Unsupervised Detection of Long-Term Idle Periods in Large-Scale On-Premises Server Fleets

Ahmed Javed[1], **Haneul Yang**[1], **Zohaib Shahid**[2]
[1] OnitsAI Inc., South Korea
[2] Loughborough University London, United Kingdom
aj@onits.ai, hnyang@onits.ai, z.shahid@lboro.ac.uk

Abstract

As on-premises GPU and server fleets scale to meet AI workload demands, substantial hardware assets remain underutilized, resulting in prolonged, high-cost "idle" periods. Detecting these segments in large-scale environments is inherently difficult due to the absence of ground-truth labels and the high volatility of modern workloads. We propose an unsupervised pipeline for identifying long-term idle intervals in unlabeled multivariate utilization time series. By leveraging daily volatility vectors across CPU, memory, GPU, and storage metrics (`/data` space and `/root` space), our novel framework, the BGMM-HMM, employs a Bayesian Gaussian Mixture Model for state clustering followed by a Hidden Markov Model to enforce temporal consistency. Experiments on production server-fleet data show that the BGMM-HMM identifies underutilized assets $\approx 5\times$ more effectively than traditional rule-based baselines. Critically, ablation studies demonstrate that the HMM integration reduces spurious state-switching by $> 90\%$ compared to standalone clustering, providing the stable, contiguous intervals necessary for practical resource reclamation. Furthermore, robustness tests via synthetic noise injection confirm a 98.3% sensitivity to workload spikes. This framework provides a scalable and operationally stable tool for infrastructure optimization and ESG-aligned sustainable computing.

Introduction

Recent industry trend reports indicate that large organizations are increasingly repatriating workloads from the cloud to on-premises infrastructure or operating hybrid deployments, bringing the operational management of on-premises servers and hardware assets back to the forefront (Llorente 2024). In response to the surge in demand for generative AI and high-performance computing, many server-fleets have rapidly expanded their server and GPU capacities; however, prior studies have consistently reported that overall data center server utilization remains low and that idle resources constitute a substantial share (Hoßfeld 2025). Organizations are also forced to purchase additional equipment or maintain excessive capacity, leading to investment in avoidable capital and operational waste (Google 2025). Moreover, unused equipment and unnecessary power consumption increase CO_2 emissions, creating negative implications from an ESG perspective (Safari et al. 2025).

To address these challenges, deep-learning-based anomaly detection methods leveraging operational logs and utilization time series have been actively studied and used as a foundation for early failure detection and operational and capacity-management decision-making (Zamanzadeh Darban et al. 2024; Jia et al. 2025). However, a substantial portion of prior work focuses on short-lived event-like anomalies, such as failure precursors or abrupt spikes, and relatively few studies explicitly model idle intervals that persist over long periods in fully unsupervised settings (Zamanzadeh Darban et al. 2024; Jia et al. 2025). In contrast, the idle patterns considered in this paper are characterized by both low absolute utilization and low variability and are therefore easily regarded as normal under conventional anomaly criteria (Google 2025). Moreover, in real operational environments, labeling is practically infeasible, and the problem requires capturing continuous patterns over multiple days or weeks across server-fleets comprising thousands of machines. To the best of our knowledge, only a limited number of studies have reported the unsupervised detection of long, continuous idle intervals from unlabeled, large-scale on-premises operational data.

In this study, we propose a method that supports efficient IT asset management by automatically detecting idle periods from operational data in which labels are largely unavailable. We focus on a key characteristic of idle periods: rather than exhibiting low utilization levels, they tend to show extremely low variability in their utilization. Accordingly, we aggregated the hourly time series to a daily resolution and used the daily standard deviation (hereafter, volatility) of each utilization metric as the core signal. We employed a Bayesian Gaussian Mixture Model (BGMM) to infer clusters of days and subsequently applied a Hidden Markov Model (HMM) to impose temporal continuity on the daily decisions, thereby filtering transient noise. By cascading Bayesian state inference with temporal dynamics, the proposed BGMM–HMM model can robustly extract long-term idle periods without the need for labels.

The main contributions of this paper are as follows. (1) **Problem formulation**: We clearly define the problem of detecting idle periods in large-scale on-premises server fleets using only unlabeled operational time series and reformu-

late it as a daily-volatility-based state estimation problem. (2) **Modeling contribution**: We propose a BGMM–HMM ensemble model that takes the daily volatility vectors of each metric as input and combines unsupervised clustering with enhanced temporal continuity, thereby robustly deriving contiguous long-term idle periods.

The remainder of this paper is organized as follows. Section Related Works reviews related literature. Section Methods describes the proposed BGMM–HMM model and several other baselines for comparison. Section Experimental Settings presents the experimental setup of this study, and Section Results and Discussion presents the results and discusses the findings.

Related Works

Time-series Based Anomaly Detection

Research on anomaly detection in utilization time series data is extensive. However, comprehensive reviews of deep learning approaches indicate that existing methods primarily target short-term point anomalies or abrupt changes rather than sustained behavioral states (Zamanzadeh Darban et al. 2024). Similarly, (Jia et al. 2025) examined deep anomaly detection for multivariate time series and found a clear pattern: the field is dominated by a focus on short-term, event-like anomalies. These methods are typically designed to instantly flag failure precursors or workload spikes in complex systems, such as smart grids and networks (Jia et al. 2025).

