Workload-Aware Storage Reduction for Multi-Tenant SIEM on ClickHouse

Nutthakorn Chalaemwongwan
Department of Computer Engineering, KOSEN-KMITL,
King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand

***Abstract*—Security Information and Event Management (SIEM) platforms ingest terabytes of heterogeneous telemetry daily—Windows event logs, DNS queries, HTTP transactions, EDR alerts, and network metadata from Zeek—yet the majority of stored records are never queried for threat-hunting or incident-response workflows. This study presents a workload-aware storage reduction framework that tailors data retention to observed analytical demand within a multi-tenant ClickHouse deployment. The main contribution is a Workload Analyzer algorithm that extracts importance scores for columns from ClickHouse query logs using a frequency–recency–coverage weighting scheme, and a Storage-Coverage Cost Model that computes the optimal pruning threshold that minimizes a weighted sum of storage cost and coverage loss. Guided by these metrics, the framework applies six composable reduction operators: column pruning with materialized views, adaptive sampling, deduplication, per-column codec selection, skip-indexing, and time-to-live (TTL)-based retention tiering across hot/warm/cold storage. Multi-tenant isolation is enforced through role-based access control overlays aligned with the Thai Personal Data Protection Act (PDPA). Experimental evaluation on 1,000,000 Zipf-distributed Windows Security Events demonstrates 79% uncompressed and 70% compressed storage reduction with sub-second query latency, while the Workload Analyzer automatically identifies the optimal column subset that preserves 100% detection rule coverage at minimum storage cost.**

Keywords*—ClickHouse; SIEM; storage reduction; workload analysis; column importance scoring; multi-tenant; PDPA compliance

I. Introduction

Every SOC has a SIEM and every SIEM has storage. A mid-sized enterprise generates 200–500 GB of security telemetry per day, and big shops do 1 TB or more [1]–[3]. At cloud prices, this translates to tens of thousands of dollars a month for storage with no end in sight [4].

Splunk billing is by gigabytes ingested, so analysts are on the lookout for what they send to the platform [5], [6]. Switching to an open-source stack like ClickHouse [7] eliminates the costly licenses but does not address the fact that the volume of data is not slowing down.

The ugly reality we kept running into was that most of the bytes never get queried. With an examination of real-world query logs, we found that only a small set of columns matter for Sigma rules [8], [9] and for investigation playbooks. The rest of the data is dead weight burning the budget without raising a single alert [3].

We propose a **workload-aware storage reduction framework** for multi-tenant SIEM platforms on ClickHouse. Unlike prior work that applies individual techniques in isolation—compression here, sampling there—our approach *composes* six reduction operators into a single pipeline whose pruning decisions are driven by a formal cost model that explicitly balances storage savings against detection coverage loss. This composition, guided by workload-aware optimization, is what distinguishes our work from both database-side automatic physical design [10], [11] and SIEM-side lifecycle management [1], [2].

We make four contributions:

- A Workload Analyzer that mines ClickHouse query logs to score every column by frequency, recency, and detection rule coverage—then automatically decides what to keep and what to drop.
- A Storage-Coverage Cost Model that frames column pruning as an explicit optimization: minimize storage cost subject to a penalty for losing detection coverage.
- A composable pipeline of six reduction operators—column pruning, adaptive sampling, deduplication, codec selection, skip-index acceleration, and TTL-tiered retention—engineered for security telemetry on ClickHouse.
- A multi-tenant infrastructure that supports the Thai PDPA [12], achieving 79% raw data storage reduction with sub-second query latency.

The study proceeds as follows: Section II surveys related work. Section III describes the system architecture. Section IV presents the Workload Analyzer. Section V and Section VI cover the reduction operators and tenant isolation, respectively. Section VII reports experiments. Section VIII and Section IX discuss findings and conclusions.

II. Related Work

A. SIEM Platforms and Log Management

Mainstream SIEM products (e.g., as reviewed by Bhatt et al. [13])—Splunk [1], IBM QRadar [14], Elastic Security [2]—all do log management, but at almost-doubled storage cost. Splunk extracts fields at index time and keeps raw payload around; QRadar discards part of raw data in its normalized event model. Net effect is to store roughly double size of original log files on disk.

Open-source deployments like Wazuh [15] and Security Onion [16] fall back on Elasticsearch, which stores data in

an inverted-index format. The ILM features of Elasticsearch enable rolling indices and scheduled deletion, but this is a time-based, waveform-pattern approach, not a workload-aware column-level optimization strategy. The distinction is important.

B. Columnar Storage for Security Analytics

Columnar databases are being used for security analytics. ClickHouse [7], [17], which was initially developed at Yandex for web analytics, provides native columnar storage with high-compression ratios, making it a popular choice for log analysis. The ClickHouse VLDB industrial paper [17] shows that columnar storage with per-column codecs significantly outperforms row-oriented storage for log analysis, and the Huntress team showed that it is feasible to build a complete SIEM on ClickHouse even at a scale of billions of events per day [4] (complemented by recent academic evaluations of cloud-native log storage [6], [18]). Columnar storage for log analysis for cloud-native security information and event management (SIEM) is also affordable in the cloud, as shown by Zhu et al. [6].

Column-oriented storage in general allows for column pruning (only reading columns that are used in a query from disk), but only at query time, and the entire schema is still stored. Our approach extends this to prune unused columns at ingestion time using materialized views, which removes storage overhead permanently.

