Edge Analytics Solution for Online Monitoring of Building Operations Data

Warren R. Williams
Engineering Research and Development Center
U.S. Army Corps of Engineers
Vicksburg MS, USA
warren.r.williams2@usace.army.mil

Raahul C. Pathak
Department of Computer Science and Engineering
Mississippi State University
Starkville MS, USA
rahuul.pathak@gmail.com

2023 10th International Conference on Internet of Things: Systems, Management and Security (IOTSMS) | 979-8-3503-9389-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/IOTSMS59855.2023.10325815

***Abstract*— Edge based analytics solutions are an essential part of the continued shift towards creating efficient and secure IoT systems to support growing needs across every industry. These analytics often utilize machine learning models to provide informed feedback on the data being generated. As these IoT and edge systems continue to grow, the amount of data being generated is also growing exponentially. Utilizing compute heavy analytics poses many difficulties, especially in harsh environments that restrict the amount and quality of resources available to these systems. To address this, some proposed analytics solutions allow for more efficient use of these restricted systems. This paper is an investigation and implementation of one of these solutions, the QUANTS monitoring algorithm, on building operations data. This data is from many types of sensors throughout a buildings HVAC system and provides a variety of different data types to allow the QUANTS algorithm's capability with heterogenous data to be displayed. Alterations to the proposed algorithm made during this investigation were made to allow for a more decentralized method. This evaluation of the QUANTS algorithm demonstrates an efficient use of simple resources to support online monitoring and anomaly detection of heterogenous data.**

Keywords*— *Edge Computing, Internet of Things (IoT), Edge Analytics, Anomaly Detection, Predictive Maintenance, Online System Monitoring.

I. Introduction

In this paper an evaluation of an edge monitoring solution is performed utilizing building operations data. The algorithm is found to monitor its clients with a significantly low demand on computing resources. The algorithm's dynamic sampling method produces detections with a similar distribution to the detections that are found locally to the clients with full observation. This monitoring solution provides anomaly detection with low cost to network and computing resources that can be limited across IoT and edge networks. These types of networks are commonly utilized to provide increased capabilities across smart systems.

Since its first emergence with content delivery networks (CDNs) edge computing has consistently pushed to change the ways in which we handle our data [1]. This has only become more apparent as society continues to create more "smart" systems across every facet of life and industry. These systems create data rich environments that are the perfect fields to grow and explore further capabilities of edge computing solutions. Edge computing at its core is the concept of pushing the computation and analysis of data as close to the edge of a network as possible. The edge of a network is its furthest extensions where data is generated and gathered. The ideas and frameworks under edge computing allow for increases in efficiency and security by reducing the amount of data transferred and shortening the distance that transferred data is traveling [2]. The large collection of small devices that make up the edges of networks that support the various smart systems previously would not have been considered computers or part of the network at all, and these types of devices have become what is generally referred to as the Internet of Things (IoT) [3]. The data rich networks that have been created by the IoT support the continued advancement of edge computing capabilities. Decentralizing data analytics with edge computing solutions would increase the redundancy across these systems, as well as decrease the demand on network infrastructure caused by the movement of data towards the centralized locations that currently perform the bulk of the computation work for conventional data analytics.

One area that effectively displays the progression of smart systems is the management of smart homes, offices, and facilities. There is always the novelty of being able to control your home's locks, lights, or even entertainment systems from your phone without even being in town. However, there is also the possibility for significant increases in efficiency that these systems can provide to these buildings. Smart heating, ventilation, and air conditioning (HVAC) systems now have an increased number of thermal sensors throughout the entirety of a building to more accurately direct air effectively. They incorporate occupancy sensors that can allow for the prioritization of utilized areas. This can lower costs by letting unoccupied zones be set to a minimum setting instead decreasing the demand on the system, and it can increase the overall effectiveness of the system by freeing more work capacity to be directed at providing the desired temperatures in the occupied zones [4][5].

To accomplish this type of functionality, especially across large buildings such as office complexes or industrial spaces, requires numerous sensors and control systems throughout an HVAC system. This large network can generate a lot of data and needs effective processes for monitoring across the whole system while performing the necessary data analytics.