The bulk of the literature focuses on short-term anomalies, usually assuming a semi-supervised context in which 'normal' data are already labeled. However, our problem is the opposite. Idle periods are long, stable stretches of low activity levels. In a conventional model looking for spikes, these periods appear perfectly safe. As they last for days, they require a different mindset from that of standard time-series anomaly detection.

Log-based Anomaly Detection

Deep learning for log anomaly detection is well established, with (Landauer et al. 2023) systematically reviewing approaches ranging from parsing to self-supervised learning to address this issue. However, log data (unstructured text/events) and their anomalies (errors/sequence deviations) differ sharply from our context. We focus on long-term idle patterns in numerical time series to find unused assets, making log-based techniques less relevant than time-series methodologies.

Idle Detection, Energy Saving, and Resource Utilization

Research on detecting idle computing resources, from servers to GPUs, is extensive. (Lu et al. 2024), for instance, developed a prediction platform for storage I/O to optimize power and performance, while other studies focused on hardware-level power gating (Wang and Zhang 2020) or data-center-wide metrics (Safari et al. 2025). However, these approaches predominantly target the micro-idle slots or aggregate efficiency. Few studies have addressed the unsupervised detection of long-term idleness across large server fleets. Our work fills that gap by utilizing daily aggregated time series data to identify sustained low-utilization patterns that standard methods miss.

Unsupervised Time-series Segmentation, GMM–HMM, and Clustering

Our BGMM–HMM approach is based on established research on unsupervised time-series segmentation. As noted by (Wang et al. 2024), HMMs are classical yet robust tools for modeling persistent latent states, whereas broader surveys (Aghabozorgi, Shirkhorshidi, and Wah 2015) and applied studies (Jacobs et al. 2018) have demonstrated the efficacy of combining GMMs and HMMs in domains such as speech recognition and manufacturing. However, these applications rarely address large-scale IT infrastructures. We diverge from existing studies by tailoring this architecture for server fleet management. Specifically, we apply a BGMM to daily volatility vectors to cluster low-variance states and then use an HMM to smooth these assignments into contiguous and long-term idle segments.

Summary of Related Works

In short, the literature tends to focus on extremes: either detecting fleeting, short-term anomalies or optimizing high-level energy metrics. General segmentation models rarely drill down into specific operational states, such as idleness. Our research targets this unlabeled middle ground. We used a specialized BGMM–HMM framework to uncover long-term idle periods across thousands of servers, capturing the 'quiet' waste that standard methods overlook.

Methods

Problem Definition and Formalization

Problem Statement Let $\mathcal{H}$ be a set of hosts and $\mathcal{M}$ be the set of metrics (CPU, memory, GPU, `/data` space, and `/root` space). For each host $h \in \mathcal{H}$ and day $d \in \{1, \ldots, D_h\}$, we represent the system state as a multivariate feature vector $\mathbf{x}_{h,d} \in \mathbb{R}^{|\mathcal{M}|}$ consisting of daily standard deviations derived from hourly utilization $u_{h,d,m,t}$:

$$\mathbf{x}_{h,d} = \left(\sigma(u_{h,d,m,1:24})\right)_{m \in \mathcal{M}}. \tag{1}$$

The task is defined as the unsupervised inference of a latent state sequence $\mathbf{z}_h = \{z_{h,1}, \ldots, z_{h,D_h}\}$, where each state $z_{h,d} \in \{I, A, N\}$ corresponds to *Idle*, *Ambiguous*, and *Non-Idle*, respectively.

Mathematical Formulation We formulate this as a two-stage hierarchical inference problem. First, continuous vectors $\mathbf{x}_{h,d}$ are mapped to discrete observations $o_{h,d} \in \{0, 1, 2\}$ via a Bayesian Gaussian Mixture Model (BGMM) based on the cluster responsibility $r_{h,d,k}$. Second, we model the temporal dependencies of these observations using a Hidden Markov Model (HMM) defined by the parameter set $\lambda = (\pi, \mathbf{A}, \mathbf{B})$. Our objective is to determine the optimal sequence $\hat{\mathbf{z}}_h$ by maximizing the posterior probability:

$$\hat{\mathbf{z}}_h = \underset{\mathbf{z}}{\operatorname{argmax}}\, P(\mathbf{z} \mid \mathbf{o}_h, \lambda). \tag{2}$$

Symbol	Description	Symbol	Description
$\mathcal{H}, h$	Host set, index	$\mathbf{x}_{h,d}$	Feature vector
$\mathcal{M}, m$	Metric set, index	$o_{h,d}$	Discrete obs.
d, t	Day, hour indices	$r_{h,d,k}$	Cluster resp.
u, σ	H. Util., D. Std. Dev.	$\mathbf{A}, \mathbf{B}$	Trans./Emis.
D_h, K	Days, Num. states	$\mathbf{z}, \hat{\mathbf{z}}$	Latent seq.
I, A, N	Idle, Amb., Non-Idle	λ, π	Model params

Table 1: Key notations used in the BGMM-HMM pipeline.

Data Preprocessing

To ensure consistency across heterogeneous hardware, we processed the raw hourly data as follows:

GPU Aggregation For GPU-enabled hosts, multi-GPU signals are aggregated into a single metric, GPU_{avg}, defined as the total utilization divided by the count of active GPU's (utilization > 0). If no GPU is active, the value is set to 0.

Imputation and Scaling Missing values were imputed as 0. To address varying hardware capacities (e.g., different RAM sizes or CPU cores), we apply min-max scaling to all metrics $m \in \mathcal{M}$, linearly transforming values to the range $[0, 1]$.This results in a normalized tensor $u_{h,d,m,t}$ that is ready for feature extraction.

