21, rue d'Artois, F-75008 PARIS
http://www.cigre.org

CIGRE US National Committee
2019 Grid of the Future Symposium

# Who Monitors the Monitors?
# Automated, Hierarchical Data Quality Assessment for Time Synchronized Measurements

**B. BENGFORT, S. P. MURPHY**
**PingThings, Inc.**
**USA**

**M. ANDERSEN**
**PingThings, Inc.**
**South Africa**

**K. D. JONES**
**Dominion Energy**
**USA**

## SUMMARY

Distributed sensor systems are a critical component required for the operation of a reliable, resilient, and efficient electrical grid. However, just as the grid requires monitoring, so too does the sensor system itself. Therefore, the continuous assessment of the quality of the data captured by sensors becomes an important obligation and task for transmission and distribution system operators. This paper examines the underlying philosophy of data quality assessment and then uses this perspective to craft a set of requirements for an operational system. The paper then discusses one potential system implementation that meets or exceeds the requirements posed and then examines a few ways data quality assessment could evolve.

## KEYWORDS

Time synchronized measurements, synchrophasor, data quality, distributed sensor network

sean@pingthings.io

## 1. INTRODUCTION

Death and taxes are joined by a third unavoidable certainty - data quality issues - for any organization that operates a network of sensors measuring the real world. Electric utilities, stewards of transmission and distributions systems together valued at over a trillion dollars in the US, are no exception. If an organization's assets are valuable enough to be instrumented with sensors, they are valuable enough for the organization to monitor the quality of the data that is generated by those sensors. As entropy perpetually assaults such a system, data quality assessments must be done continuously. It is not something that, done once, is forever resolved and disappears.

The quality of the data captured by sensors can be impacted by a number of different factors that span disciplines, from the field of metrology itself to the installation and configuration of the actual hardware, to the communications channel, to the networking layer of the system, to the computer systems and software receiving and persisting the data. No single engineer or subject matter expert has deep knowledge in all of these fields, potentially making data quality problems especially pernicious. Further, as a result of its interdisciplinary nature, responsibility for data quality can often fall through the organizational cracks of a system operator and no individual or no group "owns" data quality.

Over the last 5 years, the authors of this paper have examined over a petabyte of synchrophasor data from utilities across North America, including both the transmission and distribution grids. While this total data volume examined may sound large, it represents less than one percent of the total synchrophasor data available if all deployed phasor measurement capabilities were activated. Using a conservative estimate of 20 channels of data per sensor, each at 30 samples per second, we estimate that over 150 Petabytes of time synchronized, phasor measurements would be generated per year if all of the hundreds of thousands of sensors were simply turned on. At either scale, the need for data quality assessment becomes obvious as does the futility of human-based, ad-hoc assessments. The quality of terabytes, petabytes, or even larger data sets must be interrogated, assessed, and captured in an automated fashion. Further, given the magnitude of this analysis, it should be done once and only once in a standardized fashion and included with data sent to each downstream consumer of data, be it an application, an analytic, or an end user.

The modern utility operates an enormous, distributed sensor network composed of up to millions of sensors of different types--smart meters, synchrophasors, power quality meters, point on wave, and more--and reporting at varying data rates from once every 30 minutes to over a hundred thousand samples per second. This paper will explore the requirements for a system capable of monitoring the quality of the data originating from such a system at scale. We then examine particular aspects of a universal sensor analytics platform that enable continuous data quality surveillance, attempting to meet or achieve all requirements set forth. Finally, the paper will conclude with a glimpse into the future of data quality assessments. With this effort, the authors of this paper hope to answer the question, who (or what) monitors the monitors and how.

## 2. DATA QUALITY ASSESSMENT REQUIREMENTS

The high-level objective of such a system is to maximize the amount of "usable" data generated by the sensor network while minimizing the consumption of resources to accomplish this goal. To make this more tangible, consider a fictional grid measured by 100 sensors, each providing 10 channels of data with 100Hz sampling per channel. This sensor network generates 100 x 10 x 100 = 100,000 measurements describing the grid per second. The objective of the associated data quality monitoring system is to ensure that as many of these 100,000 measurements per second are "usable" by as many downstream consumers as possible.

The first general requirement that we have of the system is that it must be capable of assessing the data quality of the distributed sensor system for two different time periods. The first is historical, starting

with the current moment and stretching back in time to when the sensor first captured data. This period allows those responsible for the sensor data to understand what problems occurred in the past, what problems that were intermittent might still be impacting data quality, and what problems may still be persisting today. The second time period is the present; the system must be able to examine and analyze the quality of data as it arrives. This allows an alert to be triggered as bad data arrives, enabling the system or its operators to take corrective action quickly to decrease the length of time bad data pollutes the system in pursuit of one hundred percent good data.