This significant amount of data could be used to better inform facilities management personnel about the health of the system and status of the building. This would enable more effective operations and maintenance decisions from improved maintenance scheduling to more accurate predictions for the remaining useful life (RUL) of parts allowing replacements to be obtained and installed before significant problems occur [6]. Enabling these informed decisions is the goal of predictive maintenance, which is the next step in the modernization of maintenance techniques. Historically maintenance utilized a framework of reactive maintenance that only responded to problems as they arose. Then there was a shift to preventative maintenance, which is the current conventional standard across most industries and applications. This framework looks to provide maintenance regularly in such a way as to prevent any major issue from occurring. With the growing capabilities and information generated with edge computing and IoT methods, we are beginning to increase the possibilities for effectively predicting the necessity of various maintenance operations. Predictive maintenance allows for cost reductions as well as increased reliability and availability of systems and resources [7]. These predictions are to be informed by historical data, as well as data across the system as it operates. The methods for monitoring these online systems for useful information typically utilize some form of anomaly detection.

Anomaly detection is a data analytics method that helps identify deviations from expected behavior. Anomaly detection helps enables improved understanding of what is going on with a system. This data and information can provide useful feedback that can improve predictive maintenance capabilities and further inform maintenance decisions. These analytics are often done by utilizing machine learning (ML) models [8]. ML models can be time consuming to prepare because they require large datasets for training, and the training itself is a computationally heavy process. Additionally, as the application of sensors continues to expand and vary in these IoT systems, finding historical datasets for these applications that can be utilized for training may prove difficult. These models may also be trained for single datatypes, which would then require utilizing distinct models for each type of data throughout a system. This could exacerbate the time and computational demands for generating and utilizing ML models. The compute power needed to effectively run ML models may exceed the capabilities of the low powered devices that typically make up IoT and edge networks, and if the devices are intended to execute tasking beyond running these models, the performance of all the processes would be negatively impacted. There may also be limitations to the compute and network resources that are available due to other demands within the facility or harsh environmental factors. As an alternative to the use of ML models, some anomaly detection methods opt for statistics-based techniques instead. These methods typically reduce the demand for computational power to perform their analysis. The size of datasets prior to implementation may also be reduced compared to those that are needed for effective ML models.

As edge and IoT networks continue to enable the implementation of the smart systems that are emerging from every industry, the need to enable effective data analytics across these networks grows. The capability to perform effective analytics across these systems will allow for improved efficiency and operational management. As part of these analytics real-time anomaly detection is important for supporting predictive maintenance efforts and implementations.

The remainder of this paper will present an introduction of the anomaly detection technique utilized in our work in Section 2. An explanation of the resources and data that enabled our work is detailed in Section 3, and a description of the implementation of the described anomaly detection technique on the available resources is provided in Section 4. The resulting data and discussion will be presented in Section 5, and concluding thoughts and future work will be laid out in Section 6.

II. QUANTILE-BASED NONPARAMETRIC MONITORING AND THOMPSON SAMPLING (QUANTS) ALGORITHM

The increasing number of sensors and data generated across modern smart systems creates a few barriers to implementing conventional anomaly detection methods. The large amount of data can put strain on network resources as the monitoring algorithms pull data from every data source for analysis. The larger amount of data also increases the demand on computing resources to perform analysis on each of the data streams as well. This increased demand for resources could be handled in several ways. One method is to upgrade the available hardware resources to higher performance alternatives that would allow the system to perform at the levels necessary to keep up with managing the increased network traffic, monitoring, and data analysis. This can be financially costly, but also may include time and labor costs if other aspects of the system or facility need to be altered to function on the new infrastructure. Complications with limited available space to house the new hardware and other limitations due to hazardous environments could complicate this method as well. Another method for handling with this increased demand would be to use the existing resources more efficiently. This idea is one of the motivations that led researchers at the University of Wisconsin-Madison to develop the QUAntile-based Nonparametric Monitoring and Thompson Sampling (QUANTS) algorithm [9].

The QUANTS algorithm is developed to enable more efficient use of limited resources across online edge and IoT networks while effectively monitoring these systems. Instead of sampling data from all the data streams across the system simultaneously, the algorithm dynamically selects a subset of the available data streams to observe by sampling their data. This dynamic sampling would significantly lower the amount of data moving over the network, therefore avoiding any heavy network traffic across these systems. This method allows for more frequent observations to be made from the system. The more frequent observations enable detections of anomalous behavior to occur sooner, allowing faster response times to problems as they arise.