Feature Engineering

Our exploratory analysis revealed that volatility, rather than absolute values, is the defining characteristic of idle periods. While non-idle workloads fluctuate substantially owing to user activity and diverse workloads, idle intervals exhibit negligible changes over time.

As defined in Section Problem Definition and Formalization, our input feature vector to the model is $\mathbf{x}_{h,d}$, which is constructed from the daily standard deviations of each metric. We selected the standard deviation after evaluating other features, such as the mean and first-order difference. Various tests verified that volatility provides superior separability, as shown in Figure 1, consistently remaining near zero during idle periods while spiking during non-idle periods, whereas the mean often failed to distinguish low-usage active workloads from true idleness.

Baseline Models

To evaluate the efficacy of the BGMM-HMM pipeline, we implement three heuristic and machine learning baselines. All models utilize identical daily volatility vectors $\mathbf{x}_{h,d}$; while clustering methods are post-hoc aligned via CPU-volatility mapping (Section Experimental Setup and State Alignment), remaining baselines employ model-specific thresholds.

Rule-based Model This baseline represents conventional enterprise IT asset management. A host h is labeled *Idle* on day d only if all five resource metrics fall below their respective 10^{th} percentile fleet-wide volatility thresholds (τ_m). To filter transient noise, we apply a 7-day minimum run-length constraint; idle sequences shorter than one week are reverted to *Non-Idle*.

K-means Clustering We employ K-means ($k = 3$) using the Lloyd algorithm and k-means++ initialization. This baseline partitions the feature space based on Euclidean distance, allowing us to quantify the benefits of probabilistic cluster-specific covariance estimation when handling the high-variance distributions typical of production workloads.

Isolation Forest (IF) To test an anomaly-centric hypothesis, we use an Isolation Forest ($n\_estimators = 200$) to identify the stablest-regime days. We define an idle threshold as the lower 10% quantile of the anomaly score distribution among inlier training samples. A day is classified as *Idle* only if it is not flagged as an outlier and its score falls below this learned threshold. While these baselines provide a robust comparative foundation, we recognize the potential for more sophisticated statistical thresholding strategies and plan to develop additional complex methods in future work.

Proposed Model: BGMM–HMM

Although the several baselines discussed earlier offer interpretability, they lack the nuance required to handle the messy reality of large-scale fleets. Real-world data is characterized by heterogeneous hardware, varying workloads, and fuzzy boundaries between idle and active states. To address this, we introduce a probabilistic framework that combines a Bayesian Gaussian Mixture Model (BGMM) with a Hidden Markov Model (HMM). This architecture allows us to learn local volatility patterns in an unsupervised manner, while enforcing temporal consistency across long time horizons. The workflow proceeds in two stages: (1) probabilistic clustering of daily volatility vectors, and (2) temporal smoothing.

Stage 1: Probabilistic Clustering (BGMM) We model the density of these standardized vectors $\tilde{\mathbf{x}}_n$ using a BGMM with $K = 3$ components. We chose the BGMM over the standard GMM to improve stability. By placing prior distributions over the mixture parameters, the model helps mitigate overfitting and better reflects the uncertainty in parameter estimation. To control the model complexity and ensure robust convergence, we restricted the model to a spherical covariance structure, $\Sigma_k = \sigma_k^2 \mathbf{I}$. Eq. 3 shows the resulting mixture density:

$$P(\tilde{\mathbf{x}}) = \sum_{k=0}^{K-1} \pi_k \mathcal{N}(\tilde{\mathbf{x}} \mid \boldsymbol{\mu}_k, \Sigma_k), \quad \sum_{k=0}^{K-1} \pi_k = 1, \quad \pi_k \geq 0. \tag{3}$$

Once trained, the BGMM provides an initial "soft" belief about the underlying state of each host-day via the posterior responsibilities:

$$r_{h,d,k} = P(C_{h,d} = k \mid \tilde{\mathbf{x}}_{h,d}), \quad k \in \{0, 1, 2\}. \tag{4}$$

We assign each host-day to a discrete cluster k based on the maximum posterior responsibility ($\text{argmax}_k\, r_{h,d,k}$). However, these clusters are mathematically distinct and semantically unlabeled. To map them to our operational regimes $\{I, A, N\}$, we rely on a domain-driven heuristic. We calculate the mean CPU volatility for each cluster and

