Proceedings of ASME Turbo Expo 2019
Turbomachinery Technical Conference and Exposition
GT2019
June 17-21, 2019, Phoenix, Arizona, USA

GT2019-91259

TURNING DYNAMIC SENSOR MEASUREMENTS FROM GAS TURBINES INTO INSIGHTS: A BIG DATA APPROACH

Roman Karlstetter*
Robert Widhopf-Fenk
Jakob Hermann
Driek Rouwenhorst
IfTA GmbH
IfTA Ingenieurbüro für Thermoakustik GmbH
82194 Gröbenzell, Germany
Email: roman.karlstetter@ifta.com

Amir Raoofy
Carsten Trinitis
Martin Schulz
Department of Informatics
Chair for Computer Architecture and Parallel Systems
Technical University of Munich
85748 Garching, Germany

ABSTRACT

Gas turbine power plants generate an ever growing amount of high frequency dynamic sensor data. One of the applications of this data is the protection against problems induced by combustion dynamics, as, e.g., with the ArgusOMDS system developed by IfTA. In the light of digitalization, this data has the potential to also be used in other areas and ultimately transform maintenance, repair and overhaul approaches. However, current solutions are not designed to cope with the large time windows needed for a general analysis and this can hinder development of advanced machine analysis algorithms. In this work, we present an end-to-end approach for large scale sensor measurement analysis, employing data mining techniques and enabling machine learning algorithms. Our approach covers the complete data pipeline from sensor measurement acquisition to analysis and visualization. We demonstrate the feasibility of our approach by presenting several case studies that prove the benefits over existing solutions.

1 INTRODUCTION

A large number of gas-fired power plants provide the backbone for varying power demands worldwide. In the context of regenerative power production they are key to stabilizing power grids. This is particularly critical for the European energy market and the increased need for flexibility in the energy supply. With their ability to quickly adapt the power generation to the electricity demand over a broad load range, gas turbines are therefore highly competitive. However, this flexibility comes at a cost: power plants are no longer continuously operated at base load, requiring adaptive tuning, and they are increasingly prone to combustion dynamics. To monitor and actively counteract these combustion dynamics, power plants are typically highly instrumented and monitored through a variety of different—in many cases independent—systems with sampling rates from one to several ten thousand Hz.

Consequently, the generated data volumes vary significantly (few MBs/day of operating data vs. many GBs/day for rotor and combustion dynamics sensor data) posing different processing requirements. Current approaches often only transfer operating and summarized combustion dynamics data to the cloud, constituting a significant loss of information. This design decision, though, is driven by the fact that when managing a complete fleet of gas turbines, even storing this reduced data set is already challenging. Expanding this concept to non-aggregated data or to include high frequency data typically makes transfer, storage, and analysis infeasible.

Despite the challenges, this data, covering a detailed state of the machine during its entire operation, has great potential to aid

*Address all correspondence to this author.

in a deeper understanding and further optimization and improved operation of gas turbines. Understanding how machines and machine behavior change over seasons, over lifetimes or how they differ within a fleet of identical machines is crucial for detecting problems early and avoiding unplanned standstills. This also makes it possible to identify problematic machines or machine components and helps to keep focus on improving the relevant assets. Similar to approaches in an increasing share of industries, it can also enable better maintenance strategies. This means that a transition from "run to failure" to "condition-based maintenance", where maintenance intervals are scheduled according to the actual state of the machine, is possible. Finally, it can help in the development of new generations of gas turbines by understanding issues of older designs.

In this paper, we discuss the challenges and requirements for an end-to-end solution of large scale sensor measurement analysis and present an architecture that enables such analysis in a real-world installation. After a detailed discussion of the challenges in Section 2, we scope our problem to a real-life case study in Section 3 and present theoretical concepts and application requirements regarding data storage in Section 4. In Section 5, we present an analysis architecture and highlight the components of our solution. After that, in Section 6, we compare our solution with existing approaches before we demonstrate the superiority of our approach by means of selected examples. Section 7 sketches ideas and challenges on how data analysis might be further automatized. We wrap up with conclusions and possible future work in Section 8.

2 DATA PIPELINE CHALLENGES

The instrumentation of today's gas turbines produces vast amounts of sensor measurements combined with a wide array of control information. Access to such data in its entirety, in order to enable advanced analysis techniques spanning all data sources as well as long time ranges, leads to several key challenges that must be tackled when developing a big data analysis platform for gas turbine sensor measurements. Moreover, such a platform must consider the entire data pipeline from the sensor data acquisition, to data storage and transmission, to security aspects as well as the required analysis interpretation skills.

2.1 Data Storage

Heavy duty gas turbines are equipped with a variety of different sensors to monitor and control their operation. For monitoring the combustion process of a typical can-type gas turbine with 14 cans/baskets, a pressure sensor is usually installed in each basket, and the dynamic pressure is sampled at a frequency of around 25 kHz in order to be able to observe high-frequency phenomena.

Assuming a sampling frequency of 25.6 kHz, as given in our use case, every day one machine produces

$$14 \times 25600 \, \frac{\text{samples}}{\text{sec}} \times 4 \, \frac{\text{Byte}}{\text{sample}} \times 86400 \, \frac{\text{sec}}{\text{day}} \approx 124 \, \frac{\text{GByte}}{\text{day}}$$

of single precision floating point (32-Bit) raw sensor sample data alone.

