, Roberto Zicari, and Richard Göbel. 2013. Evaluating the Energy Efficiency of OLTP Operations. In *Availability, Reliability, and Security in Information Systems and HCI*, Alfredo Cuzzocrea, Christian Kittl, Dimitris E. Simos, Edgar Weippl, and Lida Xu (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 28–43.

[41] Ehsan Pakbaznia and Massoud Pedram. 2009. Minimizing data center cooling and server power costs. In *Proceedings of the 2009 ACM/IEEE International Symposium on Low Power Electronics and Design* (San Fancisco, CA, USA) *(ISLPED '09)*. Association for Computing Machinery, New York, NY, USA, 145–150. doi:10.1145/1594233.1594268

[42] Meikel Poess, Raghunath Othayoth Nambiar, Kushagra Vaid, John M. Stephens, Karl Huppler, and Evan Haines. 2010. Energy benchmarks: a detailed analysis. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking* (Passau, Germany) *(e-Energy '10)*. Association for Computing Machinery, New York, NY, USA, 131–140. doi:10.1145/1791314.1791336

[43] Meikel Poess, Da Qi Ren, Tilmann Rabl, and Hans-Arno Jacobsen. 2018. Methods for Quantifying Energy Consumption in TPC-H. In *Proceedings of the 2018 ACM/SPEC International Conference on Performance Engineering* (Berlin, Germany) *(ICPE '18)*. Association for Computing Machinery, New York, NY, USA, 293–304. doi:10.1145/3184407.3184429

[44] PostgreSQL Global Development Group. 2026. PostgreSQL Documentation: Parallel Query. https://www.postgresql.org/docs/current/parallel-query.html. Accessed: 2026-05-09.

[45] Feitong Qiao, Yiming Fang, and Asaf Cidon. 2025. Energy-Aware Process Scheduling in Linux. *SIGENERGY Energy Inform. Rev.* 4, 5 (April 2025), 91–97. doi:10.1145/3727200.3727214

[46] Ana Radovanovic, Ross Koningstein, Ian Schneider, Bokan Chen, Alexandre Duarte, Binz Roy, Diyue Xiao, Maya Haridasan, Patrick Hung, Nick Care, Saurav Talukdar, Eric Mullen, Kendal Smith, MariEllen Cottman, and Walfredo Cirne. 2021. Carbon-Aware Computing for Datacenters. arXiv:2106.11750 [cs.DC] https://arxiv.org/abs/2106.11750

[47] Manuel Rodriguez-Martinez, Harold Valdivia, Jaime Seguel, and Melvin Greer. 2011. Estimating Power/Energy Consumption in Database Servers. *Procedia Computer Science* 6 (2011), 112–117. doi:10.1016/j.procs.2011.08.022 Complex adaptive sysytems.

[48] Nicolas Schiper, Fernando Pedone, and Robbert Van Renesse. 2014. The Energy Efficiency of Database Replication Protocols. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. 407–418. doi:10.1109/DSN.2014.46

[49] Ian Schneider and Taylor Mattia. 2024. Carbon accounting in the Cloud: a methodology for allocating emissions across data center users. arXiv:2406.09645 [cs.SE] https://arxiv.org/abs/2406.09645

[50] Ian Schneider, Hui Xu, Stephan Benecke, David Patterson, Keguo Huang, Parthasarathy Ranganathan, and Cooper Elsworth. 2025. Life-Cycle Emissions of AI Hardware: A Cradle-To-Grave Approach and Generational Trends. arXiv:2502.01671 [cs.AR] https://arxiv.org/abs/2502.01671

[51] Prasoon Sinha, Dimitrios Liakopoulos, Ruihao Li, and Neeraja J. Yadwadkar. 2025. The Utilization Fallacy and the Real Drivers of Carbon-Efficient Inference Serving. *SIGENERGY Energy Inform. Rev.* 5, 2 (Aug. 2025), 76–83. doi:10.1145/3757892.3757903

[52] Utku Sirin, Raja Appuswamy, and Anastasia Ailamaki. 2016. OLTP on a server-grade ARM: power, throughput and latency comparison. In *Proceedings of the 12th International Workshop on Data Management on New Hardware* (San Francisco, California) *(DaMoN '16)*. Association for Computing Machinery, New York, NY, USA, Article 10, 7 pages. doi:10.1145/2933349.2933359

[53] Abel Souza, Noman Bashir, Jorge Murillo, Walid Hanafy, Qianlin Liang, David Irwin, and Prashant Shenoy. 2023. Ecovisor: A Virtual Energy System for Carbon-Efficient Applications. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2* (Vancouver, BC, Canada) *(ASPLOS 2023)*. Association for Computing Machinery, New York, NY, USA, 252–265. doi:10.1145/3575693.3575709

[54] SQLite Consortium. 2026. Atomic Commit In SQLite. https://sqlite.org/atomiccommit.html. Accessed 2026-05-09.

