Supporting Application Deployment and Management in Fog Computing

Stefano Forti

Department of Computer Science, University of Pisa, Italy
stefano.forti@di.unipi.it

Abstract. Deploying and managing multi-component IoT applications in Fog computing scenarios is challenging due to the heterogeneity, scale and dynamicity of Fog infrastructures, as well as to the complexity of modern software systems. When deciding on where/how to (re-)allocate application components over the continuum from the IoT to the Cloud, application administrators need to find the best deployment, satisfying all application (hardware, software, QoS, IoT) requirements over the contextually available resources, also trading-off non-functional desiderata (e.g., financial costs, security). This PhD thesis proposal aims at devising models, algorithms and methodologies to support the adaptive deployment and management of Fog applications.

Keywords: Fog computing · IoT · QoS-aware application deployment · Application Management.

1 Introduction

Context – Connected devices are changing the way we live and work. In the next years, the Internet of Things (IoT) is expected to bring more and more intelligence around us, being embedded in or interacting with the objects that we use every day [22, 21]. Self-driving cars, autonomous domotics systems, energy production plants, agricultural lands, supermarkets, healthcare, embedded AI will more and more exploit devices and Things that are integral part of the Internet and of our existence without us being aware of them.

As a consequence of this trend, enormous amounts of data – the so-called Big Data [42] – are collected by IoT sensors and stored in Cloud data centres [35]. Once there, data are subsequently analysed to determine reactions to events or to extract analytics or statistics. Whilst data-processing speeds have increased rapidly, bandwidth to carry data to and from data centres has not increased equally fast [47]. On one hand, supporting the transfer of data from/to billions of IoT devices is becoming hard to accomplish due to the volume and geodistribution of those devices. On the other hand, the need to reduce latency for time-sensitive applications, to eliminate mandatory connectivity requirements, and to support computation closer to where data is generated 24/7, is evident [9].

* PhD Thesis Supervisor: Prof. Antonio Brogi, University of Pisa, Italy.
Fog Computing – Recent research efforts are investigating how to better exploit capabilities along the continuum from the edge of the Internet to the Cloud data centres, to support new IoT applications and their needs. Computational nodes closer to the edge will act both as filters – reducing the amount of data sent to the Cloud – and as processing capabilities – producing analytics closer to where data is being sensed or used.

Among the existing proposals, Fog computing [10, 26] aims at better supporting the growing processing demand of (time-sensitive and bandwidth hungry) IoT applications by selectively pushing computation closer to where data is produced and by relying on a geographically distributed multitude of heterogeneous devices (e.g., personal devices, gateways, micro-data centres, embedded servers) spanning the continuum from the Cloud to the IoT. A substantial amount of computation, storage and networking is therefore expected to happen closer to where data is produced and to IoT-based cyber-physical systems, contiguously to and interdependently with the Cloud. In general, Fog computing platforms are expected to guarantee that processing always occurs wherever it is best-placed for any given IoT application, thereby accelerating the velocity of decision making, by enabling prompter responses to sensed events [45].

Scope of the Thesis – Modern large-scale applications are not monolithic anymore [51]. Therefore, an application running in a Fog computing infrastructure consists of a set of independently deployable components (or services, or microservices) that work together and must meet some requirements. Deploying and managing such applications in Fog computing scenarios is, therefore, a challenging task. Indeed, it requires to dynamically map each of the (possibly many) application components (i.e., functionalities) to the computational node(s) that will host them at runtime.

Whilst some application functionalities are naturally suited to the Cloud (e.g., service back-ends) and others are naturally suited to edge devices (e.g., industrial control loops), there are applications for which functionality segmentation is not as straightforward (e.g., short to medium term analytics). Future tools for the deployment and management of IoT applications should consider application requirements (i.e., hardware, software, IoT, QoS), infrastructure capabilities (i.e., hardware, software, IoT devices, network conditions, security) and deployers’ desiderata (i.e., business and security policies, cost constraints) to efficiently support adaptive segmentation of functionalities from the Cloud to the IoT.

In this context, we are investigating the design, prototyping and validation of novel models, and predictive algorithms and methodologies which will be useful to (i) process data about the application, the infrastructure and their monitored performance so to informally suggest how to (re-)distribute application components, (ii) identify and validate the best sequence of actions to (re-)distribute components to different Fog or Cloud nodes based on specified policies, and (iii) choose when/how to (re-)deploy, (re-)configure or scale components in response to workload or network variations, churn and failures.
2 State of the Art

The problem of deciding how to deploy multi-component applications has been thoroughly studied in the Cloud scenario. Projects like SeaClouds [16], Aeolus [28] or Cloud-4SOA [24], for instance, proposed model-driven optimised planning solutions to deploy software applications across different (IaaS or PaaS) Clouds. [39] proposed to use OASIS TOSCA [17] to model IoT applications in Cloud+IoT scenarios. Also, solutions to automatically provision and configure software components in Cloud (or multi-Cloud) scenarios are currently used by the DevOps community to automate application deployment or to lead deployment design choices (e.g., Puppet [3] and Chef [2]). However, only few efforts in Cloud computing considered non-functional requirements by-design [44, 25] or uncertainty of execution (as in Fog nodes) and security risks among interactive and interdependent components [61]. With respect to the Cloud paradigm, the Fog introduces new problems, mainly due to its pervasive geo-distribution and heterogeneity, need for QoS-awareness, dynamicity and support to interactions with the IoT, that were not taken into account by previous works [56, 62, 4].