The second general requirement for the system is that the data quality assessment must be persisted so that downstream consumers do not have to replicate the analysis unless their data quality needs are extraordinary. This record of the data quality must be forever associated to the original data. As a caveat to this requirement, storage, both on premise and in the cloud, can be expensive. Amazon AWS's storage tiers range in cost from less than a penny per gigabyte per month for the slowest tier, Glacier, to $0.30 per gigabyte per month for Amazon's Elastic File Storage as of August 2019 [1]. It is quite possible that the data quality assessment for each measurement could consume at least as much space as the original value. Depending on how data is stored, the data quality assessment could inflate the total storage size by 50-100% at a minimum if full temporal resolution is kept. Thus, it may make sense to persist only an aggregated data quality assessment in lieu of full resolution.

The last general requirement for the system is to handle two broad categories of data quality assessments. The first is a generic assessment that can be applied to every time series data stream ingested by the platform, answering basic questions that are applicable to any sensor data. The second is to be flexible enough to have custom data quality assessors applied to different types of time series, some potentially specified by users.

## 3. SYSTEM IMPLEMENTATION

Assessing data quality for high frequency sensor networks should not be a bolt on after-thought but rather one of the first design requirements for any system handling real world sensor data. Sensor networks are distributed over a wide geographic area and send data through a large number of subsystems including data concentrators and intermediate caches, all of which may contribute to poor data quality. Because of this, the problem of data quality assessment is closely related to the problem of how to use sensor data effectively. These design requirements mean that the data quality assessment must occur as far downstream and as close to the end consumer as possible, ideally where the sensor data is centrally and ultimately stored.

The PredictiveGrid™ platform was designed from the ground up to be a third-generation data management, analytics, and artificial intelligence platform specifically for high-frequency time series (e.g. dense telemetry generated from sensor data) [2]. Because the platform was architected to store, process, and analyze dense telemetry faster than real-time, it is well-suited to both historical data analysis and model generation as well as applying computation to live, streaming data. Because of this dual role, the PredictiveGrid platform is the last stop in the automatic data processing pipeline before data is presented to users, and therefore also is required to provide data quality assessment and assurance.

### 3.1 Enabling General Purpose Sensor Data Quality Assessment: The Berkeley Tree

The data structure at the heart of the PredictiveGrid is the Berkeley Tree [3] - a *time-partitioning, multi-resolution, versioned k-ary tree.* An artist's rendering of the tree is shown in Figure 1. The leaf nodes of the tree store fixed blocks of (time, value) pairs for a specific time-range. These blocks are summarized by statistical aggregates, which are themselves stored in fixed time-ranges and are also summarized by higher levels of the tree. This creates a hierarchical data structure where different levels of the tree represent different time resolutions, from all epoch representable time at the root node to leaf nodes containing nanoseconds of data through granularities representing years, months,

weeks, hours, etc. Finally, links between nodes are annotated with a monotonically increasing version number when the underlying data is modified, ensuring that the tree is consistent even as data is being written.

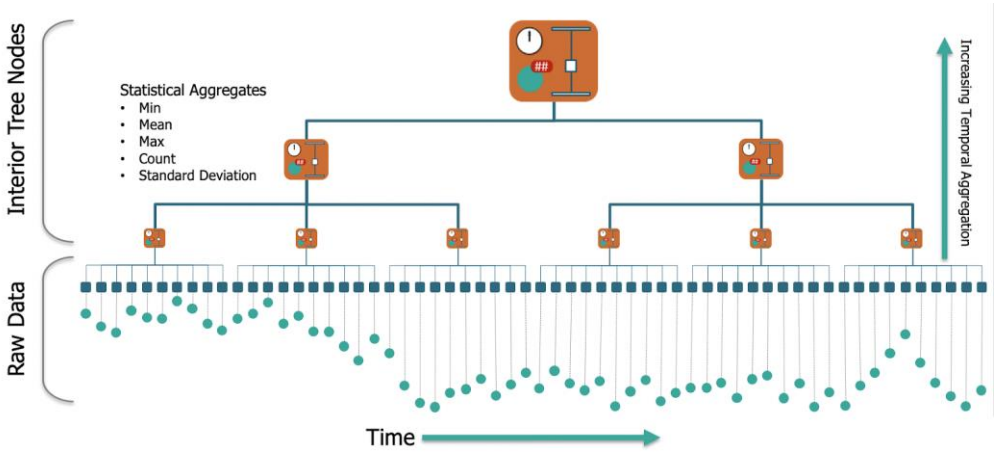


Figure 1: Representation of the Berkeley Tree is a time-partitioning, multi-resolution, versioned k-ary tree. This novel data structure allows users to rapidly make both historical and online data quality assessments.

