Advancing Workload Management with Foundational Models: Challenges in Time Series Similarity and Interpretability

Tiemo Bang, Sergiy Matusevych, Yuanyuan Tian*
Microsoft Gray Systems Lab

Ioannis Roumpos†, Georgia Christofidi† ‡, Thaleia Dimitra Doudali†
† IMDEA Software Institute,
‡ Universidad Politécnica de Madrid

Abstract

Workload management (WLM) is essential for cloud providers to balance performance, reliability, and cost. Many WLM tasks rely on understanding workload behavior through time series similarity analysis, but traditional approaches face scalability challenges due to manual feature engineering and computational overheads. Foundational time series models promise to address these limitations by learning reusable representations with minimal supervision. This paper evaluates their practical potential for WLM through a focused case study on time series similarity. We present concrete use cases, characterize a real-world query arrival dataset from Microsoft Fabric Warehouse, and compare the foundational model *MOMENT* against conventional similarity methods. Our findings reveal that while foundational models offer computational efficiency, they produce overly generalized similarities with limited interpretability compared to hand-engineered features. We identify key challenges and research directions needed to make foundational models practical for workload management.

CCS Concepts

• **Information systems** → **Data management systems**; • **Computing methodologies** → *Machine learning*; • **Computer systems organization** → Cloud computing.

Keywords

Workload Management, Workload Analysis, Workload Embeddings, Query Embeddings, Time Series, Microsoft Fabric, Query Dataset

1 Introduction

Cloud service providers face the challenge of balancing customer demands (adequate response times, reliable service availability, etc.) with operational costs. In cloud environments, resources such as computing power, storage, and bandwidth must be dynamically allocated to meet varying customer needs. Efficient workload management (WLM) is essential to avoid resource wastage during low demand periods and resource shortages during peak times. Examples of workload optimization include resource oversubscription, which allocates more virtual resources than the available physical resources based on predictable time-series workload patterns, and scheduling database maintenance tasks during periods of low query load by learning customer query submission behavior (more examples in Section 2.1).

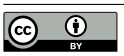
Workload management improves system performance and customer satisfaction by optimizing for workload characteristics. At its core, this requires understanding how workloads behave over time—a process we refer to as workload intelligence. Because system metrics like query arrival rates, CPU usage, and cache hit ratios evolve as time series, effective workload management depends on analyzing and comparing these signals. For example, time series of query arrivals can reveal predictable low-traffic periods, for scheduling maintenance tasks like index rebuilds or statistics updates. Similarly, comparing system performance under current query arrival patterns to historical observations helps distinguish between typical and anomalous behavior. Many workload intelligence tasks aim to identify recurring patterns, detect anomalies, or match new workloads to known behaviors. Assessing time series similarity is therefore essential for workload management.

A core challenge for cloud providers is the superlinear growth of workload management overhead with the customer base. The overhead of traditional time series analysis methods underlying workload management grows with workload volume and diversity. On one hand, automatic unsupervised methods are computationally expensive—for example, workload pattern matching using Dynamic Time Warping (DTW) can have quadratic complexity in both the number and length of time series [24]. On the other hand, supervised approaches require human feature engineering and careful tuning of algorithms for each new workload pattern. At the hyperscale of Microsoft Azure and other clouds, these overheads create a significant burden and slow the advancement of sophisticated workload management.

Foundational time series models promise significant advantages for workload management compared to traditional approaches. The core idea is that extensive pretraining allows foundational models to capture semantically meaningful features in compact vectors. In this way, foundational models promise to adapt to various data characteristics and support various tasks with little or no human supervision. As a result, foundational time series models represent an exciting direction for improving the efficiency and capabilities of workload management in clouds like Azure.

This paper calls for continued research to make foundational time series models practical for cloud workload management. We show that currently, limited generalization across data characteristics and interpretability of foundational models hinder their deployment in production. We therefore motivate further research on foundational models as follows. In Section 2, we outline practical workload management use cases based on time

*Corresponding author's email: tiemobang@microsoft.com



MIDAS '25, Berlin, Germany

ACM ISBN 979-8-4007-1960-8/2025/06
https://doi.org/10.1145/3737412.3743491

series similarity, offering concrete scenarios for follow-up research. In Section 3, we characterize our dataset of real-world query arrival rates on Fabric Data Warehouse [18]. We plan to publish our dataset in the Azure Public Repository[1] to support future research. In Section 4, our case studies on time series similarity and clustering highlight the limitations of foundational models in delivering concise and interpretable similarity measures. Finally, in Section 5, we reflect on the lessons learned and the open challenges for foundational models to serve as versatile, efficient, and reliable tools for workload management at scale.

Workloads tell stories,
Embeddings speak in riddles—
Trust needs clear answers.

2 Background

2.1 Time Series Similarity for Workload Management

Time series similarity is a fundamental tool in workload management. Many tasks across performance optimization, system planning, and operational efficiency depend on understanding how workloads behave and evolve over time. These behaviors are captured in time series—such as query arrival rates, resource consumption, or system metrics—either as real observations or as predictions. We outline key similarity-driven tasks and their corresponding use cases below:

- **Identifying Representative Patterns.** Grouping workloads by similarity enables systems to select representative patterns for each cluster. These representatives support the creation of new benchmarks and offline optimization such as configuration tuning via frameworks like MLOS [2, 13]. Grouping by similarity can also help with model selection, such as in scenarios with many coexisting models for workload prediction [5–9, 21].
- **Tracking Change Over Time.** Comparing time series segments over time supports trend analysis and evolution tracking. This is useful for assessing workload growth, refining system features, or triggering reoptimization pipelines when workloads diverge from previously optimized patterns [12, 26].
- **Outlier Detection.** Similarity can reveal unusual workload patterns or queries that deviate significantly from known behavior. Such outliers may indicate potential system incidents or emerging workload classes—informing both operational responses and long-term planning. Work like Griffon [25] illustrates data-driven approaches to detecting job anomalies, while open challenges remain in industry [19].
- **Pattern Matching for Scheduling.** Similarity also supports alignment or anti-alignment of workloads for resource allocation and scheduling of maintenance tasks. For example, workloads with non-overlapping demand patterns can be colocated to improve resource utilization. Likewise, predictable low-traffic periods can guide the timing of background operations such as index rebuilds or statistics refreshes [5, 21, 22].

