Monitoring Electrical Power Consumption with Kieker

Sören Henning
soeren.henning@email.uni-kiel.de
Kiel University, Germany

Abstract

Detailed knowledge about the electrical power consumption is a prerequisite for reducing it in manufacturing enterprises. Therefore, a continuous monitoring and analysis of the power usage of the overall business as well as of its individual devices, machines, and production plants is demanded.

In this paper, we present an approach for such a monitoring infrastructure. It facilitates the integration of different types of sensors in order to make their measurements comparable. Automatically and in real time, it analyzes and visualizes recorded data in a number of ways. Supported by patterns and technologies for scalable, big data processing systems, this approach utilizes the software performance monitoring framework Kieker in large parts. We implement a prototype of this approach and inspect the power consumption of servers in a medium-sized enterprise.

1 Introduction

Legal as well as self-imposed regulations such as ISO 50001 force enterprises to reduce and optimize their electrical power consumption. In particular, load peaks should be reduced as those are significantly more expensive. This is particularity challenging for manufacturing businesses, which typically operate a variety of different devices, machines, and production plants.

To discover saving potential it is necessary to monitor all consumers and to analyze and visualize their consumptions. The data should be monitored as detailed as possible in order to analyze individual consumers or the consumption at particular points in time intensively. However, also summarized and simplified analyses are necessary to make data comprehensible so that decisions can be made quickly.

We present how a monitoring infrastructure for power consumption can be realized in software. The remaining paper starts by listing the functional and non-functional key requirements such an infrastructure has to meet in Section 2. Based on that, we present an architecture for such a software system in Section 3. Section 4 shows how a prototype of it can be realized using technologies from software performance monitoring and, in Section 5, we evaluate this. Finally, Section 6 concludes this paper and discuses future work.

2 Requirements

In order to gain knowledge about the power consumption, we deem the following four consecutive steps necessary:

Data Acquisition Recording the data is the fundamental step of analyzing power consumption. Caused by increasingly powerful hardware, more and more devices are equipped with sensors and network capabilities. Since not every device or machine is able to monitor its power consumption itself, it is also possible to use utility sensor devices such as a monitoring power socket. The data that is produced by such devices are manifold and power consumption can be measured from various points of view. In our approach, we only consider active power as measured by conventional electricity meters.

Data Cleaning and Integration Devices and machines in production environments mostly come from different manufactures located in different business domains. Furthermore, they are likely to differ in their age and originate from different generations of technological evolution. That leads to the fact that also the way they supply data varies widely. Most notably, this is due to the protocols and data formats they use but also to the way they measure. Parameters such as precision, sampling rate, or measurement units may vary from domain to domain.

In order to compare data of different sensors and to consider the data analysis from a higher level, data first have to be brought into a common format. This also includes converting measurement units or splitting up individual measurement that are sent together. Moreover, it is possible that not all measurements are of interest and only specific values have to be selected. As the amount of data may be too large to be analyzed, it is often reasonable to aggregate measurements at first.

Data Analysis The individual consumption values of devices are often too detailed to draw conclusions about the entire production environment. Instead, it may often be more reasonable to evaluate data for an entire group of devices. That is even more significant in cases where devices have more than one power supply and those are monitored individually. It is likely that in such cases, only the summed data is of inter-

est. Therefore, users of our infrastructure can arrange all sensors in nested groups. Then, our approach automatically computes the consumption values of those groups.

Apart from the hierarchically aggregation, we perform further analyses to enable advanced visualizations. Therefore, the corresponding visualizations define which analyses these are.

Data Visualization In order to make monitored and analyzed data perceivable, it has to be visualized. That enables a user to draw conclusion about the current state of the overall consumption and to decide about the further operation.

For the realization of this approach, also non-functional requirements have to be addressed. Both the architecture and a corresponding implementation should be horizontally scalable in order to provide a solution for small-scale application environments as well as for arbitrary large ones. Also within the same application scenario the amount of monitoring data can vary – on the one hand during the operation (e.g., depending on the time of day), but on the other hand also if more and more departments are integrated in the monitoring.

