Streamlining Real-Time Temperature Prediction with Apache Kafka and Online Machine Learning Models

Dhanush Hebbar
Indian Institute of Information Technology, Kottayam
Pala, Kerala
dhanushhebbar18@gmail.com

Dr. E Silambarasan
Department of Cyber Security
Indian Institute of Information Technology, Kottayam
Pala, Kerala
silambarasan@iiitkottayam.ac.in

***Abstract*—Thermal management in hyper-scale cloud data centers presents significant challenges due to rising host temperatures and non-uniform thermal distributions, leading to higher cooling costs and reduced system reliability. Accurate temperature prediction is crucial for efficient resource management. However, conventional approaches have notable limitations—analytical models are often computationally intensive yet lack accuracy, while static machine learning methods, though more precise, fail to adapt to evolving operational patterns in modern data centers. To address these challenges, we propose an online machine learning framework that leverages Apache Kafka for real-time data streaming. Our approach, based on the Adaptive Random Forest algorithm, achieves an average predictive error of just 0.70°C and an R² score of 96.7%. This performance closely matches that of existing static models while offering while offering the significant advantage of rapid adaptability to changes in usage patterns without the need for lengthy and resource-intensive retraining processes.**

***Index Terms*—Cloud Computing, Green Computing, Apache Kafka, Temperature Prediction, Machine Learning, Deep Learning, Data Centers, Energy Management**

I. Introduction

The rapid expansion of hyper-scale cloud data centers has fundamentally transformed modern computing while introducing significant thermal management challenges. Data centers consumed an estimated 460 terawatt-hours (TWh) of energy in 2022 and are predicted to reach up to 1,000 TWh by 2026 [1]. According to JLL, hyper-scale data centers are expected to increase their rack density at a compound annual growth rate (CAGR) of 7.8% [17], rising from the current average of 36 kW per rack to 50 kW by 2027. As these centers scale to meet increasing computational demands, elevated host temperatures create uneven temperature distributions that escalate cooling costs and compromise hardware reliability. The report from Uptime Institute [2] shows that the failure rate of equipment doubles for every 10°C increase above 21°C, highlighting the critical importance of effective thermal management within the data center Resource Management System (RMS). Hence, accurate prediction of temperature fluctuations is crucial for resource management and energy efficiency.

Traditionally, Computational Fluid Dynamics (CFD) models have been employed to simulate and predict temperature distributions within data centers. Guo et al [**?**] developed a three dimensional model and CFD simulations to analyze airflow and temperature distribution alongside installing adjustable underloor deflectors and and replacing the side of the floor grille with a fan floor to improve efficiency but were still only able to get an maximum temperature reduction of 2.81°C and it only focuses on full-load scenarios, potentially limiting its applicability to varying operational conditions in data centers. Hence, theoretical models for temperature estimation are inaccurate, computationally expensive, and vary from one data center to another.

Modern data-driven machine learning approaches have demonstrated great accuracy and efficiency in predicting the temperatures of hyper-scale data centers. These models can integrate data from numerous sensors - such as CPU and GPU utilization, fan speeds, RAM consumption, outside temperatures, etc - to accurately understand and predicts temperatures. Deep Learning models, especially those designed for time series analysis like Recurrent Neural Networks or Long Short-Term Memory Networks (LSTMs) are great at understanding sequential data and can effectively predict future temperatures based on historical observations and can help eliminate thermal hotspots and improve system reliability.

While these models have demonstrated flexibility and accuracy in temperature predictions, their effectiveness relies on historical training data that may quickly become outdated as the data center usage scales up leading to a complex, lengthy and resource-intensive retraining process, hindering their ability to adapt in real time.

Online machine learning models continuously update their parameters with each incoming data point, enabling immediate adaptation to evolving usage patterns. This real-time learning eliminates the need for extensive retraining, ensuring models remain accurate even as operational conditions change and their ability to dynamically adjust to new patterns make them particularly suited for managing the complex, shifting patterns of data centers.

Our approach utilizes Apache Kafka, a distributed streaming platform, to ingest and stream resource usage data from the data center environment to online learning models in real time. We perform a comprehensive comparative evaluation of various online learning models, demonstrating the superior performance of online learning models like Adaptive Random Forests over traditional methods.

The rest of the paper is structured as follows: Section II reviews related work and existing methods. Section III presents the proposed framework and its integration with real-time data streaming. Section IV details the experimental setup and performance evaluation. Section V concludes the paper and discusses future work.