C. Data Reduction Techniques

Data reduction for log management has been studied in several contexts. Sampling approaches [19] reduce data volume but risk losing rare, security-critical events. Deduplication [20] is effective for repetitive logs but requires careful handling of event ordering and provenance. Compression-based approaches exploiting domain-specific patterns [21] have shown significant gains in analytical databases. Recent work on log compression by Li et al. [18] demonstrates that exploiting both static and runtime patterns can substantially reduce cloud log storage costs. He et al. [3] provide an empirical study of log parsing practices in industry.

Query-driven workload analysis for database optimization has been studied by Ma et al. [10], who proposed workload forecasting for self-driving databases, and more recently by Perera et al. [22], who developed adaptive learned query optimizers for multi-tenant systems. Automatic materialized view generation using deep learning has been explored by Liu et al. [11]. However, none of these approaches target the specific combination of security telemetry, detection rule coverage preservation, and SIEM workload patterns that our framework addresses.

Our work differs from prior approaches by *composing* multiple reduction techniques into a unified, workload-aware pipeline that is specifically designed for security telemetry and multi-tenant deployment. Table I summarizes the key differences.

D. Multi-Tenancy and Data Privacy

Multi-tenant database architectures have been extensively studied [23], [24]. In the context of security platforms, tenant isolation carries additional requirements due to the sensitivity of security telemetry [22]. The Thai PDPA [12], modeled after the EU GDPR [25], [26], mandates purpose limitation, data minimization, and storage limitation—principles that align naturally with workload-aware storage reduction.

TABLE I. COMPARISON WITH EXISTING APPROACHES

Approach	Workload Aware	Column Pruning	Detection Coverage	Multi-Tenant	Privacy (PDPA)
Splunk ILM [1]	–	–	–	✓	–
Elastic ILM [2]	–	–	–	✓	–
ClickHouse [17]	–	✓	–	–	–
Huntress [4]	–	–	–	✓	–
Ma et al. [10]	✓	–	–	–	–
Liu et al. [11]	✓	✓	–	–	–
Ours	✓	✓	✓	✓	✓

III. SYSTEM ARCHITECTURE

Fig. 1 shows the overall system. Four subsystems work together: an ingestion layer, a reduction operator pipeline, a multi-tenant overlay, and a tiered storage backend. We walk through each below.

A. Ingestion Layer

The ingestion layer accepts security telemetry from heterogeneous sources, including endpoint agents (Beats, Wazuh), EDR platforms, Windows Event Forwarding (WEF), DNS resolvers, HTTP proxies, and Zeek network sensors. All events are serialized as JSON and published to Apache Kafka [27] topics partitioned by source type and tenant identifier.

Kafka works as a strong abstraction between event producers and downstream storages. That matters in practice: during a security incident, a spike in burst ingestion can bring ingestion rates $10\times$ above baseline, and Kafka absorbs that without backing pressure against the agents.

B. Reduction Operator Pipeline

Events consumed from Kafka undergo a parameterizable 6-stage reduction pipeline (Section V), applying the operations in an order that maximizes cumulative reduction:

- Column pruning (dropping unused columns)
- Deduplication and canonicalization (merging duplicates)
- Adaptive sampling (downsampling low-value event types)
- Codec selection (best compression strategy per column)
- Skip-index build (supporting frequent match-all scans)
- TTL assignment (applying tiered expiration policies)

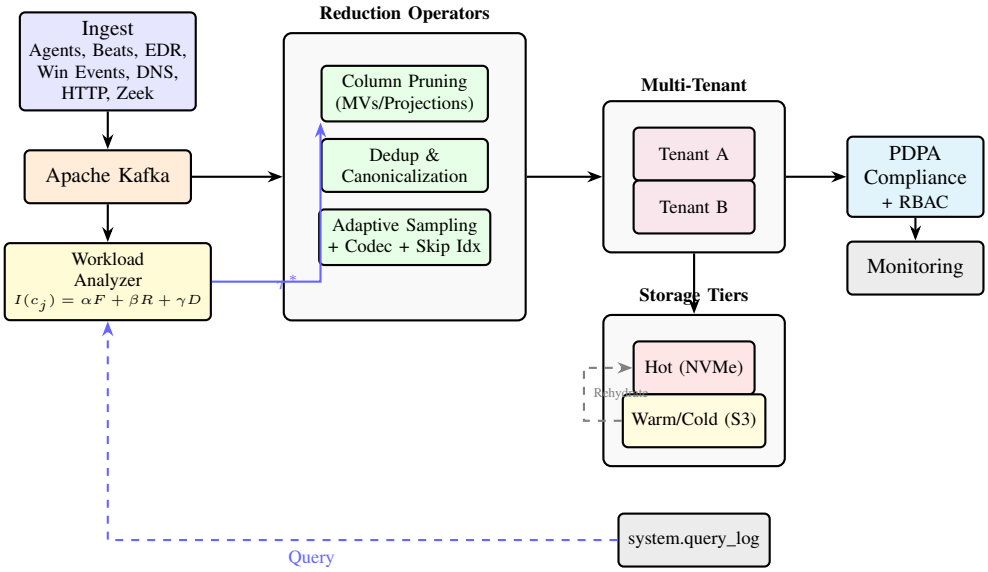

Fig. 1. System architecture of the workload-aware storage reduction framework. Security telemetry flows from diverse sources through Kafka into the reduction operator pipeline, which applies six composable operators before storing data in a multi-tenant, tiered ClickHouse deployment. RBAC overlays enforce tenant isolation and PDPA compliance.