Systems like IfTA's ArgusOMDS [1] provide such high resolution sensor capabilities. As they are installed in hundreds of power plants around the world, it makes it potentially possible to monitor a complete fleet of turbines. Together with analysis results and other operational data, like electrical power output, static pressures and temperatures, the data volume quickly becomes intractable (an estimated 12 PB of high resolution data per year) to manage using conventional and existing data pipelines, even only considering a single turbine, let alone a fleet of turbines.

The current state-of-the-art solution for handling this kind of high frequency sensor data is to consider only a limited time window. Data is stored directly at the power plant, managed in a ring buffer and contains only the most recent measurement values. As a consequence—depending on the analysis settings and how many sensors are installed for a machine—at most only few weeks of turbine data can be stored in full resolution on site using this approach. As processing resources are currently very limited at plant sites, no advanced data analytics can take place, losing valuable opportunities for turbine behavioral observation and analysis.

2.2 Data Transfer

In a perfect world all data should be transferred to and be accessible from some kind of compute cloud. This would enable correlation and comparison between machines of a fleet as all data from all the different turbines could be accessed from a single access point. Additionally, making use of the full resolution data requires an environment with less resource constraints, and—aside from a dedicated HPC center, which is not realistic for this sector—only cloud based solutions are able to meet these demands.

However, this vision is currently limited by technical constraints: In order to transfer the 124 GBytes/day from the example above, a dedicated and sustained connection speed of at least ≈ 11.5 MBit/s per turbine would be required. While connection bandwidths are generally increasing, many power plants do not have sufficient bandwidth capacity to transfer all the data to a centralized cloud server, and only a small fraction of plants can provide the necessary bandwidth.

For that reason, to monitor and compare data from different machines, currently only a tiny fraction of sub-sampled and aggregated analysis results is transferred to the cloud. Conse-

quently, the insights this reduced dataset can provide are naturally very limited.

2.3 Security Considerations

In addition to data storage and transfer related problems, we also want to touch on challenges concerning data security when working with big data analytics in an industrial setting. Power plants are critical infrastructure and thus are subject to higher computer security requirements when compared to other applications. As long as no network connections to the outside of the power plant are allowed, physical access restrictions are a very effective measure against most security problems. However, once data analytics applications are deployed in some sort of compute and storage cloud, network and application security aspects have to be considered.

First, the plant network itself and incoming connections need to be secured against unauthorized access. Next, when measurement data are transferred to the cloud, the data should be secured against eavesdropping as well as unauthorized modification. The same security concerns still apply once the data is stored in a data center – it should not be possible for unauthorized parties to modify or access the data. Additionally, many of the challenges in this area are not only technical, but also motivated by trust issues towards data center operators. Solutions in this area are widely researched topics and are generally orthogonal to the design of the data pipeline, as long as all individual segments of the pipeline can be secured. A detailed discussion would therefore go beyond the scope of this paper.

2.4 Required Technical Expertise

The deployment of data processing and analytics faces several non-technical hurdles, which also comprises technical expertise for analyzing the data. As we will present in the remainder of this paper, a vast amount of sensor data is already being collected, for which we have developed solutions for storing and analyzing. Within this context, knowledge in computer science, statistics and, most importantly, the application domain, is of crucial importance. Data mining promises to extract hidden, unknown, but potentially valuable information from big data, but initially and naively often only exposes straightforward correlations, which are trivially known by gas turbine engineers, but not computer scientists lacking the required application knowledge. Finding new insight, though, is comparable to finding a needle in a haystack and ultimately requires the hidden knowledge and intuition of engineers and turbine customers alike when analyzing the data. Only this will allow us to meet the promises of big data, but also requires a framework that is a) easy to use also by non computer scientists, b) is powerful enough to go beyond simple correlations, and c) provides the needed interactivity to allow engineers to explore data in the search for insights.

3 APPROACH AND SETUP

In this work, we present a workflow specifically designed for the scenario of gas turbines and that creates a complete end-to-end pipeline capable of guiding the user from data acquisition to visualization and final analysis. Our work is based on a real-world production setting: We acquire and analyze data from two gas turbines from Stadtwerke München (SWM), the municipal power supplier in the city of Munich, Germany. The data acquisition is performed by IfTA's ArgusOMDS system [1], which is installed on both turbines. Each instance captures measurement values of 14 high frequency pressure sensors at 25.6 kHz and transforms this data into spectral information. Several other systems provide over 100 additional low frequency signals at roughly 2 Hz, which are integrated into the final sensor measurement stream. These signals measure operating data like static pressures, temperatures, valve positions, electrical power output, etc.

Due to technical constraints, the acquired data cannot be streamed over the Internet into the cloud for further processing. We therefore transfer all sensor measurement data via external hard disks, which are exchanged at the power plant in regular intervals and then carried physically to both IfTA and TU Munich for analysis. As a side effect, this approach not only solves all data transfer challenges, but also addresses all security concerns, as the disks can be physically protected. Once stored on the analysis servers, the data is protected by state-of-the-art firewall and identity management procedures.

While this approach addresses all issues for our current framework presented in this work and enables us to offer an end-to-end pipeline, it, of course, does not scale to deployments where we want to monitor a complete fleet of gas turbines. However, as discussed above, the data transfer challenges are orthogonal, and any solution can be merged into the framework discussed here without loss of generality.