Among the first proposals investigating this new lines, [34] proposed a Fog-to-Cloud search algorithm as a first way to determine an eligible deployment of (multi-component) DAG applications to tree-like Fog infrastructures. Their placement algorithm proceeds Edge-ward, i.e., it attempts the placement of components Fog-to-Cloud by considering hardware capacity only. An open-source simulator – iFogSim – has been released to test the proposed policy against Cloud-only deployments. Building on top of iFogSim, [40] refines the Edge-ward algorithm to guarantee the application service delivery deadlines and to optimize Fog resource exploitation. Limiting their work to linear application graphs and tree-like infrastructure topologies, [60] used iFogSim to implement an algorithm for optimal online placement of application components, with respect to load balancing. An approximate extension handling tree-like application was also proposed. Recently, exploiting iFogSim, [33] proposed a distributed search strategy to find the best service placement in the Fog, which minimizes the distance between the clients and the most requested services, based on request rates and available free resources. Their results showed a substantial improvement on network usage and service latency for the most frequently called services. [36] proposed a (linearithmic) heuristic algorithm that attempts deployments prioritising placement of smaller applications to devices with less free resources. Along the same line, [54] proposed an Edge-ward linearithmic algorithm that assigns application components to the node with the lowest capacity that can satisfy all application requirements.

From an alternative viewpoint, [57] proposed the design of a framework for application deployment in Fog computing, based on Integer Linear Programming (ILP). [5] in addition to proposing a Fog architectural framework, gave a Mixed-Integer Non-Linear Programming (MINLP) formulation of the problem of placing application components so to satisfy end-to-end delay constraints. The problem is then solved by linearisation into a Mixed-Integer Linear Programming (MILP), showing potential improvements in latency, energy consumption
and costs for routing and storage that the Fog might bring. Skarlat et al. designed a hierarchical modelling of Fog infrastructures, consisting of a centralised management system to control Fog nodes organised per colonies ([48, 50, 49]). Particularly, [48] adopted an ILP formulation of the problem of allocating computation to Fog nodes in order to optimise (user-defined) time deadlines on application execution, considering IoT devices needed to properly run the application. A simple linear model for the Cloud costs is also taken into account. Similar solutions were proposed, attempting to optimise various metrics such as access latency, resource usage, energy consumption or data migrations cost ([64, 32, 65, 53, 7, 37]). [41] described instead a fuzzy QoE extension of iFogSim – based on an ILP modelling of users expectation – which achieved improvements in network conditions and service QoS.

Regrettably, none of the discussed ILP/MILP approaches came with the code to run the experiments. Conversely, [58] proposed a software platform to support optimal application placement in the Fog, within the framework of the CoSS-Mic European Project [1]. Envisioning resource, bandwidth and response time constraints, they compare a Cloud-only, a Fog-only or a Cloud-to-Fog deployment policy. Additionally, the authors of [18, 20, 19] released S-ODP, an open-source extension of Apache Storm that performs components placement with the goal of minimising the end-to-end application latency and the availability of deployed applications. Finally, also dynamic programming (e.g., [46, 52], genetic algorithms (e.g., [48, 50]) and deep learning (e.g., [55]) were exploited to tackle the placement of application components with some promising results.

After the first deployment, the management of applications in the Fog is also time-consuming and error-prone to be tuned manually, lacking adequate support. [43] proposed a MAPE-K loop to identify action plans to minimise SLA violations while maximising the use of allocated resources by simulating different strategies to manage deployed applications. [30] highlighted the need to check for inconsistencies that can arise within or between different stages of a deployment plan. [30] proposed a deployment management system model to enable the automated generation of deployment plans for distributed infrastructures after identifying (with static analysis techniques) possible flaws in deployment plan specifications. The use of formal models to verify properties of application deployments to Cloud infrastructure has been advocated by various authors. [38] for instance defined a process calculus to specify deployment, migration and security policies of virtual machines (VMs) across different Clouds, in order to enable the verification of security policies after live VM reconfigurations. [6] proposed a similar approach to preserve data consistency when migrating deployed applications in Fog scenarios. [29] proposed a pseudo-dynamic testing approach, which combines emulation, simulation, and existing real testbeds, whilst leveraging multiple methodologies to test complex and large Fog infrastructures taking into account also scalability and churn conditions. While various proposals exist to automate the management of applications, to verify the correctness of deployments to the Cloud, to the best of our knowledge, none of the existing approaches addresses the validation of application management for the Fog.
3 Thesis Objectives

