Some Placement Techniques of Test Components Inspired by Fog Computing Approaches
Moez Krichen, Wilfried Yves Hamilton Adoni, Tarik Nahhal

▶ To cite this version:
Moez Krichen, Wilfried Yves Hamilton Adoni, Tarik Nahhal. Some Placement Techniques of Test Components Inspired by Fog Computing Approaches. 2019. hal-02299922

HAL Id: hal-02299922
https://hal.science/hal-02299922
Preprint submitted on 28 Sep 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Some Placement Techniques of Test Components
Inspired by Fog Computing Approaches

Moez Krichen¹, Wilfried Yves Hamilton Adoni², and Tarik Nahhal²

¹ FCSIT, Albaha University, Albaha, Saudi Arabia
ReDCAD Laboratory, University of Sfax, Tunisia
moez.krichen@redcad.org
² Hassan II University of Casablanca, Morocco
adonwilfri@gmail.com, t.nahhal@fsac.ac.ma

Abstract. In this work we are interested in placing test components for Internet of Things (IoT) and Smart Cities. Our work is inspired by similar works aiming the placement of application components in Fog computational nodes. First we give an overview about the decision variables to consider. Then, we define several types of constraints that may be included in the placement problem. Moreover, We list a set of possible Objectives Functions to maximize or minimize. Finally, we propose some algorithms and techniques to solve the considered Test Component Placement Problem (TCPP) taken from the literature.

Keywords: Internet of Things, Smart Cities, Test Components, Placement, Optimisation, Constraints, Objective Functions, Algorithms, Fog.

1 Introduction

The city of the future is more sustainable, cleaner, safer and smarter than ever. Full networks and smart cities offer versatile solutions to current challenges such as environmental protection, growing traffic volumes and increasing urbanization. In particular, the Internet of Things (IoT) plays a key role here, as it links devices, programs and users together. In Smart Cities cars communicate with houses, houses with digital devices, and these in turn communicate with the city’s inhabitants. Most errors and problems of IoT applications only become apparent during actual use by the customer. This is why a classic integration test is usually not enough. In addition, it is hardly possible in the laboratory to reproduce the variety of devices and operating systems combinations. Moving testings to the real world [22, 20, 16, 18] also helps counteract operational blindness. Because some usability problems are not noticed by the developers, because they deal with the product every day. Often, real users interact with the application differently than thought, resulting in unexpected bugs. That is why it is important to test under real conditions with potential end users.

Test Components can be placed either locally on the same computational node under test or remotely on the cloud. An other half-way alternative consists
in using Fog computing techniques which allow to distribute application components -test components in our case- on the different available computational nodes in the Fog. In some of previous works [13, 11, 12] we proposed a formal work for testing security aspects of IoT and Smart Cities. In [25, 21, 15], we were interested in placement problems of test components for distributed and dynamic distributed systems in general. In the present work we concentrate on test component placement problem (TCPP) for IoT and Smart Cities. We are mainly inspired by the work of Brogi et al. [4]. The remaining part of the paper is organised as follows. Section 2 provides some preliminaries about IoT, Testing, Fog computing and placement problem. In section 3, we deal with decision variables and different types of constraints used in the placement problem. Section 4 lists a set of possible Objective Functions. Section 5 proposes a set of algorithms and techniques to solve TCPP. Finally Section 6 concludes the paper.

2 Preliminaries

2.1 Fog Computing

After Cloud Computing [19, 17] now Fog Computing is to come. The goal of Fog Computing is to bring the analysis, processing and storage functions from the cloud back to the edge of the network. The more networked our world becomes, the more data is processed. The communication path into a cloud computing center and then back to the terminal leads to high latencies - for example for autonomously driving cars whose information processing must be guaranteed in real time.

Some advantages of Fog Computing are as follows:

- Faster data processing due to reduced network traffic.
- Networked devices in the IoT also work when the Internet goes down or the cloud connection causes delays.
- Sensitive company and customer data would not have to be transferred to the cloud and can remain on the spot.

2.2 Test Component Placement Problem (TCPP)

Let $TC$ be the set of test components we aim to distribute. The set of requirements to satisfy by the placement strategy is called $Req$ and the distributed architecture inside which the test components will be placed is called $Infr$. Possible Solutions to the Test Component Placement Problem (TCPP) are mappings from the set of test components $TC$ to the set of computational nodes $Infr$, satisfying the requirements fixed by $Req$ and maximizing/minimizing a set of objective functions $Obj$ used to measure their quality. The adopted Solutions are many-to-many. That is a test component can be assigned to one or many nodes and a computational node can execute one or many test components.
2.3 Decision Variables

The values of these variables need to be calculated during the optimisation procedure. They also define the requirements $Req$ to satisfy. Values of the variables which do not satisfy the constraints are rejected and can not be considered as acceptable solutions. For TCPP, they may be binary decision variables which indicate if a test component of $TC$ is allocated to a computational node of $Inf$.r.

3 Different Constraints

3.1 Energy Constraints

The authors of [2] characterised the fog nodes with their energy capacities. Devices with batteries are considered in order to guarantee lifetime constraints. The authors of [30] introduced the concept of energy cells to calculate the energy consumed by the computational devices. The optimisation procedure considers the number of available energy cells. The goal is to minimise energy consumption in constrained nodes.