4 STORAGE CONCEPTS

The first question in the design of an analysis platform is how to store the acquired data in a form that supports and simplifies additional data analysis, including machine learning. Central to an answer is the fact that (almost exclusively) we are dealing with time series data. This means that all data points are elements of a signal that varies over time and has a timestamp associated with each new value (see Fig. 1). The timestamps for these signals might either be sampled at regular intervals (e.g., exactly 25.6 kHz), but they also might be sampled at irregular time intervals.

Each sensor produces a stream of raw samples. Further, additional analysis algorithms produce derived signals, e.g., peak-to-peak, RMS, spectrum, frequency band values, etc. The data can be thought of as a table where each column is a signal and each row a specific point in time with its related signal values

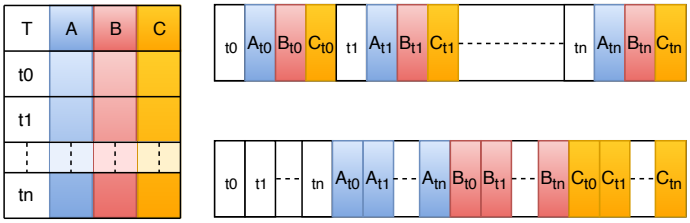

FIGURE 1. *Left)* Sensor measurement time series data in tabular form: for each time stamp, there is a value for each signal. *Top-right)* Row-oriented storage. Values are grouped by time stamp. This provides optimal efficiency for writing data to persistent storage. *Bottom-right)* Column-oriented storage. Values are grouped by signal. This is optimal for sequential reading of a single signal.

(see Fig. 1 *left*). Writing this data stream row by row to persistent storage is very efficient, as the necessary memory requirements are very small and disk-based hard-drives handle the resulting sequential writes very efficiently. In the resulting row-oriented storage scheme, the signals are interleaved within the stream (depicted in Fig. 1 *top-right*).

Once the data is stored permanently, the ability to access the data in the time series efficiently is crucial for any further processing pipeline. In virtually all of the analysis and visualization cases, when a certain signal value is needed, its preceding and following values are also required. On the other hand, typically only a subset of signals is required for a specific analysis. These considerations make it obvious that for analysis workloads, the data should be stored in a way that sequential reads of a single signal can be executed efficiently. This can be achieved by using a column-oriented storage scheme as illustrated in Fig. 1 *bottom-right*, where all values of one signal are stored consecutively on persistent storage.

Unfortunately, the downside of the scenario described above is that, while reading is efficient, writing of the data in a column-oriented format is problematic and will lead to reduced efficiency. More generally, writing of streamed sensor measurements and efficient reading of a single signal for data analysis are contradictory. One of the main reasons why large scale machine analysis so far has been time consuming, is indeed the lack of a storage format that has been designed for efficient writing *and* reading. We will describe how we deal with this conflict in the data processing pipeline in Section 5.

Another aspect related to data storage is data compression. Here, three optimization targets conflict with each other: compression efficiency (the amount of disk space that is needed to store a given sequence of values) as well as compression and decompression throughput (how much time it takes to compress or decompress a certain amount of data). Researching and developing compression algorithms for sensor measurement time series data is a research topic in its own right [2], and hence we only touch the surface of the challenges here. It is important to recognize that there is a hard real time requirement on the compression throughput in a production deployment, i.e., data must be compressed at least as fast as it is produced. For the other two aspects (compression ratio and decompression rate) a satisfying trade-off has to be determined. For the case of offline data analysis, data is usually written just once but read often. The conflict here is between compression ratio and read throughput, i.e., decompressing the data should not limit the speed of accessing it.

The logical data organization (cf. Fig. 1) also has an impact on the performance of data compression. In the case of sensor measurement time series, data stored in a column-oriented layout has consecutive data values that are similar to each other, so the information entropy is smaller and thus, a higher compression efficiency can be achieved.

All the above-mentioned considerations apply to both lossless as well as lossy compression algorithms. For lossy compression, the tolerated amount of information loss is another factor that has to be considered. Typically, the compression ratio can be improved by allowing a higher information loss. In addition to that, application specific lossy compression algorithms generally achieve much higher compression ratios, as they exploit domain knowledge that is embedded in the compression algorithm.

A less technical aspect for storing data is the type of storage system itself and the ecosystem around it. One option is a full database system with integrated query handling and efficient data ingestion and automatic deletion of old data. Some of these tools also feature built-in data analysis capabilities. On the other hand, data can be manually organized in files, avoiding the overhead of a database system, but consequently also lacking its features. In that case, the entire file management must be done manually, including the deletion of files containing old time windows, which cannot be stored infinitely due to limitations for storage capacity and cost.

Data analysis tools can be selected independently, provided that these tools are able to read the data format. Additionally and according to Gorenflo et al. [3], besides the raw performance of a potential storage and analysis system, having a good ecosystem around it is equally crucial.

5 CREATING AN END-TO-END WORK FLOW

Based on the observations and requirements discussed above, we have designed an end-to-end data pipeline that provides the necessary foundation to turn acquired measurement data into insights for the operator. The most important parts of the resulting analysis data flow are sketched in Fig. 2 and described in more detail below.