One function of the Berkeley Tree is to provide highly performant dense telemetry writes and aggregate reads. The tree structure allows data to be inserted out of order unlike other append-only systems. This ensures that sensor systems subject to variable network latencies and intermittent communication outages can apply writes out of order or across overlapping windows as efficiently as possible. Each write updates a version, ensuring that analysts can read at a specific version even while writes are ongoing, giving their computations consistency through snapshot isolation [4]. Because most time series analytics consider aggregate windows of data at varying resolution, the tree allows users to query exactly the level of the tree where the aggregates are already stored without computation. Secondarily, but just as importantly, the platform also allows for rich data quality assessments using similar features: versioning, aggregates, and ingest distillers.

Data streaming into BTrDB (the Berkeley Tree Database) is generally written in blocks to the underlying tree, causing a new version of the tree to be created. When the tree is read, the user must specify a version, often the most recent, and only data belonging to that version is returned even as the tree is being concurrently updated, e.g. they are seeing a specific snapshot of the data. Snapshot isolation also ensures that results are repeatable as data changes - a key requirement for consistent reporting. From a data quality perspective, versioning and snapshot isolation allows us to be more aggressive when addressing data quality both because anomalous data can be recovered without interrupting live streams and because the tree makes it possible to identify only what has changed during automatic data cleaning. The *ComputeDiff()* function returns a list of time ranges where data has changed between two versions, without requiring a query of raw data. By itself, simply tracking the number of ranges that are changing between subsequent versions may indicate a problem (data is being updated as much as it is being inserted). Tracking changes between more distant versions allows data quality assessments to zoom directly into regions of question without having to query the entire raw data set.

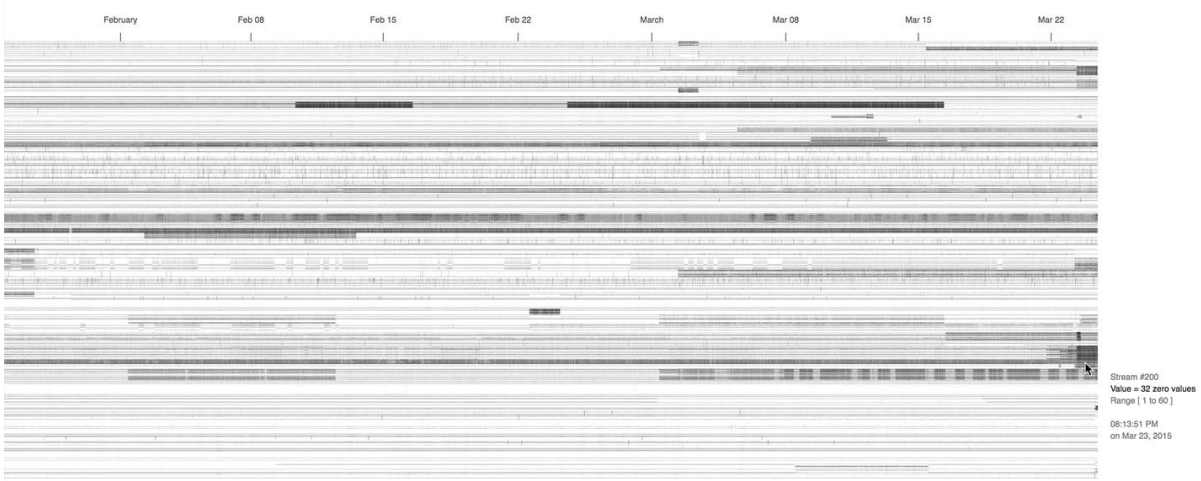


Figure 2: A snapshot of an interactive data quality heatmap over several months where each row is a stream and streams from the same synchrophasor are adjacent. Darker regions indicate areas with a higher percentage of bad data (missing data, null values, etc). This interactive visualization is made possible by only querying the aggregate values at a high-enough time resolution for each stream, making interactive data quality visualization possible.