This section aims at illustrating the objectives of the thesis work, seeking to suitably support automated application deployment (and functionality allocation) in Fog computing. The provision of adequate support to adaptively deploy applications and manage their components in Fog scenarios is among the crucial steps for the success of Fog computing. In this context, we intend to design, prototype and validate novel models, and predictive algorithms and methodologies, which will improve the decision-making process related to the life-cycle management of Fog applications.

In what follows, we detail the research goals we intend to accomplish during this research, from the point of view of modelling (Section 3.1), design of algorithms and methodologies (Section 3.2), and prototyping and validation (Section 3.3).

3.1 Modelling

First, we aim at contributing to the modelling of the Fog scenario with a particular focus on:

1. describing arbitrary multi-component applications topologies considering their processing (e.g., hardware, software and IoT devices), QoS (e.g., latency, bandwidth, security) requirements and component inter-dependencies, along with the possibility for their components to scale both vertically and horizontally, according to workload demand and behaviour models,
2. describing accordingly fog infrastructures in terms of their capabilities (i.e., Cloud data-centres, Fog nodes, Things) and previous performance/utilisation (e.g., QoS of communication links, historical data on nodes utilisation, reliability of nodes and links), considering IoT-Fog, Fog-Fog and Fog-Cloud interactions,
3. accounting for dynamicity and churn of the infrastructure (e.g., variations in the QoS of communication links, mobility of IoT devices and Fog nodes, failures) and in the users’ demand, as well as for application scalability on heterogeneous devices so to be able to plan for scalable, reliable and dependable application deployments,
4. including the possibility of expressing preferences on application deployment that have to be enforced due to particular end-user targets (e.g., QoS-assurance, financial budget, resource usage) or deployment needs (e.g., security, trust, reliability, energy consumption),
5. identifying and devising appropriate metrics and performance indicators (e.g., QoS-assurance, resource consumption, reliability) to characterise eligible application deployments and plans, also considering their behaviour over time, as well as financial costs and energy consumption to keep the application up and running.

Naturally, to support the deployment of applications to Fog infrastructures, we intend to accompany the devised models with novel algorithms and methodologies that exploit them as illustrated in the next section.
3.2 Algorithms and Methodologies

To exploit the models described in the previous section, we intend to devise algorithms and methodologies in order to:

1. efficiently determine eligible context- and QoS-aware deployments of application components to Fog infrastructures, according to different strategies and by adopting proper heuristics to reduce the search space, whilst selecting cost-/energy-aware matchings between application requirements (viz., hardware and software) and available Fog/Cloud offerings,

2. simulate and predict the (expected) behaviour of different eligible deployments under the proposed metrics at varying (i) QoS of available communication links, (ii) available resources in the current state of the infrastructure, (iii) workload and users demand, also considering historical data about the monitored infrastructure and feedback about previously enacted deployments,

3. compare and recommend and/or automatically select best candidate deployments — among the eligible ones — based on predicted metrics, expressed targets and historical data, by plotting results to empower experts to make informed choices, and by exploiting multi-objective optimisation or learning techniques,

4. determine and optimise plans that take into account dependencies between components so to perform application deployment to a given infrastructure, envisioning deployment (vertical and horizontal) scalability on heterogeneous devices and optimal resources exploitation (e.g., hardware, energy), and considering alternative backup deployments to tackle dynamicity issues (e.g., increasing workload, mobility, QoS variations, churn and failures),

5. understand when to trigger and how to (optimally) perform reconfiguration actions (e.g., enactment of an alternative plan), scaling of application components, or components re-allocation to different nodes so to guarantee QoS or SLA constraints will be met by enacted deployments, whilst avoiding (or minimising) the likelihood of service disruption.

3.3 Prototyping and Validation

To provide some validation to our approaches we aim, when possible, at providing formal properties (e.g., correctness, completeness) of the proposed methodologies, along with a systematic evaluation of their computational complexity. Then, we plan to prototype all proposed models and methodologies in open-source tools, so to show feasibility, utility and practicality of the devised solutions.

Finally, with the purpose of testing and demonstrating our prototypes at work, we aim at designing lifelike use cases and testbeds, by implementing meaningful IoT applications and deploying them to experimental Fog infrastructures.
4 First Results

The first results of this work have been already published in some conferences and journals. In this section, we briefly summarise them and the research they triggered in the community.