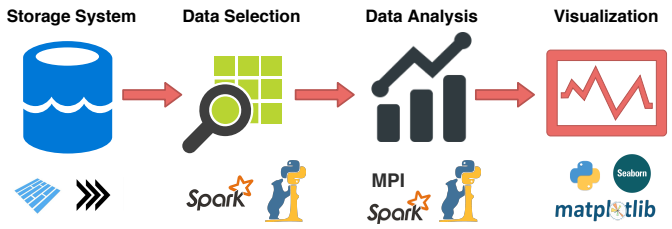

FIGURE 2. Prototype of analysis data flow: data is stored in Apache Parquet files (on a NAS-server). Using a Jupyter Notebook, it is loaded and filtered by Apache Spark or pandas and further processed with these tools. There, data mining algorithms are applied and the results are visualized via matplotlib, seaborn, ipyvolume or other frameworks.

5.1 Data Acquisition and Storage

Sensor measurement and analysis result data is generated by the IfTA ArgusOMDS System [1], located directly within the power plant. In order to avoid data bandwidth and security issues, this data is transferred to our storage system using external hard-disks that are physically delivered by mail. Shipping an 8 TB hard disk provides higher bandwidth compared to the upload speed of a standard VDSL connection. The induced latency, however, is extremely poor, i.e., we get new data only every couple of weeks, which is acceptable when building a first prototype. The data on these external disks is stored in a row-oriented storage format, which is unsuitable for data analysis workloads, and we therefore convert it into a column oriented format.

5.1.1 Data Storage We aim at storing all data in a column-oriented format[1], so we first investigate two potential tools that can be used for storing data.

First, we looked at schema-less storage solutions. *InfluxDB* [4] "*is a time series database designed to handle high write and query loads*" and it has many features that meet our requirements. It features automatic compression of values and timestamps, is optimized for append-only workloads (stream processing) and can automatically delete old values. Unfortunately, there are limitations and issues of InfluxDB that make adoption for our use-case harder. First of all, it only provides limited support for different data types, especially no support for 32 bit floating point values. In addition to this, as the InfluxDB storage engine is schema-less, we cannot exploit the data we are dealing with follows a specified schema. This also means that the time column is unnecessarily duplicated and cannot be queried on its own.

[1] Another alternative optimized for storing time series data, TimescaleDB, is a solution based on PostgresQL, thus implementing a row-oriented storage format. A short test showed that not only does this solution have worse write and read performance, but the data also consumes more disk storage space than with the solutions presented here, so it is not considered in this paper.

Measurement type	Size on Disk	Actual Size	Ratio
Dynamic data	1432.1 MB	3458.1 MB	41.4%
Operating data	5.2 MB	33.8 MB	15.5 %

TABLE 1. Comparison of compressed data size on disk and actual data size when loaded in memory. The sizes on disk represent the size of Parquet files with dictionary compression, containing dynamic or operating data for one day.

Second, we investigated schema-driven storage options. In particular, we examined a file-based approach using the *Apache Parquet* file format [5]. Apache Parquet "*is a columnar storage format available to any project in the Hadoop ecosystem, regardless of the choice of data processing framework, data model or programming language*" [5]. It is an open source implementation of the ideas from Melnik et al. [6] and is also compatible with processing frameworks in the Python data processing ecosystem. It exploits a predefined schema and has built-in support for different compression algorithms. Thus, it supports fast sequential reading of individual signals, can filter data using simple statistics on the columns and uses the available storage space efficiently. One of the main drawbacks of this approach is that file handling has to be done manually. Furthermore, creating Parquet files from a data stream is memory intensive. As the two mentioned issues are not critical for our use-case, yet the read and analysis performance clearly outperformed other options, we decided to base our analysis platform on the Parquet format and use it as target for the data conversion, which is described next.

5.1.2 Data Conversion When a new external hard disk is received from the power plant, the data is first converted into a set of Parquet files. This is done such that one Parquet file contains all the values for a subset of signals for one complete day. This means that for each day, a small number of Parquet files is created, each of them with reasonable file size (up to $\approx$ 1500 MB). These Parquet files use lossless dictionary data compression to reduce the required data storage size on disk (cf. Tab. 1). The sets of files are then organized into folders, such that for every day of turbine data, there is one folder containing the respective Parquet files.

5.2 Data Processing

Once the conversion is complete, the resulting Parquet files serve as input for any further analysis step, requiring processing options capable of handling Parquet files, while providing high performance and ease-of-use for the end user. For this, we settled on two tools, namely Apache Spark and the Python framework *pandas*.

Apache Spark™, is "*a unified analytics engine for large-scale data processing*" [7]. Spark has an easy SQL-like interface for simple data filtering and processing, can work on streaming data, and has an integrated machine learning library. It provides a convenient interface for data stored in the Apache Parquet file format, and can be used to filter data and perform simple data aggregation. Its main advantage comes into play when data is too big to be stored or processed by one machine: it can be deployed on a cluster of machines, which all store data and perform computations on all nodes in the cluster. However, it also works on a single machine, which makes it a perfect tool for prototyping.

To complement the functionality of Spark for data analysis, we use the Python framework pandas [8]. pandas is a "*[. . .] library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language*" [9]. It can read Parquet files, perform simple data filtering and aggregation tasks, but can also be used to create simple visualizations.