The interior tree nodes contain statistical aggregates, capturing the statistical properties and behavior of all data points below them in the tree. Currently, the platform has implemented the following statistical aggregates:

- count - the number of data points;
- max - the maximum value of the measurements;
- min - the minimum value of the measurements; and
- mean - the mean value of the measurements

for all data points below that specific interior node. In reality, any associative function can be used as a statistical aggregate and the standard deviation has already been implemented in a mathematically associative algorithm.

From a data quality perspective, these statistical aggregates create an almost ideal mechanism to assess time series data quality in a generic way. For example, count allow you to infer from the sample frequency if data is missing from a specific time range. This missing data metric is relevant to all regularly sampled sensor data. The range of the window, defined by the min and max value of the raw data, may indicate if there are any domain errors for even just a single point. Finally, standard deviations may express abnormal changes in variability between windows that could also indicate systemic problems or other types of changes that need to be addressed. As the tree is built in memory upon data ingest and maintained by the database, all of the capabilities are “free” from the end user’s perspective (and do not cost in terms of performance or latency). This allows for interactive data visualizations like the one shown in Figure 2: a heatmap of bad data that relies on the performance of querying statistical points to display groups of related streams or windows of time with poor data quality.

### 3.2 Enabling Custom, Sensor-Specific Data Quality Assessment: Ingest Distillers

Timeliness in discovering and repairing data quality issues is essential to the PredictiveGrid platform. Because the platform deals with very large datasets of dense telemetry, timeliness must be achieved through Big Data analytics using distributed computing to parallelize computation across multiple machines. The platform provides several options for distributed computing, including open source options such as Apache Spark for general purpose machine learning and Google’s TensorFlow for deep learning. However, to support automated data quality assessment with adaptive requirements, the platform primarily relies on the DISTIL analytics framework [5].

DISTIL provides an environment for materializing analytical streams given a dataflow graph, a directed acyclic graph that describes how data flows from input sources through different algorithmic outputs before finally being stored as output streams in BTrDB. Each node in the dataflow graph (a distiller) represents a stateless, idempotent computation that can be applied to a window of raw values or statistical points (with some before and after the computational window). While seemingly simple, the dataflow abstraction can be used to compose arbitrarily complex computations and, importantly, in such a way that computations can be scheduled in parallel across a distributed computing cluster.

When a distiller is added to the platform, it is immediately applied to all new incoming data that match the distiller inputs. Note that BTrDB supports two first class functions, *GetNearestValue()* and *ComputeDiff()* that ensure the distiller is applied efficiently to only changed data, even if that data arrives out of order. Further, the distiller also begins to back-process historical data in the stream while prioritizing computation on incoming data. Given sufficient computational resources, the distillers will eventually be applied to the entire input dataset while continuously making computations on incoming streaming data. Because of this behavior, DISTIL is ideal for implementing an adaptive data quality assessment process that can evolve with changing environment conditions and new error conditions.

Consider a distiller that takes as input a voltage or current phasor group measured by a PMU and produces a per-second data quality score between 0 and 1 such that higher scores indicate higher data quality. A first pass at this distiller might simply use the known sample rate of the PMU to measure the number of missing points or to apply rules such as there can be no two consecutive, non-zero but identical values for a stream. Later, after a few months of measurement, we may feel confident about applying statistical thresholds to reduce our data quality score if there are measurements outside of a few standard deviations of a computed mean. As more data enters the system, we can begin to add depth to our data quality measurement by using equipment-specific flag streams or ensuring that the magnitudes between phases are within a physical model based on their angle differences. As the data quality system evolves, the distiller not only applies these new analytics to new, incoming data but also to historical data, ensuring that a system-specific data quality assessment can be developed and automatically applied organically. Note that versions in the data quality stream also allow analysts to compare and understand how data quality changes as the assessment evolves, also deepening the understanding of the underlying data.

## 4. REAL WORLD DATA QUALITY ISSUES

In this section, we will explore two case studies of data quality assessments that rely on time synchronized measurements: data loss and outage detection and configuration change detection. We will see how the error due to drift causes uncertainty in these assessments. Finally, we'll explore how timestamp jitter can lead to other data issues like decreasing the effectiveness of compression mechanisms or causing variability across aggregated windows during analytics.

### 4.1 Data Loss and Outages

Given enough time or a big enough system, some data simply will not make it from the sensors to the analytical data concentrator. This leads to the first and perhaps most fundamental data quality question: are we receiving all of the data we should be? If the sample rate of the device and the timestamp the first data point was received are known, it is easy to compare the total number of points stored to the expected number of points received. When, inevitably, this ratio falls below one, new questions arise: what is the distribution of outage durations, do they occur at regular intervals, are these outages correlated with other devices, etc. For example, the analysis shown in Figure 3 demonstrates that there are both device-independent outages as well as routine outages that occur across groups of devices. This may be related to maintenance windows or other processes that could be adapted to keep better caches.