When anomalous behavior is detected, the algorithm then actively monitors the anomalous data streams to investigate the anomalous behavior. This allows for more data to be gathered about the detected behavior for verification and further analysis. Figure 1 shows an illustration comparing this dynamic sampling concept to the more conventional sampling method that samples every data stream at each interval. This illustration displays the potential difference of observation frequency between the two sampling strategies. The dynamic method samples at every interval, while the conventional method only samples every five intervals. Both methods sample the same amount of data on average, with the conventional method sampling the full set of data streams every five intervals, and the dynamic method sampling a fifth of the data streams at each interval. In the illustration the higher frequency allowed for the anomalies to be observed sooner by the dynamic sampling that is implemented by the QUANTS algorithm. The active and continued observation in reaction to detecting the anomalies is also illustrated.

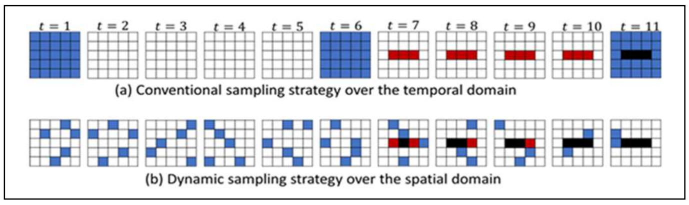
t = 1
t = 2
t = 3
t = 4
t = 5
t = 6
t = 7
t = 8
t = 9
t = 10
t = 11
(a) Conventional sampling strategy over the temporal domain
(b) Dynamic sampling strategy over the spatial domain

Fig. 1. Comparison of the QUANTS algorithm's dynamic sampling and conventional sampling. Each square in the 5x5 grid represents a data stream. White: not being sampled. Blue: being sampled. Red: anomalous but not being sampled. Black: anomalous and being sampled.

The data streams to be sampled are selected utilizing some statistics-based evaluation. The first step enabling this evaluation is the collection of in-control data. Historical data is not necessary for this step, the in-control data can be gathered from an active online system without requiring the system to be taken offline. This data should be gathered during normal operations at a time when the system can most likely be assumed to be in good health. The amount of in-control data necessary will vary depending on the application. Familiarity with the data of interest by the practitioner is useful in making this decision, and consideration of the sampling rate is also worthwhile. The in-control data is utilized to build a statistical model that represents the distribution of non-anomalous input data. Therefore, data and environments that are more susceptible to regular fluctuations will need an increased amount of in-control data to effectively represent these fluctuations.

Following the collection of in-control data, a series of quantiles is generated from it. These quantiles are used for the evaluation of incoming sampled data. The sampled data is compared to each of the quantiles to generate a matrix that represents if the sampled value is greater than or less than each of the quantiles, this matrix is referred to as an interval indicator. These interval indicators are utilized in some CUSUM analysis for the sampled data streams [14][15]. CUSUM analysis is a technique developed for monitoring the detection of changes. This analysis represents the current data as well as previous data, and specifically detects a deviation from the expected value. This analysis results in a value that is utilized to represent how anomalous the sampled data and the behavior of its respective data stream is. This analysis is done twice to ensure sensitivity to both upward and downward shifts in the data. The larger of the two resulting values is selected to represent how anomalous that data stream is and is referred to as its local statistic going forward.

Data streams that are not sampled generate an estimation of their expected data with a Bayesian update calculation. This method utilizes previously received data to infer what the next expected data point should be. This estimated value is then used to calculate an estimate local statistic with same process described previously. The data streams that have generated the largest statistics, from both sampled data and estimated values, are considered the most anomalous, and are then selected to be sampled for the next interval.

The QUANTS algorithm also proposes a global statistic, or monitoring statistic, that is used to represent the overall health of the system. This global statistic is generated by finding the summation of the largest local statistics. If the global statistic exceeds a set threshold, then an alarm is raised. Figure 2 provides a visual illustration of the QUANTS algorithm's workflow. The workflow starts at receiving the incoming sampled data and goes through the previously described steps to the selection of data streams to be sampled next. It also contains the ending condition when the alarm is raised.

