A Microservice architecture for monitoring, processing and predicting climate data in animal husbandry

Alexander Stein, Marcel Zillekens, and Marius Khan

[1] University of Applied Sciences and Arts Dortmund,
[2] Institute for Digital Transformation of Application and Living Domains, 44227 Dortmund, Germany
{firstname.lastname}@fh-dortmund.de

Abstract

In this work we present a microservice architecture that offers a reliable and scalable system to monitor and process incoming climate data from sensors of hog houses. Through data processing our system predicts trends about long term climate data. By using machine learning algorithms and based on the trends, the system recommends parameter values for climate computers installed in hog houses.

1 Introduction

In the field of intensive animal husbandry there are increasing issues regarding animal diseases induced by poor air conditions [1]. Thus, farmers are forced to increase medication which produces additional costs and diminishes the quality of meat [2]. We discuss an intelligent and flexible software to permanently monitor the noxious gas emissions of pigs in intensive animal husbandry which is able to reduce these emissions by long term data acquisition and analysis. Statistical methods and machine learning algorithms allow the prediction of trends. This in turn enables the extraction of recommendation values for installed climate computers in the barns for an early adjustment of the overall climate condition.

2 Climate data processing pipeline

The cloud application is based on a microservice architecture to establish the flexibility, scalability, expandability and modularity needed for the system to allow continuous data processing of hog houses in parallel. Thus, the architecture is decomposed into several components on different logical layers. Figure 1 shows the software architecture with each layer being represented by a distinct background color. Infrastructure services are depicted as uncolored boxes. They correspond to those being mentioned in [3]. Message brokers handle the backbone communication between services. The *Data Reception* service receives sensor values from the hog houses and checks its validity. The data is being stored in the *Persistence* and forwarded to the *Data Analysis and Regression*. With these services our system provides a pipeline architecture with multiple processing stages. In its core, machine learning algorithms perform self-optimization and predictions on climate data. To archive this we used a time series analysis with statistical techniques like moving average on historical data of sensor values to smooth out short-term fluctuations in the data curves [4]. In the next stage we derived our predictions by using a regression model [5]. The *Rule Engine* then uses fuzzy-logic to extract recommended settings for the ventilation system controlled by the climate computers [6]. The Engine loads predefined rules for the different sensor types and compares the current sensor values with its set point values. By additionally taking the determined climate predictions into account, this system is able to operate in a preventive manner.