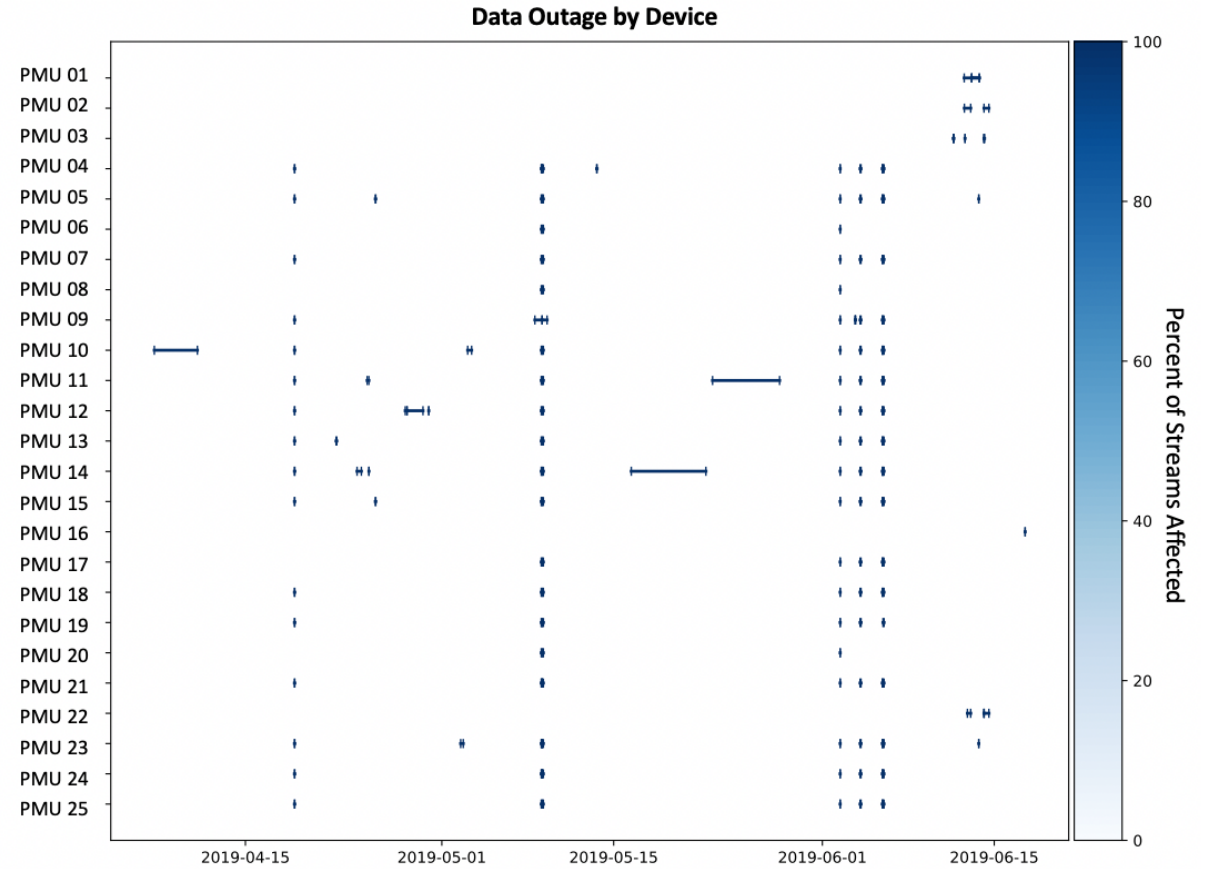


Figure 3: An analysis of the per-device outages across 2 months of data collection for a system operator. This analysis was conducted on approximately 153B data points but completed in less than one minute via tree pruning that reduced the outage search space.

A brute force approach to finding outages would be to scan every point and determine if the time delta between those two points is greater than some minimum threshold (e.g. the sample interval). Besides the obvious problem of the time required to compute this approach on large dense telemetry datasets, it is also ineffective to apply this approach to online data as it streams in: data may arrive out of order (e.g. it's not lost, it's simply late) or there could be a drift in the periodicity of the sample rate that appears as an outage that is really just a time compression of two consecutive measurements.