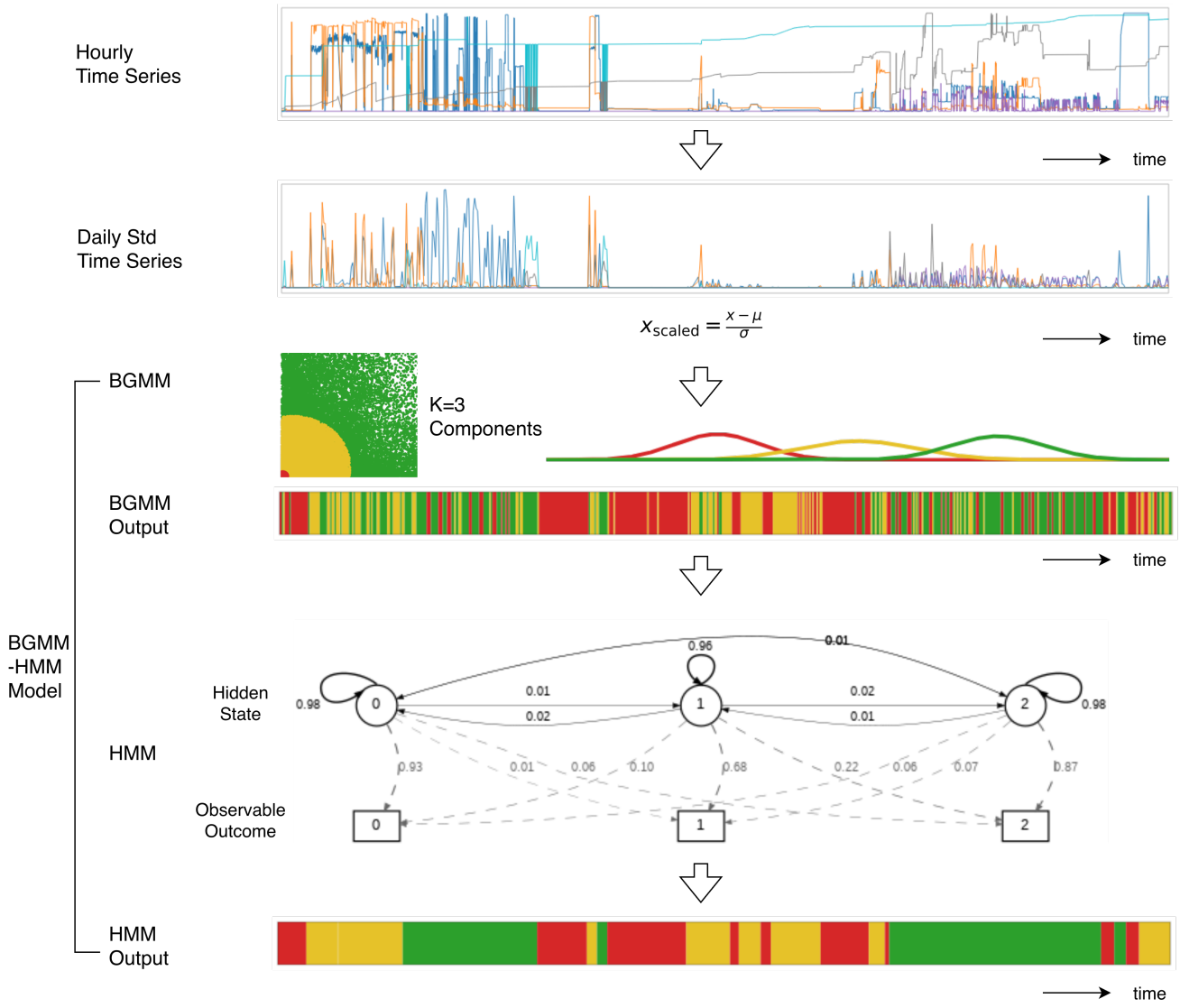

Figure 1: BGMM-HMM model steps with inferred state labels (Red: Idle, Yellow: Ambiguous, Green: Non-Idle).

rank them; the cluster with the lowest mean volatility is defined as *Idle*, the highest as *Non-Idle*, and the remainder as *Ambiguous*:

$$k_{\text{Idle}} = \underset{k}{\operatorname{argmin}}\, \bar{\mu}_{k,\text{CPU}}, \quad k_{\text{Non-Idle}} = \underset{k}{\operatorname{argmax}}\, \bar{\mu}_{k,\text{CPU}}. \quad (5)$$

We prioritized CPU volatility for this ranking because our empirical analysis identified it as the most discriminative signal for separating Idle, Ambiguous and Non-Idle regimes in our operational data.

Stage 2: Temporal Smoothing (HMM) Relying solely on BGMM yields overly fragmented results. A single day of slight volatility change can break a long idle streak, causing the system to "flicker" between states. To resolve this, we treat the sequence of BGMM cluster assignments as the noisy emissions of an underlying Hidden Markov Model. We trained a single fleet-wide HMM to learn a common transition matrix and emission model and then applied it to each host sequence. This helps capture the temporal persistence of the idle state and filter out transient noise. Learning a shared transition structure also reduces the number of parameters involved and improves estimation stability compared to fitting a per-host HMM model. Let $\mathbf{z}_{h,d} \in \{I, A, N\}$ be a latent state. We use the discrete cluster indices resulting from the BGMM, $o_{h,d} \in \{0, 1, 2\}$ as the observed sequence. We then define the HMM with the initial distribution $\boldsymbol{\pi}$, transition matrix $\mathbf{A}$, and emission matrix $\mathbf{B}$. Given the discrete observations $o_{h,d} \in \{0, 1, 2\}$, the joint probability is factorized as:

$$P(\mathbf{z}_h, \mathbf{o}_h) = \pi_{z_{h,1}} \prod_{d=2}^{D_h} A_{z_{h,d-1}, z_{h,d}} \prod_{d=1}^{D_h} B_{z_{h,d}, o_{h,d}}. \quad (6)$$

The emission model $B_{s,k}$ represents the probability of observing a specific BGMM cluster k given the true state s:

$$P(o_{h,d} = k \mid z_{h,d} = s) = B_{s,k}, k \in \{0,1,2\}, s \in \{0,1,2\}. \quad (7)$$

To ensure robust convergence, we initialized the HMM parameters using a data-driven approach. We derive the initial state distribution $\boldsymbol{\pi}$ from the empirical distribution of the first-day observations across the fleet. Similarly, we initialize the transition matrix $\mathbf{A}$ using the aggregated empirical transition counts ($o_{h,d} \to o_{h,d+1}$) across all hosts, applying light smoothing to manage sparsity.

