# IoT-Aquarium

Anna Zlámalová Aneta Volfová Petr Tománek
Kryštof Sýkora Pavel Krulec

May 2019

## 1 Introduction

The aim of this project is to create a Palladio simulation of two different software architectures that handle the same problem.

The architectures chosen are one of a monolithic approach and one that is distributed and service oriented. Some parts of the architectures will remain the same such as the resource environment.

This paper will document the simulation project and its results.

## 2 IoT-Aquarium

"You like fish? We like fish. And we like IoT Systems. Hence IoT Aquarium project¡'

The system under analysis is designed to maintain and supervise functions of various IoT devices used in domestic fish-fish life supervision and habitation.

The simulation design is generally separated in two different ways. Firstly, the system has a common part witch is an abstraction of the IoT Aquarium, representing the various devices used in Aquarium maintenance. Secondly, there is the part, of a controller software, that is implemented as a monolithic and distributed system.

These implementations of the controller system are then compared and evaluated based on their own merits.

## 3 Repositories

The simulation is grounded in repository abstractions, that showcase the system basic functions and dependencies. Figure 1 illustrates the repository housing the IoT Aquarium system, that is common to both distributed and monolithic systems. This is implemented as a Palladio composite component.

First itteration of the controller implementation is the monolithic one. Here the controller is represented by one component containing and maintaining all the functions required by the system. figure 2

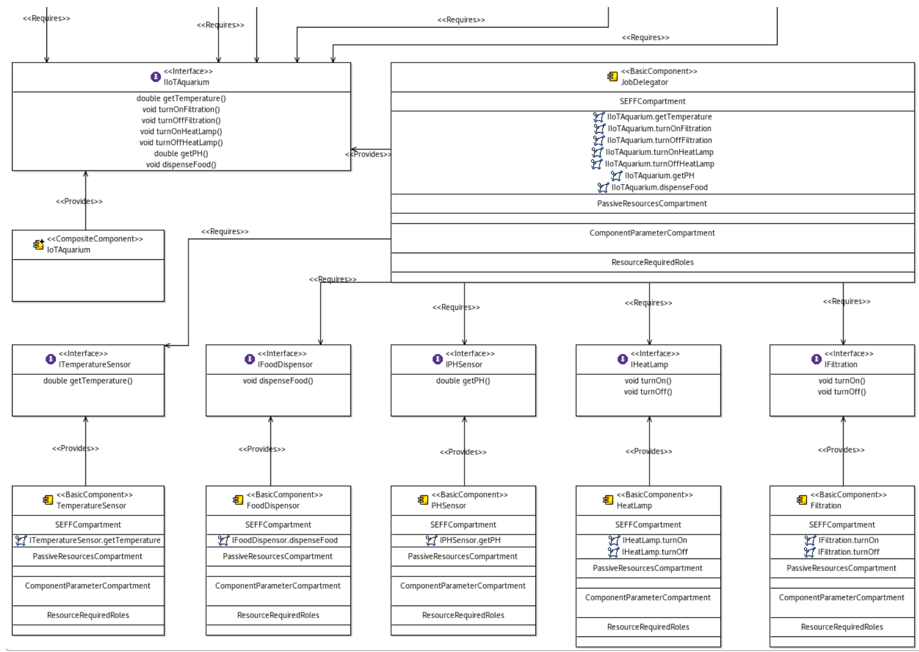


Figure 1: Repository diagram of IoT-Aquarium.

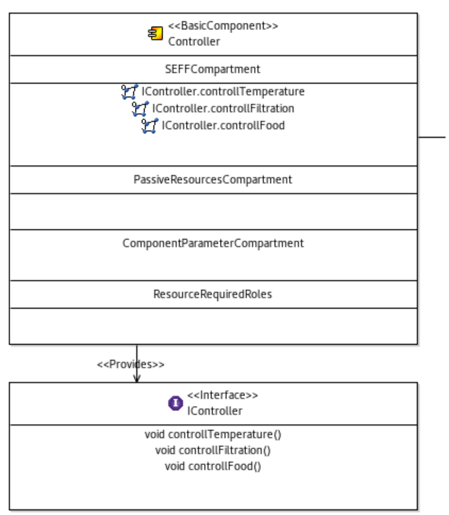


Figure 2: Repository diagram of monolithic controller.

Secondly in figure 3, the distributed controller is modeled. The responsibilities of the monolithic application are distributed into services. The system is implemented as a subsystem component.