These similarity tasks underpin both workload intelligence and optimization—characterizing workloads and adapting systems to them. While prediction methods play a complementary role by enabling forward-looking analysis [11, 16], this paper focuses exclusively on the practical analysis of time series similarity for both observed and predicted data, without addressing prediction models or optimization algorithms directly.

2.2 Foundational Time Series Models

Foundation models for time series offer a powerful new paradigm for workload management by enabling zero-shot, few-shot, and multi-task capabilities across forecasting, anomaly detection, classification, and more. Trained on diverse datasets, these models capture general temporal patterns such as seasonality, spikes, or trends, making them highly adaptable to new domains with minimal fine-tuning [10, 17]. They reduce the manual effort required for building task-specific models and eliminate feature engineering steps, thereby streamlining deployment pipelines and accelerating innovation in dynamic workload environments.

Time-series-native and LLM-based models. LLM-based models such as *Chronos* [3] treat time series as token sequences and apply language modeling techniques, benefiting from the scalability and pretraining infrastructure developed for NLP. However, they often require discretization, which may compromise numerical precision for tasks like forecasting. In contrast, time-series-native models like *MOMENT* [10] operate directly on raw numerical time series and are designed with temporal structures in mind. As a result, they tend to be more compact, interpretable, and better suited for continuous-valued signals, making them particularly effective for practical workload management scenarios.

Exemplary focus on MOMENT. This study focuses on MOMENT due to its promise for workload telemetry and system signals. MOMENT is pretrained on a broad collection of public datasets and designed to generalize across diverse domains with minimal supervision, making it ideal for handling the noisy, heterogeneous, and high-dimensional time series typical in cloud workloads [10]. Its lightweight architecture supports efficient inference and fine-tuning, and its strong few-shot performance aligns well with the practical demands of systems that observe ever-evolving workloads with limited labeled data. These properties make MOMENT a promising candidate for advancing similarity-based workload analysis in both research and production settings.

3 Real-World Query Arrival Dataset

We present a real-world dataset of query arrivals for our case study and to support future research.

3.1 Dataset and Basic Characteristics

Our dataset captures telemetry data on queries submitted by customers to the Microsoft Fabric Warehouse [18]. The dataset covers January 2025 and includes a sample of customers from one popular cloud region. It records the number of submitted queries in 15-minute intervals, per customer and query type: *insert, delete, update, aggregate, join*, and *simple select*[2]. In total, the dataset comprises over 1.4 million query submissions.

Before utilizing this dataset to study time series similarities, we need to analyze its basic characteristics: *What is the range of query*

[1] Dataset at https://aka.ms/fabric-dw-query-arrivals-dataset

[2] We classify query types by the presence of the corresponding operators with decreasing precedence, e.g., aggregates identify aggregate queries and among the remaining queries joins identify join queries.

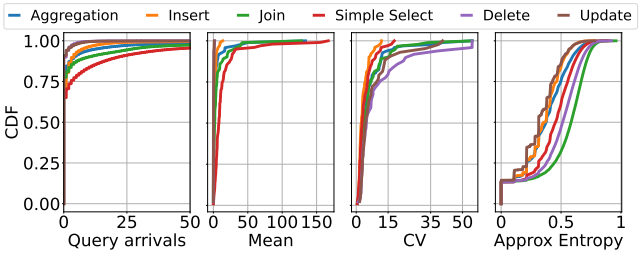


Figure 1: CDF of the absolute number of query arrivals, the mean, coefficient of variation, and approximate entropy.

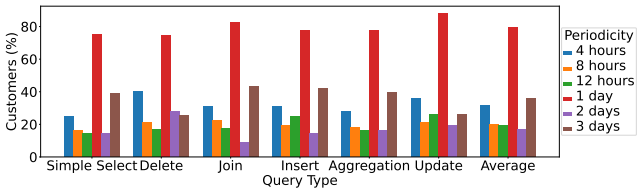


Figure 2: Number of customers with periodic patterns.

counts that arrive at a given time, depending on the type of query? Do they exhibit regular patterns or behave randomly? If they exhibit regular patterns, are these patterns periodic? In this regard, Figure 1 shows cumulative distribution function (CDF) plots of four statistics by query type: the absolute and mean query arrival rates as well as their variation (CV) and approximate entropy.

Distribution of Values. A large fraction of time steps have zero arrivals, ranging from 65% for Simple Select to 90% for Deletes and Updates. Though the long tail is omitted for clarity, the query arrivals reach values in the thousands, indicating high traffic from very few customers. The abundance of zeros in the time series skews the mean arrival rates to have relatively low values, with most time series averaging under 50; however, around 10% of Simple Select, Join, and Aggregation queries reach average arrivals between 50–170. In terms of variability (CV is the standard deviation divided by the mean), we observe longer tails for the more sporadic queries, such as Updates and Deletes, because of their higher divergence from their very low mean values.

Regularity. Approximate Entropy [20] (ApEn) is a measure of how predictable a time series is. It quantifies the likelihood that patterns in the data repeat over time. To compute ApEn, we examine sequences of m consecutive values in the time series (called the *embedding dimension*). For each such sequence, we count how many other sequences of the same length are similar to it—that is, their values stay within a specified tolerance r. We then repeat this process for sequences of length $m+1$, and compare the results. If sequences that are similar at length m remain similar at length $m+1$, then the ApEn value will be low. Conversely, if the addition of one more data point leads to dissimilarity, the ApEn value increases. That is, a low ApEn value (close to 0) indicates time series have simple structure (e.g., a flat line) while a high value indicates highly irregular time series. As shown in Figure 1, a subset of workloads across different categories exhibits strong regularity. Notably, a higher proportion of `Update`, `Insert`, and `Aggregate` workloads demonstrate regular behavior compared to the other types.

Periodicity. We use Fast Fourier Transform (FFT) to identify recurring patterns in query traffic. This reveals dominant periodic components, such as daily (1-day), multi-day (2- or 3-day), and intra-day (4-, 8-, 12-hour) cycles. Figure 2 shows the percentage of customers whose query arrivals align with these dominant frequencies. Daily periodicity is prominent across all query types, seen in 79.2% of sampled customers. Patterns repeating every 3 days (33.4%) and every 4 hours (19.2%) are also evident.