II. Literature Review

Significant research has been conducted to develop techniques for temperature prediction in cloud data centers. Zhang et al. [3] employed various machine learning techniques to minimize thermal variation and mitigate hotspots across supercomputing components. As one of the first studies to predict temperature using machine learning, it faced challenges such as prioritizing thermal management over computational performance and exhibiting limited prediction accuracy, with average errors ranging from 4.2°C to 2.9°C.

Ilager et al. [5] developed a Gradient Boosting algorithm that achieved a Root Mean Squared Error (RMSE) of 0.05 and an average prediction error of 2.38°C. Furthermore, they proposed a dynamic scheduling algorithm to minimize peak host temperatures, achieving a reduction of 6.5°C and a decrease in energy consumption by 34.5% compared to a baseline algorithm.

Zapater et al. [4] introduced a Grammatical Evolution (GE)-based methodology for predicting CPU and inlet temperatures, addressing the limitations of conventional Computational Fluid Dynamics (CFD) techniques. The proposed models are adaptable to diverse data center environments and do not require domain-specific knowledge. However, the GE methodology significantly increases the solution search space, leading to high computational costs and the potential risk of convergence to local optima.

Tabrizchi et al. [4] introduce a hybrid deep learning model using Convolutional Neural Networks (CNN) and stacking multi-layer Bi-directional Long Short-Term Memory (BiLSTM) to predict temperatures in cloud data centers. The model aims to improve the accuracy of thermal forecasts and achieve an impreessive R2 score of 97.15 %. But the study does not discuss the performance of the model under various workload scenarios and its reliance on historical data for training the model could be a major limitation if there are any changes in the configuration or operational patterns.

Bozkurt et al. [15] explores the synergy between Apache Flink and Apache Kafka for real-time data processing and analysis. It seeks to provide insights into how their integration can enable organizations to effectively utilize real-time data to improve decision-making and explains how Flink and Kafka can be used to create a robust system for managing and analyzing real-time data.

The field of online learning, where models are updated incrementally with each new data point, marks a paradigm shift from traditional batch learning. Early research in this area focused on algorithms capable of learning from continuous data streams without storing the entire dataset in memory. Decision trees, due to their ability to adapt to new patterns, were among the first to be adapted for online learning, leading to the development of algorithms like the Very Fast Decision Tree (VFDT) or Hoeffding Tree [6]. These algorithms use statistical bounds, such as the Hoeffding bound, to make splitting decisions based on limited observations, enabling rapid learning from streaming data.

Building on the success of online decision trees, ensemble methods were extended to the streaming domain to enhance prediction accuracy and robustness. The Adaptive Random Forest (ARF) algorithm, introduced by Bifet and Gavaldà in 2007 [7], significantly advanced this area by adapting the classic Random Forest algorithm for evolving data streams. ARF maintains an ensemble of Hoeffding Trees and incorporates online bagging for diversity, along with drift detection and adaptation strategies to address concept drift. By monitoring individual tree performance and replacing outdated ones, ARF ensures that the ensemble remains accurate in dynamic environments.

Pasala et al. [16] delved into optimizing real-time data pipelines for machine learning applications by comparing three stream processing architectures: Apache Kafka Streams, Apache Flink, and Apache Pulsar, addressing the gap in the comprehensive comparative analysis of these technologies in practical scenarios. It focuses on evaluating their efficiency in handling high-throughput data, delivering real-time predictions, and managing resource utilization.

While prior work has applied machine learning for temperature prediction in data centers and established online learning as effective for streaming data, there is a need to further investigate adaptive online learning methods, such as Adaptive Random Forest (ARF), for real-time temperature prediction and learning under evolving data center workloads. This gap is critical because data center workloads are inherently non-stationary, with fluctuations in resource demands (e.g., CPU, memory usage) caused by factors like user traffic changes, virtualization, or hardware aging, leading to concept drift, which refers to the changes in statistical properties of data over time, making historical models less accurate in gauging thermal patterns.

Online learning models have built-in mechanisms to handle issues like concept drift allowing them to stay up to date with the current patterns in data center usage which may differ from historical data. This paper addresses this gap by evaluating various online learning algorithms, including ARF, Hoeffding Trees, Passive Aggressive Regressors, to predict server temperatures using real-world resource utilization data streamed via Apache Kafka, utilizing its low-latency data pipelines to efficiently capture and pass data to these models for live temperature predictions.