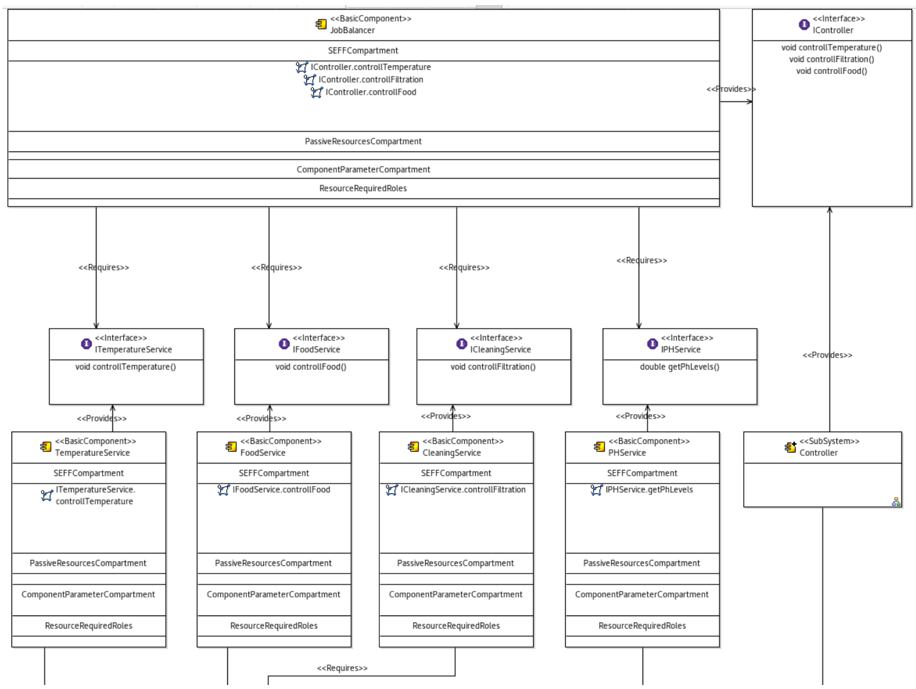


Figure 3: Repository diagram of distributed controller.

# 4 System

In this simulation, two types of composite systems are used. One (Figure 4) for the IoT Aquarium composite component and the second (Figure 5) for the distributed controller subsystem.

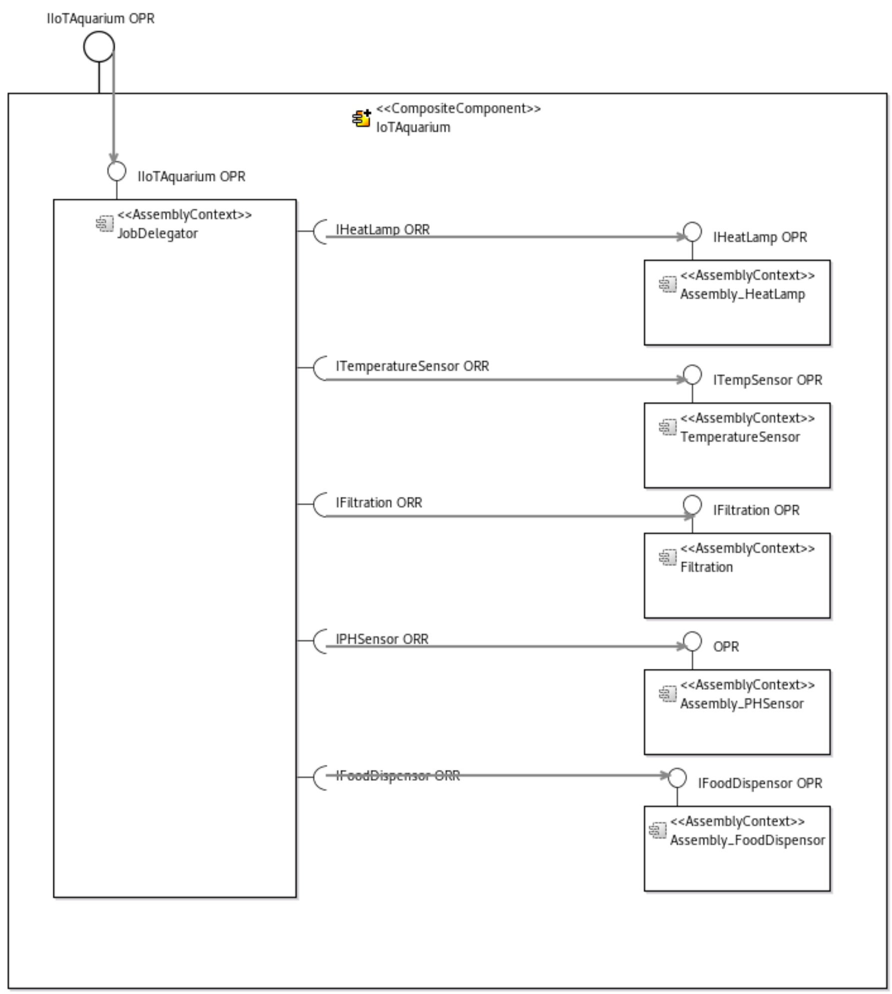


Figure 4: IoT-Aquarium system diagram.

