SAND2021-12503C

# Pattern and Anomaly Detection in UX (PADUX)

## A System to Capture and Utilize User Signals for Better UX

Joshua Mitchell and Alan Gross

josmitc@sandia.gov and ajgros@sandia.gov

***Abstract***—The rise of remote work in historically on-premise companies leaves less margin for error in software deployment processes. Further, it requires that developers and system administrators obtain automatic feedback on deployment success or failure more quickly than in the past. How can DevOps-oriented teams, concerned about the user experience of the workforce, receive more immediate, data-driven feedback to reduce outages, know when to perform rollbacks, and create more robust deployment strategies overall? The User Experience Solutions department at Sandia National Laboratories is working on such a tool, proposing that an automated outage detection system based on web analytics software and real time message streaming could be a key to identify widespread user experience issues.

## 1 INTRODUCTION

Many industries have historically optimized enterprise IT for an on-premise user experience. However, a sudden acceleration of remote work situations due to the COVID-19 pandemic have increased the overall burden on these "brick-and-mortar-first" enterprise IT organizations. The question of how remote workers experience enterprise IT is crucial, as these knowledge workers often cannot deliver on their mission if IT systems are not functioning effectively. Measuring and responding to negative IT experiences (and responding at mission speed) is therefore an important capability. This paper outlines an experimental system prototyped to address such a problem. First, we briefly explore the problem space, focusing on some of the challenges IT organizations face with a larger-than-normal remote population. Second, we outline how traditional IT monitoring occurs using a signal-and-measure approach (i.e. software that imitates a client interacting with a web site) and that, while there is great value in this, we hypothesize about a more granular approach that involves capturing and measuring the signals originating from *real* clients. Third, we lay out a system architecture (PADUX) that was built to digest these client signals and produce real time insights. Fourth, we lay out the details of how the underlying PADUX algorithm functions and the resulting model from this algorithm. Fifth, we discuss the results of our experiment by analyzing historical Major IT Incidents (MITIs) against PADUX data and discuss the accuracy of the prototype and its potential uses.

## 2 PROBLEM SPACE

Remote work among knowledge workers has been on the rise for years, but many industries, especially prior to the COVID-19 pandemic, maintained a primarily "brick-and-mortar" approach in terms of knowledge worker locality. The arrival of the pandemic sent many of these workers into remote roles, placing strain on systems (such as VPNs) that might not have been optimized for this increased load. Enterprise IT organizations, such as the one these authors belong to, scrambled to reconfigure resources for this new, remote dynamic.

Workers who shifted to home offices also face an increased number of variables in accessing enterprise IT systems. There are now multiple ISPs, geographies, time zones and weather patterns involved (consider hurricanes on the US east coast and forest fires on the US west coast). There is also non-standard equipment, such as personal routers and modems, to consider. Remote work, with its many upsides, contains many variables. This creates a complex environment to navigate.

The combined shift in burden from on-premise IT systems to remote IT systems, along with the increased complexity of a remote worker's environment, indicates that many organizations might struggle to maintain a good user experience for knowledge workers. The impact of such a struggle could be serious: the poorer the collective user experience of enterprise IT, the greater the threat to the workforce's ability to deliver on its mission effectively. The challenge has never been greater to deliver excellent enterprise IT.

In order to handle these complexities, enterprise IT organizations must be aware of how their deployments and systems are faring. In the past, the colocation of employees made identifying issues easier. However, because of the distribution of employees across multiple geographies and time zones, it can be difficult to determine if a few calls to a helpdesk represent an isolated incident or a broader pattern. The PADUX system was undertaken with all of these concerns in mind.

## 3 HYPOTHESIS AND GOALS

Here we outline the fundamental hypothesis of PADUX as well as the encompassing goals we attempted to achieve through deployment of the system.

### 3.1 System Access as a Foundational UX Task

When considering whether enterprise IT systems are usable, we should first consider this fundamental definition of usability from ISO 9241-11:2018: "[the] extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" (International Standards Organization). The scope of the PADUX project involves a prerequisite to all other usability considerations: system access. The basic idea is that user access to a system is the first step in measuring their experience. If users cannot access the system, the remaining measurements do not matter.[1]

### 3.2 Hypothesis

Our hypothesis can be stated as follows:

> (1) If the burden of sending real time signals that mirror user behavior can be acquired by client systems and (2) that signal pattern can be translated into a model, then anomalies in basic system access can be detected and used to determine if system

[1] Note that by "system access" we are *not* referring to authentication and authorization mechanisms. Instead we are referring to whether a user is simply able to reach the outmost, expected bounds of a system.

access by users is currently impaired or has been impaired in the past.

### 3.2.1 *Assumptions*

Our hypothesis depends on two assumptions. The first assumption is that clients can take on the burden of signaling in real time. This requires that each system broadcasts a signal to a listening endpoint at some level of granularity great enough to accurately state that the client is connected (or "up", as the common vernacular goes). This assumption itself presumes the broadcast signals mirror real user behavior. The second assumption is that these signal patterns can be modeled in such a way that an anomaly (e.g. weak client signal patterns) can be detected in real time and then be utilized to identify a system outage.