**QoS-aware Deployment of Fog Applications** – In [11], we proposed a simple, yet general, model of multi-component IoT applications and Fog infrastructures. After proving that the problem of determining eligible deployments is NP-hard, we devised a heuristic backtracking search algorithm to solve it and we run it on a motivating example from smart agriculture (viz., 3 application components, 2 Clouds, 3 Fog nodes). The heuristic attempts the placement of components sorted in ascending order on the number of compatible nodes (i.e., *fail-first*), considering candidate nodes one by one sorted in decreasing order on the available resources (i.e., *fail-last*).

In [12], we combined an exhaustive version of our search algorithm with Monte Carlo simulations so to consider variations in the QoS of communication links (modelled by probability distributions) and to predict how likely a deployed application is to comply with the desired network QoS (viz., latency and bandwidth) and how much Fog resources it will consume. In [13], we further enhanced the proposed methodology by proposing a cost model that extends Cloud cost models to Fog scenarios and integrates them with costs coming from the IoT. It is worth noting that, with respect to the majority of related works, our approach works on arbitrary application and infrastructure graph topologies.

All proposed predictive methodologies have been implemented in an open-source prototype\(^1\), FogTorch\(\Pi\), and are described in detail in [14], which also offers a comparison with one of the first tools for simulating Fog scenarios (iFogSim [34]). FogTorch\(\Pi\) can be used to determine, simulate and compare eligible deployments of applications to given infrastructures in a QoS- (with respect to network variations), context- (with respect to the considered resources), and cost-aware (estimating monthly revenues and outflows) manner, meeting all deployers’ desiderata. Despite exploiting worst-case exponential-time algorithms, the prototype has been shown to scale [14] also on the larger VR game example (viz., 3 to 66 app components, 1 Cloud, up to 80 Fog nodes) proposed in [34].

Inspired by FogTorch\(\Pi\) models and algorithms, Xia et al. [63] proposed a backtracking solution to FAPP to minimise the average response time of deployed IoT applications. Two new heuristics were devised. The first one sorts the nodes considered for deploying each component in ascending order with respect to the (average) latency between each node and the IoT devices required by the component. The second one considers a component that caused backtracking as the first one to be mapped in the next search step. Despite discussing improved results on latency with respect to exhaustive backtracking and first-fit strategies, no prototype implementations were released. Finally, FogTorch\(\Pi\) was also modularly extended by De Maio et al. [27] to simulate mobile task offloading in Edge computing scenarios.

\(^1\) Available at: https://github.com/di-unipi-socc/FogTorchPI/
Mimicking Fog Application Management – CISCO FogDirector [23] is among the first available management tools for large-scale production deployments of Fog applications. It provides centralised management services that span the entire lifecycle of Fog applications, and it can be used via REST APIs that enable integration with client programs implementing application management.

In [31] we presented a simple operational semantics of all basic functionalities of FogDirector, describing the effects of the operations that client programs can perform to publish, deploy, configure, start, monitor, stop, undeploy and retire their applications in a Fog-Director-managed infrastructure. Based on the given formalisation, we implemented a prototype\(^2\), FogDirMime, which is the core of a simulator environment for FogDirector. The prototype also simulates probabilistic (hardware and network QoS) variations of the infrastructure that happen independently from the considered application management.

On one hand, the proposed semantics constitutes a concise and unambiguous reference of the (basic) behaviour of FogDirector that can be used to quickly understand its functioning and to check the correctness of management scripts at design time. On the other hand, FogDirMime can be fruitfully exploited to experiment and compare different application management policies, so to predict their effectiveness and tune them in a simulated environment, according to user-defined metrics. The prototype was used over a smart building use case.

5 Conclusions & Future Work

We consider our preliminary results and prototypes the first promising steps to support decision-making when deploying or managing IoT applications to Fog infrastructures. Yet, such results clearly present some limitations with respect to the objectives of this thesis, as set in Section 3.

In our future work, we intend to:

1. extend our methodologies to include more aspects of the life-cycle of application management, including new features such as components upgrade, reconfiguration and scaling, while envisioning the possibility for components to be deployed in different flavours like in Osmotic Computing [59],
2. consider new metrics and dimensions that will be important in Fog scenarios (e.g., security, mobility, energy consumption) and propose ways to automatically and efficiently select best candidate (re-)deployments – i.e., matching deployers’ desiderata – using (explainable) probabilistic AI [8] or multi-objective optimisation, and
3. prototype, validate and assess all new methodologies as extensions to our prototypes or as new open-source tools that can synergically work with them, and assess them in controlled settings (e.g., over the simple Fog application we proposed in [15]) as well as, possibly, in lifelike Fog environments.

Naturally, we plan to validate the proposed approaches by formally proving the correctness and completeness of the proposed algorithms, when possible.

\(^2\) Available at: https://github.com/di-unipi-socc/FogDirMime/
References

27. De Maio, V., Brandic, I.: First hop mobile offloading of dag computations (2018), In press