Weather the subsystem is used or not, the overall system diagram remains the same for both monolithic and distributed versions as shown in figure 6.

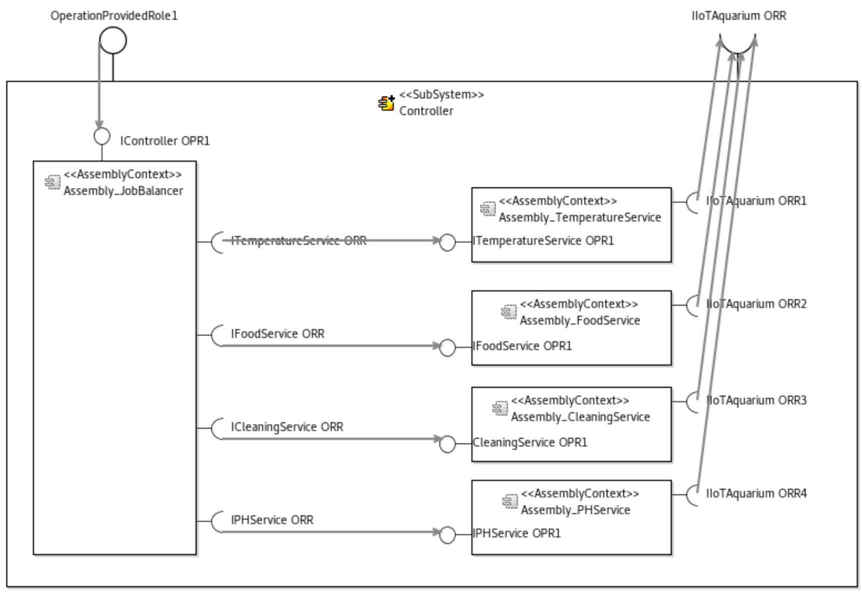


Figure 5: IoT-Aquarium system diagram.

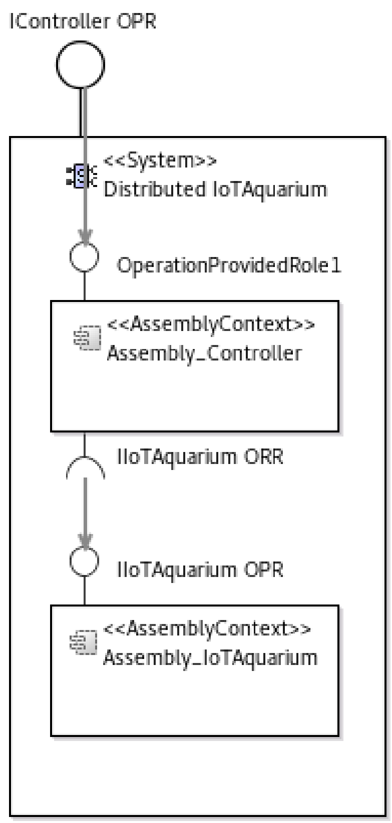


Figure 6: System diagram.

# 5 Environment

To maintain a an experiment result reflecting the architectural changes and not the HW used, the resource enviroment is shared by both monolithic and distributed version of the system (see figure 7).

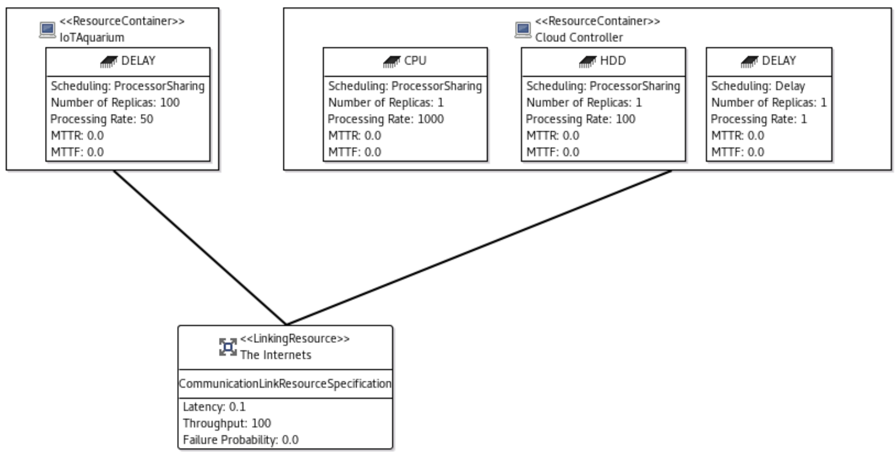


Figure 7: Resource environment diagram.