Take-ways: Our global characterization reveals that the dataset exhibits diverse and complex structure across customer workloads, some of which relates to rich periodic patterns with overlapping daily, multi-day, and intra-day cycles. This complexity highlights the challenges inherent in our real-world query arrival time series data and underscores the need for robust and generalizable representations and similarity analysis for effective workload management.

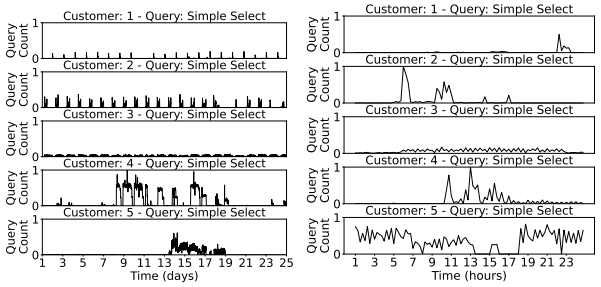
3.2 Diverse Subset for Evaluation

To support our case study on time series similarity and modeling, we select a subset of five customers with diverse arrival behaviors and human-perceived similarities and dissimilarities. This section details their key characteristics.

Diversity of Query-level Patterns. Figure 3a shows normalized monthly `simple select` query arrivals for the five customers, revealing a mix of regular, bursty, and irregular patterns. We first visually interpret the structure of these time series:

- *Customer 1* shows sparse, short daily bursts with high regularity, serving as the reference for comparison.
- *Customer 2* has similar daily bursts but is more spiky and temporally misaligned.
- *Customer 3* exhibits continuous low-magnitude signals with weak periodicity.
- *Customer 4* shows mid-month bursts with semi-regular structure.
- *Customer 5* is mostly inactive except for a short, bursty interval around the middle of the month.

Figure 3b zooms into a single day (Jan. 15) and confirms the complex and diverse structure ranging from sustained usage to sporadic bursts also at daily resolution.



(a) January 2025. **(b) January 15 2025.**

Figure 3: Arrivals of simple select queries over a month and a day for five customers. Query count (y-axis) is min-max normalized separately for monthly and daily arrivals.

	Month-wide			Day-wide		
Customer	Mean	CV	Entropy	Mean	CV	Entropy
1	0.002	5.839	0.503	0.012	4.346	1.009
2	0.016	2.927	1.909	0.059	2.804	1.863
3	0.017	0.963	3.203	0.066	0.693	3.200
4	0.124	1.713	2.465	0.079	2.151	2.093
5	0.026	2.710	2.209	0.366	0.643	3.859

Table 1: Mean, coefficient of variance (CV) and entropy for the normalized month-wide and day-wide arrivals of simple selects for the 5 selected customers.

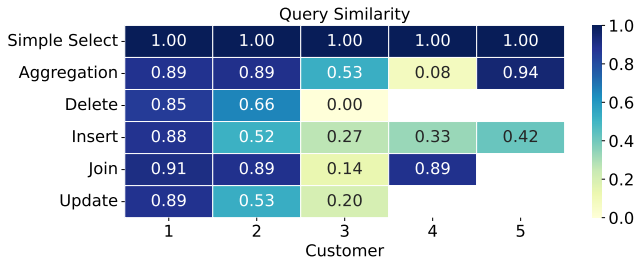


Figure 4: Similarity of arrival times across query types for the 5 selected customers.

Table 1 quantifies this diversity using the coefficient of variation (CV) and entropy, highlighting variability across customers. Interestingly, some customers exhibit high CV, indicating values that deviate significantly from the mean, yet have low entropy, suggesting predictable patterns. For example, Customer 1 demonstrates this behavior. Conversely, Customer 3 shows the opposite trend, with low CV and high entropy, illustrating the dataset's diversity.

Cross-Query Similarity Within Customers. We also examine whether different query types follow similar temporal patterns within each customer. Figure 4 presents a heatmap of normalized similarity (via reversed Euclidean distance) between the `simple select` query and other query types. We observe distinct behavioral profiles. For example, Customer 1 issues all query types concurrently, while Customer 3 exhibits noisy and unaligned activity. Customers 4 and 5 primarily execute read-only queries, suggesting different styles of database usage.

Take-ways: The analysis of five customers with diverse yet structured query arrival patterns reveals several *visual* and *analytical* insights. Customers 1 and 2 exhibit strong daily periodicity and bursty arrivals, while Customers 3 to 5 show weaker or no clear periodicity. Customer 3 has a low but continuous signal, in contrast to Customers 4 and 5, who display bursty arrivals concentrated around mid-month. These patterns suggest visually and intuitively identifiable groupings: specifically, a high similarity between Customers 1 and 2 due to their daily bursts, and a marked dissimilarity between them and Customers 3 to 5. Next, we will see whether these *human-interpretable* expectations align with similarity derived from time series analysis (Section 4).

Domain	Transformation	Metric
Time	None, Min-Max, Z-Score	Euclidean, DTW [24]
Frequency	FFT	Euclidean, DTW [24]
Frequency & Time	Wavelet [14]	Euclidean, DTW [24]
Embedding	Foundation Model (FM) [10]	Euclidean, Cosine

Table 2: Time series similarity by common transformations and metrics in the time, frequency, and embedding space.

4 Time Series Similarity: Case Study

In this section, we conduct a focused case study to evaluate the practicality of different similarity approaches — conventional and foundational — for workload management on our dataset. Engineers must be able to understand and trust similarity scoring of workloads, since these scores directly determine how downstream tasks interpret these workloads. We hence start by examining how similarity approaches align with intuitive, human-understandable patterns in real workload time series. Then, we evaluate clustering as a concrete downstream task where similarity scores directly impact outcomes like the identification of workload patterns.

4.1 Time Series Similarity: Interpretability

Similarity between time series is a core building block for many workload management tasks in clouds like Microsoft Azure. Crucially, engineers and operators must be able to understand, trust, and explain why two workloads are considered similar. Without this, similarity metrics risk becoming black-box scores with limited practical utility. Our goal is to test the interpretability of time series similarity approaches for practical purposes.