The platform provides an effective solution to this problem because at higher levels of time granularity, stat point aggregates mask the drift of time synchronization. Said another way, at lower time resolutions (e.g. at the microsecond scale the number of points per period is more variable than the number of points per period at the hourly scale. Because of the multi-resolution tree structure, the platform ensures that outage analysis can occur in both offline, whole dataset analyses, and during online applications because the tree ensures that only the time areas with probably outages are inspected - pruning the search space and making the assessment more efficient.

A sketch of the algorithm is as follows. In a depth-first fashion traverse, the tree such that at each level of the tree only nodes are kept that meet particular filtering criteria. First, determine the number of expected points per node at the time-resolution of the level. For each consecutive pair of nodes at the level, determine if they contain the expected number of points; if not, expand the next level of the tree by selecting the aligned windows from the start time of the first node to the end time of the second. If they do contain the expected number of points, compare end time of the first node to the start time of the second to determine if the time delta between those nodes is greater than twice the expected sample interval at that level. If so, an outage has occurred, and the duration of the outage can be computed by subtracting the timestamp of the last point in the first node and the first point of the second.

This mechanism works because the Berkeley Tree does not materialize nodes for data that does not exist. If there is an hour-long outage, there will be a missing stat point at a pointwidth of 41 (statistical points that cover a period of approximately 36.65) and the two consecutive stat points on either side will have less than their required number points. The search will zoom in on this window of time and ignore hours at the previous level that have complete stat points, pruning the tree only to the areas of interest. Because we consider consecutive stat points, we also minimize the variability of the periodicity that may mean that one statpoint has more than expected points while the one after has fewer than expected or vice versa.

This data quality assessment can be conducted in both automated offline analyses and online applications. In the offline analysis, the worst-case scenario is that every other point is missing, which would result in an analysis equivalent to the brute force approach. In most cases, system providers require a 95% uptime SLA, therefore tree pruning will identify gaps in logarithmic time and only query the data rangers where gaps exist. In the online case, the hierarchical nature of the analysis means that windows can be monitored at multiple granularities that create missing data alarms which increase in severity as the outage continues. Because the distiller applies the data outage algorithm on every insert or update, the alarm state will be eliminated if the data comes in out of order and the gap is repaired or if the system returns to normal operation without repair, the outage and its severity will be logged.

### 4.2 Detecting Configuration Changes

In the outage detection example, the algorithm required the sample rate to be known a priori. For many time series analytics, particularly those that use windowing, the sample rate must also be known in advance. This leads to the data quality question: are the devices configured at the expected sample rate? For many devices the sample rate is a configuration with common settings at 15Hz, 30Hz, 60Hz, and 120Hz; therefore, the corollary to this is to determine when a device configuration has changed, which can have a significant, but hard to detect effect on downstream analytics. Moreover, if devices are configured at too small a sample rate, events in very narrow time windows may be missed by being sampling around.

The sample rate is usually expressed in Hz; however, loss of frequency may be a data loss problem, not a configuration issue. Moreover, smaller aggregate windows like seconds may be affected by jitter or other time synchronization issues. However, the platform makes this problem trivial to solve since each interior node in the tree is defined by a point count and a time duration, such that frequency can easily be computed and plotted as shown in Figure 4. At least initially, a visual analytics approach is well-suited to this application, allowing the user to choose different time resolutions to see if they can quickly spot changes in the frequency while minimizing the variability of the sample rate.