III. Proposed Methodology

Our proposed methodology comprises a structured, multi-stage process designed to deliver real-time temperature predictions for data center servers. The approach begins with the acquisition of live sensor data, including temperature, CPU usage, memory utilization, and other pertinent system resources. Data collection is performed using industry-standard monitoring tools such as collectd and Prometheus. This raw data is subsequently aggregated and streamed in real time through Apache Kafka, ensuring reliable, high-throughput data transmission across the system.

A. Apache Kafka

Apache Kafka is an open-source distributed streaming platform designed for handling high-throughput, real-time data streams. It follows a publish-subscribe model while incorporating elements of traditional messaging queue systems. Kafka enables scalable and fault-tolerant data ingestion and processing by organizing data into topics and distributing it across multiple brokers [8].

1) Architecture: Kafka's architecture consists of several key components: producers, topics, partitions, brokers, consumers, and ZooKeeper. The detailed architecture diagram is displayed in Fig (1).

2) Topics, Partitions, and Records: A Kafka topic represents a logical stream of messages, categorized into partitions to support parallelism. Each partition is assigned to one or more brokers and contains an ordered sequence of immutable records. The use of partitions enables Kafka to scale horizontally by distributing load across multiple brokers.

Kafka ensures fault tolerance by replicating partitions across brokers. Each partition has a leader, responsible for handling read and write operations, and follower replicas that synchronize data from the leader. If a leader fails, a follower is elected as the new leader, maintaining availability [8].

3) Brokers and ZooKeeper: Kafka operates as a distributed cluster of brokers, with each broker responsible for managing partitions. Brokers store data persistently and handle client requests. They coordinate with ZooKeeper, which manages cluster metadata, performs leader election, and monitors broker health [8].

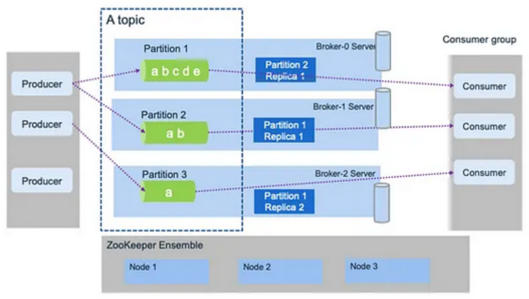

Fig. 1. Apache Kafka Architecture

ZooKeeper plays a crucial role in ensuring cluster synchronization, detecting broker failures, and reassigning partitions when necessary. It maintains a registry of active brokers and helps in distributing metadata efficiently.

4) Producers and Consumers: Producers are responsible for publishing data to Kafka topics. They can choose a partitioning strategy to determine how records are distributed among partitions, either randomly or based on a key to ensure order within a partition.

Consumers retrieve messages from Kafka topics. They can operate as part of a consumer group, where each consumer reads from a subset of partitions, ensuring load balancing. Kafka guarantees that each partition is consumed by only one consumer within a group, preventing duplicate processing [8].

5) Message Retention and Performance: Kafka employs a log-based retention mechanism, storing messages for a configurable period or until storage limits are reached. This enables applications to reprocess data and ensures fault tolerance without impacting performance.

Hence, Apache Kafka's distributed architecture, fault tolerance, and high scalability make it an essential component in modern data pipelines. By leveraging topics, partitions, brokers, and ZooKeeper, Kafka enables real-time data processing with minimal latency and high reliability.

Message retention policies allow Kafka to store data for a configurable period, balancing between performance and resource usage. This design enables efficient handling of large data volumes with low latency and high availability.

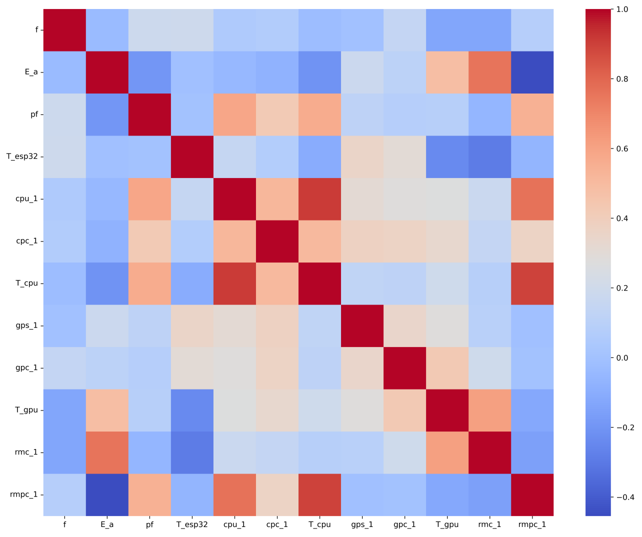

Fig. 2. Confusion Matrix of All Features from the Dataset

Therefore, we use Kafka to enable reliable integration between monitoring tools and online learning models, ensuring accurate real-time temperature prediction in data center environments.