# 6 SEFF

To showcase the internal logic of the system, three SEFF diagrams are included expanding on the functions behind filtration, food and temperature controll (Figures 8, 9, 10). The implementations are simmilar in both architectures, especialy when it comes to ballancing a similar simulated load for all basic functions.

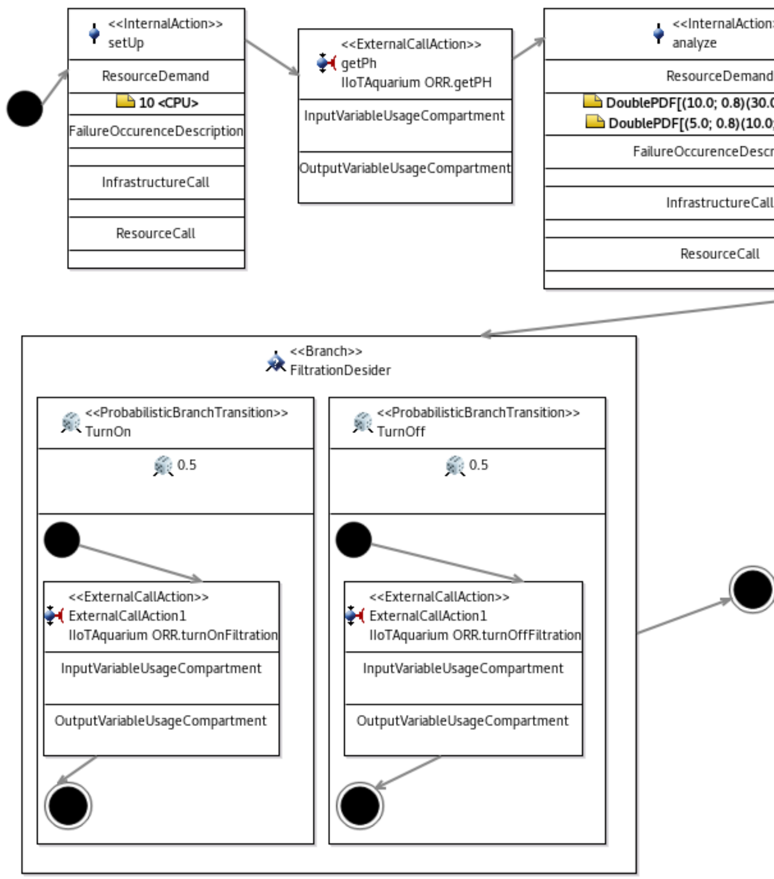


Figure 8: SEFF digram of filtration control.