To guide model convergence, we initialized the emission matrix $\mathbf{B}$ as a diagonally dominant matrix, setting the diagonal entries to approximately 0.90. This configuration encodes a strong prior belief that the BGMM cluster assignments, $o_{h,d}$, are reliable proxies for the true latent states (Idle, Ambiguous, and Non-Idle). By starting with these noisy assumptions, we enabled the EM algorithm to subsequently refine the specific label confusion probabilities from the data. We re-estimate $(\boldsymbol{\pi}, \mathbf{A}, \mathbf{B})$ using the Baum–Welch (EM) algorithm. Finally, we recover the most probable sequence of states $\hat{\mathbf{z}}_h$ using the Viterbi algorithm:

$$\hat{\mathbf{z}}_{h,1:D_h} = \underset{\mathbf{z}_{h,1:D_h}}{\operatorname{argmax}} P(\mathbf{z}_{h,1:D_h} \mid \mathbf{o}_{h,1:D_h}). \quad (8)$$

The resulting Viterbi-decoded sequence serves as the final Idle, Ambiguous, and Non-Idle labels produced by the BGMM–HMM model, effectively smoothing out short-term variability to reveal sustained idle periods. The complete steps are summarized in Algorithm 1.

Algorithm 1: The BGMM-HMM State Inference Model

Input: Hourly utilization $u_{h,d,m,t}$, Metrics Set $\mathcal{M}$
Output: State sequence $\hat{\mathbf{z}}_{h,1:D_h}$

Input Transformation
Compute daily std. dev. $\sigma_{h,d,m}$ from $u_{h,d,m,t}$ for each $m \in \mathcal{M}$.
Construct feature vector $\mathbf{x}_{h,d} = (\sigma_{h,d,m})_{m \in \mathcal{M}}$.
Standardize $\mathbf{x}_{h,d}$ to obtain $\tilde{\mathbf{x}}_{h,d}$ (zero mean, unit variance).
Stage 1: Latent State Discovery (BGMM)
Fit BGMM to $\{\tilde{\mathbf{x}}_{h,d}\}$ to learn parameters.
Calculate posteriors $r_{h,d,k} = P(C_{h,d} = k \mid \tilde{\mathbf{x}}_{h,d})$.
$o_{h,d} \leftarrow \operatorname{argmax}_k r_{h,d,k}$.
Assign labels based on cluster CPU means $\bar{\mu}_{k,\text{CPU}}$.
Stage 2: Temporal Smoothing (HMM)
Set π and $\mathbf{A}$ from empirical frequencies of $\mathbf{o}_h$.
Initialize $\mathbf{B}$ as diagonally dominant (e.g., $B_{ii} \approx 0.9$).
Refine $\lambda = (\pi, \mathbf{A}, \mathbf{B})$ via Baum-Welch on $\mathbf{o}_h$.
$\hat{\mathbf{z}}_{h,1:D_h} \leftarrow \text{Viterbi}(\mathbf{o}_h, \lambda)$.
return $\hat{\mathbf{z}}_h$ mapped to {**Idle**, **Ambiguous**, **Non-Idle**}

Experimental Settings

Dataset and Selection Criteria

Our study utilizes operational telemetry from a fleet of approximately 6,500 on-premises assets. We identified a high-quality subset of 509 production hosts based on two criteria: (1) **Continuity**, requiring $\geq$ 90 days of uninterrupted telemetry to ensure the HMM reliably estimates state transition probabilities ($A_{i,j}$); and (2) **Metric Diversity**, incorporating a mix of general-purpose and memory-intensive hosts to test the BGMM against varying baseline noise levels (σ).

The analysis unit is the host–day pair, derived from hourly utilization (CPU, memory, GPU, and two disk volumes) aggregated into daily volatility vectors, $\mathbf{x}_{h,d}$. To ensure generalization and prevent information leakage, we employed a host-stratified split, partitioning the 509 hosts into disjoint groups: 80% (407 hosts) for training and 20% (102 hosts) as a holdout set. This ensures the model is validated exclusively on hardware configurations distinct from those used during parameter estimation.

Metric	Formulation
Idle Usage Stats	$\mu_{h,m} = \frac{1}{\lvert I_h \rvert} \sum_{d \in I_h} u_{h,d,m}$
State Change Count	$SC_h = \sum_{d=2}^{D_h} \mathbb{I}(z_{h,d} \neq z_{h,d-1})$
Idle Coverage Ratio	$Cov_h = \frac{\lvert I_h \rvert}{D_h}$

Table 2: Host-level evaluation metrics. Here, I_h denotes the set of idle days for host h.

Evaluation Metrics

Quantitative Evaluation In fully unsupervised settings, quantitative evaluation is inherently challenging because of the lack of ground-truth labels (Campos et al. 2016). Unlike prior studies that rely on synthetic benchmarks or proxy datasets, we operated in an environment without complete labels (Google 2025). Consequently, instead of prioritizing traditional metrics such as Accuracy or ROC-AUC, we evaluate the detected segments using three operational proxy metrics that capture their characteristics and operational utility (Table 2). **Idle Segment Statistics** (μ, s) verify that the detected segments exhibit low variability. The **State Change Count** (SC_h) quantifies temporal stability, favoring coherent segments over fragmentation. The **Idle Coverage Ratio** (Cov_h) assesses the trade-off between model conservativeness and permissiveness. All metrics are reported with 95% bootstrap confidence intervals.