5.3 Visualization

Finally, we visualize the analysis results in order to be able to understand, discuss and communicate them. Thanks to the use of pandas and the availability of the Python ecosystem, we can rely on a variety of tools and frameworks that provide visualization capabilities. Probably the most widely used of such frameworks is *matplotlib* [10], which provides substantial functionality (and which the aforementioned plotting capabilities of pandas rely on). Additionally, we use *seaborn* [11], which provides a high-level interface for matplotlib. Finally, we add *ipyvolume*'s [12] interactive 3D-visualization to present our analysis results to the end-user.

5.4 Driving the Pipeline

In order to be able to quickly and interactively apply all steps of data analysis, we decided to use *Jupyter Notebooks* [13]. They make it possible to quickly develop and interactively execute Python code and with that enable us to make analysis and visualization steps available to the end user in a tangible manner. The resulting documents or notebooks perfectly combine algorithmic descriptions as well as analysis results including sophisticated visualizations. This approach furthermore provides a very valuable way of communicating and presenting results. As a consequence, we also use it for prototyping new algorithms.

6 EXAMPLE DATA ANALYSES

In the following we highlight three of the exemplary analyses that are enabled by our proposed framework. All results are obtained on an Intel® Xeon® Gold 6136 CPU @ 3 GHz. As the storage requirements are significant (and growing continuously), we located all Apache Parquet data files on a Synology NAS with sufficient storage capacity and, more importantly, a 10 GBit/s link to the server. On a much smaller data set, these analyses would likely also have been possible with the intuitive IfTA Trend [14] visualization and analysis software. However, the analyses would have taken significantly longer to create, both in terms of actual working hours as well as computational time required for data processing.

6.1 Comparison of Workflows

First, we want to highlight the difference in work flow between the traditional, event based way of data analysis and the new *big data* based analysis.

In the former case, the analysis is triggered by observing some problem with a machine, e.g., some kind of turbine failure. Often, this event based and manual approach is only done if there is an immediate need for answering a specific question (*"Why did the compressor of my machine fail?"* or *"Why did our combustion monitoring system cause a turbine shutdown on date ...?"*) about a certain event, so the time window in question is known. By utilizing an aggregated data set with a reduced resolution, the data is limited to a manageable size and can be visualized interactively. While not showing all details, in most cases, the user can further narrow down the time window and iteratively load and visualize time windows with higher resolution. Looking at a list of appropriate signals (in the iteratively refined time window in question) reveals insight into the data and allows the user to draw conclusions.

Almost all of this analysis is performed with the help of interactive visualization for several reasons: first, as the data of only a limited time window is investigated, it is the most straightforward way to analyze data visually. In most cases, interactive visualization is comfortably feasible performance-wise for the limited amount of data that needs to be analyzed. In addition to giving the user immediate cues on which parts of the data are interesting, this also allows easy communication of the analysis results. On the other hand, the time window of accessible data is quite constrained, so there are few better alternatives to visualization, as many of them would require a lot more data.

As soon as more data can be accessed, the analysis work flow naturally changes. In addition to the old event based analysis requests (which can still be answered in the same way), in many cases, there are now further questions that are not as specific (*"Are all of the combustion cans working equally well?"* or *"How much did the efficiency change over the lifetime of the machine and why?"*). This means that they cannot be answered by looking only at a certain time period of the data or an aggregated view of the data. Hence, the pure visualization based way of data analysis quickly breaks down for clear reasons.

Consequently, it is necessary to filter the data according to some analysis driven criteria that is useful for the question to be answered. These filter conditions range from simple manually

selected signal thresholds to automated, shape based signal selection. Since the resulting data set might still be too large to be visualized immediately, it needs further processing. Again, techniques range from very simple data aggregation operations, to statistical analysis or to more advanced approaches like machine learning algorithms (cf. Section 7). Furthermore, it is often required to correlate different signals of the data set or intermediate analysis results with each other, which is also made feasible through the use of Parquet files.

In the rest of this section, we present example data analysis tasks that can now be carried out efficiently using our proposed analysis architecture.

6.2 Example 1: Frequency Distribution Of Signals

In this first example, using our presented approach, we show that loading data of roughly 18 months of two machines in full resolution and performing analysis on this data is feasible in a strikingly efficient way. The data has an average sampling rate of almost 2 Hz, resulting in $\approx 6.6 \cdot 10^7$ samples, or almost 800 MB (8 byte timestamp + 4 byte value per sample). It should be noted that when we would have loaded this amount of data using the original row-oriented storage scheme, it would have been necessary to read several dozens of TB from disk. Even with a bandwidth of 10 GBit/s, this would have taken hours just for a single turbine. Using our new analysis platform, loading and grouping data for two turbines takes less than 15 seconds.

For the plot in Fig. 3, we group the electrical power output data weekly and calculate the histogram for each week. We then plot these histograms color-coded, visualizing the relative frequency of the respective power output state for the respective week.