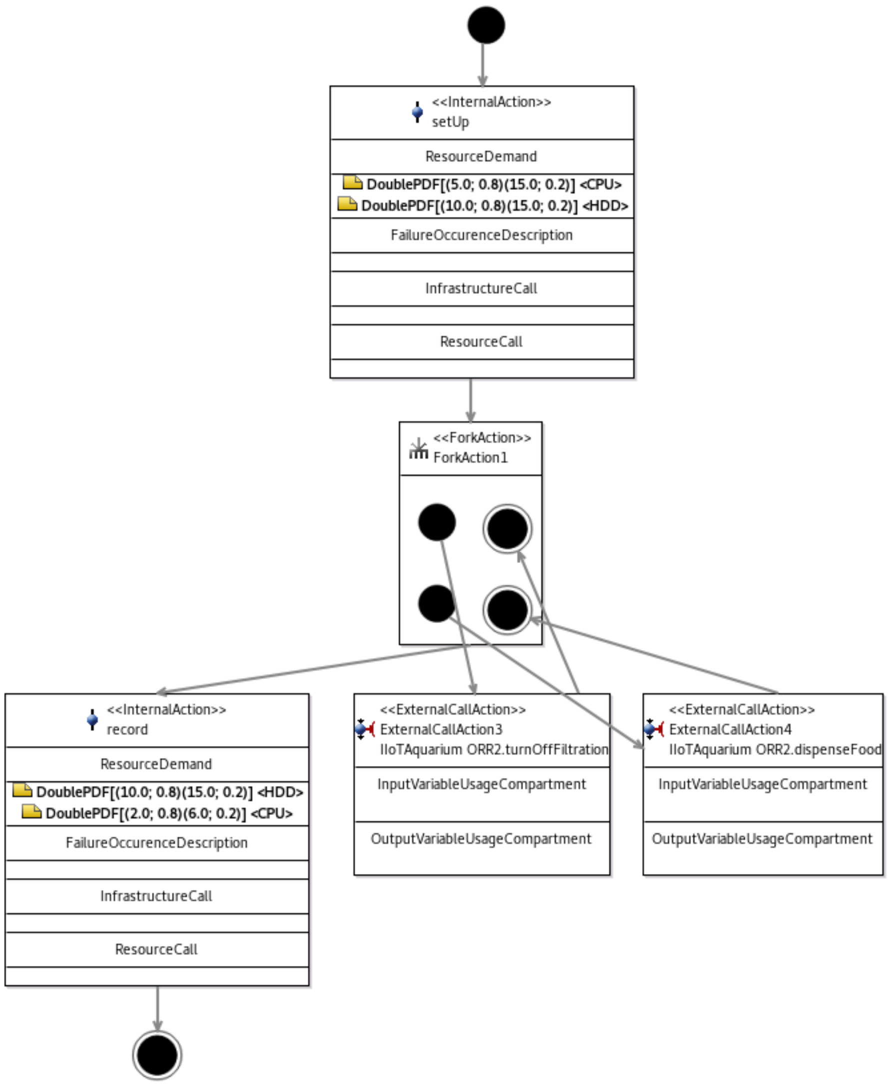


Figure 9: SEFF digram of food control.

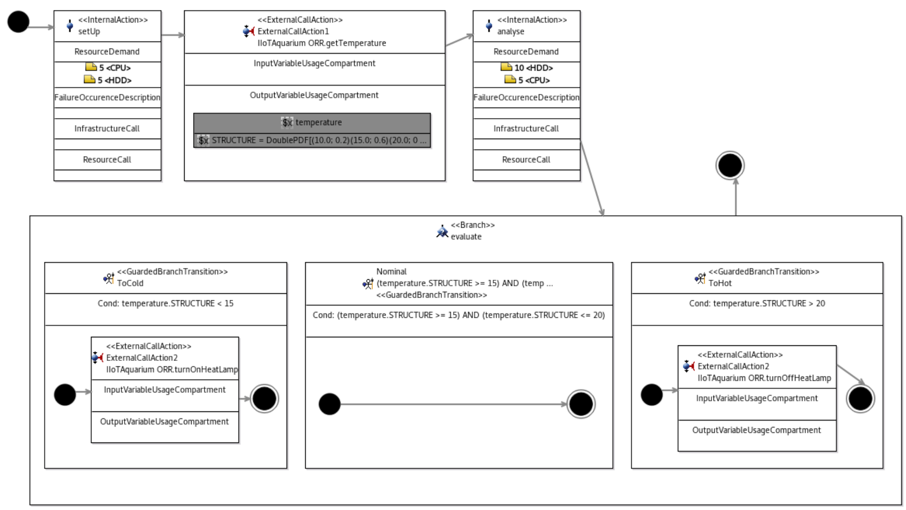


Figure 10: SEFF digram of temperature control.

# 7 Allocation

The system components are allocated to the respective HW abstractions from the resource environment. The only difference is, that Palladio simulations require for the subsystem to be serves as only a reference and not be actually deployed to the resource container as seen in figure 11 for the distributed architecture. This is not the case for the monolithic architecture (figure 12).

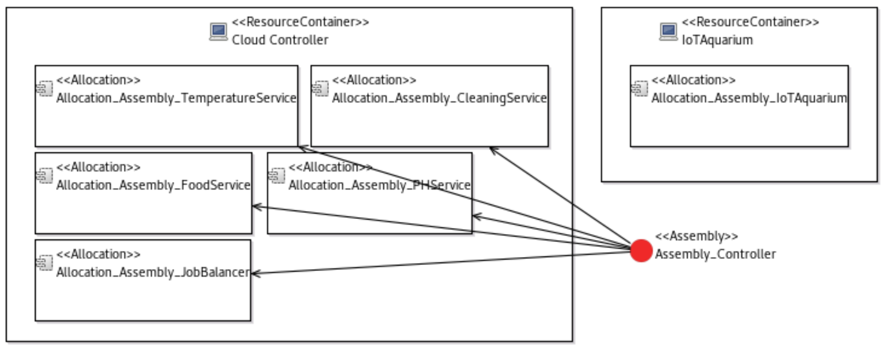


Figure 11: Allocation diagram for distributed system.