The first assumption, that real time signals can be sent from clients, is answered by the use of a broadly adopted, open source, web analytics tool called Matomo (https://matomo.org). Matomo, through embedding of a short JavaScript snippet into client code, measures user behavior at a highly granular level. Further, it was implemented as an enterprise service within our organization and broadly adopted into client applications. Because Matomo tracks extremely granular user behavior such as clicks, input field values, out links, page load times, and many other details, this satisfies the second part of the first assumption.

The second assumption, that these signal can be translated into a model, is satisfied because a high volume of data was captured over a several year period, allowing for some consistent behavior modeling. For example, in Figures 1A and 1B below, two different dates of user behavior are modeled. The full explanation of how to interpret these figures is discussed in section 5, *Algorithms.* Suffice to say, Figures 1A and 1B represent signal levels of approximately fifteen-thousand users interacting with organizational enterprise IT systems over the course of one day.

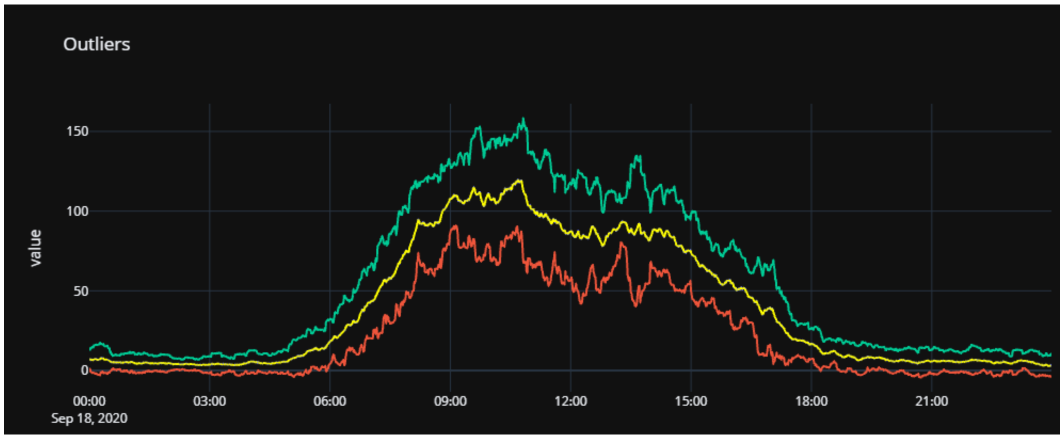

***Figure 1A***—Visitor patterns for September 18, 2020. The yellow (middle) line is the 15-minute, rolling mean count of active visits, while the green (upper) and red (lower) lines represent 3 standard deviations from the mean, positive and negative respectively.

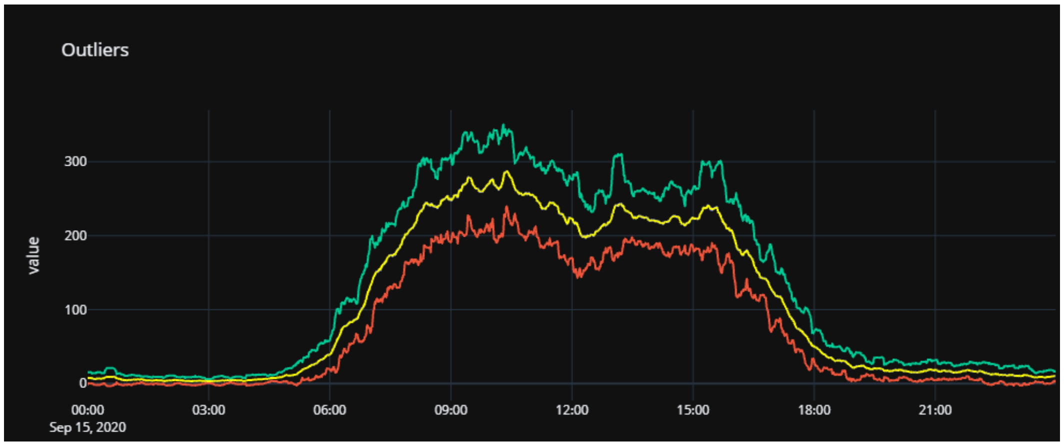

***Figure 1B***—Visitor patterns for September 15, 2020.

### 3.2.2 *Traditional Monitoring Tools*

The approach explored in this paper differs significantly from traditional monitoring techniques. Oftentimes, instead of placing the burden on the client, monitoring systems either (1) act as a single client themselves or (2) utilize privileged access to the system, perhaps monitoring database connections or internal logs. The approach we are recommending does not seek to disqualify these monitoring patterns. We do suggest, however, the following addendum to our hypothesis above:

> To maximally measure a population's user experience in real time, it is essential that measurements be taken as close to the physical user as possible.

In this case, we are utilizing signal generators *within each user's browser,* which is much closer to the user than a distant monitoring client[2] or even the backend logging system monitoring the system's inner workings.

### 3.2.3 *Inspirations*

The basic idea of broad-scale signal monitoring is inspired largely by work done in photovoltaic research with solar panels. "When the output of a solar electric system is graphed… it forms a traditional bell curve" (pveducation.com). This makes intuitive sense. As the sun "rises" (to use the colloquial term), the output of the solar panel will gradually increase throughout the day. Eventually it will hit some peak and then decline as the sun "sets."

In much the same way, companies with the vast majority of their workforce in nearby time zones will see the same bell curve. As employees arrive on-site or login from remote locations the activity levels recorded will follow a predictable pattern. There will be a peak, some valleys (e.g. lunch breaks), and then an eventual decline as the workday ends. This pattern was observed in Figures 1A and 1B.