Adaptability is another requirement of our approach. The presented prototype realizes only a few analyzes and visualizations. It is likely that this has to be extended or existing ones will be revised later.

Furthermore, the data should be processed in real time. That means, at any time, the results of the analyses and the visualization show the current state of the monitored system. This is the only way to react to unexpected events in time or to evaluate the current operation.

3 Microservice-based Architecture

Considering the defined requirements, we designed a microservice-based architecture [2] for the desired monitoring infrastructure. Following the microservice pattern allows us to encapsulate components according to business capabilities, which enhances the adaptability [6]. Moreover, this pattern facilitates the realization in a distributed system, in which the single services can be scaled individually [3].

Figure 1 shows a graphical representation of our architecture. It contains the three microservices *Record Bridge*, *History*, and *Configuration*. Here, the Record Bridge functions as a placeholder for multiple microservices, where each Bridge service integrates a specific sensor type. The History service continuously aggregates incoming monitoring records and stores both the aggregated and the monitored records in a database. Using the stored data, it is also able to execute analyses on them. The Configuration service maintains a model representing the way how sensors can be grouped. Whereas the Record Bridge solely contains application logic, the History and Configuration services additionally contain a data storage subcomponent.

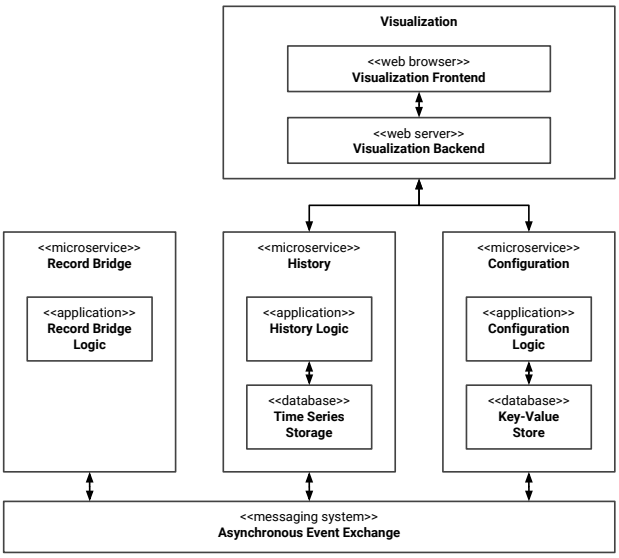

Figure 1: Microservice-based architecture

As the amount of monitoring data can become immense, our architecture design features an asynchronous messaging system to distribute and forward records. It allows to send and receive monitoring records asynchronously and, moreover, handles fault-tolerance and scaling by deploying several instances.

The *Visualization* component is not a typical microservice as it does not represent an own business function but instead serves as an integration of different business functions. Following the *Backends for Frontends* pattern [2], it consists of two parts, a backend and a frontend. Our approach features several different forms of visualization [8], for instance, a chart showing the consumption's evolution over time.

Since the individual services only require normal network connections between them, they can be deployed in totally different contexts. For example, it may be reasonable to deploy the analysis and storage components in an elastic cloud infrastructure, but, following the concepts of edge computing, the Record Bridges physically close to the sensors.

4 Implementation using Kieker

Based on the previously described architecture, we developed a prototypical realization [8]. In order to enable different services working on the same monitoring data, there must be a common data format that is understood by every service. Even if our prototype only supports active power as a metric, there should be a generic format to exchange data.

The Kieker monitoring framework [1] faces similar challenges, although in the context of software performance monitoring. Even if large parts are not relevant, we decided to adapt Kieker for our implementation as it provides two major benefits:

1. The Instrumentation Record Language (IRL) [4] enables us to define record types in a generic way. Besides record types for active power, we can, therefore, later define record types for other metrics and process them.