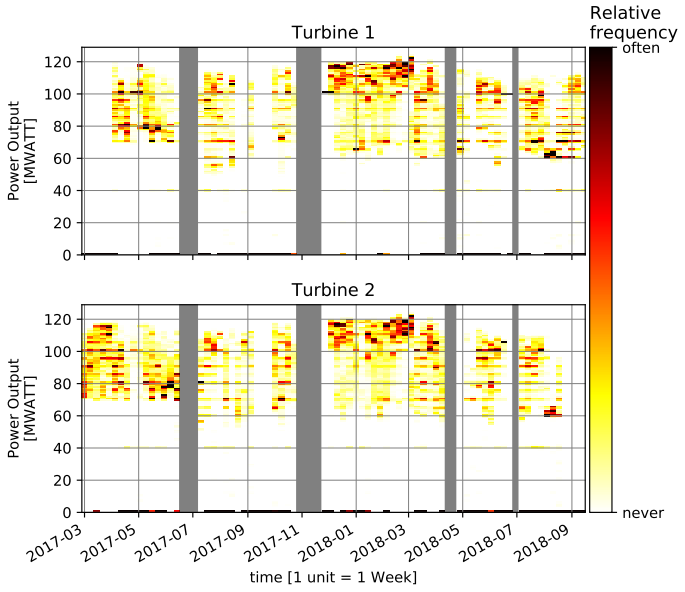

FIGURE 3. Histogram of relative frequency of power output over roughly 18 months, grouped by week. The gray areas indicate times when no data is available. This analysis provides an overview over a long time window and enables visual comparison of two gas turbines of one power plant. Furthermore, it shows that during winter in central Europe, the power output is higher due to an increased electricity demand and additional need for district heating (the respective power plant is also operated for district heating mode).

6.3 Example 2: Comparison Of Signal Shape

Next, we look at the turbine speed data of the complete database, but now use a different approach for data selection: we search for a specific time series pattern shape. As we know that a slower sampling rate is enough to detect the phenomena we are interested in, we first subsample to 0.05 Hz and average the values. We filter on the subsampled data by manually selecting prototype patterns (behavior of turbine speed for startup and shutdown), and then find similar occurrences in the data. An efficient way to do so is, e.g., by calculating the distance profile using the MASS algorithm by Mueen et al. [15]. Having the distance profile, we empirically determine an appropriate threshold for the distance to get all matching time windows. After excluding false matches by examining the absolute turbine speed values of startup and shutdown, we have reduced the data set to the time windows of interest.

Lastly, we simply plot all the selected startup and shutdown sequences into a single plot. As one can see in Fig. 4, most of the time the behavior is indeed very similar. Only for the shutdown process of the second turbine in our example data set, there are larger deviations, which are easy to detect by the proposed visualization.

Since the selection of the time windows is based on the shape of the time series, it is not necessary to manually align the data, this is a feature of the automatic selection of the time window. As before, loading the 18 month of data takes a little less than 10 seconds for the data of two turbines in our setup. Actually selecting the relevant time windows by querying for the signal shapes is also remarkably efficient: it takes less than 2 seconds for both machines and both patterns. It is worth noting that this selection process is independent of the signal shape, as the employed algorithm does not optimize with respect to data values. There is no other end-to-end solution that we know of which provides the capability of selecting real world gas turbine sensor measurement data by signal shape.

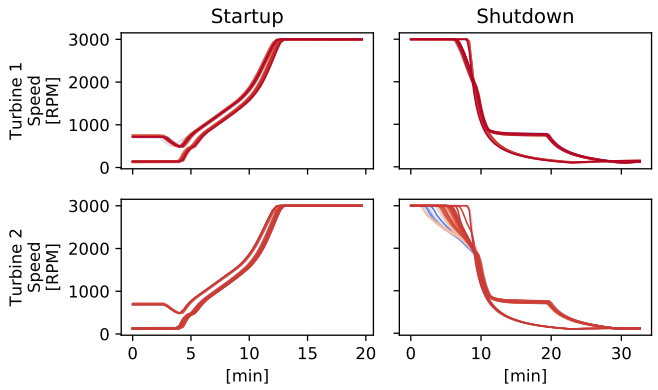

FIGURE 4. Visualization of the shaft speed of multiple startup and shutdown sequences for two machines of the same type. All extracted sequences are automatically aligned and plotted on top of each other. While the startups look very similar for both machines, the shutdown process of the second machine is subject to noticeable fluctuations.

6.4 Example 3: Visually Checking Signal Quality

In contrast to the previous examples, for this analysis, we use the high frequency pressure sensor data, sampled at 25.6 kHz and transformed to the frequency domain using an FFT of length 320 ms. In particular, we are looking at the time windows of all the startups of one machine (cf. Fig. 4) and display the spectrum as spectrogram plot over the time of a startup. By showing all combinations of cans/baskets and startups, we get an overview of the measurements of all signals and how the behavior of the monitored values changes over time. Figure 5 shows an excerpt from such an overview plot with only 25 such combinations. Even in this small part of the overview, a person can effortlessly identify the normal and abnormal behavior, e.g., realize that the signal amplitude for basket 5 is always too low. Also, the signals for Basket 2 and 3 stopped looking normal after startup 3. After investigation and discussion with the operator of the turbines, it turned out that this was after a maintenance stop – likely a reassembly problem in the measurement chain. Looking only at a few restricted frequency band values (as available in a reduced data set in the state-of-the-art approach) instead of the full frequency spectrum is often not sufficient.

Due to space limitations, we here limit ourselves to a grid of 5x5 startup spectra. We have created a similar plot for a turbine with 14 cans and all startups that the data for this turbine covers. This results in a grid of 14x75 subplots. The resulting plot was shown to the engineering staff of the turbine, which identified faulty sensors and ultimately led to the replacement of these sensors.