## 4 PADUX ARCHITECTURE

In this section we discuss the full PADUX architecture. An inventory of the full system can be laid out as follows:

- A single-node Kafka cluster
- A Python consumer script
- Various Python modeling scripts
- A MSSQL database

[2] Example: an automated script making HTTP calls to a web application every minute. If a status code of 200 is returned, the application is assumed to be running correctly.

- Plotly dashboards

Each of these layers is discussed below, but the high-level architecture is displayed in Figure 3.

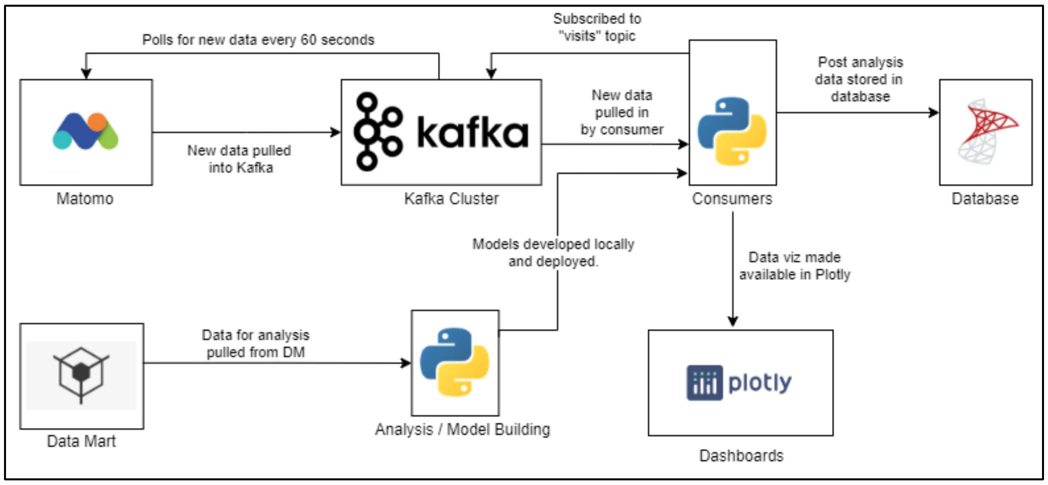
Polls for new data every 60 seconds
Subscribed to "visits" topic
Post analysis data stored in database
New data pulled into Kafka
New data pulled in by consumer
Matomo
kafka
Kafka Cluster
Consumers
Database
Models developed locally and deployed.
Data viz made available in Plotly
Data for analysis pulled from DM
Data Mart
Analysis / Model Building
plotly
Dashboards

***Figure 3***—PADUX architecture design

Note in Figure 3 the Data Mart and Matomo are considered external to PADUX. They are shown for reference.

## 4.1 Kafka Cluster

The Kafka cluster was deployed to an internally developed Platform as a Service (PaaS) and managed using Docker Compose. The Kafka instance used was built with a default configuration sourced from Confluent's Docker Hub account (see https://hub.docker.com/u/confluentinc). By using Confluent, we were able to harness a preconfigured cluster with various conveniences such as a web client for managing the background services. A Java database connector (JDBC) was added and a connector instance was instantiated to poll for new data from Matomo every sixty seconds.

Utilizing Kafka offered several helpful capabilities for our use case. First, its JDBC connector minimized load on the Matomo database by only requesting recent messages. Second, the system is highly scalable and even though for our purposes only a single node was used, there is the option to scale the system horizontally later. Third, currently only Matomo data is being streamed, but additional data sources can be easily added using a plethora of different Kafka connectors.

### 4.2 Analysis and Model Building

The analysis and model building components are mostly contained in a single Python script and can be run either manually or via automation. The script parameters require a start and end date to specify the bounds of the model's data. Other parameters of the model will be discussed in detail in the algorithm section.

### 4.3 Consumer Script

The consumer script is written in Python and leveraged the Kafka Consumer library, Kafka-Python. The consumer has several notable points:

- It uses an Auto Offset Reset (AOR) policy of "latest" so only the latest messages are retrieved, thereby optimizing consumer performance (see section 4.7 on the Retroactive Data Script for more information on the rationale of this policy choice).
- It utilizes a consumer group, which allows for scaling the consumer horizontally.
- The primary logic of the consumer is to pre-load a model and compare real time signal data against a model of past signal data. If the real time signals are significantly weaker than past signals, an anomaly is registered. Regardless of whether an anomaly is noted, signal strength is recorded in the MSSQL database. Details of this algorithm are discussed later in the paper.