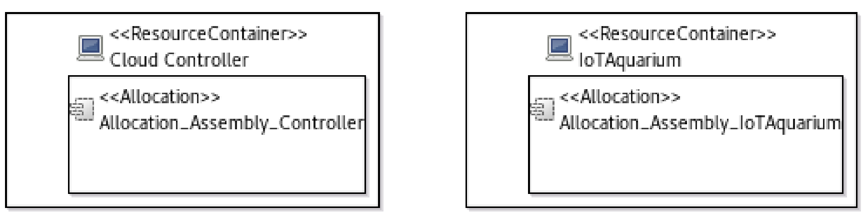


Figure 12: Allocation diagram for monolithic system.

# 8 Usage Scenarios

Scenarios in figure 13 are designed to run concurrently to simulated the three basic system responsibilities. In this case, the actor of the scenarios would be the system.

Wait time between different scenarios reflects the frequency of individual methods.

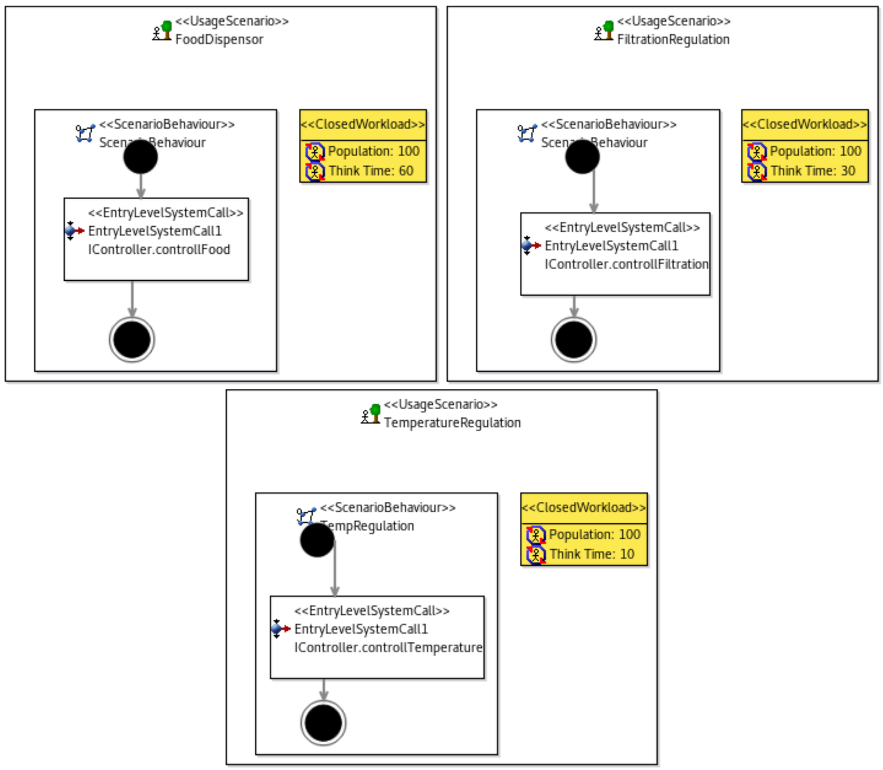


Figure 13: Usage scenarios

# 9 Experiment Results

Experiment results are shown in figures 14, 15, 16, 17, 18.

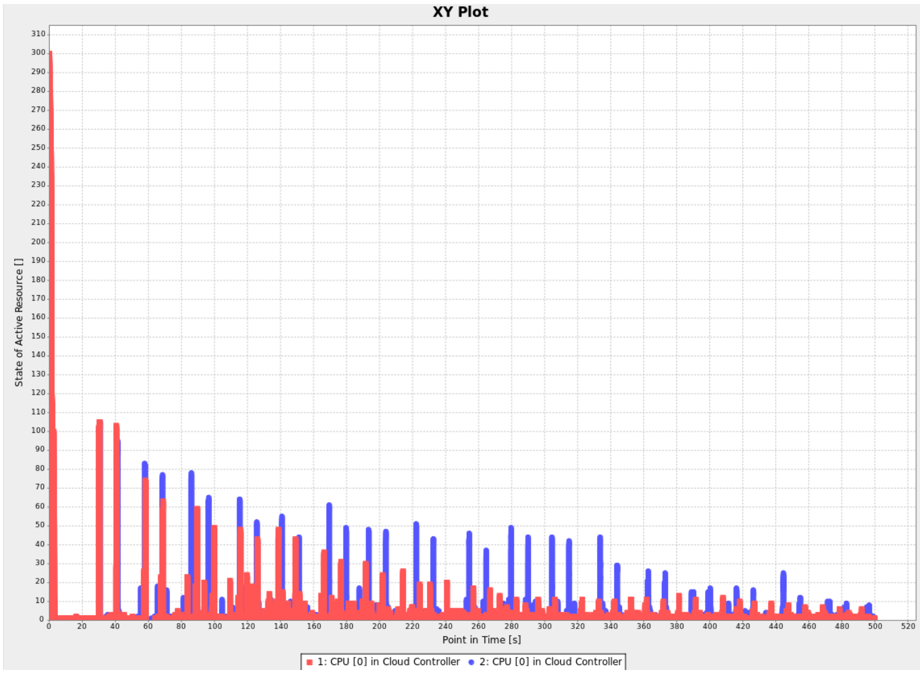


Figure 14: CPU active resource use at Controller (1. monolithic, 2. distributed).