Section Quantitative Performance and Stability presents the detailed results for each metric and discusses the relative performance of the models.

Qualitative Evaluation To complement the quantitative metrics, we analyzed eight representative hosts with diverse usage patterns, ranging from clear-cut idle spans to complex, blended activity. We highlight two characteristic cases in Figure 2, where inferred regimes are mapped as color-coded indicator bars beneath the hourly utilization plots. This vi-

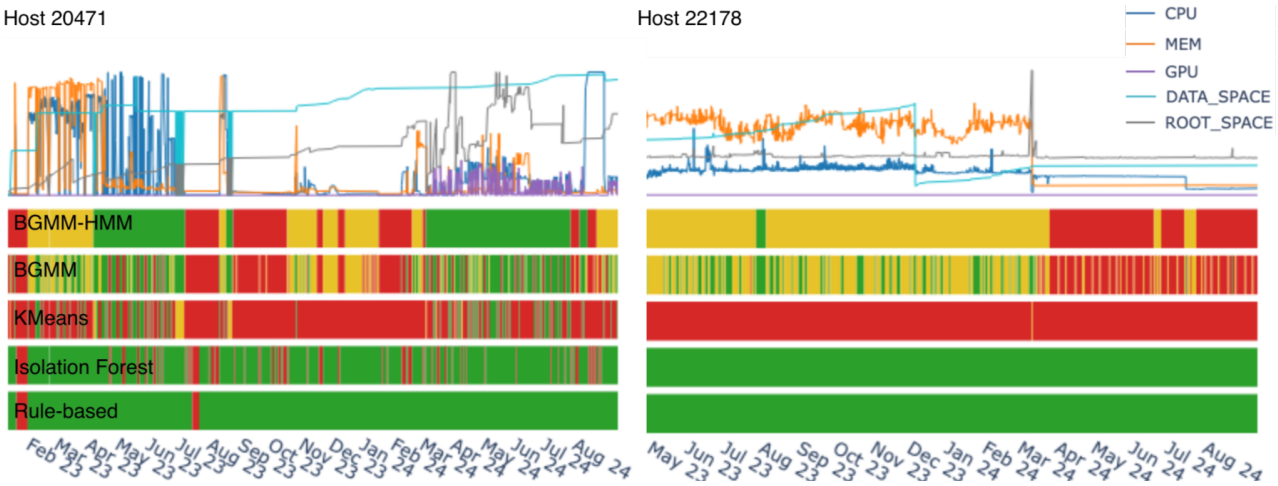

Figure 2: Qualitative evaluation. Hourly utilization time series with daily state labels inferred by BGMM–HMM and various baselines (Red: Idle, Yellow: Ambiguous, Green: Non-Idle).

sualization allows for a direct evaluation of model-specific performance based on the continuity and temporal alignment of the detected idle segments. Section Qualitative Analysis presents the detailed discussion of few selected cases.

Implementation and Reproducibility

The experimental framework was developed in Python using `scikit-learn` and `hmmlearn`. To ensure that the reported results reflect genuine generalization, we locked a single configuration based on training-set pilots prior to holdout evaluation.

Preprocessing and Split Features were aggregated into daily windows requiring a minimum of 6 observations for statistical validity. We employed a host-stratified split (80% train, 20% holdout), ensuring specific production hosts with known workload patterns were included in the holdout set for qualitative validation.

BGMM-HMM Configuration The BGMM utilized $K = 3$ components with a spherical covariance structure and a Dirichlet process prior ($\gamma = 0.1$). The HMM was initialized with transition and start probabilities derived from BGMM cluster counts, incorporating a Laplacian smoothing factor of 10^{-2}. To reflect the initial clustering fidelity, the emission matrix used a strong diagonal initialization of 0.90. The training utilized the Baum-Welch algorithm for 50 iterations with a convergence tolerance of 10^{-4}, updating the start, transition, and emission parameters.

Baseline Parameters The rule-based model utilized 10^{th} percentile fleet-wide thresholds with a 7-day minimum run-length constraint. The Isolation Forest ($n\_estimators = 200$) utilized a 10% inlier-quantile threshold for idle labeling, while K-means was implemented with $k = 3$ and 300 maximum iterations. Final daily predictions for all models were merged back to hourly telemetry using standardized date keys to maintain temporal alignment across the fleet.

Results and Discussion

Experimental Setup and State Alignment

To facilitate a rigorous comparison, all models were evaluated using an identical holdout set and daily multivariate feature vectors. We performed a post-hoc alignment for the K-means, Isolation Forest, and standalone BGMM baselines, as these models do not natively generate semantic operational labels. For the BGMM and K-means, we mapped clusters to *Idle, Ambiguous,* and *Non-Idle* states based on the mean daily CPU standard deviation (σ) within each group similar to BGMM-HMM and based on Eq. 5.

The Isolation Forest required a more nuanced alignment. Because IF outputs anomaly scores rather than state labels, we defined an idle threshold using the lower 10% quantile of the anomaly score distribution among the "normal" samples in the training set. A day was subsequently classified as *Idle* only if it was not flagged as an outlier and its anomaly score fell below this threshold. This approach ensured that the IF baseline was specifically tuned to identify only the most stable regimes, paralleling the intent of our rule-based baseline.