### 4.4 MSSQL Database

The database was designed using an extremely simplified schema. Below is the entity relationship diagram (ERD) of the database:

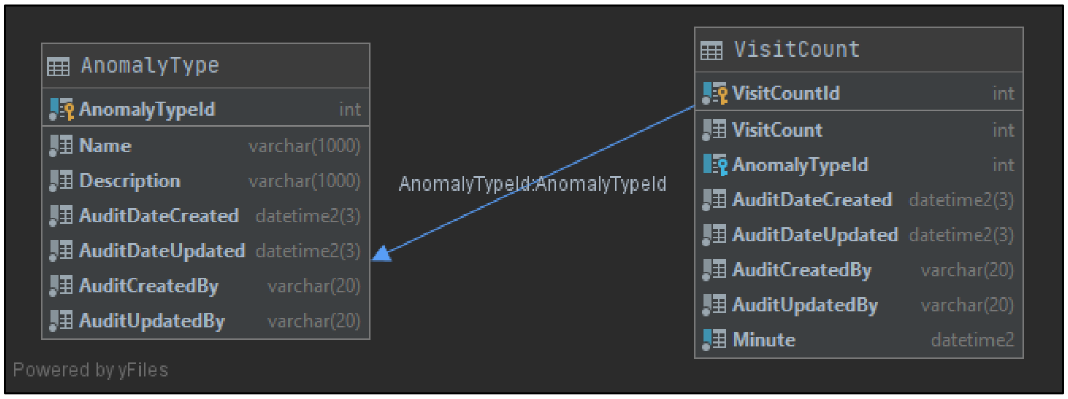
AnomalyType
AnomalyTypeId int
Name varchar(1000)
Description varchar(1000)
AuditDateCreated datetime2(3)
AuditDateUpdated datetime2(3)
AuditCreatedBy varchar(20)
AuditUpdatedBy varchar(20)
AnomalyTypeId:AnomalyTypeId
VisitCount
VisitCountId int
VisitCount int
AnomalyTypeId int
AuditDateCreated datetime2(3)
AuditDateUpdated datetime2(3)
AuditCreatedBy varchar(20)
AuditUpdatedBy varchar(20)
Minute datetime2
Powered by yFiles

***Figure 4***—PADUX database ERD

Each time new data is polled and processed by the consumer the results are stored within the VisitCount table, which can be thought of as the record of signal levels. This occurs regardless of anomaly detection. If an anomaly is detected, the appropriate anomaly type is included within the VisitCount record. Determining the appropriate AnomalyType is done within the consumer algorithm directly.

### 4.5 Plotly Dashboards

Dashboards to display data were developed within Plotly's Dash framework, as shown in Figure 4 (see https://plotly.com/). The dashboard runs within the same Docker compose stack as the consumer. The dashboard is populated by pulling data from the PADUX database (the signal data, shown in yellow as the "real time" line) and compared against another dataset loaded from a CSV file. This second file contains the model (which are the lines "smooth", "under", "over"). These details are discussed in detail in the algorithm section. It is worth pointing out that there is also a historical option (the "Past Visits" tab in Figure 5), which allows for historical data exploration of the signal data against different models.

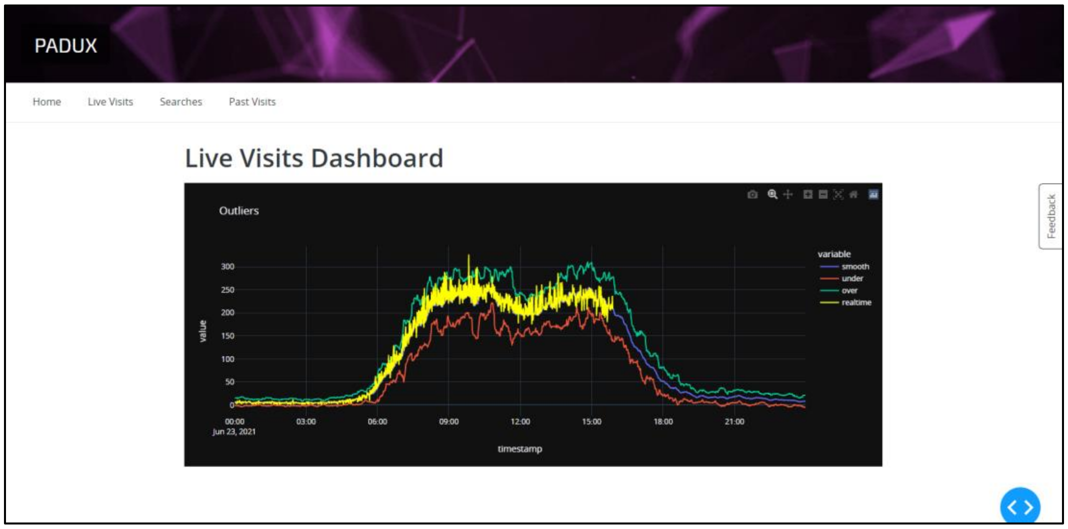

*Figure 5*—PADUX dashboard showing signals in real time

## 4.6 Deploying the Consumer and Dashboard

The consumer and dashboard are deployed using Docker Compose. Each instance (any consumer and the dashboard) uses the same image, with "python" specified as the entry point. Within the Docker Compose a different script is called for the same image, depending on which service needs to be started. This has allowed for scaling to add more consumers easily, while still streamlining local development.[3]

## 4.7 Retroactive Data Script (not shown in Figure 3)

An additional component that was developed was a Python script that retroactively populates the MSSQL database. This was designed for situations where unexpected issues cause outages within the PADUX infrastructure itself. It also allows for back-populating pre-PADUX time periods for historical analysis. The script can be run for a date range and will retroactively populate data in the

[3] The core code repository contains the analysis modules, the consumers, and the dashboards. This makes the system somewhat of a monolith. However, this was an intentional design as the analysis portion, consumers, and dashboard all work closely together. It would have been highly inconvenient to split these into a microservice pattern.

MSSQL database. Currently it is only configured to be run manually; ideally, this will be deployed as an automated background process.