Evaluation Setup. We compare the similarity rankings to human intuition derived from the visible patterns in the raw time series. Specifically, we focus on comparing the similarity of Customer 1's simple select query arrival time series against the corresponding time series of Customers 2–5. Table 2 provides an overview of the similarity approaches that we describe and evaluate in the following. The evaluated methods are grouped into four domains: time, frequency, frequency-time, and embedding space. For embedding space, we use a foundational model, *MOMENT*, whereas the other domains use conventional approaches. Notably, we use the small *MOMENT* fine-tuned to our dataset, which shows significantly improved representation accuracy on our workload time series compared to the pretrained and large version (see Appendix 6.1 for details). Despite this improvement, we note that the model still struggles with abrupt changes characteristic of query arrival patterns. This sets the stage for our evaluation of how well such models capture workload similarity from a practical perspective.

For each similarity method, we describe how well the similarity aligns with visual intuition based on the normalized time series (Figure 3a) and discuss key takeaways. Figure 5 visualizes the resulting similarity scores across methods. We begin with conventional similarity methods. These approaches offer transparency and control, since users can select desired transformations and similarity metrics to emphasize specific characteristics of the time series, such as shape, burstiness, or periodicity.

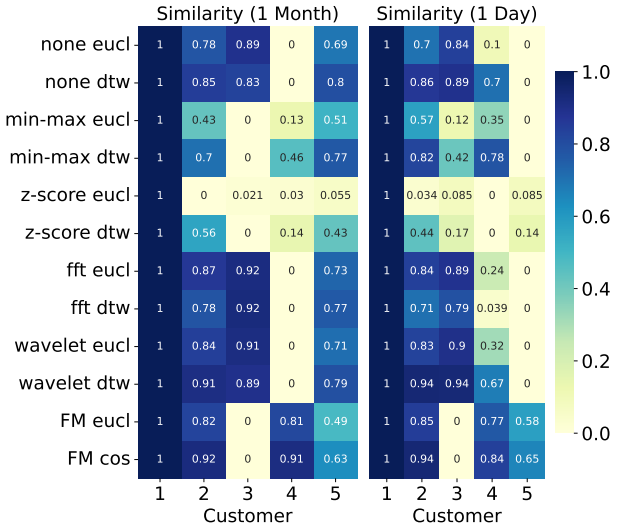


Figure 5: Similarity of *simple select* query arrivals of Customer 1 versus Customers 1–5; highest = 1, lowest = 0.

4.1.1 Conventional Methods: Time Domain. **No transformation:** Euclidean and DTW (Dynamic Time Warping) both rate Customers 2, 4, and 5 as similar to Customer 1. This is largely due to similar volume and periods of inactivity. DTW smooths misalignment but also overstates similarity to Customer 5, which is visually distinct. These results show that raw-value comparisons can be misleading when dominated by shared low-activity phases.

Min-max normalization: These methods emphasize shape over scale. DTW aligns well with intuition—highlighting high similarity of Customer 2 (similar bursts) and Customer 4 (moderately bursty). However, Customer 5 is rated too highly, likely due to low-activity phases. Euclidean with min-max fails to capture time-shifted bursts and underrates Customer 4.

Z-score normalization: Both Euclidean and DTW metrics perform poorly under Z-score normalization. The transformation removes signal shape in favor of deviation from the mean, producing low interpretability.

4.1.2 Conventional Methods: Frequency and Time-Frequency Domains. FFT and wavelet-based similarity correctly group Customers 1–3 as similar, due to the strong periodic structure observed previously. However, they ignore temporal alignment, overrating Customer 3 despite its irregular spikes. These methods are useful for coarse trends but struggle with fine-grained behaviors.

4.1.3 Foundational Model Approaches. We compute embeddings using the fine-tuned small *MOMENT* model and compare similarity via Euclidean and cosine distance. The similarity scores are consistent across metrics: Customers 2 and 4 are most similar to Customer 1, followed by Customer 5, with Customer 3 least similar. This matches high-level visual similarity—Customer 2 shares bursty shape; Customer 3 is flat and continuous.

However, some results are hard to explain—Customer 4's bursts differ significantly in timing and density. The embedding scores suggest meaningful structure, but it remains unclear what dimensions influence the similarity judgments. Compared to conventional methods, embedding similarity aligns with intuitive results for Customers 2 and 3, but its reasoning is opaque.

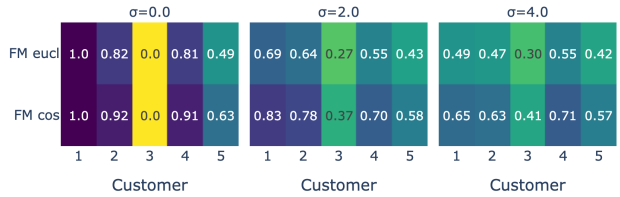


Figure 6: Embedding similarity of monthly *simple select* query arrivals of Customer 1 versus Customers 1–5 under Gaussian noise with 0 mean and σ; highest = 1, lowest = 0.

Sensitivity to Noise. In Figure 6, we inject Gaussian noise ($\sigma = 2, 4$) into Customers 1–5 and recompute similarity to the original Customer 1. Moderate noise reduces similarity for Customers 2 and 4, suggesting embeddings rely on burst structure. Stronger noise causes all scores to converge, indicating loss of structure. Notably, Customer 3's similarity remains stable, implying its embedding lacks sensitivity to additional noise.

Take-ways: Among conventional methods, *min-max DTW* best matches human intuition by capturing recurring burst patterns. Frequency-based methods capture recurring patterns but miss timing. Embedding-based similarity offers consistent, plausible rankings and some robustness to noise, but lacks interpretability. Its results cannot be easily explained or diagnosed—posing challenges for practical workload management.

4.2 End-to-end Time Series Clustering

We now examine how the similarity approaches affect hierarchical clustering for workload pattern mining—contrasting our isolated interpretability findings with practical end-to-end performance. Since clustering algorithms group time series based solely on calculated similarities, the choice of similarity approach determines both cluster composition and quality. Effective similarity measures for cloud workload management must balance four critical factors: (1) precisely separating distinct workload patterns, (2) correctly grouping similar behaviors, (3) maintaining reasonable computational costs, and (4) being interpretable. While we could evaluate interpretability in isolation, the remaining three factors require evaluation in a downstream task. We hence quantify these using clustering, which is common in many workload management tasks.