TABLE I
SENSOR DATA FEATURES WITH ASSIGNED SYMBOLS

Feature	Definition
wd	weekday
V	Voltage (V)
I	Current (A)
P_a	Power (PA) (W)
f	Frequency (Hz)
E_a	Active Energy (KWh)
pf	Power factor
t_{esp32}	ESP32 temperature (°C)
cpu_1	CPU consumption (%)
cpc_1	CPU power consumption (%)
guc_1	GPU consumption (%)
gps_1	GPU power consumption (%)
t_{gpu}	GPU temperature (°C)
rmc_1	RAM memory consumption (%)
$rmpc_1$	RAM memory power consumption (%)
t_{cpu}	CPU temperature (°C)

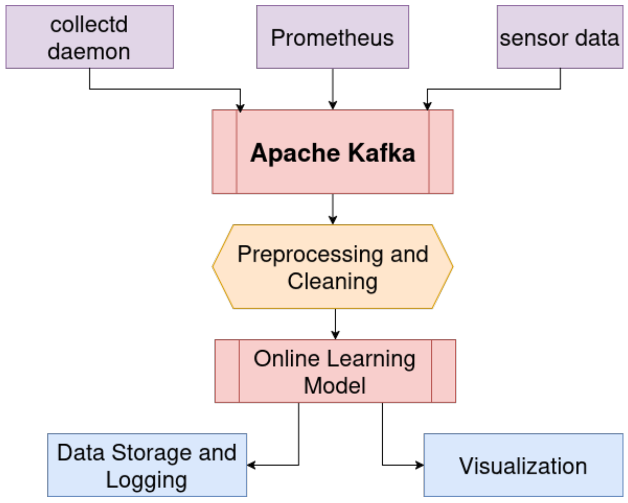

Fig. 3. Experiment Workflow

B. Dataset

The dataset on which the models were trained on was obtained from a Data Server that is running in the facilities of the Information Technology Center (CTI) of the Escuela Superior Politécnica del Litoral (ESPOL) [9]. Data was collected from an HP Z440 workstation for 245 days (35 weeks) with a sampling rate of one value per second. The dataset was down-sampled to compute average values every minute, as the original sampling rate of one value per second introduced frequent repetitions that degraded data quality. Additional preprocessing steps involved normalization, scaling and outlier removal were applied to better understand the patterns. The dataset spans diverse workload conditions and temperature variations, capturing both steady and fluctuating operational states. It was partitioned into training and testing subsets to evaluate generalization and resilience to concept drift. After data filtration and cleaning, the final dataset consisted of 83,379 tuples with each tuple having 16 features including resource and usage metrics and power and thermal measurements.

C. Prediction Algorithms

Standard deep learning methodologies predominantly employ batch learning, wherein model training occurs over entire, static datasets. This paradigm necessitates substantial computational resources, contributing significantly to the energy demands of data centers—an issue of growing importance given scaling operational loads and evolving data characteristics. The periodic nature of batch retraining presents limitations in dynamic environments, as models may exhibit suboptimal performance on current data due to the latency between updates when handling continuous data streams [10]. Consequently, for applications involving streaming data exhibiting potential non-stationarity, online learning approaches are imperative.

To identify effective online prediction mechanisms for the given dataset, a range of algorithms capable of incremental learning were evaluated. The selection aimed to encompass diverse methodologies, including efficient tree-based techniques, linear models, ensemble methods robust to concept drift, and deep learning architectures, thereby facilitating a comprehensive assessment of their suitability for the prediction task. The evaluated algorithms include:

1) *Hoeffding Trees*: An incremental decision tree algorithm designed for high-volume data streams. It utilizes the Hoeffding bound to make statistically significant split decisions with a limited number of samples, enabling real-time learning.

$$\epsilon = R\sqrt{\frac{\ln \frac{1}{\delta}}{2N}} \tag{1}$$

In Equation (1), ϵ represents the Hoeffding bound, quantifying the number of samples N needed to estimate the true mean of a variable within ϵ accuracy with probability $1 - \delta$. R denotes the range of the random variable. This bound allows the tree to determine split points efficiently.