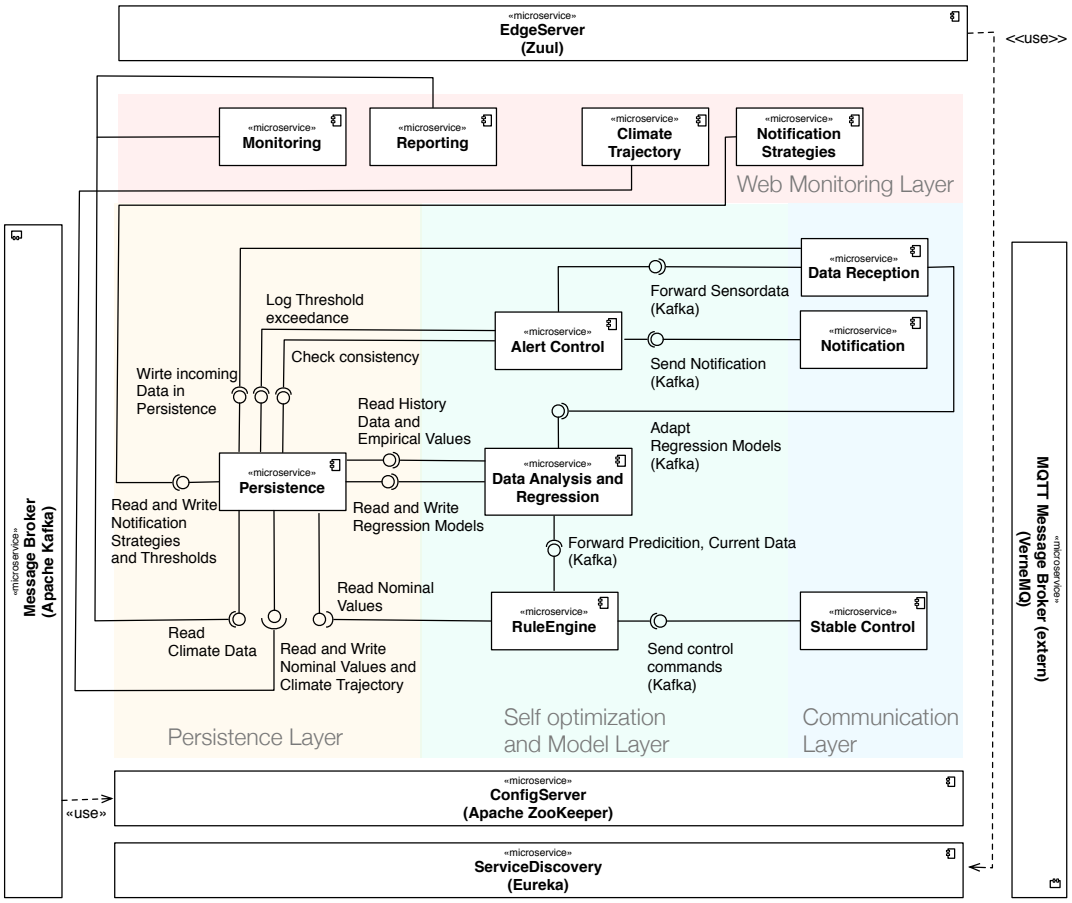

Figure 1: Diagram of the microservice architecture

3 Conclusion and Discussion

The established microservice architecture described in section 2 gave us the option to deploy additional core services on demand for load balancing and scaling using *Docker*[7]. This underlying container virtualisation software also allowes a clustered operation of the infrastructure over different hardware platforms which further enhanced the scalability. The use of message brokers like *Apache Kafka*[8] added stability and reliability to the backbone communication by its Quality of Service features. The architecture itself is based on the *Netflix OSS*[9] which provided modularity that enabled the exchange and integration of different technologies within a single continuous system. This was an important reason for us to use the microservice approach because we tried to archive as much flexibility as possible for our machine learning pipeline due to the quick progression and widespread amount of technologies in this domain. At the beginning of the project we used the Java framework *MLlib* from *Apache Spark*[10] because it is optimised for big data analysis. During development new features for the modern *TensorFlow*[11] framework where released and we found that this technology suited better to our

needs regarding the derivation of predictions from regression models. In contrast to *MLlib* the *TensorFlow* framework is written in C++ and Python. Therefore some of our machine learning services are written in Python, while other services of our system are based on Java. This let to a heterogeneous technology landscape which the microservice architecture is predestinated for.

References

[1] A. Haeussermann, T. Jungbluth, and E. Hartung. Nh3 emission from pig husbandry in relation to ventilation control and indoor air cooling. Workshop on Agricultural Air Quality, 2008.

[2] Q. Chang et al. Antibiotics in agriculture and the risk to human health. 2014.

[3] A. Balalaie, A. Heydarnoori, and P. Jamshidi. Microservices architecture enables devops: Migration to a cloud-native architecture. *IEEE Software 33, 3*, page 42–52, 2016.

[4] J. D. Hamilton. *Time Series Analysis. Moving Average Processes.* Princeton University Press, 1994.

[5] G. A. F. Seber and A. J. Lee. *Linear regression analysis.* Wiley series in probability and statistics, 2 edition, 2012.

[6] F. M. McNeill and E. Thro. *Fuzzy Logic: A Practical Approach.* Academic Press, 2014.

[7] Inc. Docker. Docker. https://www.docker.com/, 2013.

[8] Apache. Apache Kafka. https://kafka.apache.org/, 2011.

[9] Netflix. Netflix OSS. https://netflix.github.io/, 2015.

[10] Apache. Apache Spark. https://spark.apache.org/, 2014.

[11] Google Brain team. TensorFlow. https://www.tensorflow.org/, 2015.