In this first ever approach to analyze spectrum data in a large scale, the spectrum data for 14 cans requires ≈ 672 MB of memory, which in our framework and due to the Parquet format, takes under 3 seconds to load.

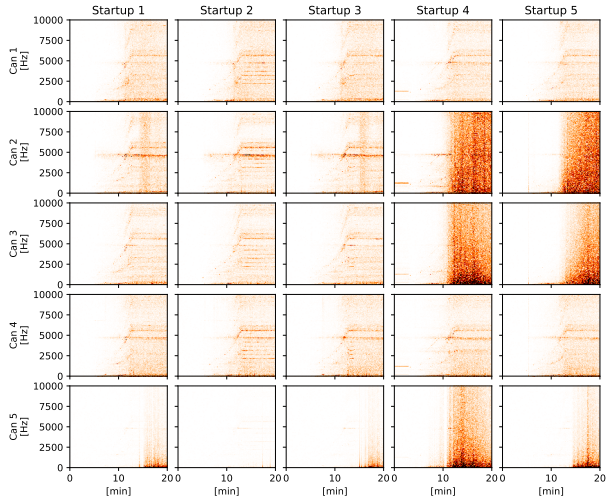

FIGURE 5. Every subplot in this 5x5-grid represents the intensity plot of the spectrum of one can/basket over the time of one startup process (cf. Fig. 4) of the turbine. All subplots of one row show the measurements of one particular combustion can, so one row reveals the changes in process over time of that particular can. One column represents one startup process (ordered by time), so all plots in one column show the same startup process for different cans. It can be clearly seen that the data for the last can does not match the *normal behavior* (other four cans, first three startups). For cans two and three, the measured data changes significantly for startups four and five.

6.5 Summary: New Capabilities

The presented analyses show only a small subset of possibilities that can now be realized using our proposed end-to-end analysis platform. Nevertheless, all of this already helps gas turbine engineers and teams operating gas turbines to gain insight into the instrumentation and operation of their machines. To the best of our knowledge, no other end-to-end approach exists that analyzes data at the given high time and frequency resolution for time windows of more than one year.

7 ADVANCED ANALYSIS ALGORITHMS

All examples in the last section use rather simple filtering, data processing and visualization techniques. This end-to-end data analysis approach, based on an efficient file format and when applied to long time windows of sensor measurement data, yields very meaningful results close to interactive handling. However, there is huge potential to also apply more advanced analysis algo-

rithms on top of the proposed analysis platform to further expand the capabilities of the system. Thus, in this section, we present a concept on what type of tools and algorithms might yield even more insights into the data or make analysis more robust.

7.1 Algorithms

One weakness of all the examples presented in Section 6 is the need for human interpretation of the results. While often the only solution as user expertise is needed, it is feasible only for a small number of machines or when analyses are only performed rarely. When scaling to a fleet of machines or demanding regularly repeating analysis, the required ability to interpret the results quickly becomes the bottleneck if performed by a human. Due to recent advances in machine learning, some of these potential analysis tasks can be automated, especially those that do not require much creativity. In particular, repetitive tasks like anomaly detection or operating mode classification can be easily performed by algorithms trained with historical data.

Anomaly detection algorithms range from simple threshold based algorithms to sophisticated statistical or neural network based machine learning algorithms. IfTA's FleetMonitor software already offers the basic automated analysis capabilities for a centralized and automated analysis of the measurements of a fleet of gas turbines. Many of the advanced algorithms are implemented in, e.g., the scikit-learn library [16]. Other examples using neural networks include auto encoders for anomaly detection or recurrent neural networks for classification of behavior in time series.

Regarding time series analysis, similar to how we select signals of a specific shape to examine the startup and shutdown behavior (cf. Fig. 4), it is interesting to detect repeating patterns (motifs) and novelty/anomalies (discords) in the sensor time series measurements. Furthermore, it is interesting to segment and classify the time series. Using the Matrix Profile data structures developed by Keogh et al. [17], all of this is now computationally feasible [18].

7.2 Data and Meta-Data Quality

One aspect that has not been covered for all of the abovementioned analysis techniques, but which is essential for good analysis results, is data quality. Concerning measurement values themselves, problems might be in each stage of the measurement chain, starting with the analogue path of this chain, which consists of different kinds of sensors, charge amplifiers, cables and connectors between the components. In each of these components, we have to deal with gradual degradation over their lifetime caused by wear (frequent temperature changes, heat, physical stress). Besides that, abrupt changes in signal quality are also an issue, e.g., after maintenance stops when turbine engineers made mistakes while reassembling the measurement chain (cf. Fig. 5).

Similarly, we must also carefully address problems in the digital section of the chain. Different digital systems have to work together reliably in order to have one integrated database of sensor measurement values. When one of this systems is down or does not work correctly, no or even wrong measurement values might be the consequence. In addition to that, having consistent timestamps for data generated by different systems is also an issue that has to be taken care of.

Beyond that, there are problems that do not directly concern the actual sensor measurements, but still are of great importance for an analysis platform. These *meta-problems* cover the description of data and how it changes over time as well as the assessment of signal quality discussed in the previous paragraph. It might be necessary to exclude time windows where some signals are missing or have faulty signal values, so data for such time windows might need to be annotated as such.