2) *Adaptive Random Forest (ARF)*: An online ensemble method leveraging multiple Hoeffding Trees to enhance predictive accuracy and robustness [11]. ARF incorporates mechanisms to detect and adapt to concept drift by monitoring the performance of individual trees and replacing those deemed to be underperforming on recent data.

3) *Passive-Aggressive Regressor (PAR)*: An online linear model algorithm belonging to the family of margin-based algorithms, suitable for regression tasks [12]. It adopts a conservative update strategy, adjusting its weight vector only when the prediction error for an instance exceeds a predefined sensitivity margin (ϵ), balancing model stability with responsiveness to significant errors.

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + C \cdot L(\mathbf{w}; (\mathbf{x}_t, y_t)) \tag{2}$$

$$L(\mathbf{w}; (\mathbf{x}_t, y_t)) = \max(0, |y_t - \mathbf{w} \cdot \mathbf{x}_t| - \epsilon) \quad (3)$$

The update rule in Equation (2) minimizes the change to the weight vector $\mathbf{w}_t$ while correcting the error on the current instance $(\mathbf{x}_t, y_t)$, controlled by the regularization parameter C. The hinge loss function L in Equation (3) defines the error only when the prediction $|y_t - \mathbf{w} \cdot \mathbf{x}_t|$ exceeds the insensitivity margin ϵ.

4) *Feedforward Neural Network (FNN)*: A standard deep learning architecture was adapted for the online setting using deep_river. The implementation utilized sequential learning, processing instances individually. The network weights were updated using an online optimizer (SGD) based on a suitable loss function (Mean Squared Error)

All online algorithms were implemented using the Python libraries `river` [13] and `deep_river` [14].

D. *Model Training*

Due to a lack of access to the data center, the dataset itself was passed to the producer to simulate real-time streaming of data. Before model training, the data were preprocessed using a Standard Scaler, which normalizes each feature by removing the mean and scaling to unit variance (mean = 0, standard deviation = 1). The pipeline, as shown in Fig. (3), used an online evaluation strategy using Apache Kafka running in simulation mode to pass the data from the test dataset to the online learning models. A rolling validation scheme was used to assess performance over successive time windows, calculating average prediction error every 100 iterations, ensuring that the models remained accurate and were able to handle issues like concept drift correctly.

IV. Results

A. *Deep Learning*

For our deep learning model, we implemented a feed-forward neural network. This model comprised an input layer and three hidden layers with 128, 64, and 32 neurons, respectively, each followed by a ReLU activation function. The SGD optimizer and MSE loss function were also utilized in the model.

Deep learning models, while capable of eventually matching the performance of online machine learning counterparts, were found to be suboptimal in our experiments. They were found to take significantly more time to reach similar accuracy levels of their online machine learning counterparts, making them impractical for real-time or production environments. They exhibit limited adaptability to evolving patterns, resulting in performance degradation over time. Hence, their use in dynamic settings is less favorable compared to more lightweight and adaptive online learning models.

Fig. (3) shows the RMSE of the model calculated every 100 iterations for over 20,000 iterations. After this point, the accuracy improvements reduce significantly. The final calculated RMSE value after 20,100 iterations was 1.80°C.

Online Long Short-Term Memory (LSTM) models were also explored as an alternative approach for real-time temperature prediction in data centers. While LSTMs are theoretically well-suited for capturing complex temporal dependencies in streaming sensor data, their practical implementation revealed several challenges. Unlike ensemble methods like ARF that can adapt quickly to concept drift, LSTMs require substantial amounts of sequential training data to properly learn the underlying time-series patterns. This is because LSTMs rely on learning long-term dependencies through their gated architecture (forget, input, and output gates), which necessitates extended exposure to evolving data distributions before achieving stable performance.

In our experiments, the online LSTM model eventually matched the prediction accuracy of other methods, but converged significantly slower—often requiring 2-3x more data points to stabilize its error rates. This lag in adaptation poses a problem in dynamic data center environments, where rapid workload changes demand near-instantaneous model adjustments. Additionally, LSTMs incurred higher computational overhead during both training and inference due to their sequential processing nature, making them less suitable for low-latency streaming applications compared to tree-based ensembles.