Evaluation Setup. We evaluate hierarchical clustering (HDBSCAN [4]) using the similarity approaches and dataset from the previous section. For comparison, we establish as reference the Euclidean distance on hand-engineered feature vectors—24 time-domain features [15] combined with 10 frequency components. We quantify clustering quality by: (1) Adjusted Rand Index (ARI), measuring cluster agreement with our reference (higher value is better); (2) Silhouette scores in both feature and native similarity spaces to measure cluster cohesion/separation (higher value is better); (3) Noise ratio, indicating the fraction of time windows not assigned to any cluster (lower value is better). We create a challenging test by dividing our 30-day dataset into 5.333-day windows (512 samples), which intentionally misaligns periodic patterns and forces similarity approaches to detect structural relationships rather than exact matches. For these 25 time windows, we configure HDBSCAN with minimum cluster size of 2 [1].

Similarity Approach	ARI	Mean Silhouette Feat.	Mean Silhouette Native	Noise (%)	Clus -ters	Time (ms)
hand eucl	**1.00**	**0.49**	**0.49**	**4**	**5**	0.9
none dtw	0.74	0.31	0.44	20	4	3.4e+3
fft dtw	0.66	0.19	0.25	24	5	3.4e+3
z-score dtw	0.63	0.25	0.39	20	4	3.5e+3
fft eucl	0.62	0.22	0.41	16	3	0.8
min-max dtw	0.54	0.19	0.21	28	4	3.7e+3
wavelet dtw	0.46	0.19	0.24	40	3	3.7e+3
min-max eucl	0.26	-0.02	0.38	16	2	0.8
FM eucl	0.17	0.15	0.62	0	2	1.2
FM cos	0.17	0.15	0.61	0	2	1.6
none eucl	0.00	–	–	100	0	0.8
z-score eucl	0.00	–	–	100	0	0.8
wavelet eucl	0.00	–	–	100	0	0.7

Table 3: Performance of hierarchical clustering under different similarity approaches. ARI measures agreement with the hand-engineered feature similarity reference—1: perfect, 0: worst. Silhouette scores measure cluster data separation in the hand-engineered feature space (Feat.) and native similarity space (Native)—1: strong separation, 0: little separation, -1: overlapping clusters [23]. Noise indicates the fraction of unassigned data points—lower is better.

Clustering Results. Table 3 shows the clustering results (additionally Figure 9 in the appendix). To set the stage, the hand-engineered feature vector approach (*hand eucl*) proves as the gold standard, with low clustering time and the highest clustering quality: highest cluster consistency (Silhouette [23] score in feature space) and at the same time the least noise. In contrast, the lower number of discovered clusters and the lower Adjusted Rand Index (ARI) score indicate that compared to hand-engineering, the remaining approaches do not find as much structural commonality between the time windows. Visual inspection of Figure 9 confirms this more coarse-grained clustering. However, a clear trade-off between fast and consistent approaches becomes evident, besides the hand-engineering approach.

Fast Approaches. The Euclidean and Cosine based approaches are fast but cause low quality clustering, due to their limited ability to detect structural similarity. *fft eucl* achieves clustering quality comparable to some of the DTW approaches but with much lower overhead, since the FFT transformation exposes the periodic structure. Instead, the remaining fast Euclidean approaches with different transformations perform poor clustering, either classifying the data as pure noise (*none eucl*, *z-score eucl*, *wavelet eucl*) or identifying only two coarse clusters (*min-max eucl*, *FM eucl*, *FM cos*). Notably, the foundational model approaches (*FM eucl* & *FM cos*) assign too high similarity to the time windows, as evidenced by the high Silhouette score in the embedding space but low Silhouette score in the feature space. This reveals a limitation we could not detect in our interpretability analysis: while embeddings produced plausible similarity rankings, they actually over-generalize similarities between time windows, making them less discriminative for detailed pattern identification. Similar behavior applies to *min-max eucl*.

DTW Approaches. The Dynamic Time Warping (DTW) approaches achieve the highest clustering quality, after the hand-engineered features. DTW without transformation (*none dtw*) achieves the clustering most similar to the hand approach—0.74 ARI. Transformations prior to DTW, such as FFT (*fft dtw*), Z-score (*z-score dtw*), and min-max (*min-max dtw*), lower clustering quality. Consistent with our interpretability findings, this shows that DTW successfully captures structural similarity between time windows even with temporal misalignment and DTW performs best when this structure is not distorted by transformations. In comparison to hand-engineering, however, these approaches still do not identify structurally complex similarity (see high noise) and incur >3000x higher compute overhead.

Take-ways: For workload pattern mining, hand-engineered features remain the gold standard, offering superior clustering quality with minimal computational overhead. Among automated approaches, frequency domain transformations (FFT) with Euclidean distance offer the best balance of quality and efficiency, highlighting the importance of periodic patterns in our workload data. DTW methods achieve high quality but at prohibitive computational cost (3000x slower). Foundation models offer computational efficiency but produce overly generalized clusters with limited utility, a weakness not apparent in our manual interpretability evaluation. This demonstrates why rigorous interpretability benefits practical cloud workload management; not only does it inspire confidence for practitioners, but it will also allow to decouple designing similarity metrics/embeddings from (future) downstream tasks.

5 Lessons Learned And Future Directions

This paper exposes practical challenges in analyzing time series patterns for cloud workload management. It studies time series similarity approaches for capturing patterns in workload signals like query arrivals. It emphasizes the need for interpretable approaches and highlights limitations in current methods, including foundational time series models.

5.1 Lessons Learned for Time Series Similarity

Diverse & Complex Patterns in Real-World Data. Our detailed characterization of real-world query arrival telemetry reveals a wide range of time series patterns and the inherent complexity they present. While common patterns such as periodic query arrivals are evident at a broad level, closer examination uncovers significant variations, including offsets in periodic arrivals and interleaved sporadic activity.

Limitations of Traditional Similarity Approaches. Through comprehensive experiments, we demonstrate how different traditional similarity approaches emphasize different time series features, such as amplitude, alignment, and frequency. The wide diversity in time series structures and workload management tasks, however, requires complex customized feature engineering. Such featurization, along with the interpretation of the resulting similarities, poses challenges for non-experts, pointing to the opportunities for foundational time series models.

Promise & Challenges of Foundational Models. Foundational time series models, such as transformer-based architectures, offer a promising solution to address the above complexities by

automatically capturing key time series features in compact embeddings. These models thus facilitate efficient similarity searches and various downstream tasks, such as forecasting, anomaly detection, and optimization. While decoding mechanisms provide partial insights, a comprehensive understanding of whether these similarities reflect meaningful characteristics or merely representational limitations remains elusive. This highlights the necessity for continued research aimed at enhancing interpretability and robustly assessing the model's perception of time series.