One of the advantages of this second script is it allows the primary consumer to operate with a "latest" policy. In practice this policy means only the newest messages will be consumed and older messages will be ignored in the case where the consumer goes down and comes back up at some later point. It was observed that when the offset was set to a policy like "earliest", the system struggled to catch up on messages due to the sheer volume requiring processing.

## 5 ALGORITHMS

This section covers PADUX's various algorithms in depth. First, we will provide some background on Matomo analytics. Second, the Kafka connect query will be shown and explained. Third, the model used to determine if there is an anomaly in signals is discussed. Finally, the implementation of the anomaly algorithm within the PADUX consumer is briefly reviewed.

### 5.1 Matomo Analytics Background

Signals, for the purposes of this paper, are the same as the Matomo concept of a "visit." A visit in Matomo can be defined as follows:

> A sustained period of activity on a web site by a single user where there is no more than 30 minutes between any two actions[4] (see https://matomo.org/faq/general/faq_36/ for more details).

With some exceptions, this holds true even if the user closes their browser and accesses the site again within the 30-minute window. "Actions" can be defined less formally as there is a very broad set. For examples, actions include out-links, clicks on the page, typing into form fields, and any inner-site navigation. **These are the low-level signals about the user that Matomo captures.** When a user initiates a visit to a site, two important details are recorded: the visit's first action time and the visit's last action time. When this visit record is initially created, those two values are equivalent. As the user moves about the site and performs actions, the

[4] This is the default time interval shipped with Matomo. It is configurable, however.

last action time is updated. The implications of this record keeping is that it provides high granularity in measuring user activity as a signal on any site where Matomo is installed.

### 5.2 Kafka Connect Query

Kafka connectors are a type of abstraction that allows for the Kafka cluster to connect to different data sources using various protocols. For example, one type of connector might be for a web REST API, while another connector might be specific to a MongoDB system. In this case, we decided to connect directly to the Matomo database using a JDBC connector. This connector includes a "query" feature, which allows for standard SQL syntax against a MySQL database (MySQL is the specific Matomo database technology). The essential elements of the query are as follows:

- It selects the unique ID of the visit, the time of the first and last action, the site metadata such as the site's recorded unique identifier, and the total time of the visit.
- The query was constrained to visit records that had a last action time occurring within 1 minute of the time of the query.

Simply put, the connector selects all visit records that were *active* within the last minute. By doing so, the records coming into the PADUX consumer at every minute represent the number of individuals actively utilizing web-based resources during that minute and gives the consumer a high-granularity signal to utilize for model comparison.

### 5.3 PADUX Model

#### 5.3.1 *Fundamentals*

The signal coming into the PADUX consumer must have a comparison model. The basic model proposed fundamentally uses the same concept discussed in section 5.2: one that measures active visits for each minute in a 24-hour time period. However, the variableness in these measures could be significant enough that detecting anomalies would be difficult. Instead, the following was calculated: (1)

the rolling, 15-minute mean of active visits at every minute of a twenty-four hour period and (2) the +/- standard deviation (again, rolling 15-minutes) where Σ = 3. The end result is shown in Figure 6 below.

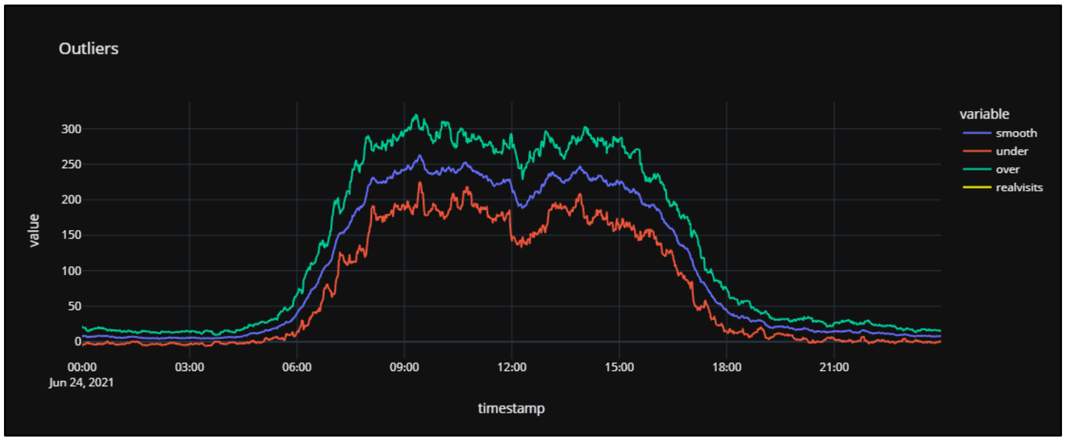

*Figure 6*—Rolling standard deviation of visit data

### 5.3.2 *Intuition Check of Model*

In general, Figure 6 illustrates expected active signals coming from employee use of web systems. The tails (which never go to zero), illustrate overnight periods when automated bots, such as the monitoring tools discussed previously, perform tests, and checks of the systems. During the morning hours, there is a sudden climb in activity. In the middle of the day, when many employees take lunch, a dip in activity is seen with a spike approximately an hour later. Finally, as the day ends, there is a decline that eventually returns to near-zero levels of activity, except for the bots and the occasional employees burning the midnight oil.

