VRIJE
UNIVERSITEIT
BRUSSEL

Graduation thesis submitted in partial fulfilment of the requirements for the degree of bachelor in de Wetenschappen: Computerwetenschappen

PYTHON ON THE EDGE

Gérard Lichtert

May 15, 2024

Promotors: Prof. Dr. Joeri de Koster and Prof. Dr. Wolfgang de Meuter.
Advisor: Mathijs Saey

Sciences and bioengineering sciences

Contents

1 Introduction

In a world of electronics and machines where power consumption is always increasing, we must do our best to optimize power consumption. This because electricity is expensive and while there is research being done to make energy generation more efficient, we can still do our part in optimizing the energy consumption to put less strain on the power generation. In the world of distributed computing, according to Pesce, 2021 cloud computing takes 1% of worldwide energy consumption. While this may not seem like much, it is still a significant amount of energy, and we want to do our part to decrease this percentage. While the end goal is to reduce the energy consumption for distributed programs, or more specifically IoT systems, where the generated data gets aggregated on the server in the cloud. We will be looking at reducing the energy consumption of the network traffic caused by transmitting data from the IoT devices to the cloud.

In IoT systems, the IoT devices, or sensors, are responsible for generating data. This data is then sent to the server for processing. Naturally this means that there is extra network traffic caused by the IoT devices. Grossetete, 2020 predicted that "IoT devices will account for 50% of all networked devices by 2023". This also means that there is an increase in energy consumption due to the necessary infrastructure required for the increase in data volume, as stated in J. and Klarin, 2021. What if we could reduce the network traffic by applying the edge-computing principle to the IoT systems? Meaning that prior to the IoT device sending the data to the cloud, it processes it locally first and then sends the preprocessed data to the cloud. This would mean that not all the data has to be sent but rather a subset of it. Logically this should reduce the total network traffic and the required energy. But what if this is not the optimal configuration? What if it is more beneficial to have some parts processed on the IoT devices and some part processed on the server? Or what if it is more beneficial to have some sensors process the data and then send it to the cloud and other sensors to send the data directly to the cloud? To try these different configurations we require a tool that allows us to easily deploy the distributed system and easily change the configuration of the deployment as well as configure where what gets processed. In this thesis we will be focusing on this deployment tool. We will start with the following goals:

1. We want to be able to easily deploy the distributed system
2. We want to be able easily allow us to change the configuration of the deployment
3. We want to be able easily allow us to declare which data gets processed where

However, prior to achieving these goals, we need to do some preparatory work. We need a suitable distributed computing paradigm, a programming language of implementation and possibly a framework in which we can create the tool. To start off with, we will look at the distributed computing paradigm.

2 Preparation

2.1 Choosing a distributed computing paradigm

To start with distributed computing, or our IoT systems we have to look for a suitable distributed computing paradigm that allows our system to be deployed from the cloud without much manual configuration, yet is able to do what we require it to do. We have several options, such as:

1. Message Passing Interface (MPI) as described in "MPI: A message passing interface", 1993
2. Remote Procedure Call (RPC) as described in Tay and Ananda, 1990
3. Shared Memory Model as described in Herlihy et al., 2013
4. The Actor Model as described in Hewitt, 2015
5. Publish/Subscribe (Pub/Sub) as described in Lin et al., n.d.

While each paradigm has their strengths and drawbacks, we will be using the Actor Model. Primarily because its modularity. Each actor encapsulates its state and behavior, which makes it a prime candidate to encapsulate the behavior of our IoT devices. Furthermore, using the Actor Model we can split our data processing pipeline into segments, each segment being captured by a different Actor. This allows us to easily move part of the data processing pipeline to the IoT devices or back to the server. This contributes to one of our previously stated goals: We want to be able to easily allow us to declare which data gets processed where. Other than the modularity, we also choose it for its maintainability. Since each part of the system is encapsulated in an Actor, we can easily change the behavior of parts of the system, instead of having to change our entire system. Another reason is scalability. We are able to distribute actors across different systems and machines. Which makes it perfect for IoT systems. Fault-tolerance makes it easier to contain errors and within individual actors and not propagate this to the entire system. The asynchronous nature of the Actor Model allows us to sparely use resources as needed, which will presumably reduce the required energy of the system. Lastly, through the message-based communication between Actors, we can easily decouple components, which in turn makes it easier to debug and test our systems. This is beneficial because distributed systems can quickly become very complex.

2.2 Choosing a programming language

The next step is to choose a programming language. Before we start looking for a suitable programming language, we need to look at what requirements the language has to fulfill. First and most importantly we need a programming language that can run on IoT devices. Otherwise, we would not be able to deploy our system on the IoT devices. Secondly, we need a programming language that supports the Actor Model, since we have chosen this as our distributed computing paradigm.

References

Grossetete, P. (2020). *Iot and the network: What is the future?* https://blogs.cisco.com/networking/iot-and-the-network-what-is-the-future

Herlihy, M., Rajsbaum, S., & Raynal, M. (2013). Power and limits of distributed computing shared memory models [Structural Information and Communication Complexity]. *Theoretical Computer Science*, *509*, 3–24. https://doi.org/https://doi.org/10.1016/j.tcs.2013.03.002

Hewitt, C. (2015). Actor model of computation: Scalable robust information systems. https://arxiv.org/abs/1008.1459

J., L., & Klarin, Z. (2021). How trend of increasing data volume affects the energy efficiency of 5g networks. *Sensors (Basel, Switzerland)*. https://doi.org/https://doi.org/10.3390/s22010255

Lin, W.-T., Wolski, R., & Krintz, C. (n.d.). *A programmable and reliable publish/subscribe system for multi-tier iot*. https://sites.cs.ucsb.edu/~ckrintz/papers/canal21.pdf

Mpi: A message passing interface. (1993). *Supercomputing '93:Proceedings of the 1993 ACM/IEEE Conference on Supercomputing*, 878–883. https://doi.org/10.1145/169627.169855

Pesce, M. (2021). *Cloud computing's coming energy crisis*. https://spectrum.ieee.org/cloud-computings-coming-energy-crisis

Tay, B. H., & Ananda, A. L. (1990). A survey of remote procedure calls. *SIGOPS Oper. Syst. Rev.*, *24*(3), 68–79. https://doi.org/10.1145/382244.382832