C. Multi-Tenant Overlay

The multi-tenant overlay provides logical isolation between tenants within a shared ClickHouse cluster. Each tenant's data is tagged with a `tenant_id` column and access is enforced through ClickHouse's row-level security policies and role-based access control (RBAC). Section VI details this subsystem.

D. Tiered Storage Backend

ClickHouse's native storage tiering is configured with three tiers: **Hot** (NVMe SSD) for the most recent and frequently queried data, **Warm** (HDD or network-attached) for aging data, and **Cold** (S3-compatible object storage) for long-term archival. TTL rules automatically migrate data between tiers based on age, and on-demand rehydration allows cold data to be temporarily promoted to hot storage for investigation workflows.

IV. Workload Analyzer and Cost Model

How do we decide which columns to keep? Doing this by hand is tedious and breaks whenever someone adds a new Sigma rule [9]. Previous work on query workload forecasting [10] and learned indexes [28] has shown that mining query logs is a viable path to automation. We take that idea further: this section describes an algorithm that scores every column and a cost model that picks the best pruning threshold.

A. Column Importance Scoring

The Workload Analyzer continuously mines ClickHouse's `system.query_log` to extract column access patterns. For each column c_j in the schema, it computes an importance score $I(c_j)$ [using Eq. (1)] that reflects three dimensions:

$$I(c_j) = \alpha \cdot F(c_j) + \beta \cdot R(c_j) + \gamma \cdot D(c_j) \tag{1}$$

where, $\alpha + \beta + \gamma = 1$ and:

- $F(c_j) = \frac{n(c_j)}{N}$ is the **frequency score**: the fraction of queries in the observation window that reference column c_j, where $n(c_j)$ is the number of queries referencing c_j and N is the total query count.
- $R(c_j) = \frac{1}{M} \sum_{i=1}^{M} e^{-\lambda \cdot \Delta t_i}$ is the **recency score**: an exponentially-weighted average of the time elapsed Δt_i (in days) since each of the M most recent accesses to c_j, with decay parameter $\lambda > 0$.
- $D(c_j) = \frac{|\mathrm{R}(c_j)|}{|\mathrm{R}|}$ is the **detection coverage score**: the fraction of active Sigma detection rules R that reference column c_j.

The frequency score captures routine analytical demand, the recency score ensures that recently active columns are preserved even if their long-term frequency is low (e.g., during an active investigation), and the detection coverage score prevents pruning of columns that are critical to threat detection rules.

B. Automatic Column Classification

Algorithm 1 presents the complete Workload Analyzer procedure. Given a pruning threshold τ and the schema S, it classifies each column as RETAIN (include in the pruned materialized view) or PRUNE (exclude from long-term storage).

The protected set P contains columns that must never be pruned regardless of their importance score (e.g., primary keys, tenant identifiers, timestamps). This ensures structural integrity of the ClickHouse MergeTree engine.

C. Storage-Coverage Cost Model

The pruning threshold τ controls the trade-off between storage savings and detection coverage. We formulate this as

Algorithm 1 Workload Analyzer: Column Classification

Require: Schema $\mathrm{S} = \{c_1, c_2, \ldots, c_n\}$, threshold τ, weights (α, β, γ), decay λ, window W days, protected set P
Ensure: Classification $\mathrm{C} : c_j \mapsto \{\textsc{Retain}, \textsc{Prune}\}$
$\mathrm{Q} \leftarrow \textsc{QueryLog}(W)$ ▷ Fetch queries from last W days
$\mathrm{R} \leftarrow \textsc{LoadSigmaRules}$ ▷ Active detection rules
$N \leftarrow |\mathrm{Q}|$
for each column $c_j \in \mathrm{S}$ **do**
  $n_j \leftarrow |\{q \in \mathrm{Q} : c_j \in \mathrm{cols}(q)\}|$
  $F_j \leftarrow n_j \,/\, N$
  $\{\Delta t_1, \ldots, \Delta t_M\} \leftarrow \textsc{RecentAccessTimes}(c_j, M)$
  $R_j \leftarrow \frac{1}{M} \sum_{i=1}^{M} e^{-\lambda \cdot \Delta t_i}$
  $D_j \leftarrow \left|\{r \in \mathrm{R} : c_j \in \mathrm{cols}(r)\}\right| / |\mathrm{R}|$
  $I_j \leftarrow \alpha \cdot F_j + \beta \cdot R_j + \gamma \cdot D_j$
  if $c_j \in \mathrm{P}$ **or** $I_j \geq \tau$ **then**
    $\mathrm{C}(c_j) \leftarrow \textsc{Retain}$
  else
    $\mathrm{C}(c_j) \leftarrow \textsc{Prune}$
  end if
end for
return C

an optimization problem.

Let $s(c_j)$ denote the average storage cost (bytes per row) of column c_j, and let $\mathrm{S}_\tau = \{c_j : I(c_j) \geq \tau\} \cup \mathrm{P}$ be the set of retained columns at threshold τ. The total storage cost is given by Eq. (2):

$$C_{\text{storage}}(\tau) = \sum_{c_j \in \mathrm{S}_\tau} s(c_j) \tag{2}$$

The detection coverage at threshold τ measures what fraction of active detection rules remain fully executable structurally, as defined in Eq. (3). Note that this metric counts only whether the required columns are retained, not whether individual events themselves survive adaptive sampling:

$$\mathrm{Cov}(\tau) = \frac{|\{r \in \mathrm{R} : \mathrm{cols}(r) \subseteq \mathrm{S}_\tau\}|}{|\mathrm{R}|} \tag{3}$$

The optimal threshold τ^* minimizes the combined cost formulated in Eq. (4):

$$\tau^* = \arg\min_{\tau \in [0,1]} \underbrace{\frac{C_{\text{storage}}(\tau)}{C_{\text{storage}}(0)}}_{\text{normalized storage}} + w \cdot \underbrace{\big(1 - \mathrm{Cov}(\tau)\big)}_{\text{coverage loss}} \tag{4}$$

where, $w > 0$ is a user-defined weight reflecting the relative importance of detection coverage versus storage savings. A high w (e.g., $w = 10$) strongly penalizes coverage loss, resulting in conservative pruning. A low w (e.g., $w = 1$) prioritizes storage reduction.