### 5.3.3 *Caveats of the Model*

Some caveats had to be considered. First, activity varies significantly depending on the day of the week. For example, a Tuesday and a Thursday can have very different patterns. Further, due to the fact that the 9/80, standard 40 and 4/10 schedule are all supported within our organization, Fridays are an outlier among working days. To complicate matters further, subsequent Fridays differ, because employees largely favor certain 9/80 schedules over others due to how they line up with US holidays.

In order to account for these caveats, the model takes the following parameters: (1) day of the week and (2) minute of the day. For this experiment, the specific Friday in question was not used as a parameter. Future versions of PADUX will account for this nuance.

## 5.4 Implementation in Consumer

Implementation in the consumer involves bringing together the real time query used in Kafka and the model, both discussed above. The key points of the algorithm are as follows:

- Every minute, the consumer checks for updates to the Kafka topic where the visit data is waiting to be consumed.
- If data is available, the consumer pulls it.
- Once the data is pulled, it goes through a basic parsing process that puts it into JSON format.
- After parsing, the consumer examines the data to determine what day and what minute of the day the visit data represents (for example, Monday at 12:06 PM MST). It then sums the number of active visits for that minute
- The model is already loaded during startup, so the consumer finds the matching day and minute in the model (Monday at 12:06 PM MST) and pulls the expected mean, high, and low counts from the model.
- The consumer then compares the real time visit count against the lower rolling count. If the current is less than the expected count, then it is flagged and noted in the database. Otherwise the consumer simply stores the real time count without an anomaly reference.

Figure 7 illustrates what this looks like in real time:

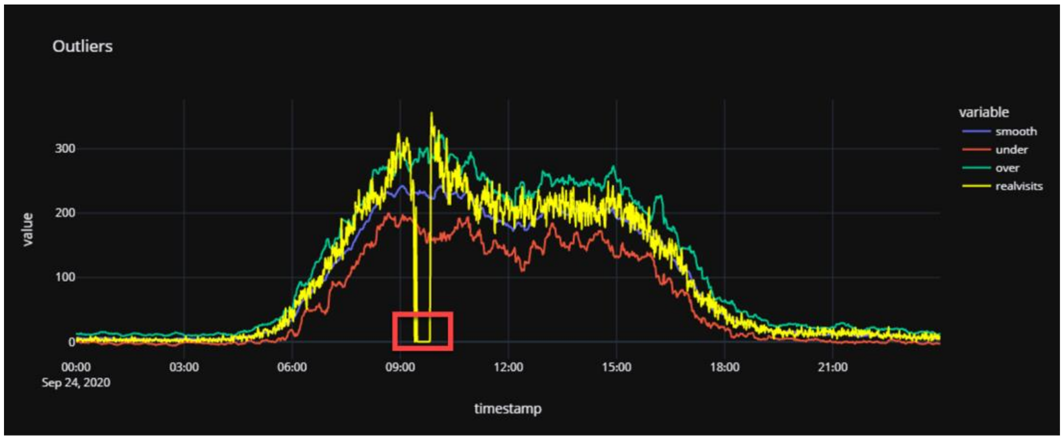

***Figure 7***—Rolling standard deviation of visit data

In Figure 7 there was a significant drop in activity signals (visits) around 9:30 AM, and the signal went to zero at certain points, triggering several anomaly events. Interestingly enough, these correspond to a power failure within an IT data center.

## 6 RESULTS

### 6.1 MITIs

A "major incident" in IT can be defined as "an emergency-level outage or loss of service" (Atlassian). In order to manage such situations, organizations can create an incident management process that defines the "path we take to identify, resolve, understand, and avoid repeating incidents" (Atlassian). Within our organization, this process is formally referred to as a MITI: A Major IT Incident.

In order to cross-check the effectiveness of the PADUX system, we decided to perform historical analysis of signal data, comparing the results of each MITI incident with the PADUX signal measurements. Prior to beginning the PADUX project, it was never an expectation to detect every MITI. This is simply due to the heterogenous nature of outages. A major IT incident may or may not cause significant signal loss to the degree that it is detectable using the methodology we've described in this paper. Therefore, in order to properly evaluate PADUX performance, we evaluated each MITI according to its recorded Issue Description. The categorizations used are placed into Table 1.

While this categorization is based on intuition and not more measurable techniques, nonetheless, coming up with a non-intuitive, quantifiable approach goes far beyond the scope of this experimental system. Instead, we relied on our combined 12 years of experience within our organization to determine category. This provides us with a minimum of 11 MITI days for PADUX evaluation.

***Table 1***—MITI Categorizations for PADUX analysis

| **Category Name** | **Description** | **Count** |
|---|---|---|
| Unclear | In these cases, we were not clear on the scope of the affected systems. | 6 |
| Too localized | These were too localized to cause signal measurement anomalies. For example, power outage in an office-only building. | 7 |
| Not tracked by Matomo | This situation is not localized, but we did not have receptors for outage signals coming from the clients of these systems. | 8 |
| Detectable | We expect PADUX to detect signal anomalies in these affected systems. | 11 |
| Total | | 32 |

### 6.2 PADUX Comparison with MITIs

The next step in the process was to analyze PADUX data for the specific days in which the MITI's occurred (the 11 MITI days classified as "detectable" in Table 1). We then compared analysis of those days against anomaly readings on other random collections of business days. Essentially, we were looking for clear evidence that the higher the level of signal anomalies from PADUX, the more likely a MITI was to be initiated by the MITI team. Our null hypothesis, therefore, is that the mean difference of anomaly counts for any random selection of two groups of days (eleven in each group) exceeds the mean difference of the MITI days anomaly counts and any other randomly selected group of days more than 5% of the time. The alternate would indicate that MITI days have much higher levels of anomaly counts than any random group of days and, therefore, if PADUX detects a high anomaly count this should indicate a widespread IT issue worth investigating.