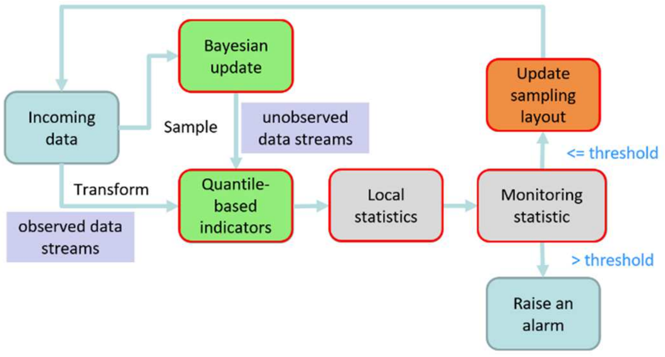

Fig. 2. Diagram of the QUANTS workflow.

Implementation of the QUANTS algorithm requires setting four parameters to enable its functionality. The first of these parameters specifies the number of intervals the in-control data is separated into when generating quantiles and is standardly denoted as "d". The performance variation is minimal when the number of intervals is within the range of ($5 \leqslant d \leqslant 15$). The second parameter is an allowance parameter that is denoted as "k". The allowance parameter is derived based upon the set "d" parameter as in (1). The allowance parameter is utilized during the calculation of local statistics for deciding when the CUSUM calculation should be restarted.

$$k < \min\left(\begin{matrix} \max\limits_{1\le i\le d-1}\left(\sum_{l=1}^{i}\frac{l^2}{d(d-1)}+\sum_{l=i+1}^{d-1}1-\frac{l}{d}\right), \\ \max\limits_{1\le i\le d-1}\left(\sum_{l=1}^{i}\frac{l}{d}+\sum_{l=i+1}^{d-1}\frac{(d-1)^2}{dl}\right)\end{matrix}\right) \tag{1}$$

The third parameter is the control limit and is denoted as "h". The control limit serves as the threshold for the monitoring statistic. When the monitoring statistic exceeds the threshold, it triggers the alarm. The control limit can be determined by simulations and bootstrap techniques. Further detail for this derivation can be found in these papers referenced within the original QUANTS paper [10][11]. The fourth parameter is denoted as "r" and is the number of local statistics that are used when calculating the monitoring statistic. The monitoring statistic is calculated by the summation of the top-r local statistics. The value for this parameter should be smaller than the number of data streams that are to be sampled with each iteration through the monitoring process. This is to help limit the amount of estimated data being used to represent the global health status of the system. Smaller "r" values allow for more robust systems.

The QUANTS algorithm allows for the detection of a variety of anomaly types with its statistics-based monitoring and sampling technique that relies on the data's distribution during healthy operation. It also allows for anomaly detection across systems with heterogenous data. Each data stream has its own collection of in-control data for comparison with current data values. The interval indicators that are used in the analysis contain only 1's and 0's from the comparison of sampled data to the quantiles. This allows the algorithm to remove the reliance on specific data types that would need to be accounted for in other anomaly detection methods. The algorithm's analysis is computationally lightweight when compared to other conventional methods that utilize ML models.

III. Materials and Methods

A. The Network

The network utilized in this paper's work is a closed IoT network that consists primarily of Raspberry Pi 4 Model B's with 4GB of RAM. The network is managed by a residential style wireless router, and all communication between the Pi's is done over that wireless connection. Many of the Raspberry Pi's within the network are outfitted with a PiSugar 2 Plus Battery that provides functionality comparable to an uninterrupted power supply (UPS). The PiSugar batteries come with their own board that connects under the Pi's board. PiSugar also provides tools that allow for various types of battery information and data to be accessed, such as the current battery level.

B. The Data

The building operations data that is utilized in the work described by this paper is HVAC data from a facility primarily consisting of offices and collaborative work areas. The data was collected through Schneider Electric EcoStruxure Building Operation software. Direct access to the live data from this system was not provided in compliance with security standards that keep the system isolated. Access to data archives from the system was provided in lieu of that. The data that is archived is a small subset of the large amount of data that the system monitors and is capable of recording. This subset consists of data from one of several air handler units (AHU) and two of its associated variable air valves (VAV). The specific data types that are being archived were not chosen with any intention to be informative to any specific analysis, but were selected at random as a proof of concept in testing the archive functionality of the system.