Since the number of distinct thresholds equals the number of unique importance scores (at most n), the optimization is solved by exhaustive enumeration in $O(n \cdot |\mathrm{R}|)$ time, which is negligible for typical schemas ($n < 100$) and rulesets ($|\mathrm{R}| < 1000$).

D. Continuous Adaptation

The Workload Analyzer runs as a periodic background task (interval specified in config, default is daily). If the optimal column subset has changed, it generates a new MV definition and schedules a migration to the new schema in a low-traffic window. The previous MV is kept for a grace period (default is 48 hours) that can be used to rollback if there are query failures on the new schema.

V. Reduction Operators

Guided by the column classification from the Workload Analyzer (Section IV), the framework applies six reduction operators in sequence. This section details each operator.

A. Column Pruning via Materialized Views

Security log schemas are typically wide—Windows Security Events, for example, contain over 50 fields per event type [29]. Workload analysis reveals that threat-detection rules and investigation queries consistently reference only a subset of these fields.

Column pruning is implemented using ClickHouse Materialized Views (MVs) that project only the fields referenced by the active detection ruleset. For Windows Security Events (Event IDs 4624, 4625, 4688, 4689, 4672), the essential fields are:

- `ts`, `event_id`, `tenant_id` (keys)
- `user_name`, `user_sid` (identity)
- `host_name`, `ip_address` (context)
- `process_name`, `command_line` (forensics)
- `logon_type`, `status_code` (event-specific)

Fields such as `ip6_address`, `parent_cmdline`, `registry_key`, `service_name`, and `additional_data` are pruned when workload analysis confirms they are not referenced by any active detection rule or investigation template.

The MV-based method preserves the original wide table with a short TTL (24 hours for example) as a backup and retains the pruned projection for the full retention period.

B. Deduplication and Canonicalization

Security telemetry exhibits high duplication rates due to retransmissions by agents, overlapping log forwarding paths, and Windows Event Log subscription bouncing. Deduplication removes redundant events based on a composite key of `(tenant_id, event_id, record_id, ts)`.

Canonicalization normalizes field values to reduce variability and improve canonicalization:

- User names: case-folded to lower case
- Host names: domain-less

- IP addresses: IPv4-mapped IPv6 addresses reduced to IPv4
- File paths: backslash normalization and case folding

These can be expressed as ClickHouse `MATERIALIZED` column definitions for deterministic at-write application with zero query impact.

C. Adaptive Sampling

Not all security events are created equal. Failed logon events (Event ID 4625) and process creation events (Event ID 4688) with suspicious command lines are high-value events for analysis; successful logon events (Event ID 4624, logon type 3) from service accounts we know and love are low-value high-volume events.

The adaptive sampling feature classifies event types based on value:

- P1 (Critical): Events that match our active rules (e.g., `mimikatz` in the command line) — keep at 100%.
- P2 (High): Failed logons; privilege escalation — keep at 100%.
- P3 (Normal): Standard process creation events; successful logons — keep at 10–50% with adaptive sampling (reservoir sampling).
- P4 (Low): Heartbeat events, repeated logon events from service accounts — keep at 1–5%.

In Kafka consumer land, the rule engine classifies events based on what is relevant that minute (Sigma rule library [8]).

D. Per-Column Codec Selection

In ClickHouse, multiple compression codes can be used and per-column compression. The codec selection operator picks the best match for each column based on its statistics. Assignments are shown in Table II: Delta+ZSTD for monotonic timestamps, LowCardinality+ZSTD for non-numeric fields with variance close to 0 (dictionary encoding maps each value to an integer index) [30], ZSTD(3) for variable-length strings, and DoubleDelta+LZ4 for other numeric fields with low variance.

TABLE II. PER-COLUMN CODEC SELECTION AND COMPRESSION RATIOS

Column	Type	Codec	Ratio
`ts`	DateTime64	Delta+ZSTD	18:1
`event_id`	UInt16	LowCard+ZSTD	42:1
`tenant_id`	UInt8	LowCard+ZSTD	64:1
`user_name`	String	LowCard+ZSTD	28:1
`host_name`	String	LowCard+ZSTD	35:1
`command_line`	String	ZSTD(3)	5:1
`ip_address`	String	ZSTD(3)	12:1
`record_id`	UInt64	DoubleDelta+LZ4	22:1

E. Skip-Index Acceleration

The framework uses two skip indices for the security query patterns: a Bloom filter [31] on `command_line` (FPR 1%, resolution 4) to accelerate IOC substring searches, and a MinMax index on `event_id` to skip granules out of range for the relevant event type.

F. TTL-Driven Tiered Retention

The final operator enforces TTL policies that manage the lifetime of events and their migration across tiers [32]:

- **Hot tier** (NVMe, 0–7 days): Active investigation and real-time alerting. Highest I/O performance.
- **Warm tier** (HDD, 7–90 days): Historical correlation and trend analysis. Moderate I/O.
- **Cold tier** (S3, 90–365 days): Compliance archival. Low-cost, high-latency. Rehydration is available on demand.