To identify MITIs more accurately, PADUX readings were generated for the specific MITI days classified as "detectable" using a Python script that backfills the PADUX database for a specific date range. The script backfills PADUX readings for the date ranges of interest (approximately September 2020 through July 2021). To verify the statistical significance of PADUX readings on MITI days vs. non-MITI days the distribution of the data readings was analyzed and found to be non-normal. As a t-test assumes normally distributed data, this method was not appropriate. Instead, permutation testing was performed as an appropriate alternative for this data (Pellegrina, L., & Vandin, F. (2020), Bruce & Bruce, 2017). See *9.1 Appendix A: Q-Q Probability Plot*, for more information.

Permutation testing required a script for initial selection of all the MITI days being analyzed (11 total) as well as counts for the number of anomalies that occurred within business hours for each day. This represents the "test group." A selection of 11 random days was taken from the full set of possible days for which PADUX signal data was generated (September 2020 through July 2021).[5] The number of anomalies during business hours was calculated for the random 11-day set, representing the "control group." The mean was calculated for both the test group and the control group and the absolute difference was then taken. To complete the permutation testing, the script selected (again at random) pairs of 11-day groups from the allowed day pool (22 days total, half in a test group, half in a control group, selected each time), and the absolute mean-difference was calculated for

[5] Sandia National Labs, like others in the aerospace industry, has a shutdown period annually in mid-to-late December. Therefore, data from mid-December through the first few days in January were excluded from the analysis. Further, all holidays which would affect Sandia Labs workforce were removed.

each iteration.[6] This was done 118 times, besides the original test/control iteration. The results of the analysis are shown below in Figure 8.

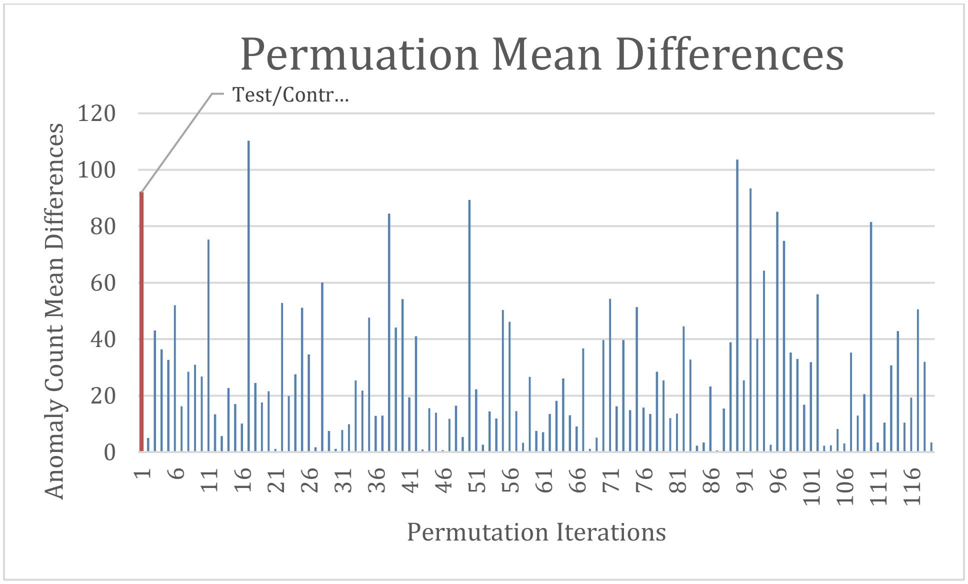

***Figure 8***—Bar chart showing mean difference of the permutation iterations using the experiment script.

As can be seen, in the vast majority of cases, the mean-difference of any randomly selected pairs of 11 days will not exceed that of the MITI days with any randomly selected control group. A closer look gives a p-value of 2.54%, where only 3 of the 118 cases were greater-than-or-equal the mean-difference of the original test/control group. This indicates it is very unlikely the null case could happen by random chance. A summary of these findings is in Table 2 below.

[6] Selection was completed without replacement of the days for any particular group of 11. While the same days might appear in the *pair* of test and control groups, in any one group of eleven, duplicate days were not allowed.

***Table 2***—P-value findings

| **Description** | **Count** |
|---|---|
| Greater-than-or-equal test/control mean difference | 3 |
| Less-than test/Control mean difference | 115 |
| Total iterations | 118 |
| **Test/Control Mean Difference:** | **92 anomalies** |
| **P-Value** | **2.54%** |

## 7 CONCLUSION

### 7.1 Revisiting the Original Hypothesis

In section 3.2, we laid out our hypothesis which, assuming the built-in assumptions could be satisfied, states that "anomalies in basic system access can be detected and used to determine if system access by users is currently impaired or has been impaired in the past." Did we prove our hypothesis?