Cloud workloads surge fast,
Black box models cannot see—
Hand-craft still prevails.

5.2 Call For Future Research

The complexities identified in this work underscore several critical open questions and opportunities for future investigation.

Interpretability. The critical demand for interpretability in embeddings and similarity scores remains unmet for practical use in workload management. While embeddings may emphasize specific time series characteristics, their meaning often remains opaque—making it difficult to trust their output, even when embedding accuracy is high. The same challenge extends to similarity comparisons and other downstream tasks: without interpretability, users lack insight into why certain patterns are considered similar, which undermines confidence in automated decisions and limits adoption in real-world systems.

Model Generalization vs. Specialization. When facing complex and diverse time series, one key open question is whether to improve the generalization of foundational models or specialize them. Our empirical study reveals saturation of training loss and insufficient representation capabilities for spiky signals. This issue could be addressed through model architecture and training procedures that enhance the generalization of foundational models. Alternatively, we could specialize these models through fine-tuning or customized loss functions tailored to specific signal types, such as bursty query arrivals or periodic CPU utilization patterns. Our observations suggest that specialization may be the easier approach to improve representation accuracy. However, this comes with operational overheads of training and maintaining multiple models instead of a single generalized model. Therefore, we urge the research community to weigh the trade-offs between accuracy and operational overheads and recommend including characterization of training data to understand model applicability, providing guidance or automated tools for model specialization.

Multivariate Embeddings. Multivariate embeddings present exciting possibilities for richer workload characterizations by capturing complex interrelations among individual signals from different workload components. For instance, workload management in database systems typically considers the interactive query arrival signals of selects, inserts, updates, and deletes together. However, the existing challenges in evaluating and interpreting even univariate embeddings underscore the critical need for robust evaluation and interpretation strategies that need to be extended to the more complex case of multivariate embeddings.

Composable & Transformable Embeddings. Another key requirement is to develop embeddings that are both composable and transformable, enabling customization for specific workload management tasks. Observations from this study indicate that the perceived similarity between time series can vary significantly depending on the chosen time scale or transformation, each crucial for distinct use cases. Therefore, embedding methods must support composability across time ranges and transformations that allow use-case-specific emphasis on various time series features like amplitude, time warping, or frequency. A pertinent research question is how to create embeddings that faithfully reflect such transformations. This could involve transforming the input and forcing the model to reflect those changes, fine-tuning transformation layers, or even instructing the model on specific transformations. A practical solution must also aim to minimize operational overheads.

Foundation cracks show,
Build anew with clarity–
Scale meets understanding.

Acknowledgments

We thank the anonymous reviewers for their constructive feedback. We also thank our colleagues at Microsoft, especially Andreas Müller and Venkatesh Emani, for their insightful discussions.
This work has been partially supported by the Madrid Regional Government (César Nombela Grant 2024-T1/COM-31302), who also funds the DATIA project alongside FEDER funds from the EU, and the grants PID2022-142290OB-I00 and TED2021-132464B-I00, funded by MCIN/AEI/10.13039/501100011033, FEDER, and NextGenerationEU/PRTR funds.

Disclaimer. We have utilized GenAI technology in creating this work: GitHub Copilot for implementing experiments, Copilot and ChatGPT to assist with writing, and Deep Research to cross-validate and compose related work.

References

[1] 2025. *Sklearn HDBSCAN.* https://github.com/scikit-learn/scikit-learn/blob/98ed9dc73a86f5f11781a0e21f24c8f47979ec67/sklearn/cluster/_hdbscan/hdbscan.py#L426

[2] Rana Alotaibi, Yuanyuan Tian, Stefan Grafberger, Jesús Camacho-Rodríguez, Nicolas Bruno, Brian Kroth, Sergiy Matusevych, Ashvin Agrawal, Mahesh Behera, Ashit Gosalia, Cesar Galindo-Legaria, Milind Joshi, Milan Potocnik, Beysim Sezgin, Xiaoyu Li, and Carlo Curino. 2025. Towards Query Optimizer as a Service (QOaaS) in a Unified LakeHouse Ecosystem: Can One QO Rule Them All? *Proc. CIDR* (2025). https://www.vldb.org/cidrdb/papers/2025/p5-alotaibi.pdf

[3] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. 2024. Chronos: Learning the Language of Time Series. doi:10.48550/arXiv.2403.07815 arXiv:2403.07815 [cs]

[4] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. 2013. Density-Based Clustering Based on Hierarchical Density Estimates. In *Advances in Knowledge Discovery and Data Mining* (Berlin, Heidelberg, 2013), Jian Pei, Vincent S. Tseng, Longbing Cao, Hiroshi Motoda, and Guandong Xu (Eds.). Springer Berlin Heidelberg, 160–172.

[5] Georgia Christofidi and Thaleia Dimitra Doudali. 2024. Do Predictors for Resource Overcommitment Even Predict?. In *Proceedings of the 4th Workshop on Machine Learning and Systems* (Athens, Greece) *(EuroMLSys '24)*. Association for Computing Machinery, New York, NY, USA, 153–160. doi:10.1145/3642970.3655838

[6] Georgia Christofidi, Konstantinos Papaioannou, and Thaleia Dimitra Doudali. 2023. Is Machine Learning Necessary for Cloud Resource Usage Forecasting?. In *Proceedings of the 2023 ACM Symposium on Cloud Computing* (Santa Cruz, CA, USA) *(SoCC '23)*. Association for Computing Machinery, New York, NY, USA, 544–554. doi:10.1145/3620678.3624790

[7] Georgia Christofidi, Konstantinos Papaioannou, and Thaleia Dimitra Doudali. 2023. Toward Pattern-based Model Selection for Cloud Resource Forecasting. In

Proceedings of the 3rd Workshop on Machine Learning and Systems (Rome, Italy) *(EuroMLSys '23)*. Association for Computing Machinery, New York, NY, USA, 115–122. doi:10.1145/3578356.3592588

[8] Thaleia Dimitra Doudali, Sergey Blagodurov, Abhinav Vishnu, Sudhanva Gurumurthi, and Ada Gavrilovska. 2019. Kleio: A Hybrid Memory Page Scheduler with Machine Intelligence. In *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing* (Phoenix, AZ, USA) *(HPDC '19)*. Association for Computing Machinery, New York, NY, USA, 37–48. doi:10.1145/3307681.3325398