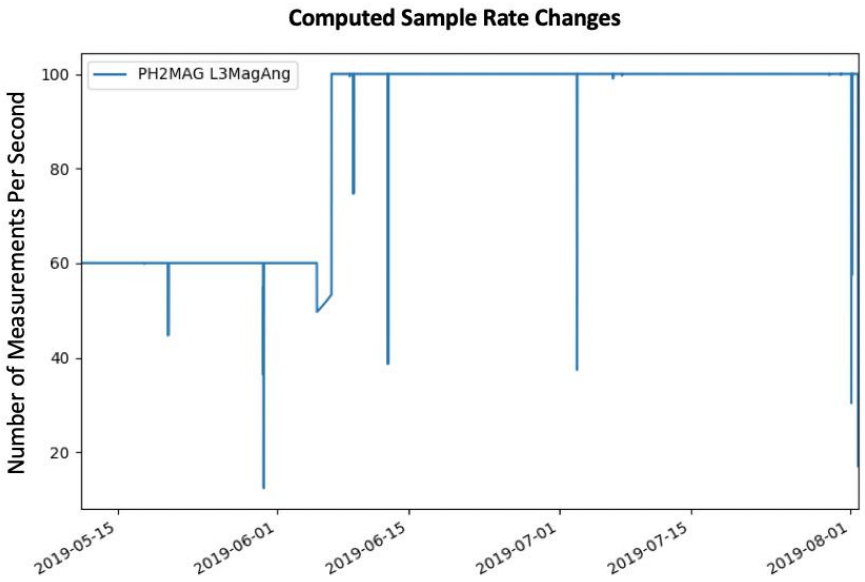


Figure 4: The frequency of this device is plotted at a time resolution of 1.63 days (pointwidth 47). In shows a clear configuration change around June 9, where the sample rate is set to 60Hz, then again on June 11 where the sample rate is reconfigured to 100Hz. The spikes in the graph also indicate where there may have been outages.

To automate this analysis, we note that the configuration changes usually change in step functions and that it is rare to reconfigure devices more than once a day. Therefore, the automated analysis of this stream operates at a daily time resolution and detects if two consecutive windows have a higher frequency than the previous window. If so, this is likely not an outage, but rather an actual change. Other approaches such as a rules based approach that uses standard configuration values may also be applied to normalize the data quality alerts, or if there is a change in frequency, that window can be expanded with the stat points at the lower level to determine if there were multiple configuration changes in the same day.

### 4.3 Sample Rate Jitter

In the previous two case studies, we used the unique data structure of the platform to efficiently and effectively diagnose data quality problems. In both cases, we noted that sample rate jitter is the cause of a large number of analytical issues and used statistical aggregates to smooth the variability of the data. More practically, data compression algorithms work better on less variable data, especially the platform's compression algorithm, which is designed for low variance sample rates. More compression means less data storage, reducing the expense of the overall system.

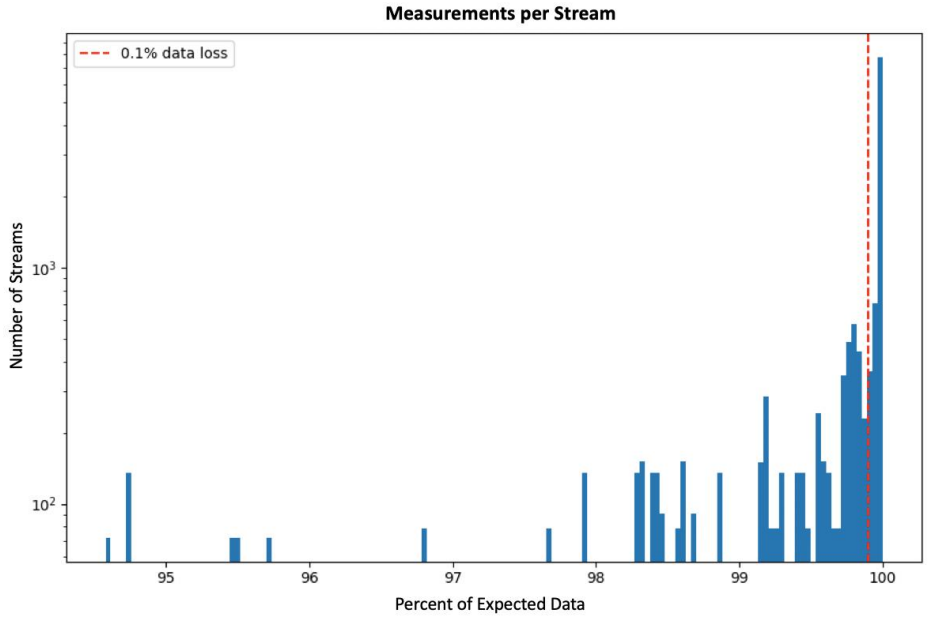


Figure 5: Data quality issues due to timestamp jitter are only observed at a very low time resolution. This analysis that covers over 14k streams for an hour's worth of data shows that most streams are missing at least a few points, many of which may have been sampled in either the previous or next periods meaning that there was no actual data loss.

It is important to conduct a thorough analysis of the sample interval jitter so that algorithmic solutions such as truncating to the nearest microsecond or other configurable solutions may be employed. Many sensor specifications require a sample rate at the millisecond or microsecond scale allowing nanosecond time measurements to be aligned simply by rounding to the nearest microsecond as shown in Figure 6. Higher resolution sensors such as Point on Wave sensors, which operate at kilohertz sampling rates and beyond, still operate above the 50-200ns average network clock drift and can be normalized in the same way. If sample rates are still too variable even after truncation or rounding, distribution analysis allows for fuzzy alignment techniques such as binarization or linear interpolation to smooth the dataset's variability.