We would argue that we did prove it in part. Certainly, PADUX can be used to identify outages from the past; it gives clear indications that users had difficulty accessing systems on the 11 MITI days. Nonetheless, it is unclear if PADUX can alert system owners of issues in real time. The analysis done here takes *full days* into account, not windows of time within a given day. Despite this uncertainty, there is enough promise within this analysis to indicate that more granular modeling and research could add a level of sophistication to PADUX such that it could be proficient in detecting real time outages.

## 7.2 Utilizing PADUX

This analysis does give strong evidence that low-level signal readings from a broad network of clients can indicate significant issues in enterprise IT systems. Many of these issues may not even be measurable by traditional monitoring methods. It cannot be reiterated enough, however, that this is not intended to be a replacement for these traditional monitoring techniques. Instead, this should be viewed as an additional protection layer or heuristic to be used in conjunction with other monitoring tools. If, for example, a real-time warning system were configured to trigger when certain anomaly patterns occur, this method could aid enterprise IT organizations in shortening response times.

Another useful application is in post-event network forensics. Essentially, this data can be used to highlight and prioritize where forensic teams *should start.* Throughout a given calendar year many IT outages might go unnoticed due to the qualitive labeling methods currently employed, e.g. an individual person (or small team of people) must decide there is sufficient evidence to call a MITI. Much of the evidence of deprecated IT services may not be apparent to those in charge of the MITI process, and so those events may never be captured. Capturing these events via a system like PADUX would allow forensic teams to prioritize specific days to investigate and then analyze granular data sets such as server logs.

## 7.3 Future Work

Currently the PADUX system has several shortcomings: (1) models are not automatically regenerated based on new data, (2) there is no automatic notification system, (3) granularity is highly limited (all sites are viewed in aggregate), (4) there is limited capability in terms of external exposure via something like a REST API, (5) many of the deployment steps are manual and (6) the system's underlying architecture is not currently configured to be highly-available. All of these issues could be addressed and would make the system more usable.

## 8 REFERENCES

1. Atlassian. *How to run a major incident management process.* https://www.atlassian.com/incident-management/itsm/major-incident-management

2. Bruce, Peter and Andrew Bruce (2017). *Practical Statistics for Data Scientists.* Chapter 3, Statistical Experiments and Significance Testing. O'Reilly, Sebastopol, CA.

3. International Standards Organization. Ergonomics of human-system interaction— Part 11: Usability: Definitions and concepts. https://www.iso.org/obp/ui/#iso:std:iso:9241:-11:ed-2:v1:en

4. Kafka-Python. KafkaConsumer https://kafka-python.readthedocs.io/en/master/apidoc/KafkaConsumer.html

5. Pellegrina, L., & Vandin, F. (2020). Efficient mining of the most significant patterns with permutation testing. Data Mining and Knowledge Discovery, 34, 1201-1234. https://link.springer.com/content/pdf/10.1007/s10618-020-00687-8.pdf

## 9 APPENDICES

### 9.1 Appendix A: Q-Q Probability Plot

In order to determine if a t-test was an appropriate approach to check for the statistical significance of the PADUX findings, a Q-Q probability plot was generated. This Q-Q plot indicated that the data was not distributed normally, but positively skewed. Therefore, permutation testing was used instead. For the full data set and analysis, please see the accompanying spreadsheet: *data_analysis_final_1632321371.1069791.xlsx*

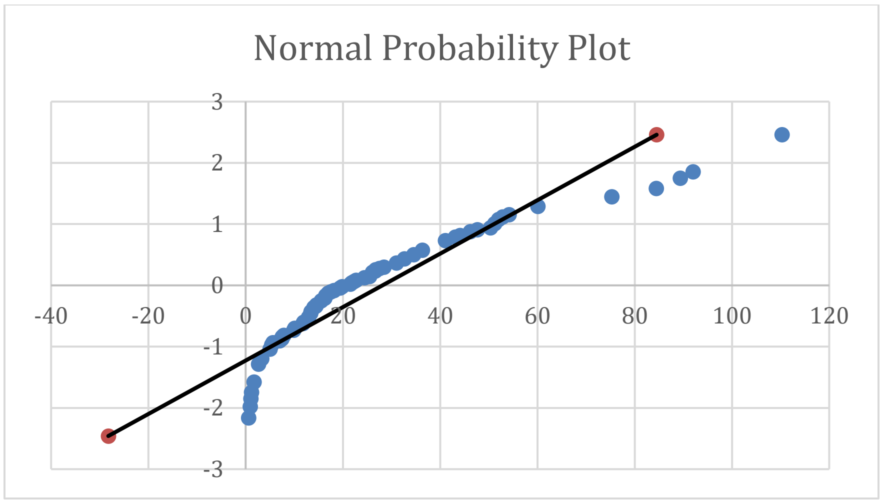

***Figure 8***—Normal Probability Plot of the data showing this set is most likely not normal.

## ACKNOWLEDGEMENTS

We want to thank John Miner (Sandia National Laboratories) for his mentorship and for sparking the idea to this project by pointing out the analogy between photovoltaic patterns and the concepts explored in this research. We would also like to thank Scarlett Marklin (Sandia National Laboratories) for helping advise us in the statistical analysis portions through informal, but very helpful, reviews (we hoped we didn't fail her too badly). Paul Salazar (Sandia National Laboratories) provided a review of our software stack and we would like to thank him for always asking good questions. Finally, we would like to thank our manager, Tameka Huff (Sandia National Laboratories), for funding us to pursue these very interesting projects.