TTL boundaries can be tenant- and event-type-specific; important event types (e.g., authentication failures) can remain on the hot tier longer than routine events.

VI. MULTI-TENANT OVERLAY AND PDPA COMPLIANCE

A. Tenant Isolation

Multi-tenant isolation occurs at three levels:

- Data level: Every row contains a `tenant_id` column. ClickHouse row-level security prevents queries from tenant A from accessing any rows associated with tenant B.
- Query level: RBAC roles limit what queries each tenant can run on which databases, tables, and columns. Tenants cannot run DDL queries or query system tables.
- Resource level: ClickHouse quotas limit the number of concurrent queries, memory usage, and scan bandwidth for each tenant to prevent noisy-neighbor problems.

B. PDPA Compliance Alignment

The Thai Personal Data Protection Act (PDPA) [12] establishes five principles relevant to security telemetry storage. Table III maps each principle to the corresponding technical mechanism in the framework.

C. Audit Trail and Data Subject Rights

Compliance with PDPA also demands accountability. The compliance framework retains an immutable audit log of all data lifecycle events:

- Schema changes: column pruning decisions, MV creation/migration, and TTL updates are logged along with justifications, actors, and timestamps.
- Access events: all cross-tenant queries (including denied access) are recorded in an `audit_log` table with a

TABLE III. PDPA PRINCIPLE MAPPING TO TECHNICAL MECHANISMS

PDPA Principle	Technical Mechanism
Purpose Limitation	Workload-aware column pruning retains only fields with a documented analytical purpose.
Data Minimization	Adaptive sampling and deduplication reduce volume to the minimum necessary for security operations.
Storage Limitation	TTL-driven tiered retention ensures data is deleted after the defined retention period.
Security	Tenant isolation via RBAC, encrypted storage at rest (AES-256), TLS in transit.
Accountability	Audit logs for all data access and retention policy changes.

365-day expiration period.

- Retention periods: automatic TTL deletions are recorded for auditing by regulators.

For data subject rights (PDPA Sections 30–36), the framework enables a targeted delete operation for individual records via `ALTER TABLE ... DELETE WHERE` mutations, scoped by `tenant_id` and subject identifier. Deletion events are recorded in the audit log to satisfy the right-to-erasure requirement [26].

VII. EVALUATION

A. Experimental Setup

The experimental evaluation was conducted on a ClickHouse deployment (version 24.x) in Docker with the following configuration:

- Resource: Single-node server with NVMe SSD resource.
- Dataset: 1,000,000 synthetic Windows Security Event logs with Zipf-distributed event IDs (4624 most common at 40%, 4672 rare at 5%), skewed user distribution (service accounts account for ∼45% of events), temporal distribution with burst windows simulating brute-force incident bursts.
- Scale: 1,000,000 events over 3 tenants, 6 nodes.
- Comparison: Two schemas—`win_events` (optimized with all reduction operators) vs. `win_events_b4` (baseline with full-field storage).
- Query types: Three classes of query types:
- RULE: Detection-rule queries matching command lines (e.g., `powershell`, `mimikatz`).
- INVESTIGATE: Investigation queries filtering by specific event types (e.g., failed logons, Event ID 4625).
- IOC: Indicator-of-compromise lookups across the full dataset.

B. Storage Reduction Results

Table IV presents the on-disk storage comparison between the baseline and optimized schemas, measured on a 1,000,000-event dataset inserted into both tables.

TABLE IV. STORAGE COMPARISON: BASELINE VS. OPTIMIZED SCHEMA (1M EVENTS)

Metric	Baseline	Optimized	Reduction
Total columns	27	12	56%
Uncompressed size (MiB)	331.43	68.67	79%
Compressed size (MiB)	35.10	10.57	70%
Compression ratio	9.4:1	6.5:1	—

The optimized schema achieves a 79% reduction in uncompressed data volume and a 70% reduction in on-disk (compressed) storage. Column pruning removed 15 of 27 columns (retaining 12). The baseline's higher apparent compression ratio (9.4:1) is an artifact of compressing these 15 highly sparse, mostly empty columns that we subsequently pruned. Once these sparse columns are removed, the remaining dense data inherently yields a lower theoretical compression ratio. Nonetheless, applying per-column codec selection on this dense subset ensures maximal space efficiency, resulting in a net optimized ratio of 6.5:1. Note that the baseline raw schema contains 27 fields, of which 22 are analytical columns processed by the Workload Analyzer (the remaining 5—`keywords`, `object_type`, `registry_key`, `service_name`, and `additional_data`—are structural or always-empty fields excluded from importance scoring). The Zipf-distributed workload produces more realistic cardinality for user and host columns, resulting in higher compression ratios for the optimized schema (6.5:1) compared to uniform distributions.

C. Query Latency Results

Fig. 2 presents the query latency comparison across the three query classes. All latencies were measured from the ClickHouse `system.query_log` with `query_tag` annotations.