Furthermore, over time, sensor measurements or analysis results might be added or removed from the set of signals. Moreover, configuration parameters (e.g., sampling frequency) might be changed, and thus, the semantic of analysis results also slightly changes. When such a change in data schema or configuration happens, this might complicate the problem, as data analysis now also has to take care of these kinds of changes in the data.

8 CONCLUSIONS & FUTURE WORK

When IfTA was founded in 1996, it was virtually impossible to store, transfer and process the data returned from a machine. However, in the last decade, storage systems and computers have evolved considerably and cloud computing has become a game changer and has made it possible to cope with an ever growing amount of high resolution data.

We presented an end-to-end approach for a large scale sensor measurement analysis platform. Meaningful examples that would not have been possible with currently available solutions demonstrate the feasibility of our approach for the complete data path from sensor value acquisition to visualization of analysis results. Real-world actions in response to one of our analysis results show that the insights can provide direct benefits to machine operators at the respective power plants.

The traditional approach of sending an external hard disk apparently has an unacceptable high latency and does not scale for a complete fleet of turbines. Consequently, in future work, we plan to further optimize data storage and tackle the data transfer challenge by improving data compression and exploiting lossy compression algorithms. Ideally, data is already transformed into a format suited for analysis when it is generated at the power plant. Furthermore, distributed data storage and processing will make large scale fleet monitoring viable and with that open a door to a new generation of monitoring, analysis, and maintenance scheduling in the light of digitalization.

ACKNOWLEDGMENT

This work is funded by the research project *Optimierung von Gasturbinen mit Hilfe von Big Data* (AZ-1214-16) by Bayerische Forschungsstiftung. Furthermore, we thank Stadtwerke München GmbH providing us the data that the analyses in this paper are based on.

REFERENCES

[1] IfTA GmbH, 2018. IfTA OMDS - Oscillation Monitoring & Diagnostics System. Last accessed October, 2018. URL https://www.ifta.com/en/products/automation/omds.

[2] Blalock, D., Madden, S., and Guttag, J., 2018. "Sprintz: Time Series Compression for the Internet of Things". *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.,* ***2***(3), Sept., pp. 93:1–93:23.

[3] Gorenflo, C., Golab, L., and Keshav, S., 2017. "Managing Sensor Data Streams: Lessons Learned from the WeBike Project". In Proceedings of the 29th International Conference on Scientific and Statistical Database Management, SSDBM '17, ACM, pp. 1:1–1:11.

[4] InfluxData, Inc., 2018. InfluxDB. Last accessed October, 2018. URL https://docs.influxdata.com/influxdb/.

[5] Apache Software Foundation, 2018. Apache Parquet. Last accessed October, 2018. URL https://parquet.apache.org/.

[6] Melnik, S., Gubarev, A., Long, J. J., Romer, G., Shivakumar, S., Tolton, M., and Vassilakis, T., 2010. "Dremel: Interactive Analysis of Web-Scale Datasets". In Proc. of the 36th Int'l Conf on Very Large Data Bases, pp. 330–339.

[7] Apache Software Foundation, 2018. Apache Spark. Last accessed October, 2018. URL https://spark.apache.org/.

[8] McKinney, W., 2011. "pandas: a foundational python library for data analysis and statistics". *PyHPC*.

[9] Pandas, 2018. Python Data Analysis Library. Last accessed October, 2018. URL https://parquet.apache.org/.

[10] Hunter, J. D., 2007. "Matplotlib: A 2d graphics environment". *Computing In Science & Engineering,* ***9***(3), pp. 90–95.

[11] Michael Waskom, 2018. seaborn: statistical data visualization. Last accessed October, 2018. URL https://seaborn.pydata.org/.

[12] Maarten Breddels, 2018. ipyvolume. Last accessed October, 2018. URL https://github.com/maartenbreddels/ipyvolume.

[13] Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., and Willing, C., 2016. "Jupyter Notebooks – a publishing format for reproducible computational workflows". In Positioning and Power in Academic Publishing: Players, Agents and Agendas, F. Loizides and B. Schmidt, eds., IOS Press, pp. 87 – 90.

[14] IfTA GmbH, 2018. Measurement Data Visualization & Analysis with IfTA Trend. Last accessed October, 2018. URL https://www.ifta.com/en/products/analysis/trend.

[15] Mueen, A., Zhu, Y., Yeh, M., Kamgar, K., Viswanathan, K., Gupta, C., and Keogh, E., 2017. The fastest similarity search algorithm for time series subsequences under euclidean distance, August. http://www.cs.unm.edu/~mueen/FastestSimilaritySearch.html.

[16] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E., 2011. "Scikit-learn: Machine Learning in Python". *Journal of Machine Learning Research,* ***12***, pp. 2825–2830.

[17] Yeh, C. C. M., Zhu, Y., Ulanova, L., Begum, N., Ding, Y., Dau, H. A., Silva, D. F., Mueen, A., and Keogh, E., 2016. "Matrix profile i: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets". In 2016 IEEE 16th International Conference on Data Mining (ICDM), pp. 1317–1322.

[18] Zhu, Y., Zimmerman, Z., Senobari, N. S., Yeh, C. C. M., Funning, G., Mueen, A., Brisk, P., and Keogh, E., 2016. "Matrix profile ii: Exploiting a novel algorithm and gpus to break the one hundred million barrier for time series motifs and joins". In 2016 IEEE 16th International Conference on Data Mining (ICDM), pp. 739–748.