Environmental Monitoring of Libraries with MonTreAL

Marcel Großmann[1(✉)], Steffen Illig[2], and Cornelius Matějka[1]

[1] Computer Networks Group, University of Bamberg, 96047 Bamberg, Germany
marcel.grossmann@uni-bamberg.de,
cornelius-lucian.matejka@stud.uni-bamberg.de
[2] University Library of Bamberg, 96047 Bamberg, Germany
steffen.illig@uni-bamberg.de

Abstract. An ever-increasing amount of devices connected over the Internet pave the road towards the realization of the 'Internet of Things' (IoT) idea. With IoT, endangered infrastructures can easily be enriched with low-cost, energy-efficient monitoring solutions, thus alerting is possible before severe damage occurs. We developed a library wide humidity and temperature monitoring framework MonTreAL, which runs on commodity single board computers. In addition, our primary objectives are to enable flexible data collection among a computing cluster by migrating virtualization approaches of data centers to IoT infrastructures.

We evaluate our prototype of the system MonTreAL at the University Library of Bamberg by collecting temperature and humidity data.

Keywords: IoT · Single board computer · Container · Virtualization · Monitoring · ARM · Sensor

1 Introduction

The environment in archives and libraries should be maintained within specified tolerances in order to guarantee cultural heritage, e.g. books, magazines, electronic media to be conserved and prevented from serious damage [1]. Our university library maintains several depots to store stocks and equipment, which are partly hosted in buildings under monumental protection. Occurring problems range from simple things, such as broken lights, to more problematic ones, like water damages which could lead to mold formation. In that case a regular supervision of a depot is absolutely essential. To overcome severe issues and to avoid health risks while doing manual data collection, the endangered depot was equipped with temperature and humidity sensors attached to a single board computer (SBC). The relation of both, humidity and temperature, is absolutely necessary for an appropriate climate control in depots [1].

These requirements lead to the idea of creating a distributed sensor environment that covers more areas within a single depot and can be distributed over multiple depots. The data captured by those sensors should then be aggregated

J. Kamps et al. (Eds.): TPDL 2017, LNCS 10450, pp. 599–602, 2017.
DOI: 10.1007/978-3-319-67008-9_52

to a single point, where it can be processed and used, e.g., for creating graphs and analyzing trends. Also the captured data should be made accessible via a web interface once the sensor network is fully operational.

2 IoT Monitoring Solutions in Other Areas of Application

Already existing monitoring solutions are provided in several other areas. For instance, Lewis *et al.* [2] propose an environmental monitoring in a quality-controlled calibration laboratory. In the e-health sector, Jassas *et al.* [3] implemented a prototype with e-health sensors attached to the RPi. Medical sensors measure patients' physical parameters and the RPi collects and transfers them to the cloud environment, as real-time data. However, the former prototypes miss an architectural concept and are not considering the privacy of sensor data. Instead, they send, process, and store the data on remote cloud servers. Only Hentschel *et al.* [4] introduce a concept with local supernodes, which are sensor enhanced RPis. In their model the nodes behave autonomously and carry out simple tasks like processing data or communication with other devices. Unfortunately, each supernode needs to be managed and thus, attaching new devices increases maintenance complexity.

In contrast to those approaches, we focus on transferring data center technology to energy-efficient devices, to enable an easily updatable, scalable, and manageable framework. Moreover, obtained sensor data are processed locally and are saved on the RPis without the need to send them to cloud services.

3 Realisation of MonTreAL

We built MonTreAL (Monitoring Treasures of all libraries) to measure temperature and humidity in our library. Due to our initial goal to perform the measurements with SBCs, MonTreAL takes advantage of Docker[1] and Docker Swarm as underlying technologies.

The system consists of two main components as depicted in Fig. 1. Devices that act as *Workers* are connected to sensors and are distributed within a building. Their tasks is a regular interaction with their environment to gather and process information. Therefore, a *sensor* container running an application to interact with the sensors is responsible to correctly process and send the gathered data to a messaging *queue* container. *Sensor* containers are able to address several different sensor types, which are connected via *GPIO* or *USB* interfaces to a SBC and will port the diversity of manufacturers' dependent data formats into a more uniform format for further processing. Every *sensor* container is configured to add a unique ID to every package it sends to the queue to distinguish and match the data with their origin later. The sensors we are using are simple temperature and humidity sensors like the *DHT-22*[2], which is directly

[1] https://www.docker.com/.
[2] https://www.adafruit.com/product/385.