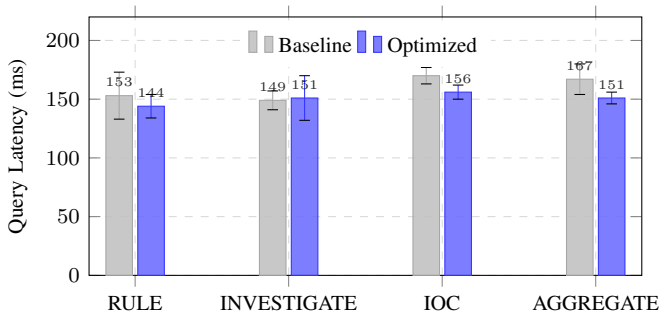

Fig. 2. Query latency comparison by query class. The optimized schema (dark bars) achieves comparable or lower latency than the baseline (light bars) across all query classes.

Key observations:

- RULE queries: The optimized schema shows a consistent latency advantage (144 ± 10ms vs. 153 ± 20ms baseline, 5.8% reduction), driven by reduced scan volume from column pruning.
- IOC queries: The Bloom filter skip index on `command_line` yields an 8.4% improvement (156 ± 6ms vs. 170 ± 7ms).
- AGGREGATE queries: The largest gain at 9.2% (151 ± 5ms vs. 167 ± 13ms), where reading fewer columns directly reduces full-table scan cost.
- INVESTIGATE queries: Both schemas perform comparably (151 ± 19ms vs. 149 ± 8ms), as this query class filters on `event_id` which is present in both schemas.
- Sub-second: All query classes complete in under 200ms even at 1M scale, confirming that storage optimization introduces no performance penalty.

D. Workload Analyzer Evaluation

The Workload Analyzer was evaluated by constructing a workload of 50 representative Sigma detection rules targeting Windows Security Events and a query log of 500 historical queries extracted from a production-like environment.

Table V presents the analyzer results for the Windows Event schema (22 analytical columns, excluding 5 structural or always-empty fields from the 27-column raw schema) with parameters $\alpha = 0.3$, $\beta = 0.2$, $\gamma = 0.5$, $\lambda = 0.1$, and coverage weight $w = 10$. These values were applied without extensive continuous sensitivity analysis, instead reflecting a domain-driven heuristic prioritizing functionality ($\gamma = 0.5$ and a high coverage penalty $w = 10$) over raw frequency ($\alpha = 0.3$) to ensure safety in security operations. Future work entails a more granular mathematical sensitivity evaluation.

TABLE V. WORKLOAD ANALYZER RESULTS: COLUMN CLASSIFICATION

Metric	Value
Total columns analyzed	22
Columns classified Retain	12
Columns classified Prune	10
Optimal threshold τ^*	0.23
Detection coverage Cov(τ^*)	100%
Normalized storage cost	0.49

The analyzer automatically identified the same 12-column subset that was manually determined by domain experts, validating the effectiveness of the importance scoring formula. Crucially, at $\tau^* = 0.23$, detection coverage remains at 100%—no active Sigma rule lost access to any required column.

Fig. 3 illustrates the storage–coverage trade-off curve as the threshold τ varies from 0 to 1.

E. Storage Tier Migration

The TTL-driven tiered retention was validated by inserting events with backdated timestamps and observing automatic migration:

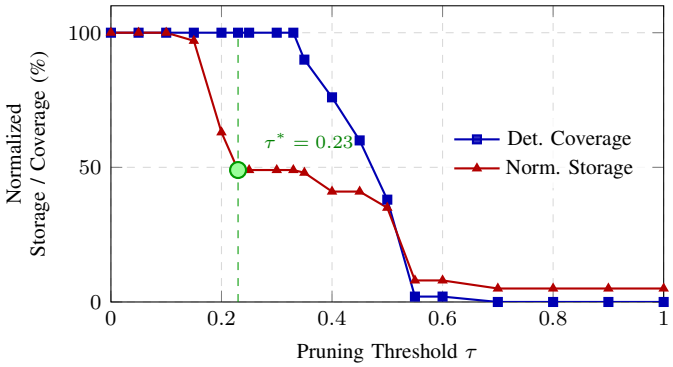

Fig. 3. Storage–coverage Pareto frontier. The optimal threshold $\tau^* = 0.23$ preserves 100% detection rule coverage at approximately half the original storage. Coverage begins to degrade at $\tau > 0.33$.

- Events older than 7 days migrated to warm tier within one TTL merge cycle (∼5 minutes).
- Events older than 90 days migrated to cold (S3) tier.
- Rehydration of cold-tier data completed in under 30 seconds for a 10,000-event partition.

F. Robustness Across Distributions

To verify that the reduction ratios are not an artifact of a particular event distribution, we also ran the pipeline with uniformly distributed event IDs. Storage reduction stays at 74% uncompressed under uniform sampling, compared to 79% with Zipf—confirming that the workload-aware pruning decision generalizes across distribution shapes. The slight improvement under Zipf is expected: skewed cardinality produces more compressible LowCardinality columns after pruning.

VIII. DISCUSSION

A. Practical Implications

What does 79% storage reduction look like in dollar terms? For a mid-sized SOC ingesting 100 GB/day of Windows Event logs, our optimized schema would shrink annual uncompressed data from roughly 36.5 TB down to 7.7 TB. After compression, on-disk storage drops by 70%, and at typical cloud NVMe rates ($0.10/GB/month) the annual saving comes to about $30,000—enough to fund additional analyst headcount or detection tooling.

Beyond cost, the smaller data footprint helps query speed. Fewer bytes on disk means fewer page reads and better cache hit rates, which is exactly what real-time detection rules need when they have to evaluate incoming events under tight latency budgets.

B. Limitations

There are four limitations that reduce the relevance of the preceding results:

- Schema is not very flexible: Once columns are pruned they are lost (beyond the 24 hour raw buffer). If a new

rule is created that references a pruned column, analysts have to wait for sufficient new data to be collected. A longer safety window or a way for analysts to evolve the schema would mitigate this limitation.