The archives contain temperature, pressure, active setpoints, and hardware status data from various locations throughout the AHU and VAVs. The archives consist of data points that are sampled at 15-minute intervals, and the full dataset spans over a 6 month time frame. The archives consisting of data across this long length of time allows the effects of seasonal weather changes to create some variation and noise in the data. Additionally, since the archive's data points consist of only values with timestamps, there is no significant indication as to what portion of the datasets could have been considered healthy for use in producing in-control data during experiments. To address concerns about this, the archive data was processed and filtered to remove an amount of outlier data. This filtered subset was then used to simulate in-control data during testing with QUANTS.

A clustering technique was selected to filter the archives. The Uniform Manifold Approximation and Projection (UMAP) technique is a dimension reduction algorithm that has emerged as a go-to clustering method. UMAP essentially constructs a high dimensional graph representation of the data then optimizes a low-dimensional graph to be as structurally similar as possible [12]. To utilize UMAP for clustering the archives a few parameters had to be set. The two primary parameters are the number of neighbors and minimum distance. The number of neighbors sets the number of connections used to create the initial high-dimensional graph, and the minimum distance sets the separation between points in the low-dimensional space [13]. The same parameters were set for each of the data types in the archives. The number of neighbors was set to 30, and the minimum distance was set to 0.

IV. QUANTS Implementation

The implementation of the QUANTS algorithm that is utilized in this paper consists of two primary Python scripts. One of which runs a server, and the other runs clients with the capability for a few variations at runtime. The standard configuration has the server handle most of the analysis and decision making. This includes selecting which data streams to sample, requesting data from the respective clients, and calculating the updated statistics and estimations. The server is also responsible for generating and evaluating the monitoring statistic. The server and client have local readouts that display recent data, statistics, and other current information. The client primarily collects data locally. The client initially collects and stores the set amount of in-control data. After that initial phase is completed, the client generates the quantiles and informs the server that it has finished this set up process. From there the client collects data and generates the interval indicator by comparing the sampled data with the quantiles. When the server requests data the client sends the interval indicator. This is a slight variation to the proposed algorithm's workflow, which has the interval indicators generated alongside the estimation and local statistic calculation. This was done to decentralize the workflow, allowing the clients to pick up some of the compute load. This alteration also allows this implementation to never send real data values away from the client.

The client allows for various configurations that can be flagged at runtime. The primary variation is the selection of the data type to be utilized by this instance of the client. This allows for the selection of all the data types available in the HVAC archives as well as data from the Raspberry Pi and PiSugar battery. When a client is set to utilize the archives, it iterates through the archive files at every time interval that would otherwise be sampling live data. This enables the client to simulate the HVAC data as if it were data from an online system.

The client also provides an option that skips the initial gathering of in-control data by utilizing a specified file containing some historical data to generate the quantiles instead. This alteration was implemented to save time during testing and to simplify the utilization of the filtered archive data. The client can also be set to calculate its own local statistic. This configuration does not change the data sent to the server, because it was implemented for comparison and verification of the server's findings. This alteration was strictly intended for enabling the evaluation of the dynamic sampling method in comparison with statistic results from data that was fully observed throughout the tests.

To evaluate the performance of this implementation, the server logs CPU utilization over every interval, and it logs "anomalies" anytime a local statistic exceeds a certain level. For the anomaly logs, the server logs the local statistic value, the respective client for this data stream, and a timestamp. The client similarly logs CPU utilization over every interval, and when the client is set to calculate its own local statistic, it will log "anomalies" in a similar manner to the server.

V. Results

The data gathered is accumulated over 10 tests of varying lengths of time and configurations. Only data types from the archives were utilized by the clients during these tests. This removed any necessary steps in considering that the original data from the archives was gathered at 15-minute intervals. The types of data being pulled from the archives were varied between the different clients during these tests to ensure the utilization of all the available data and prevent any notable performance changes originating from the data. The sever was set to sample 3 of the clients each interval, and the number of clients across the tests ranged from 3-10. Each client utilized during these tests was running a separate Raspberry Pi devices on the network. During six of the tests the server was running on a device that was also running a client, while the remaining four tests had the server running on its own device.

The CPU utilization remained very low which met the expectation for this algorithm and the statistics calculations that it employs. In several runs there were notable outliers for max utilization, so instead the 99th percentile is the metric presented here. The minimum similarly had various readings too low that returned zero in the logs, so for a more representative metric the 5th percentile is presented in these results. The mean and median values are also provided in the following. Figure 3 shows these metrics for the percentage of CPU utilization by all the clients across all the test runs. The y-axis of this figure is showing the percent values of CPU utilization, but as the values were so low only the bottom ten percent is shown.