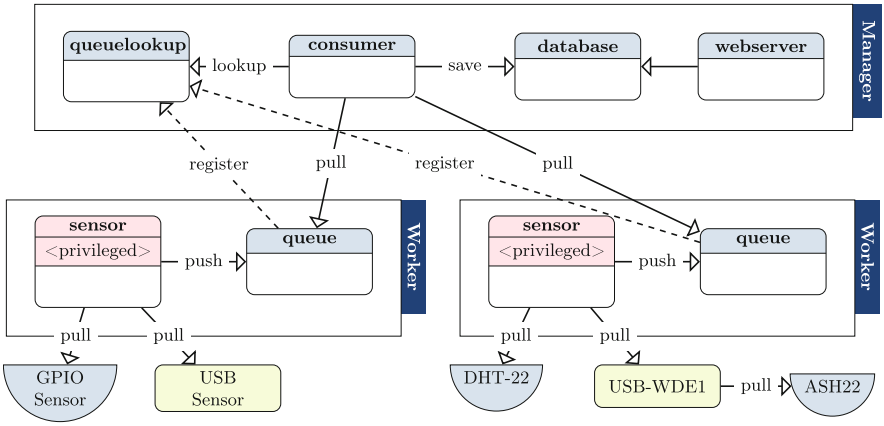

Fig. 1. Architectural overview of MonTreAL

connected to the GPIO interface of the RPi. Or, a more complex sensor system, which consists of a receiver (*USB-WDE1*[3]) that is connected via USB and several autonomous sensor devices (*ASH22*[4]), which regularly send their data to the receiver by radio.

The second main component of MonTreAL is one (or possibly more) device(s) representing the *Manager*. Its task is to accumulate and store the gathered data and provide interfaces to allow user interaction. To fetch the gathered sensor data from the queues, a *consumer* container running a dedicated consumer implementation periodically requests the addresses of all running queues from a *queuelookup* container and fetches all available data. The consumer then stores the data in a *database* container for long-term storage. MonTreAL also provides the user with the possibility to view the gathered data in a suitable way. It runs a simple *webserver* container, which allows the user to view gathered data of every single sensor in a graph and to filter it by date. Furthermore, the front end indicates, which sensors are currently online or offline, or when problems with devices occur, e.g., inoperable devices, broken sensors, empty batteries, etc.

To summarize, MonTreAL allows to set up a distributed system of IoT devices equipped with specific sensors and to gather, accumulate and store the collected data without the need of a powerful backend. It also features a simple front end to view the data. MonTreAL relies on the existence of a network infrastructure (LAN, WLAN) and operates with only a few needed configuration options. We constantly evaluate our prototype at the University Library of Bamberg by collecting temperature and humidity data with the capacity of three RPis.

[3] https://www.elv.de/output/controller.aspx?cid=74&detail=10&detail2=44549.
[4] https://www.elv.de/elv-funk-aussensensor-ash-2200-fuer-z-b-usb-wde-1-ipwe-1.html.

4 Conclusion and Future Work

Our IoT prototype MonTreAL implements a distributed temperature and humidity monitoring framework, which is delivered by Docker images for several Docker supported SBC architectures like ARM, AARCH64 and x86_64. It is made publicly available at the Github[5] repository *unibaktr/MonTreAL* containing the source code. By defining those images, virtualization is achieved at IoT level and by these means MonTreAL offers an easily manageable solution. Furthermore, deploying the system to a container enabled cluster achieves reliability and resilience through the underlying Docker Swarm paradigm. Even, if services are unavailable or are scaling up or down, collected data remain in the distributed message queue and is processed as soon as the consumer recognizes the queue again. Data collected from message queues are processed and stored in a database for long term evaluations. However, if sensors, services, the underlying network, or in the worst case SBCs fail, the consumer notifies the user by alerts.

By now, the RPis, more precisely the plugged in SD cards, provide a bottleneck for MonTreAL. Their limited lifetime is evoked by a relatively small number of write cycles and stands in contradiction to a persistent storage solution. This might be a less relevant problem for a stateless container that only produces runtime data. However, for the database we plan to evaluate the possibility to persist data on a reliable storage solution, which is attached to the database container. Moreover, MonTreAL currently supports only one depot with several sensors attached to multiple RPis. In the future, more containers should enhance the framework to remotely manage multiple depots within one web service.

Acknowledgement. The authors would like to thank the Hypriot team (https://blog.hypriot.com/crew/) for their great effort to port Docker to ARM platforms and for providing Hypriot OS.

References

1. Glauert, M.: Klimaregulierung in Bibliotheksmagazinen. In: Hauke, P., Werner, K.U. (eds.) Bibliotheken bauen und ausstatten. Institut fü Bibliotheks- und Informationswissenschaft (2009). http://edoc.hu-berlin.de/docviews/abstract.php?id=30204
2. Lewis, A., Campbell, M., Stavroulakis, P.: Performance evaluation of a cheap, open source, digital environmental monitor based on the Raspberry pi. Measurement 87, 228–235 (2016). http://www.sciencedirect.com/science/article/pii/S0263224116001871
3. Jassas, M.S., Qasem, A.A., Mahmoud, Q.H.: A smart system connecting e-Health sensors and the cloud. In: 2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE), pp. 712–716, May 2015
4. Hentschel, K., Jacob, D., Singer, J., Chalmers, M.: Supersensors: Raspberry pi devices for smart campus infrastructure. In: 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud), pp. 58–62, August 2016

[5] https://github.com.