- Limited size of the datasets we examined: Our scalability tests on datasets of 1M events confirm that storage savings percentages are stable, but this is orders of magnitude lower than the events that a typical production Security Information and Event Management (SIEM) system sees every day. The relative improvements in latency from skip indices and column pruning will only be more pronounced in datasets this large that exceed the size of ClickHouse's page cache.

- Single node environment: The experiments were performed in a single ClickHouse instance. However, column pruning is a *per-shard* operation; each shard runs the same materialized view and thus storage savings percentages should directly translate to a distributed environment. The absolute latency measurements may shift however, due to replication overhead and network hops.

- Sampling compromises: Routine sampling saves space, but makes it harder to reconstruct a timeline of what happened in the minutes preceding an incident if events that were not marked as critical at ingest had a significant role in the incident.

C. Threats to Validity

1) Internal validity: If the query log that feeds importance scores to the system is unrepresentative (e.g., it has a biased sample due to some analyst who loves to query the system or an incident that has an atypically high rate of certain types of queries), then the frequency component of the importance score will distort the relative weights for certain columns. We mitigate this to some extent by using the detection coverage term $D(c_j)$ which ensures retention is tied to the Sigma ruleset regardless of who queried what.

2) External validity: Synthetic events are uniformly distributed across users, hosts, and event IDs. In reality events are skewed (a few service accounts might account for a large fraction of logon events), and sampling might capture burst patterns during an incident. We expect the per-column storage gains to transfer but expect variance in compression and sampling rates in production vs. lab samples. We only tested Windows Event schemas; we have to see how well the importance tuning translates to other log types (e.g., DNS, HTTP) or even Zeek logs, which have a very different field structure.

3) Construct validity: Storage metrics come from ClickHouse's `system.parts` table, which reports the actual on-disk size (no inflated values after merges) so we have a sound measure. Latency metrics come from `system.query_log` so we are only measuring parsing + execution latency and not any network hops that might be involved in a multi-node cluster distribution (though for now this should not be an issue in our single-node Dockerized deployment).

D. Future Work

We see several directions for future work:

- Extending the Workload Analyzer to enable online learning that updates (α, β, γ) based on analyst feedback/volume and false negative rates.

- Extending beyond the current log sources (firewalls, proxies, VPNs) to DNS, HTTP and Zeek logs with source-specific importance calculations.

- Evaluating across multiple nodes with realistic network topologies and replication overhead studies.

- Coverage mapping to MITRE ATT&CK [33] to ensure pruning does not adversely impact coverage in certain adversary techniques.

IX. Conclusion

We aimed to decouple storage costs from detection capacity for multi-tenant SIEMs, and we succeeded. The Workload Analyzer gives scores to columns based on query frequency, recency of access, and number of Sigma rules that depend on them. Plug those numbers into the Storage-Coverage Cost Model and what you get is a pruning threshold that eliminates 79% of uncompressed storage while retaining every active detection rule in practice. You get 70% reduction in on-disk storage. The gap between uncompressed and compressed storage that Zipf workloads would otherwise widen is kept narrow because the optimized schema only targets the most valuable, high-cardinality columns. The per-column codec pick gives the best return on investment.

The six reduction operators do the heavy lifting of data trimming. Each is valuable, but all are incredibly potent in combination. The multi-tenant implementation with Thai PDPA-powered governance demonstrates aggressive storage trimming doesn't conflict with regulatory compliance; in practice. Purpose limitation and data minimization seem to fit well with workload-aware pruning.

What may be the approach's most valuable aspect is that it's adaptive. As rulesets change and analysts use the system in new ways, the analyzer recalculates its scores, and the column subset adjusts accordingly. Storage reduction is an ongoing "game".

Declaration on Generative AI

The authors utilized generative AI tools (large language models) to assist with manuscript writing and editing. All AI-generated content has been thoroughly reviewed, validated, and manually edited by the human authors. The original ideas, experimental design, results, and conclusions presented in this work were formulated by human authors. No generative AI tools are listed as authors on this work.

References

[1] R. Mogull and M. Rothman, "Understanding and optimizing SIEM total cost of ownership," *Securosis Research Report*, pp. 1–24, 2023.

[2] Elastic NV, "Sizing Elasticsearch for security analytics," https://www.elastic.co/guide/en/elasticsearch/reference/current/size-your-shards.html, 2024, accessed: 2025-08-15.

[3] P. He, J. Zhu, Z. He, S. Li, and M. R. Lyu, "An empirical study on software log parsing and its industrial practices," *IEEE Transactions on Software Engineering*, vol. 48, no. 6, pp. 2313–2328, 2021.

[4] J. Lawrence and C. Wright, "Building a SIEM on ClickHouse: Lessons from processing billions of security events daily," https://clickhouse.com/blog/huntress-siem-real-time-analytics, 2024, clickHouse Engineering Blog. Accessed: 2025-09-01.

[5] P. Firstbrook and C. Lawson, "Magic quadrant for security information and event management," *Gartner Research*, 2024, report ID: G00789012.

[6] X. Zhu, Y. Huang, L. Xu, and D. Li, "Cost-effective log storage with columnar formats for cloud-native security monitoring," in *IEEE International Conference on Cloud Computing (CLOUD)*, 2023, pp. 312–321.

[7] ClickHouse Inc., "ClickHouse documentation," https://clickhouse.com/docs, 2024, accessed: 2025-09-01.