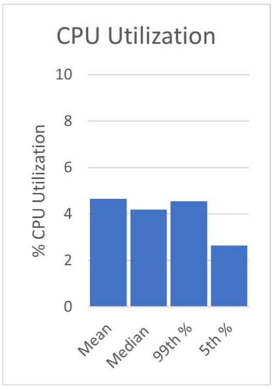

Fig. 3. CPU utilization metrics for the clients across all tests.

Removing results from the clients that were operating on the same system as the server had a negligible effect on the CPU utilization metrics for the clients.

The server performed with CPU utilization metrics lower than those of the clients. The metrics for the CPU utilization of the server across all the tests is shown in Figure 4 (left). However, where there was no notable change for the clients CPU metrics after removing the data from clients that were run alongside a server, there was a notable decrease in the metrics for the server after removing those runs from the server's results. The metrics for these runs are also shown in Figure 4 (right). The metrics for CPU utilization when the server is the only demand on the system is nearly half that of the overall metrics. As with the previous figure, the y-axis in this figure is representing from zero to 10 percent CPU utilization.

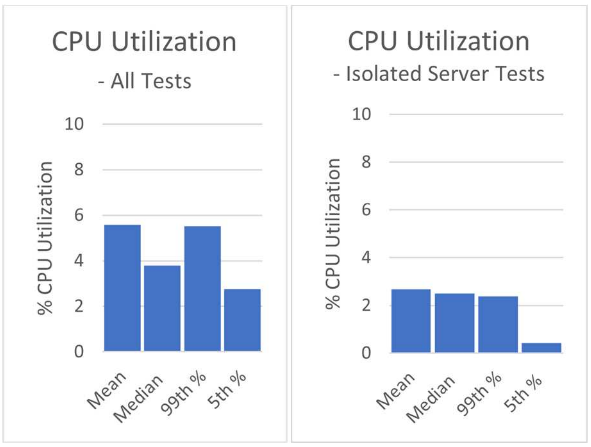

Fig. 4. The server CPU utilization metrics for all runs is shown on the left. The metrics for runs with only isolated server runs is on the right.

The server and clients were set to consider a local statistic of greater than 10 to be anomalous and to log that data. This is a low bar to be considered anomalous and likely includes minor outlier data that is not indicative of any true anomalous behavior. This is considered low because the resulting values from the CUSUM analysis are incremental, and for this configuration 10 is between the second and third increments. However, this threshold was set for two main reasons. First the algorithm was initially intended to be very sensitive to slight shifts, and so setting this low bar keeps with this original intention for the algorithm. The second reason is to increase the usefulness of the resulting data. During further analysis of these results the lower statistic values could be filtered out since the actual statistic values are part of the logged data. In practical applications this threshold could be adjusted to make the algorithm less sensitive and decrease the size of any comparable logs.

All the anomaly detection results that are presented below come from the longest test run with the clients logging, which was the tenth run. The results from this run are comparable to the of the results received on other test runs. This run provided the most data to visualize and was therefore selected to represent the overall results.

The comparison of the number of anomalies found by the clients locally and the number found by the server as it was dynamically sampling across the clients is provided in Figure 5. The server missed a notable number of anomalies that the clients flagged. This could be partially due to the high sensitivity of the logging methodology that was previously explained. Another possibility is an observed behavior where the algorithm would sometimes go long spans of time without sampling a specific client. A possible resolution to help prevent this behavior in the future would be to implement an alteration to the estimation of unsampled data streams that, after some length of time being unsampled, it would alter the estimation to include some amount of deviation that would cause the algorithm to sample the data stream. However, even with the anomalies that were missed, the distribution of anomalies logged by the server follows a similar distribution to those logged on the clients themselves.