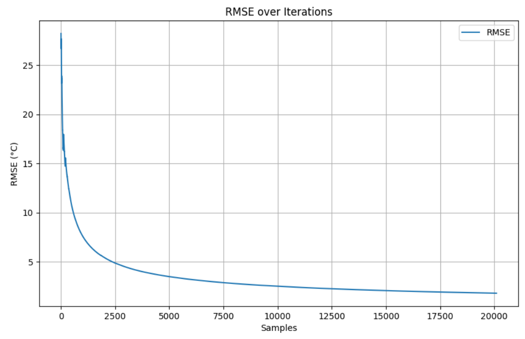

Fig. 4. RMSE over time for Deep Learning

B. *Online Machine Learning*

We tried multiple online learning models from the river library in Python including the Adaptive Hoeffding Tree Regressor, Passive Aggressive Regressor, and the Adaptive Random Forest Regressor.

The best performance was achieved by the Adaptive Random Forest Regressor with a peak RMSE value of 0.7071 °C over just 10,000 iterations.

1) Passive Aggressive Regressor: The Passive Aggressive Regressor (PAR) demonstrates unique advantages in the context of real-time temperature prediction, balancing stability and adaptability. With its regularization parameter (C) optimized to 0.01, PAR achieves a peak RMSE of 0.98°C—one of the lowest observed errors in our experiments—while maintaining consistent performance (RMSE 1.00°C) throughout the streaming evaluation. This stability stems from PAR's hybrid learning strategy: unlike purely incremental models that start from scratch, PAR first undergoes an initial batch training phase on historical data to establish a robust baseline model. Once deployed in the online phase, it then fine-tunes its weights iteratively via aggressive updates—only adjusting parameters when mispredictions occur, and doing so in a way that minimally perturbs the model (controlled by C).

Fig (4) highlights PAR's rapid adaptation to concept drift. When exposed to new test data, the RMSE initially spikes—reflecting momentary model mismatch—but quickly stabilizes as PAR reactively corrects its weights using the passive-aggressive optimization rule.

PAR's update mechanism is very effective for gradual drift scenarios (e.g., slowly rising ambient temperatures), where incremental weight adjustments suffice to maintain accuracy. However, its performance may degrade under abrupt shifts (e.g., sudden server failures), where structural model changes (like tree replacement in ARF) are more effective. This makes PAR ideal for environments with predictable, smooth variations but less suited for highly volatile workloads.

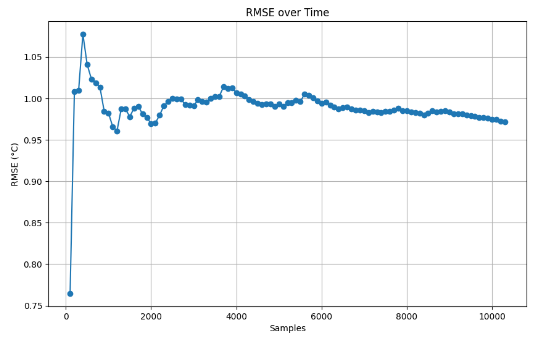

Fig. 5. RMSE over time for Passive Aggressive Regressor

2) Hoeffding Adaptive Tree Regressor: The Hoeffding Adaptive Tree Regressor (HTR) represents a specialized approach for streaming regression that combines the efficiency of incremental decision trees with built-in mechanisms to handle concept drift. Unlike batch-trained trees, HTR grows its structure on-the-fly using the Hoeffding bound—a statistical guarantee that ensures split decisions (e.g., choosing between features like CPU load or fan speed) are made with high confidence after observing a minimal number of samples. This bound dynamically adjusts based on data variance, allowing the tree to split nodes only when sufficient evidence is accumulated, thus balancing model complexity and generalization. The adaptive extension introduces two major benefits over traditional Hoeffding Trees by integrating Drift Detection using Adaptive Windowing (ADWIN), monitoring prediction errors at each node and pruning or replacing the affected subtree if drift is detected. In our tests, HTR achieved a peak RMSE of 1.78°C after 12,000 iterations—slower to converge than PAR but more robust to abrupt drift. HTR excels in memory-constrained settings but its performance settings may depend on drift detection sensitivity—overly aggressive pruning may discard valid long-term patterns.