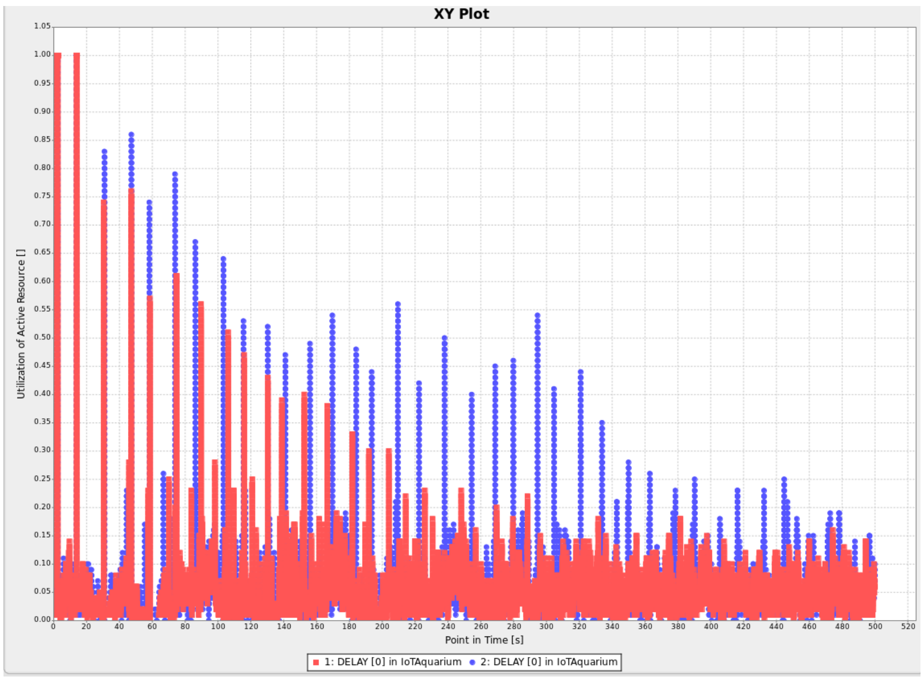


Figure 15: DELAY active resource use at IoT-Aquarium (1. monolithic, 2. distributed).

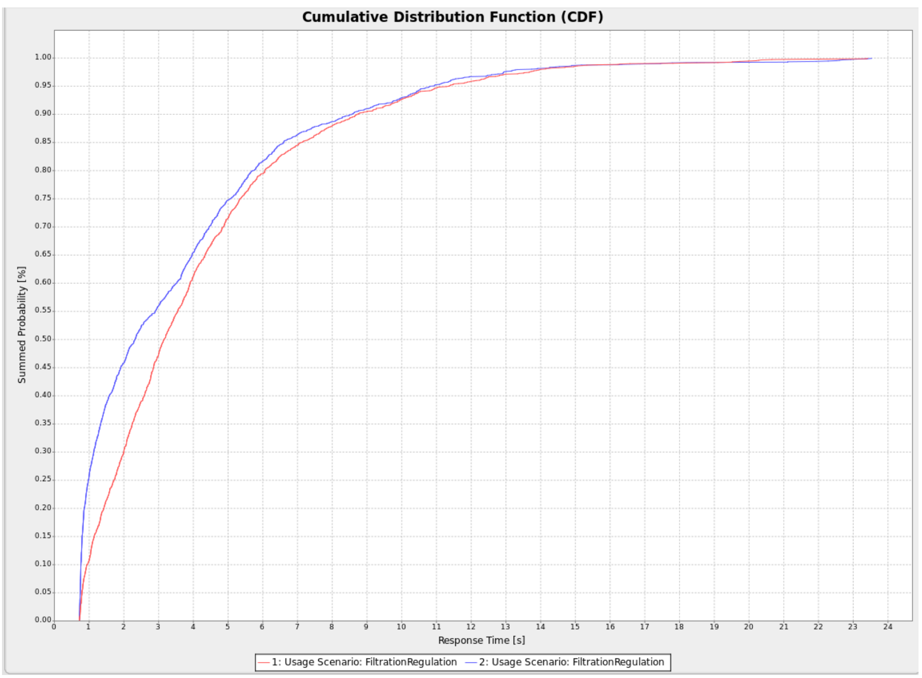


Figure 16: Response time for filtration control use case (1. monolithic, 2. distributed).