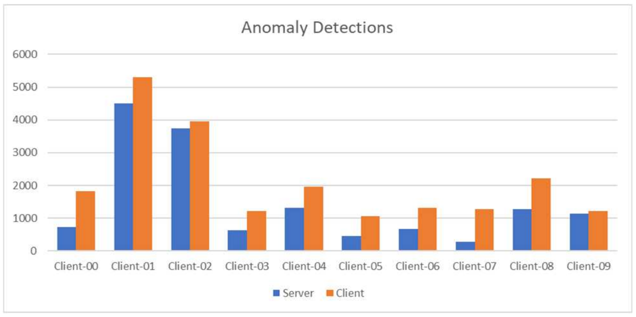

Fig. 5. Comparison of the number of anomalies detected locally by clients and by the server's dynamic sampling.

The difference in anomalies counted by the server and the clients was calculated. The clients with the largest difference, smallest difference, and the difference closest to the mean difference were selected for closer investigation.

The logged anomalies for the largest difference between the client and server counts is shown in Figure 6. This data is displayed with a scatter plot that shows anomalies over time. The local statistic value is the y-axis, and the anomalies are plotted with their timestamps on the x-axis. The test run being presented covers a 21-hour time span, and so only a subsection of these plots are presented here.

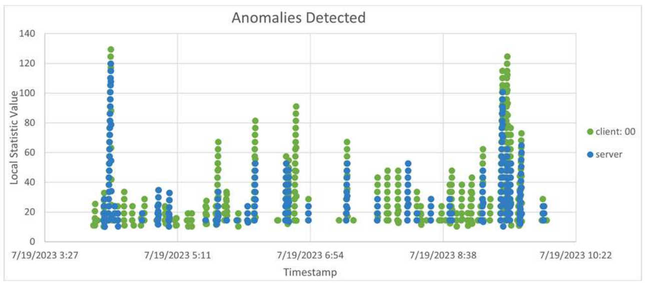

Fig. 6. Anomalies detected for the client with the largest difference in server and client detections.

The client with the smallest difference in anomalies detected between the client and server is shown in Figure 7. The structure for the plot is the same as the previous, only containing a subsection of the full plot.

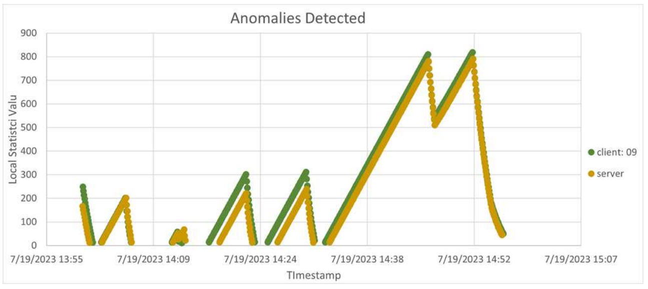

Fig. 7. Anomalies detected for the client with the smallest difference in server and client detections.

The client with the amount of difference between anomalies detected on the server and client that was closest to the mean difference is shown in Figure 8. The plotting method is consistent with the previous, only presenting a subsection of the plot.

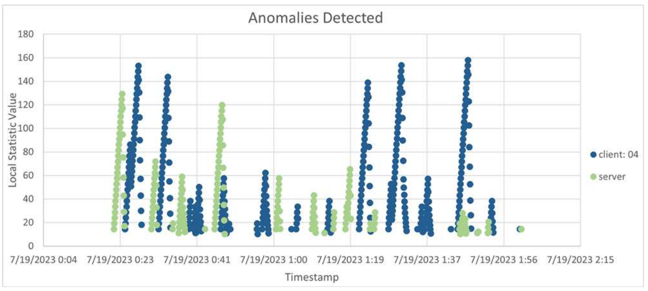

Fig. 8. Anomalies detected for the client with the difference in server and client detections that was closest to the mean difference.

VI. Conclusions

Edge and IoT networks continue to enable the advancement and growth of smart systems across a variety of fields and industries. These networks and systems generate data rich environments, and enabling efficient data analytics within these networks will allow for more informed and effective decisions. Utilizing techniques such as the QUANTS algorithm to monitor these systems is one way to enable analytics within these environments. The data resulting from these analytics will enable future predictive maintenance and expanded operations capabilities.

The results from this implementation of the QUANTS algorithm demonstrate the techniques' ability to maintain low CPU utilization while effectively monitoring a system. The monitoring results provided an effective representation of the anomalous behavior across each client or data stream. The lightweight nature of this algorithm and its ability to limit network traffic make it a good fit for utilization across IoT networks and smart systems. The alterations made to the proposed algorithm kept the primary functionality consistent, but provided some decentralized capabilities that would add some redundancy to potential applications. The alterations also demonstrated the algorithms capability to allow for variations that could further expand its applicability.