[55] SQLite Project. 2026. The SQLite Query Optimizer Overview. https://www.sqlite.org/optoverview.html. Accessed: 2026-05-09.

[56] 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*, Anna Korhonen, David Traum, and Lluís Màrquez (Eds.). Association for Computational Linguistics, Florence, Italy, 3645–3650. doi:10.18653/v1/P19-1355

[57] The PostgreSQL Global Development Group. 2026. PostgreSQL. https://www.postgresql.org/. Accessed: 2026-03-25.

[58] The PostgreSQL Global Development Group. 2026. PostgreSQL Documentation, Chapter 28.3: Write-Ahead Logging (WAL). https://www.postgresql.org/docs/current/wal-intro.html. Accessed 2026-05-09.

[59] Transaction Processing Performance Council. 2010. TPC-C Benchmark Specification, Version 5.11.0. https://www.tpc.org/TPC_Documents_Current_Versions/pdf/tpc-c_v5.11.0.pdf

[60] Transaction Processing Performance Council. 2023. TPC-H Benchmark Specification. https://www.tpc.org/tpc_documents_current_versions/current_specifications5.asp

[61] Dimitris Tsirogiannis, Stavros Harizopoulos, and Mehul A. Shah. 2010. Analyzing the energy efficiency of a database server. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data* (Indianapolis, Indiana, USA) *(SIGMOD '10)*. Association for Computing Machinery, New York, NY, USA, 231–242. doi:10.1145/1807167.1807194

[62] Yi-Cheng Tu, Xiaorui Wang, Bo Zeng, and Zichen Xu. 2014. A system for energy-efficient data management. *SIGMOD Rec.* 43, 1 (May 2014), 21–26. doi:10.1145/2627692.2627696

[63] Dam Quang Tuan, Seungyong Cheon, and Youjip Won. 2016. On the IO characteristics of the SQLite transactions. In *Proceedings of the International Conference on Mobile Software Engineering and Systems* (Austin, Texas) *(MOBILESoft '16)*. Association for Computing Machinery, New York, NY, USA, 214–224. doi:10.1145/2897073.2897093

[64] Dana Van Aken, Andrew Pavlo, Geoffrey J. Gordon, and Bohan Zhang. 2017. Automatic Database Management System Tuning Through Large-scale Machine Learning. In *Proceedings of the 2017 ACM International Conference on Management of Data* (Chicago, Illinois, USA) *(SIGMOD '17)*. Association for Computing Machinery, New York, NY, USA, 1009–1024. doi:10.1145/3035918.3064029

[65] Jackson Woodruff, David Schall, Michael F.P. O'Boyle, and Christopher Woodruff. 2023. When Does Saving Power Save the Planet?. In *Proceedings of the 2nd Workshop on Sustainable Computer Systems* (Boston, MA, USA) *(HotCarbon '23)*. Association for Computing Machinery, New York, NY, USA, Article 20, 6 pages. doi:10.1145/3604930.3605719

[66] Zichen Xu, Yi-Cheng Tu, and Xiaorui Wang. 2013. Dynamic Energy Estimation of Query Plans in Database Systems. In *Proceedings of the 2013 IEEE 33rd International Conference on Distributed Computing Systems (ICDCS '13)*. IEEE Computer Society, USA, 83–92. doi:10.1109/ICDCS.2013.21

[67] Qingxia Zhang, Zihao Meng, Xianwen Hong, Yuhao Zhan, Jia Liu, Jiabao Dong, Tian Bai, Junyu Niu, and M. Jamal Deen. 2021. A survey on data center cooling systems: Technology, power consumption modeling and control strategy optimization. *J. Syst. Archit.* 119, C (Oct. 2021), 17 pages. doi:10.1016/j.sysarc.2021.102253

[68] Jiechen Zhao, Katie Lim, Thomas Anderson, and Natalie Enright Jerger. 2023. The Case of Unsustainable CPU Affinity. In *Proceedings of the 2nd Workshop on Sustainable Computer Systems* (Boston, MA, USA) *(HotCarbon '23)*. Association for Computing Machinery, New York, NY, USA, Article 1, 7 pages. doi:10.1145/3604930.3605706