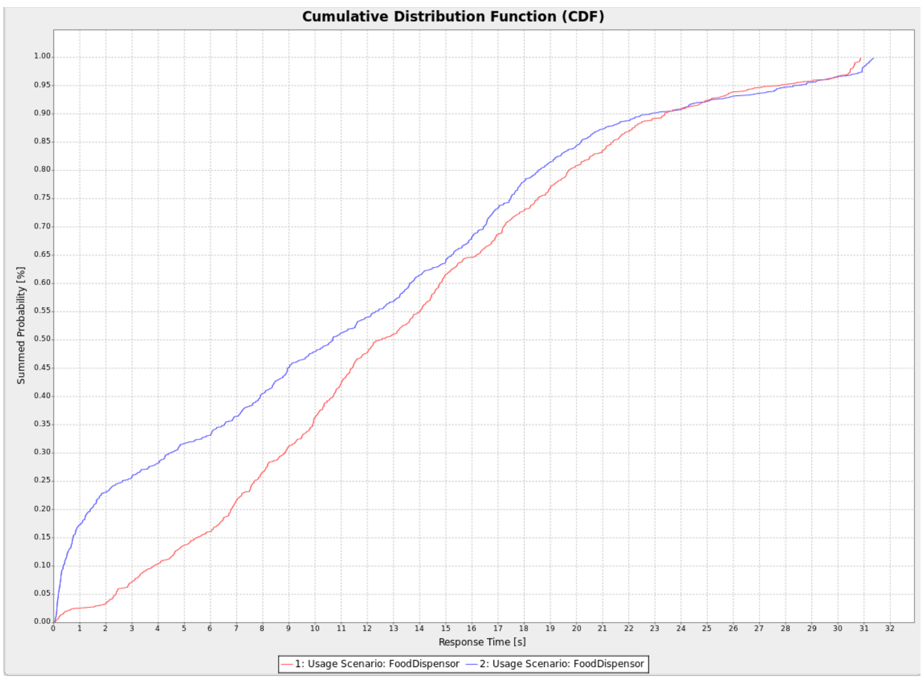


Figure 17: Response time for food control use case (1. monolithic, 2. distributed).

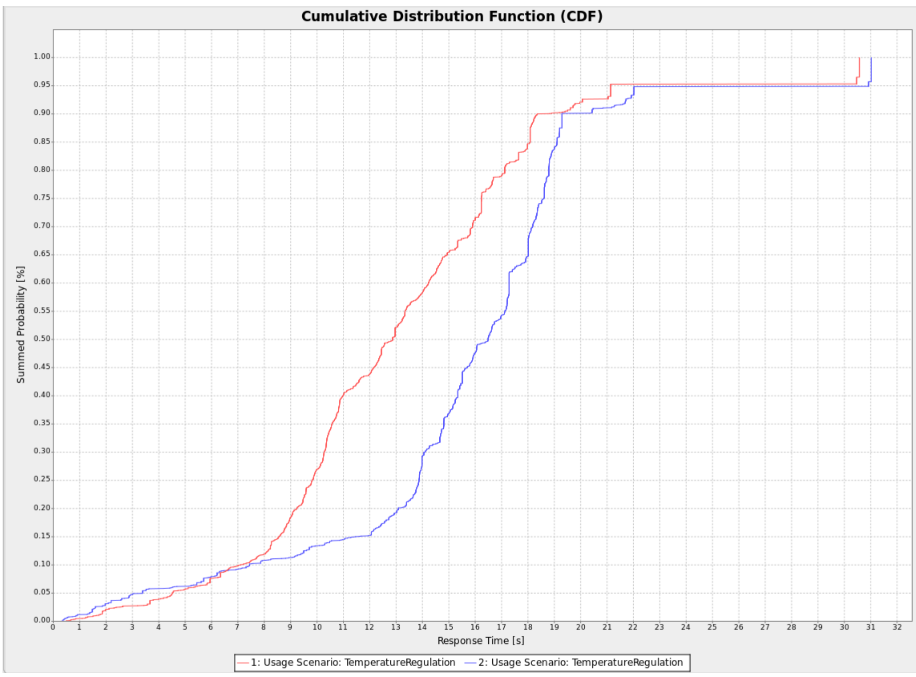


Figure 18: Response time for temperature control use case (1. monolithic, 2. distributed).

# 10 Conclusion

Is all very good.