3) Adaptive Random Forest Regressor: The Adaptive Random Forest Regressor achieves an peak RMSE value of 0.70°C

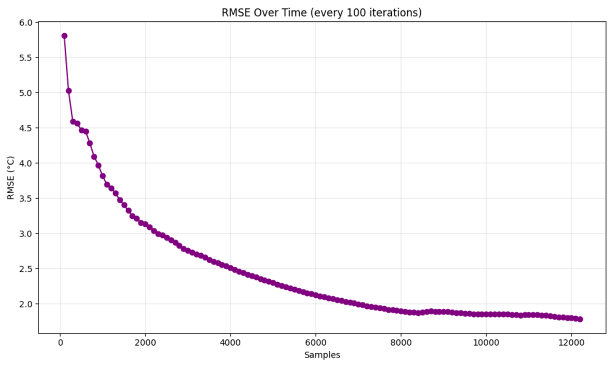

Fig. 6. RMSE over time for Hoeffding Adaptive Tree Regressor

after just 10,000 iterations and stabilizes shortly after around the same value.

This ensemble learning algorithm extends the traditional Random Forest framework to the online learning setting, maintaining an ensemble of decision trees that are continuously updated as new data arrives. Each tree in the ensemble independently processes incoming instances, and the final prediction is obtained by aggregating the predictions of individual trees.

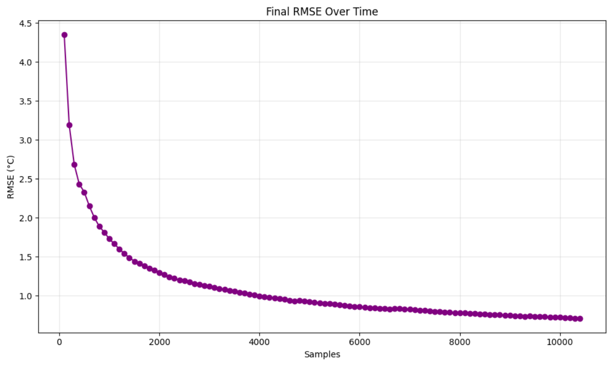

Fig. 7. RMSE over time for Adaptive Random Forest Regressor

In this study, an ensemble size of 20 trees was used to balance accuracy and speed. To manage the complexity of each individual tree and to prevent overfitting, each tree was limited to a maximum depth of 15. The predictions from the ensemble were aggregated using the median which is known to be more robust to outliers compared to the mean. The model also uses a model decay factor of 0.5 to reducing the effect of older, potentially less relevant trees and replacing them with new ones and also adds a drift detection mechanism (with a significance level of 0.001) to detect changes in the underlying data distribution and adaptation processes like pruning and regrowing to maintain accuracy as the data changes.

The model's success is primarily due to its sophisticated drift handling capabilities - where the Hoeffding Adaptive Tree relies on a single tree structure that must be pruned and regrown during concept drift (causing temporary performance drops), ARF's ensemble approach allows for gradual, parallel adaptation where only affected trees are modified while others maintain stability, letting ARF achieve lower error rates and faster convergence.

Hence, we see that ARF outperforms both HTR and PAR by combining ensemble robustness with spohiticated drift adaptation maintaining accuracy even during concept drift through parallel tree updates and using median aggregation to reduce outlier senstivity.

V. Conclusion

We have investigated the critical challenge of accurate temperature prediction in hyper-scale cloud data centers by evaluating various online machine learning models integrated with real time streaming using Apache Kafka. Our comparative analysis of various online learning algorithms, including Hoeffding Trees, Passive Aggressive Regressor, Adaptive Random Forest, and online Deep Learning models, revealed the superior performance and adaptability of the Adaptive Random Forest (ARF) regressor. The ARF model achieved an average predictive error of 0.70°C, matching the accuracy of existing static models while demonstrating the crucial ability to continuously learn and adapt to evolving data center resource usage patterns without requiring computationally expensive retraining.

The integration with Apache Kafka facilitated the seamless processing of streaming sensor and resource usage data, highlighting the feasibility of deploying such a system in a real-time data center environment. While deep learning models were explored in an online setting, our findings suggest that for this specific application and dataset, ensemble-based online machine learning methods like ARF offer a more practical and efficient solution due to their rapid training times and inherent mechanisms for handling concept drift.