2. Kieker's writing and reading infrastructure facilitates an abstraction of the record transmissions and storage process. In our implementation, we use Apache Kafka as messaging system and Apache Cassandra to store records for the long term. Kieker already provides writers and readers for both [5, 7], which only had to be adapted for our demands. Moreover, using Kieker, we can easily expand our support to various other technologies.

5 Feasibility Evaluation

In the following evaluation, we show how our prototype can integrate real sensors locate in an industrial environment. It encompasses the entire process of our approach: Integration of data via the Record Bridge, processing them in terms of writing into the database, and the visualization.

Methodology We carried out the evaluation using a power distribution unit (PDU) for server racks. It is installed in a medium-sized enterprise and power supplies severs. It is configured to record the current active outlet power for one server and pushes the measurements every second to a Record Bridge [8].

Therefore, we developed a specific Record Bridge service for this PDU. Along with all other components, it is deployed in a Kubernetes operating private cloud consisting of four nodes and one controller.

Results We conducted the evaluation over a period of two weeks and were able to record data throughout the entire test period. The screenshot in Figure 2 shows an extract of the data of about one hour. The measured consumption is mostly constant with a value of about 33 Watt but shows significant outliers of about 60 Watt. Randomly selected log messages confirm this.

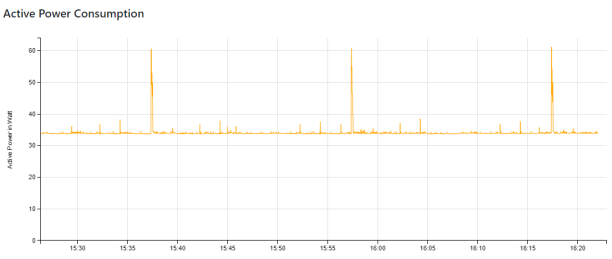

Figure 2: Screenshot of the time series chart visualization component

We conclude that the integration of the PDU is successfully implemented. In particular, we conclude that the transformation of data by the Record Bridge service works as desired. Also the storage of data was successful as we notice that records are stored with timestamps at intervals of one second. Moreover, the visualized data correspond to the one returned by the History service's API.

6 Conclusions

In this paper, we presented an infrastructure for monitoring electrical power consumption in manufacturing enterprises. For this purpose, we listed requirements for such a system and outlined a consequential architecture. We were able to realize a prototype by utilizing Kieker, which we evaluated successfully in a medium-sized enterprise.

As future work, we plan to provide more complex analyses and visualizations, for instance, to automatically detect anomalies in the consumption. Furthermore, we plan to integrate other consumption metrics and production or enterprise data in order to correlate them with each other. Another promising field of research would be the correlation of power consumption with software monitoring data.

References

[1] A. van Hoorn, J. Waller, and W. Hasselbring. "Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis". In: *Proceedings of International Conference on Performance Engineering*. 2012.

[2] S. Newman. *Building Microservices*. 1st. O'Reilly Media, Inc., 2015.

[3] W. Hasselbring. "Microservices for Scalability: Keynote Talk Abstract". In: *International Conference on Performance Engineering*. 2016.

[4] R. Jung and C. Wulf. "Advanced Typing for the Kieker Instrumentation Languages". In: *Symposium on Software Performance*. 2016.

[5] A. Moebius and S. Ulrich. "Improving Kieker's Scalability by Employing Linked Read-Optimized and Write-Optimized NoSQL Storage". In: *Symposium on Software Performance*. 2016.

[6] W. Hasselbring and G. Steinacker. "Microservice Architectures for Scalability, Agility and Reliability in E-Commerce". In: *Proceedings IEEE International Conference on Software Architecture Workshops*. 2017.

[7] H. Knoche. "Refactoring Kieker's I/O Infrastructure to Improve Scalability and Extensibility". In: *Softwaretechnik-Trends* 37.3 (Nov. 2017), pp. 17–19.

[8] S. Henning. "Prototype of a Scalable Monitoring Infrastructure for Industrial DevOps". Master's Thesis. Kiel University, 2018.