3.2 Network Constraints

Network latency, bandwidth, link reliability and topology are examples of network constraints. For instance on the one hand, the authors of [26] considered only latency constraint. On the other hand in [8], the authors considered both latency and bandwidth. Bandwidth was also taken into account in [32, 1] along with hardware requirements.

3.3 Node Constraints

This type of constraints correspond to the resource capacity of the computational nodes. Resources are usually modelled as a vectors of elements like CPU, RAM, storage, etc. For instance the authors of [8] considered only CPU and the authors of [38] considered only storage. In [1], the authors considered CPU and storage. In [6], RAM, CPU and storage were all taken into account.

3.4 Application Constraints

Three types of application constraints have been considered for the classification of the papers in the survey: constraints related to the dependencies between the modules or services of the applications; the workload generated over the applications; and if the users are able to define any kind of preference in the deployment of the services.
4 Objective Functions

4.1 Network Delay and Execution Time

For instance this objective function is considered in [39]. Similarly the authors of [37] minimised the response time to increase the number of requests which are served before a fixed deadline. Moreover, the authors of [31] diminished the computation time of the service by calculating the consumed time slots.

4.2 Energy

The optimisation of energy was considered from different aspects. For instance, the authors of [2] proposed a linear function of the cost of energy and they optimised it. Similarly in [9], the authors were interested in reducing the cost of communication energy. Their approach was mainly based on placing interrelated applications and services on the same device. This choice allows to diminish communications between computational nodes.

4.3 Cost

In Fog computing the consideration of cost aspects is at an initial phase. In [1] the authors aimed to minimise the total cost of applications deployment by assigning them to the computational nodes with have optimal costs. Similarly the authors of [35] considered cost minimisation of each device and link.

4.4 Migrations

This objective function consists in minimizing migrations number and/or the effects of migrations on the system. In [27], the authors optimised the number of migrations by minimising the network use without influencing the network latency. Similarly, the authors of [38] optimised the number of migrations along with resource usage and latency. Moreover, a technique was presented in [5] to reduce the number of migrated applications between computational nodes.

4.5 QoS-assurance

In [26], the objective of the optimisation procedure was to diminish the number of active Fog nodes. The QoS requirement considered in this work consisted in obtaining execution times which are shorter than the deadlines of the application. In addition, the authors of [34] considered QoS aspects realted to the frequency of executed requests and transactions.
5 Algorithms

5.1 Mathematical Programming

Mathematical programming is usually used to find solutions for optimisation problems by exploring the domain of the objective function. Many approaches [38, 6] tackled the placement problem using this mathematical procedure by exploiting ILP (Integer Linear Programming), MILP (Mixed-Integer Linear Programming) or MINLP (Mixed-Integer Non-Linear Programming).

5.2 Dynamic Programming

The authors of [30] modelled the placement problem as a multidimensional knapsack problem with the goal objective of optimising a particular objective function. Similarly in [28], the authors modelled the placement problem as a knapsack instance, by considering the assignment of application services to running computational nodes in a Fog infrastructure.

5.3 Search-based Algorithms

The authors of [8] proposed a search algorithm to compute a placement solution of IoT applications to a tree-structured infrastructure. Further in [7] proposed a distributed search strategy to compute the best placement solution in the Fog, which corresponds to the minimal distance between clients and requested services. In addition the authors of [3] proposed a greedy and exhaustive backtracking algorithm to solve the placement problem.

5.4 Genetic Algorithms

Genetic algorithms use meta-heuristics to deal with search and optimisation problems. In [36], the authors provided a description of the use of genetic algorithms and parallel genetic algorithms to solve to placement problems. Similarly, the authors of [29] also proposed a genetic algorithm approach solution implemented in iFogSim.

5.5 Deep Learning

The authors of [33] used recent learning techniques to solve placement problems. After defining a Markov Decision Process to optimize power consumption, communication delay and migration costs, a deep learning algorithm is provided to support migration of applications hosted in computational nodes. The proposed approach takes into consideration the mobility of the user.
5.6 Game Theory

The authors of [40] modelled the placement problem as a set of games. The first game is defined to find the number of computing blocks that users have to buy. The second game is defined for providers to set prices so to maximise financial profits. A matching game is proposed to map providers to computational nodes. Finally, matching between subscribers and computational nodes is refined.

6 Conclusion

Although the smart city is still considered as a future scenario, it will soon become a reality for future generations. How people move in cities in the future and how they live there will change significantly as a result of increasing networking.

In this work we were interested in the test component placement problem for these modern technologies. We identified existing techniques in the literature used for the placement of application components in Fog domain. In the future we need to adapt the found methods to the case of TCPP for IoT and Smart cities and to implement them. We may also use optimization techniques like in [10] and combine functional and load tests as proposed in [14, 23, 24].

References


38. Yang, L., Cao, J., Liang, G., Han, X.: Cost aware service placement and load dispatching in mobile cloud systems. IEEE Transactions on Computers 65(5) (May 2016) 1440–1452