Quantitative Performance and Stability

The results in Tables 3 and 4 reveal a clear trade-off between conservative thresholds and our probabilistic framework. While the rule-based approach is stable, it is excessively restrictive (8.6% idle ratio). Conversely, K-means and Isolation Forest identify higher idle volumes (91.1% and 19.1%) but suffer from extreme temporal instability, averaging over 50 state changes per host. This frequent "flickering" suggests that point-in-time models fail to distinguish sustained idleness from transient noise.

The HMM layer addresses this by enforcing temporal consistency via Viterbi decoding. While the standalone BGMM identifies 44.0% idle volume, it is characterized by 128.74 changes per host. The **BGMM-HMM** reduces this switching by $> 90\%$ (to 12.17 changes/host) without sac-

Model	Hosts	CPU σ	Mem σ	GPU σ	Data σ	Root σ
BGMM-HMM (Ours)	87	$\mathbf{2.32 \pm 0.92}$	$\mathbf{2.02 \pm 0.70}$	$\mathbf{0.67 \pm 0.52}$	$\mathbf{0.30 \pm 0.32}$	$\mathbf{0.54 \pm 0.29}$
BGMM (Standalone)	91	0.44 ± 0.10	0.32 ± 0.07	0.01 ± 0.01	0.01 ± 0.01	0.09 ± 0.04
K-means ($k = 3$)	102	6.25 ± 1.32	5.85 ± 1.18	1.50 ± 0.56	0.63 ± 0.25	1.07 ± 0.34
Isolation Forest	84	0.29 ± 0.08	0.18 ± 0.05	0 ± 0	0 ± 0	0 ± 0
Rule-based	34	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0

Table 3: Mean within-idle variability (σ) across metrics. Values represent host-level means $\pm$ symmetric 95% bootstrap CI. Units scaled by 10^{-3} for readability.

Model	Hosts	Changes/Host	Ratio/Host	Idle Days	Total Days	Idle Ratio
BGMM-HMM (Ours)	102	$\mathbf{12.17 \pm 2.08}$	$\mathbf{0.463 \pm 0.077}$	**21,327**	**52,345**	**0.407**
BGMM (Standalone)	102	128.74 ± 18.69	0.492 ± 0.068	23,009	52,345	0.440
K-means ($k = 3$)	102	52.15 ± 15.33	0.925 ± 0.022	47,691	52,345	0.911
Isolation Forest	102	62.98 ± 11.82	0.228 ± 0.054	9,999	52,345	0.191
Rule-based	102	4.01 ± 1.55	0.110 ± 0.047	4,492	52,345	0.086

Table 4: Comparison of state stability and fleet-wide idle coverage. Values represent host-level means $\pm$ symmetric 95% bootstrap CI and fleet-wide totals.

Window Size	Idle Coverage	Switch Rate	Runs $\geq$7d
6 Hours	0.550	0.083	4.324
1 Day (Ours)	**0.480**	**0.019**	**3.343**
1 Week	0.402	0.004	1.402

Table 5: Ablation study on window size granularity.

Feature Subset	Silhouette Score	Cohen's κ
Proposed (5-Metric)	0.072	1.000
CPU, MEM, GPU	0.069	0.830
CPU, MEM	**0.083**	**0.742**
CPU, DATA	0.038	0.667
MEM, DATA	0.069	0.612

Table 6: Ablation study on feature subset impact.

rificing coverage (40.7%). Crucially, Table 3 confirms that this stability does not compromise purity; the model maintains low variability across multiple resource metrics (CPU $\sigma = 2.32 \times 10^{-3}$), ensuring segments represent stable operational states. This high-fidelity detection enables reliable asset decommissioning and measurable energy reductions without risking the continuity of critical IT services or causing operational disruption.

Qualitative Analysis

A qualitative examination (Figure 2) of specific fleet members further underscores these structural differences. On **Host 20471**, both the standalone BGMM and K-means models generate highly fragmented labels, frequently oscillating between states at the boundaries of low-variance intervals. Conversely, **Host 22178** demonstrates the risk of over-conservativeness inherent in simpler methods; both the rule-based model and Isolation Forest fail to recognize a sustained six-month idle span, maintaining a *Non-Idle* label despite the prolonged absence of activity. The BGMM-HMM successfully navigates these extremes, identifying stable, contiguous blocks of time that are sufficiently coherent for resource reclamation tasks.

Robustness and Ablations

To verify that the identified states are not artifacts of statistical noise, we performed a stress test by injecting high-variance noise $\mathcal{N}(\mu_{act}, \Sigma_{act})$ into 10% of the identified idle segments. The pipeline correctly reclassified 98.3% of these intervals as *Non-Idle*, confirming high sensitivity to workload spikes.

Finally, ablation studies on window granularity (Table 5) and feature subsets (Table 6) justify our selection of a 1-day window as an optimal trade-off between noise reduction and temporal precision. While 6-hour windows increase granularity, they are overly susceptible to transient fluctuations; increasing the state-switching rate by $\approx 5\times$ (0.083 vs. 0.019). Conversely, 1-week windows over-smooth critical transitions. Feature analysis via Cohen's κ confirms that while CPU and Memory drive cluster separability, the inclusion of storage and GPU metrics is necessary to prevent the misclassification of data-intensive background tasks.

Limitations and Operational Integration

While our model demonstrates high precision and stability, it faces several primary constraints.