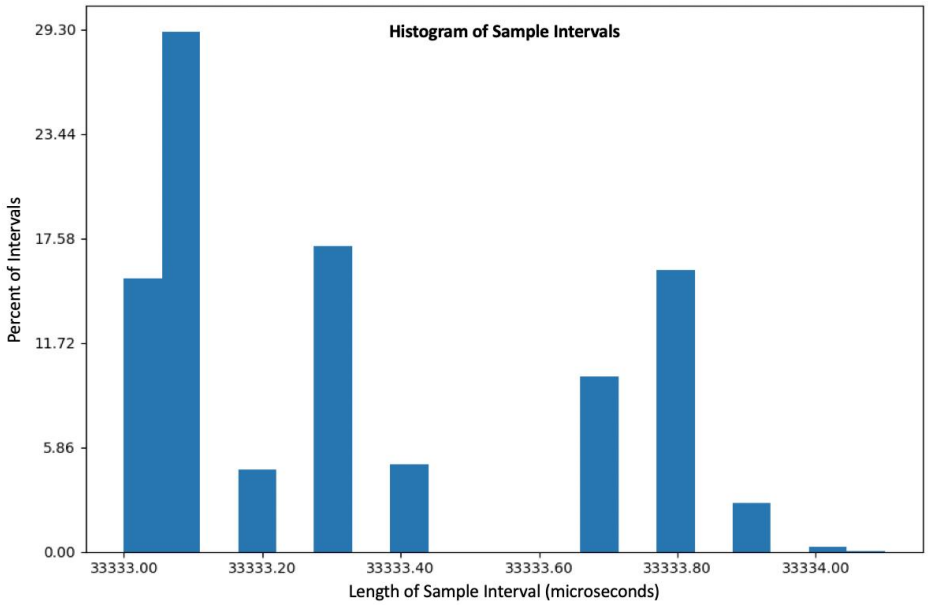


Figure 6: At the nanosecond scale, sample intervals can be highly variable, however if microsecond granularity is required, then these timestamps could be truncated so that the majority of them would align with 33333 microseconds, and only a small amount of variance would occur for those timestamps sampled at 33334 microseconds.

## 5. THE FUTURE

The overall objective function—to maximize the percentage of measurements captured by the system that are usable downstream while minimizing the resources consumed in the process—steers future directions for data quality monitoring. We will examine two efforts in progress that will help sensor network operators optimize based on this objective function: (1) reporting and (2) automation via artificial intelligence.

### 5.1 Reporting

The state of data quality for a network of sensors must be reported to multiple groups with differing interests, potentially both internal and external to the system operator. The volume of information reported and the frequency of reporting will vary greatly depending on the needs and goals of the teams. With systems such as this, there is always the concern that too frequent of alarms will desensitize the groups responsible for responding. Thus, there are multiple types of reporting that may be needed including:

- real-time alerts broadcast via email or SMS,
- historical summaries showing system behavior over hours, days, weeks, months, or years, and
- interactive visualization of both the original measurements and the data quality assessments to explore possible causes for data quality problems.

### 5.2 Automation

As a reminder, the overall objective is to maximize useful data captured while minimizing resources consumed by handling data quality. The pursuit of this goal drives us toward increasing automation where possible. While users care about the quantification of data quality, those tasked with addressing the inevitable issues that arise would rather be informed as to the probable cause of the problem and not just that there is a problem. As a result, we are using machine learning algorithms to classify the probable cause based on the data quality issues that are detected in the data. Work on this approach is very early but promising.

## BIBLIOGRAPHY


[1] https://aws.amazon.com, retrieved August 2019

[2] S. P. Murphy, M. Andersen, K. D. Jones, M. Bariya, and J. Schuman, “A Universal Sensor Analytics and Artificial Intelligence Platform for the Grid,” presented at the 2018 Grid of the Future Symposium, 2018.

[3] M. P. Andersen and D. E. Culler, “BTrDB: Optimizing Storage System Design for Timeseries Processing.,” in FAST, 2016, pp. 39–52.

[4] H. Berenson, P. A. Bernstein, J. Gray, J. Melton, E. J. O’Neil, and P. E. O’Neil. A critique of ANSI SQL isolation levels. In ACM SIGMOD Conf., 1995.

[5] M. P. Andersen, S. Kumar, C. Brooks, A. von Meier, and D. E. Culler, “DISTIL: Design and implementation of a scalable synchrophasor data processing system,” in Smart Grid Communications (SmartGridComm), 2017 IEEE International Conference on, 2015, pp. 271–277.