Deploying Fog Applications: How Much Does It Cost, By the Way?

Antonio Brogi, Stefano Forti and Ahmad Ibrahim
Department of Computer Science, University of Pisa, Italy

Keywords: Fog Computing, Application Deployment, QoS, Cost Models, Resource Consumption.

Abstract: Deploying IoT applications through the Fog in a QoS-, context-, and cost-aware manner is challenging due to the heterogeneity, scale and dynamicity of Fog infrastructures. To decide how to allocate app functionalities over the continuum from the IoT to the Cloud, app administrators need to find a trade-off among QoS, resource consumption and cost. In this paper, we present a novel cost model for estimating the cost of deploying IoT applications to Fog infrastructures. We show how the inclusion of the cost model in the FogTorchΠ open-source prototype permits to determine eligible deployments of multi-component applications to Fog infrastructures and to rank them according to their QoS-assurance, Fog resource consumption and cost. We run the extended prototype on a motivating scenario, showing how it can support IT experts in choosing the deployments that best suit their desiderata.

1 INTRODUCTION

Fog computing (Bonomi et al., 2014) aims at extending the Cloud towards the Internet of Things (IoT) to better support time-sensitive and bandwidth hungry IoT applications, by exploiting a multitude of collaborating heterogeneous devices spanning the Things[1] to Cloud continuum from IoT gateways to micro-datacentres. If architecture is about functionality allocation (Chiang and Zhang, 2016), then deciding where to deploy application functionalities (e.g., control loops, operational support, business intelligence) will be crucial in defining Fog architectures (OpenFog, 2016).

Modern applications usually consist of many independently deployable components (each with its hardware, software and IoT requirements) that interact together in a distributed way. Such interactions may have stringent QoS requirements – latency, bandwidth – to be fulfilled for the deployed application to work as expected (Dastjerdi and Buyya, 2016). Thus, when deciding where to deploy application components, one should check their hardware, software, IoT and QoS requirements against the offerings of the available context infrastructure (Iorga et al., 2017).

Determining eligible deployments of a multi-component application to a given Fog infrastructure is an NP-hard problem (Brogi and Forti, 2017). To make matters worse, variations in the QoS featured by communication links at different moments in time can cause violations to the QoS requirements of a deployed application.

Estimating deployment costs is very important to industry and businesses which aim at minimising deployment operational costs at runtime, having both to fulfil user requirements and to maximise their revenues (Niyato et al., 2016). Indeed, financial considerations can influence deployment selection, since costs can considerably vary depending on the Fog or Cloud nodes of choice. Whilst Cloud offerings are limited to few large providers, Fog computing envisions many other small and medium players (e.g., single Fog node or Things owners) that will offer virtual instances or IoT capabilities at different pricing schemes, making it more difficult to identify cost-effective deployments. Thus, the availability of cost models that account for Fog peculiarities would empower such new players to better design billing of new services for their customers and estimate their revenues and outflows beforehand.

Overall, tools to support app deployment to the Fog should desirably feature *(i) QoS-awareness* to achieve latency reduction, bandwidth savings and to enforce business policies, *(ii) context-awareness* to suitably exploit local and remote resources, and *(iii) cost-awareness* to enact cost-effective deployments.

In (Brogi et al., 2017), we developed a prototype (FogTorchΠ) that (1) determines app deployments

[1]Hereinafter, the word *Things* is used to refer to IoT devices, both sensors and actuators.

Brogi, A., Forti, S. and Ibrahim, A.
Deploying Fog Applications: How Much Does It Cost, By the Way?.
DOI: 10.5220/0006676100680077
In *Proceedings of the 8th International Conference on Cloud Computing and Services Science (CLOSER 2018)*, pages 68-77
ISBN: 978-989-758-295-0

that meet all processing, IoT and QoS requirements over a given infrastructure, (2) estimates their *QoS-assurance* and *Fog resource consumption* by simulating latency and bandwidth variations of communication links as per given probability distributions.