[9] Thaleia Dimitra Doudali and Ada Gavrilovska. 2022. Coeus: Clustering (A)like Patterns for Practical Machine Intelligent Hybrid Memory Management. In *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. 615–624. doi:10.1109/CCGrid54584.2022.00071

[10] Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. 2024. MOMENT: A Family of Open Time-series Foundation Models. doi:10.48550/arXiv.2402.03885 arXiv:2402.03885 [cs]

[11] Hanxian Huang, Tarique Siddiqui, Rana Alotaibi, Carlo Curino, Jyoti Leeka, Alekh Jindal, Jishen Zhao, Jesús Camacho-Rodríguez, and Yuanyuan Tian. 2024. Sibyl: Forecasting Time-Evolving Query Workloads. In *SIGMOD*. https://www.microsoft.com/en-us/research/publication/sibyl-forecasting-time-evolving-query-workloads/

[12] Artjom Joosen, Ahmed Hassan, Martin Asenov, Rajkarn Singh, Luke Darlow, Jianfeng Wang, and Adam Barker. 2023. How Does It Function? Characterizing Long-term Trends in Production Serverless Workloads. In *Proceedings of the 2023 ACM Symposium on Cloud Computing* (New York, NY, USA, 2023-10-31) *(SoCC '23)*. Association for Computing Machinery, 443–458. doi:10.1145/3620678.3624783

[13] Brian Kroth, Sergiy Matusevych, Rana Alotaibi, Yiwen Zhu, Anja Gruenheid, and Yuanyuan Tian. 2024. MLOS in Action: Bridging the Gap Between Experimentation and Auto-Tuning in the Cloud. *Proc. VLDB Endow.* 17 (October 2024), 4269–4272. https://www.microsoft.com/en-us/research/publication/mlos-in-action-bridging-the-gap-between-experimentation-and-auto-tuning-in-the-cloud/

[14] Gregory R. Lee, Ralf Gommers, Filip Waselewski, Kai Wohlfahrt, and Aaron O'Leary. 2019. PyWavelets: A Python package for wavelet analysis. 4, 36 (2019), 1237. doi:10.21105/joss.01237

[15] Carl H. Lubba, Sarab S. Sethi, Philip Knaute, Simon R. Schultz, Ben D. Fulcher, and Nick S. Jones. 2019. catch22: CAnonical Time-series CHaracteristics. 33, 6 (2019), 1821–1852. doi:10.1007/s10618-019-00647-x

[16] Lin Ma, Dana Van Aken, Ahmed Hefny, Gustavo Mezerhane, Andrew Pavlo, and Geoffrey J. Gordon. 2018. Query-based Workload Forecasting for Self-Driving Database Management Systems. In *Proceedings of the 2018 International Conference on Management of Data* (Houston, TX, USA) *(SIGMOD '18)*. Association for Computing Machinery, New York, NY, USA, 631–645. doi:10.1145/3183713.3196908

[17] Qianli Ma, Zhen Liu, Zhenjing Zheng, Ziyang Huang, Siying Zhu, Zhongzhong Yu, and James T. Kwok. 2024. A Survey on Time-Series Pre-Trained Models. doi:10.48550/arXiv.2305.10716 arXiv:2305.10716 [cs]

[18] Microsoft Fabric 2024. Data warehousing in Microsoft Fabric. https://learn.microsoft.com/en-us/fabric/data-warehouse. Online; accessed 2025-04-04.

[19] Andreas Mueller. 2025. Open Challenges in Time Series Anomaly Detection: An Industry Perspective. doi:10.48550/arXiv.2502.05392 arXiv:2502.05392 [cs]

[20] Steven M Pincus, Igor M Gladstone, and Richard A Ehrenkranz. 1991. A regularity statistic for medical data analysis. *Journal of clinical monitoring* 7 (1991), 335–345.

[21] Olga Poppe, Tayo Amuneke, Dalitso Banda, Aritra De, Ari Green, Manon Knoertzer, Ehi Nosakhare, Karthik Rajendran, Deepak Shankargouda, Meina Wang, Alan Au, Carlo Curino, Qun Guo, Alekh Jindal, Ajay Kalhan, Morgan Oslake, Sonia Parchani, Vijay Ramani, Raj Sellappan, Saikat Sen, Sheetal Shrotri, Soundararajan Srinivasan, Ping Xia, Shize Xu, Alicia Yang, and Yiwen Zhu. 2020. Seagull: An Infrastructure for Load Prediction and Optimized Resource Allocation. In *VLDB 2021*. VLDB Endowment, 154–162. https://www.microsoft.com/en-us/research/publication/seagull/ Top project in Azure All Hands 2021.

[22] Deepak Ravikumar, Alex Yeo, Yiwen Zhu, Aditya Lakra, Harsha Nagulapalli, Santhosh Ravindran, Steve Suh, Niharika Dutta, Andrew Fogarty, Yoonjae Park, Sumeet Khushalani, Arijit Tarafdar, Kunal Parekh, and Subru Krishnan. 2024. Intelligent Pooling: Proactive Resource Provisioning in Large-scale Cloud Service. *Proc. VLDB Endow.* 17, 7 (March 2024), 1618–1627. doi:10.14778/3654621.3654629

[23] Peter J. Rousseeuw. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. 20 (1987), 53–65. doi:10.1016/0377-0427(87)90125-7

[24] Stan Salvador and Philip Chan. 2007. Toward accurate dynamic time warping in linear time and space. 11, 5 (2007), 561–580. doi:10.3233/IDA-2007-11508 Publisher: SAGE Publications.