Future work will focus on several key areas. Firstly, we aim to explore more sophisticated drift detection and adaptation strategies within the ARF framework to further enhance its robustness to abrupt and gradual changes in workload. Secondly, we plan to investigate the impact of different feature engineering techniques and feature selection methods on the performance of online learning models. Thirdly, deploying and evaluating the proposed system on a live data center environment will be a crucial next step to validate its effectiveness and scalability in real-world conditions. Finally, exploring the potential of incorporating prediction uncertainty estimates from the online models into dynamic resource management and cooling control systems represents a promising avenue for optimizing energy efficiency and system reliability in cloud data centers.

References

[1] IEA (2024), Electricity 2024, IEA, Paris https://www.iea.org/reports/electricity-2024, Licence: CC BY 4.0

[2] ”A guide to ensuring your ups batteries do not fail from ups systems”, 2018, [online] Available: http://www.upssystems.co.uk/knowledge-base/the-it-professionals-guide-to-standby-power/part-8-how-to-ensure-your-batteries-dont-fail/.

[3] Kaicheng Zhang, Seda Ogrenci-Memik, Gokhan Memik, Kazutomo Yoshii, Rajesh Sankaran, and Pete Beckman. 2015. Minimizing Thermal Variation Across System Components. In Proceedings of the 2015 IEEE International Parallel and Distributed Processing Symposium (IPDPS ’15). IEEE Computer Society, USA, 1139–1148. https://doi.org/10.1109/IPDPS.2015.37

[4] Park I, Kim HS, Lee J, Kim JH, Song CH, Kim HK. Temperature Prediction Using the Missing Data Refinement Model Based on a Long Short-Term Memory Neural Network. Atmosphere. 2019; 10(11):718. https://doi.org/10.3390/atmos10110718

[5] S. Ilager, K. Ramamohanarao and R. Buyya, ”Thermal Prediction for Efficient Energy Management of Clouds Using Machine Learning,” in IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 5, pp. 1044-1056, 1 May 2021, doi: 10.1109/TPDS.2020.3040800. keywords: Temperature distribution;Data centers;Predictive models;Cloud computing;Data models;Temperature sensors;Cooling;Cloud computing;machine learning;energy efficiency in a data center;datacenter cooling;hotspots,

[6] Pedro Domingos and Geoff Hulten. 2000. Mining high-speed data streams. In Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD ’00). Association for Computing Machinery, New York, NY, USA, 71–80. https://doi.org/10.1145/347090.347107

[7] Bifet, Albert & Gavaldà, Ricard. (2007). Learning from Time-Changing Data with Adaptive Windowing. Proceedings of the 7th SIAM International Conference on Data Mining. 7. 10.1137/1.9781611972771.42.

[8] Khan, Faisal. (2021). Apache kafka with real-time data streaming.

[9] Adrian Bazurto, Danny Torres, Víctor Asanza, Rebeca Estrada, May 21, 2021, ”Data Server Energy Consumption Dataset”, IEEE Dataport, doi: https://dx.doi.org/10.21227/x6jw-m015.

[10] Iwashita, Adriana & Papa, João. (2018). An Overview on Concept Drift Learning. IEEE Access. PP. 1-1. 10.1109/ACCESS.2018.2886026.

[11] AlQabbany, A.O.; Azmi, A.M. Measuring the Effectiveness of Adaptive Random Forest for Handling Concept Drift in Big Data Streams. Entropy 2021, 23, 859. https://doi.org/10.3390/e23070859

[12] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online Passive-Aggressive Algorithms. J. Mach. Learn. Res. 7 (12/1/2006), 551–585.

[13] https://riverml.xyz/latest/api/overview/

[14] https://online-ml.github.io/deep-river/

[15] Bozkurt, A., Ekici, F., & Yetiskul, H. (2023). Utilizing Flink and Kafka Technologies for Real-Time Data Processing: A Case Study. The Eurasia Proceedings of Science Technology Engineering and Mathematics, 24, 177-183. https://doi.org/10.55549/epstem.1406274

[16] Pulicharla, Mohan Raja & Premani, Varsha. (2024). Optimizing Real-Time Data Pipelines for Machine Learning: A Comparative Study of Stream Processing Architectures. World Journal of Advanced Research and Reviews. 23. 1653-1660. 10.30574/wjarr.2024.23.3.2818.

[17] https://www.datacenterfrontier.com/machine-learning/article/55252311/scorecard-looking-back-at-data-center-frontiers-2024-industry-predictions