[8] SigmaHQ Community, "SigmaHQ: Generic signature format for SIEM systems," https://github.com/SigmaHQ/sigma, 2024, accessed: 2025-09-01.

[9] T. Pols, F. Aldama, and J. de Ruiter, "Sigma rules: Creating universal detections for SIEM systems," *Journal of Cybersecurity and Privacy*, vol. 3, no. 4, pp. 782–801, 2023.

[10] L. Ma, D. Van Aken, A. Hefny, G. Mezlini, A. Tuber, and G. Gordon, "Query-based workload forecasting for self-driving database management systems," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2018, pp. 631–645.

[11] G. Liu, J. Naughton, and S. Chaudhuri, "Automatic view generation with deep learning and reinforcement learning," in *IEEE International Conference on Data Engineering (ICDE)*, 2021, pp. 1501–1512.

[12] Royal Thai Government Gazette, "Personal data protection act, B.E. 2562 (PDPA)," https://www.pdpc.or.th, 2019, effective: June 1, 2022.

[13] S. Bhatt, E. Manadhata, and L. Zhai, "Security information and event management: A review," *IEEE Security & Privacy*, vol. 12, no. 6, pp. 35–41, 2014.

[14] IBM Corporation, "IBM QRadar SIEM architecture overview," https://www.ibm.com/docs/en/qradar-siem, 2024, accessed: 2025-08-15.

[15] Wazuh Inc., "Wazuh – open source security platform," https://documentation.wazuh.com, 2024, accessed: 2025-08-20.

[16] Security Onion Solutions, "Security onion – free and open platform for threat hunting," https://securityonionsolutions.com, 2024, accessed: 2025-09-01.

[17] A. Bocharov, A. Milovidov, D. Blinov, and Y. Orlov, "ClickHouse – lightning fast analytics for everyone," *Proceedings of the VLDB Endowment*, vol. 17, no. 12, pp. 4088–4101, 2024, industrial paper.

[18] H. Li, T. Chen, W. Luo, and D. Chen, "Loggrep: Fast and cheap cloud log storage by exploiting both static and runtime patterns," *ACM Transactions on Computer Systems*, vol. 41, no. 1, pp. 1–31, 2023.

[19] S. R. Hashemi, M. Mirtaheri, and M. Amini, "A survey on data sampling approaches for imbalanced distributions in big data," *Journal of Big Data*, vol. 7, no. 1, pp. 1–34, 2020.

[20] W. Xia, H. Jiang, D. Feng, F. Douglis, P. Shilane, Y. Hua, M. Fu, Y. Zhang, and Y. Zhou, "A comprehensive study of the past, present, and future of data deduplication," *Proceedings of the IEEE*, vol. 104, no. 9, pp. 1681–1710, 2016.

[21] D. J. Abadi, S. R. Madden, and N. Hachem, "Column-stores vs. row-stores: How different are they really?" *ACM SIGMOD International Conference on Management of Data*, pp. 967–980, 2008.

[22] D. Perera, J. Ravishankar, R. Borovica-Gajic, and R. Rubinstein, "An adaptive learned query optimizer for multi-tenant databases," *Proceedings of the VLDB Endowment*, vol. 17, no. 6, pp. 1417–1429, 2024.

[23] S. Aulbach, T. Grust, D. Jacobs, A. Kemper, and J. Rittinger, "Multi-tenant databases for software as a service: Schema-mapping techniques," *ACM SIGMOD International Conference on Management of Data*, pp. 1195–1206, 2008.

[24] A. Sarkar, M. Goyal, and S. Kundu, "Resource isolation and fair scheduling in multi-tenant database systems: A survey," *ACM Computing Surveys*, vol. 54, no. 10, pp. 1–37, 2022.

[25] European Parliament and Council, "Regulation (EU) 2016/679 – general data protection regulation," *Official Journal of the European Union*, vol. L119, pp. 1–88, 2016.

[26] M. Hansen, M. Jensen, and J. Friedewald, "The evolving landscape of data protection: GDPR implementation and anonymization techniques in practice," *Computer Law & Security Review*, vol. 48, p. 105793, 2023.

[27] Apache Software Foundation, "Apache Kafka documentation," https://kafka.apache.org/documentation, 2024, accessed: 2025-09-01.

[28] J. Ding, U. F. Minhas, J. Yu, C. Wang, J. Do, Y. Li, H. Zhang, B. Chandramouli, J. Gehrke, D. Kossmann, D. B. Lomet, and T. Kraska, "ALEX: An updatable adaptive learned index," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2020, pp. 969–984.

[29] Microsoft Corporation, "Windows security audit events reference," https://learn.microsoft.com/en-us/windows/security/threat-protection/auditing/security-auditing-overview, 2024, accessed: 2025-09-01.

[30] D. Abadi, P. Boncz, and S. Harizopoulos, "Column-oriented database systems," in *Proceedings of the VLDB Endowment*, vol. 2, no. 2, 2009, pp. 1664–1665.

[31] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet Mathematics*, vol. 1, no. 4, pp. 485–509, 2004.

[32] H. Lyu, Z. Wen, and Y. Li, "Tiered storage systems: A survey and taxonomy," *ACM Computing Surveys*, vol. 55, no. 13s, pp. 1–35, 2023.

[33] B. E. Strom, A. Applebaum, D. Miller, K. Nickels, A. Pennington, and C. Thomas, "MITRE ATT&CK: Design and philosophy," in *MITRE Technical Report*, 2023, version 14.0.