[25] Liqun Shao, Yiwen Zhu, Siqi Liu, Abhiram Eswaran, Kristin Lieber, Janhavi Mahajan, Minsoo Thigpen, Sudhir Darbha, Subru Krishnan, Soundar Srinivasan, Carlo Curino, and Konstantinos Karanasos. 2019. Griffon: Reasoning about Job Anomalies with Unlabeled Data in Cloud-based Platforms. In *Proceedings of the ACM Symposium on Cloud Computing* (New York, NY, USA, 2019-11-20) *(SoCC '19)*. Association for Computing Machinery, 441–452. doi:10.1145/3357223.3362716

[26] Yiwen Zhu, Subru Krishnan, Konstantinos Karanasos, Isha Tarte, Conor Power, Abhishek Modi, Manoj Kumar, Deli Zhang, Kartheek Muthyala, Nick Jurgens, Sarvesh Sakalanaga, Sudhir Darbha, Minu Iyer, Ankita Agarwal, and Carlo Curino. 2021. KEA: Tuning an Exabyte-Scale Data Infrastructure. In *Proceedings of the 2021 International Conference on Management of Data* (New York, NY, USA, 2021-06-18) *(SIGMOD '21)*. Association for Computing Machinery, 2667–2680. doi:10.1145/3448016.3457569

6 Appendix

6.1 Foundational Model Training

We now evaluate how accurate the *MOMENT* model represents the time series of our dataset. We consider four model variants: small vs. large and pretrained vs. finetuned. We finetune on 85% of time series and validate on 15% diverse timeseries, including our hand-picked customers. This setup including the finetuning procedure follows the original setup of *MOMENT*. *MOMENT* allows decoding its learned embedding back into a timeseries signal. We use this to quantify representation accuracy via Mean Squared Error (MSE) and visually inspect how the model perceives the the original time series.

Figure 7 shows the results of all model variants for our hand-picked timeseries in the validation set. Our key observations: (1) The *small pretrained* model performs poorly (MSE: 0.72–1.3); (2) the *large pretrained* model improves over small (MSE: 0.27–1.0); (3) *finetuning small model* significantly improves the accuracy (MSE: 0.17–0.63); (4) *finetuning* the *large model* yields marginal gains over pretrained (MSE: 0.26–0.81). While high-level patterns are captured, visual inspection shows *MOMENT* struggles with abrupt changes in the signal, e.g., due to spikes or the hard cut-off at 0.

To better understand the above observations, Figure 8 details the finetuning performance—the mean squared error loss over the finetuning epochs. The small model quickly reduces its loss within the

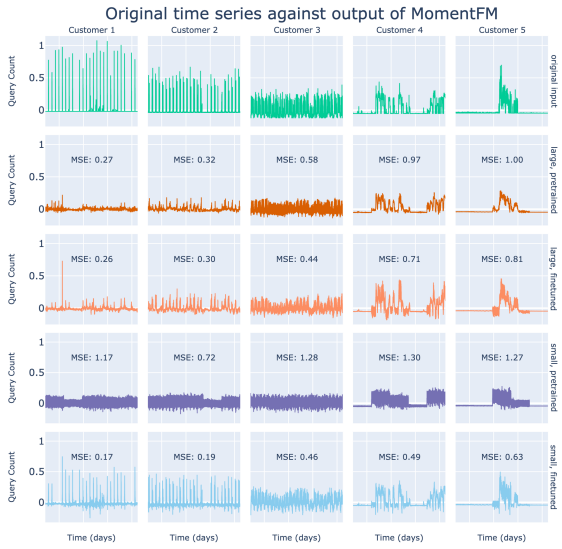


Figure 7: Visualization of original time series versus decoded output and Mean Squared Error (MSE) of the foundational time series *MOMENT*—in variants small vs. large and pretrained vs. finetuned.

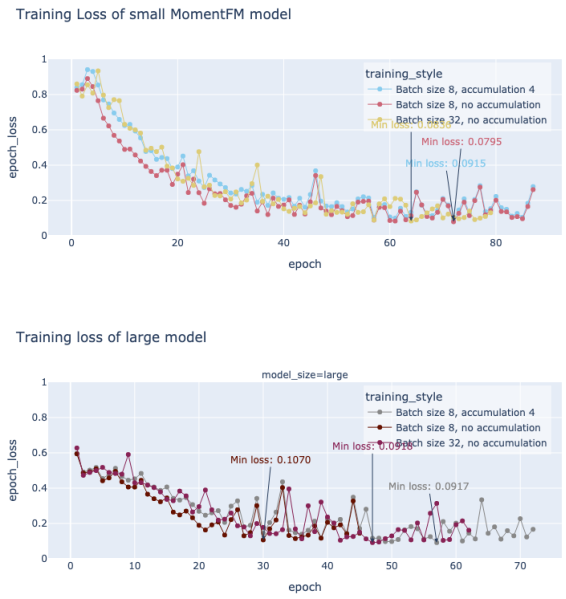


Figure 8: Training loss over epochs when finetuning *MOMENT* on our dataset.

first few epochs, but convergence is unstable—the best loss is 0.079 at epoch 72 but the loss oscillates. The large model also converges with oscillation, but more slowly and to a higher loss. This suggests diminishing returns when finetuning larger pretrained models on small, domain-specific datasets.

To improve stability, we experimented with gradient accumulation and larger batch sizes. However, neither approach led to meaningful improvement in final loss or convergence behavior.

Take-away: Finetuning enables *MOMENT* to capture the shape of our query arrival time series much better, especially the representation accuracy of the small model quickly improves. Still, both the small and large model saturate at higher loss than in the original training, presumably due to the abrupt nature of our time series signal that is not well represented in the pretraining dataset.

6.2 End-to-End Time Series Clustering Visualization

Figure 9 visualizes how different similarity approaches cluster the same set of 25 time windows (each 5.333 days long) from our dataset. Each column in the figure represents the time series of one customer. Each row represents a different similarity approach. The colors indicate cluster membership while grey indicates noise (time windows not assigned to any cluster). Identical colors across columns indicate that time windows from different customers are assigned to the same cluster, i.e., are considered as a similar time series pattern. Identical colors in the same column across rows indicate that the same time window is assigned to the same cluster by different similarity approaches.

The visualization provides several key insights: (1) hand-engineered features produce the most refined clustering with 5 distinct clusters and minimal noise; (2) DTW approaches (especially none-dtw) create meaningful clusters but with more noise; (3) FFT with Euclidean distance offers a good balance of clustering quality and efficiency; (4) foundational models (FM) produce only 2 overly generalized clusters; and (5) several approaches (none-eucl, z-score-eucl, wavelet-eucl) fail entirely, classifying all windows as noise. The figure confirms the quantitative metrics in Table 3 and demonstrates why methods with similar ARI scores can produce qualitatively different clusterings in practice. Notably, for customers with periodic time series, the clustering results are broadly consistent across different time windows, when clusters are assigned.

Figure 9: Visualization of time window clustering via different similarity approaches, cf. Section 4.2