Temporal Dependencies The HMM assumes the Markov property, which may not capture multi-week dependencies or complex seasonal workload patterns.

Concept Drift As our model uses stationary transition probabilities, it may require periodic rolling-window retraining to remain robust against shifting workload dynamics.

Granularity Relying on daily σ (Eq. 1) means sub-hour micro-bursts might be misclassified as *Idle*.

Operational Integration Deploying at scale requires addressing higher computational overhead compared to rule-based systems and integrating with existing IT orchestration for subjective validation.

Ground Truth As an unsupervised approach, we lack direct ground-truth; future work could incorporate administrator feedback in a human-in-the-loop framework to refine state boundaries.

Practical Implications for Operations and ESG

With data center electricity demand reaching $\approx$1.5% of the global total (International Energy Agency 2025), this work offers immediate value for sustainable operations.

Resource Reclamation Unlike failure-centric monitoring, the BGMM–HMM transforms passive telemetry into active capacity management. By identifying long-term idle candidates, administrators can safely downscale or reallocate underutilized assets, thereby directly reducing fleet-wide operating costs.

ESG Compliance Mitigating the energy waste of "comatose" servers is a high-impact strategy for carbon reduction (Meisner et al. 2009). This framework serves as a proactive tool for Green IT, allowing organizations to quantify and reduce their Scope 2 emissions in alignment with ESG reporting standards (ITU and The World Bank 2023).

Conclusion

This paper addresses the identification of sustained idle regimes in large-scale server fleets by formulating the task as a latent regime recovery problem rather than a momentary threshold violation. We introduced a probabilistic BGMM–HMM framework that bridges the gap between raw telemetry and actionable infrastructure insights, providing a novel, unsupervised solution for fleet-wide segmentation.

Experimental results highlight the structural advantages of this approach over traditional heuristics. While standalone clustering suffers from extreme temporal "flickering", the BGMM–HMM leverages multivariate volatility and Viterbi-driven smoothing to recover contiguous, reliable idle segments. Specifically, the model achieved a $> 90\%$ reduction in state-switching compared to standalone BGMM, ensuring the temporal persistence required for automated resource reclamation. This confirms that modeling transition dynamics is essential for distinguishing genuine low-utilization regimes from transient workload noise. Furthermore, the model's unsupervised nature allows it to scale across heterogeneous hardware environments without the need for manual, site-specific threshold tuning.

Despite these findings, several constraints remain. The Markovian assumption may not capture multi-week seasonal cycles, and the lack of ground truth necessitated reliance on proxy metrics. Future work could incorporate higher-order dependencies or human-in-the-loop refinement to calibrate state boundaries further. Ultimately, this framework provides immediate operational value for enterprise ESG strategies, enabling the decommissioning of underutilized assets to reduce Scope 2 emissions. This work advocates for a paradigm shift toward probabilistic, self-optimizing lifecycle management in sustainable computing infrastructure.

References

Aghabozorgi, S.; Shirkhorshidi, A. S.; and Wah, T. Y. 2015. Time-series Clustering – A Decade Review. *Information Systems*, 53: 16–38.

Campos, G. O.; et al. 2016. On the Evaluation of Unsupervised Outlier Detection: Measures, Datasets, and an Empirical Study. *Data Mining and Knowledge Discovery*, 30(4): 891–927.

Google. 2025. Gemini (Version 2.5 Pro). https://gemini.google.com/. Accessed: Dec. 16, 2025.

Hoßfeld, T. 2025. Energy Use in Data Centers: Current Figures and Trends. *VDE ITG News*.

International Energy Agency. 2025. Energy and AI: World Energy Outlook Special Report. Technical report, International Energy Agency, Paris, France. Accessed: 2026-01-14.

ITU and The World Bank. 2023. *Green Data Centers: Towards a Sustainable Digital Transformation – A Practitioner's Guide*. ITU and The World Bank.

Jacobs, W. R.; et al. 2018. Gas Turbine Engine Condition Monitoring Using Gaussian Mixture and Hidden Markov Models. *International Journal of Prognostics and Health Management*, 9.

Jia, X.; et al. 2025. Deep Anomaly Detection for Time Series: A Survey. *Computer Science Review*, 58: 100787.

Landauer, M.; et al. 2023. Deep Learning for Anomaly Detection in Log Data: A Survey. *Machine Learning with Applications*, 12: 1–19.

Llorente, I. M. 2024. Cloud Repatriation on the Rise: 83% of CIOs Plan Workload Shifts in 2024. EE Times Europe.

Lu, Y.-Y.; et al. 2024. A Stable Idle Time Detection Platform for Real I/O Workloads. *ACM Transactions on Architecture and Code Optimization*, 21(4).

Meisner, D.; et al. 2009. PowerNap: Eliminating Server Idle Power. In *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 205–216.

Safari, A.; et al. 2025. A Systematic Review of Energy Efficiency Metrics for Optimizing Cloud Data Center Operations and Management. *Electronics*, 14(11).

Wang, C.; et al. 2024. Unsupervised Time Series Segmentation: A Survey on Recent Advances. *Computers, Materials & Continua*, 80(2): 2657–2673.

Wang, X.; and Zhang, W. 2020. GPGPU Functional Units Power Gating for Leakage Energy Reduction. *Journal of Computer Science and Engineering*, 14(3): 102–111.

Zamanzadeh Darban, Z.; et al. 2024. Deep Learning for Time Series Anomaly Detection: A Survey. *ACM Computing Surveys*, 57(1).