Further analysis of the QUANTS algorithm in direct comparison to other anomaly detection and monitoring methods was an intended as part of this evaluation, however delays in those efforts led to its omission. An investigation of various anomaly detection methods across comparable testing configurations would allow for the identification of the advantages and disadvantages of each method. This evaluation would provide significant value to practitioners during the design of various IoT systems.

Future development into this implementation of QUANTS could be done to perform evaluations on even larger systems, to evaluate its scalability. Application of this algorithm across a full smart HVAC system instead of a simulated subsection of the system would provide a more informative evaluation for practitioners. Enabling clients to read in multiple data types instead of requiring a new client for each would expand the capabilities for utilization across more complex and distributed systems. Possibly implementing the QUANTS monitoring methodology in a nested form, having the clients monitor multiple data streams locally and then a server performing the monitoring across the full system. A comparison in the effectiveness of the monitoring scheme against various anomaly detection algorithms across the same data would also provide an interesting look at the applicability of the algorithm.

REFERENCES

[1] Edge Computing, Hewlett Packard Enterprise, https://www.hpe.com/us/en/what-is/edge-computing.html, last accessed 2023/07/16.

[2] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge Computing: Vision and Challenges," in *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637-646, Oct. 2016, doi: 10.1109/JIOT.2016.2579198.

[3] Rose, Karen, Scott Eldridge, and Lyman Chapin. "The internet of things: An overview." *The internet society (ISOC)* 80 (2015): 1-50.

[4] Sampa K. Smart CRE. https://smart-cre.com/what-is-smart-hvac-the-advantages-and-disadvantages/ last accessed 2023/07/19.

[5] Ambort Lilli. University of Minnesota, Institute on the Environment. https://environment.umn.edu/education/susteducation/pathways-to-renewable-energy/how-a-smart-hvac-system-can-increase-efficiency-while-saving-you-money/ last accessed 2023/07/20.

[6] McKinsey & Company. https://www.mckinsey.com/capabilities/operations/our-insights/a-smarter-way-to-digitize-maintenance-and-reliability. Last accessed 2023/07/19.

[7] Ran, Yongyi, et al. *A Survey of Predictive Maintenance: Systems, Purposes and Approaches*. 1, arXiv, 12 Dec. 2019. *arXiv.org*, https://doi.org/10.48550/arXiv.1912.07383.

[8] G. Muruti, F. A. Rahim and Z. -A. bin Ibrahim, "A Survey on Anomalies Detection Techniques and Measurement Methods," *2018 IEEE Conference on Application, Information and Network Security (AINS)*, Langkawi, Malaysia, 2018, pp. 81-86, doi: 10.1109/AINS.2018.8631436.

[9] H. Ye and K. Liu, "A Generic Online Nonparametric Monitoring and Sampling Strategy for High-Dimensional Heterogeneous Processes," in *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 3, pp. 1503-1516, July 2022, doi: 10.1109/TASE.2022.3146391

[10] K. Liu and J. Shi, "Objective-oriented optimal sensor allocation strategy for process monitoring and diagnosis by multivariate analysis in a Bayesian network," *IIE Trans.*, vol. 45, no. 6, pp. 630–643, 2013.

[11] W. Li and P. Qiu, "A general charting scheme for monitoring serially correlated data with short-memory dependence and nonparametric distributions," *IISE Trans.*, vol. 52, no. 1, pp. 61–74, 2020.

[12] McInnes, Leland, et al. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. arXiv, 17 Sept. 2020. *arXiv.org*, https://doi.org/10.48550/arXiv.1802.03426.

[13] A. Coenen and A. Pearce. *Understanding UMAP.* https://pair-code.github.io/understanding-umap/. Last accessed 2023/07/19.

[14] E. S. PAGE. *Continuous Inspection Schemes.* Biometrika, Volume 41, Issue 1-2, June 1954, Pages 100–115, https://doi.org/10.1093/biomet/41.1-2.100.

[15] Grigg OA, Farewell VT, Spiegelhalter DJ. *Use of risk-adjusted CUSUM and RSPRTcharts for monitoring in medical contexts.* Statistical Methods in Medical Research. 2003;12(2):147-170. doi:10.1177/096228020301200205