In this paper, we present a novel cost model for estimating the *monthly cost* of application deployments to IoT+Fog+Cloud infrastructures. We extend the FogTorchΠ prototype so to include the proposed cost model and to feature cost-awareness. The novelty of our approach is in that it extends existing pricing schemes for the Cloud to Fog computing scenarios, whilst introducing the possibility of integrating such schemes with financial costs that originate from the exploitation of IoT devices (i.e., Sensing-as-a-Service subscriptions or data transfer costs) in the deployment of applications. We show how the new version of FogTorchΠ can help IT experts (or new businesses coming onto the Fog market) in deciding how to distribute application components to Fog infrastructures in a QoS-, context- and cost-aware manner.

The rest of the paper is organised as follows. After introducing a motivating example of a smart building application (Section 2), we briefly describe FogTorchΠ (Section 3) and present the cost model extension (Section 4). Then, we present the results obtained by applying the extended version of FogTorchΠ to the motivating example (Section 5) and discuss some related work (Section 6). Finally, we draw some concluding remarks (Section 7).

2 MOTIVATING EXAMPLE

Consider a simple Fog application (Figure 1) that manages fire alarm, heating and A/C systems, interior lighting, and security cameras of a smart building. The application consists of three microservices:

- IoTController, interacting with the connected cyber-physical systems,
- DataStorage, storing all sensed information for future use and employing machine learning techniques to update sense-act rules at the IoTController so to optimise heating and lighting management based on previous experience and/or on people behaviour, and
- Dashboard, aggregating and visualising collected data and videos, as well as allowing users to interact with the system.

Each microservice represents an independently deployable component of the application (Newman, 2015) and has hardware and software requirements in order to function properly (as indicated in the grey box associated with each component). Hardware requirements are expressed in terms of the virtual machine (VM) types[2] listed in Table 1 and must be fulfilled by the VM that will host the component.

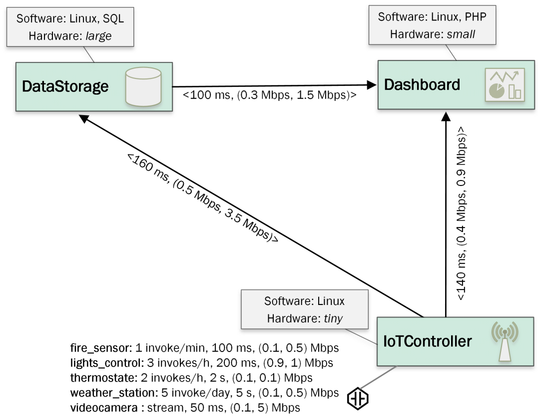

Figure 1: Fog application of the motivating example.

Table 1: Hardware specification for different VM types.

VM Type	vCPUSs	RAM (GB)	HDD (GB)
tiny	1	1	10
small	1	2	20
medium	2	4	40
large	4	8	80
xlarge	8	16	160

App components must cooperate so that well-defined levels of service are met by the application. Hence, communication links supporting component-component interactions should provide suitable end-to-end latency and bandwidth (e.g., the IoTController should reach the DataStorage within 160 ms and have at least 0.5 Mbps download and 3.5 Mbps upload free bandwidth[3]). Component-Things interactions have analogous constraints, and also specify the sampling rate at which IoTController is expected to query Things at runtime.

Figure 2 shows the infrastructure – two Cloud data centres, three Fog nodes and nine Things – selected by the system integrators in charge of deploying the smart building application for one of their customers. The deployed application will have to exploit the Things connected to Fog 1 and the weather_station_3 at Fog 3. Furthermore, the customer owns Fog 2, what makes deploying components to that node cost-free for the system integrators.

All Fog and Cloud nodes are associated with pricing schemes either to buy an instance of a certain

[2]Adapted from OpenStack Mitaka flavours: https://docs.openstack.org/.

[3]Arrows on the links in Figure 1 indicate the upload direction.

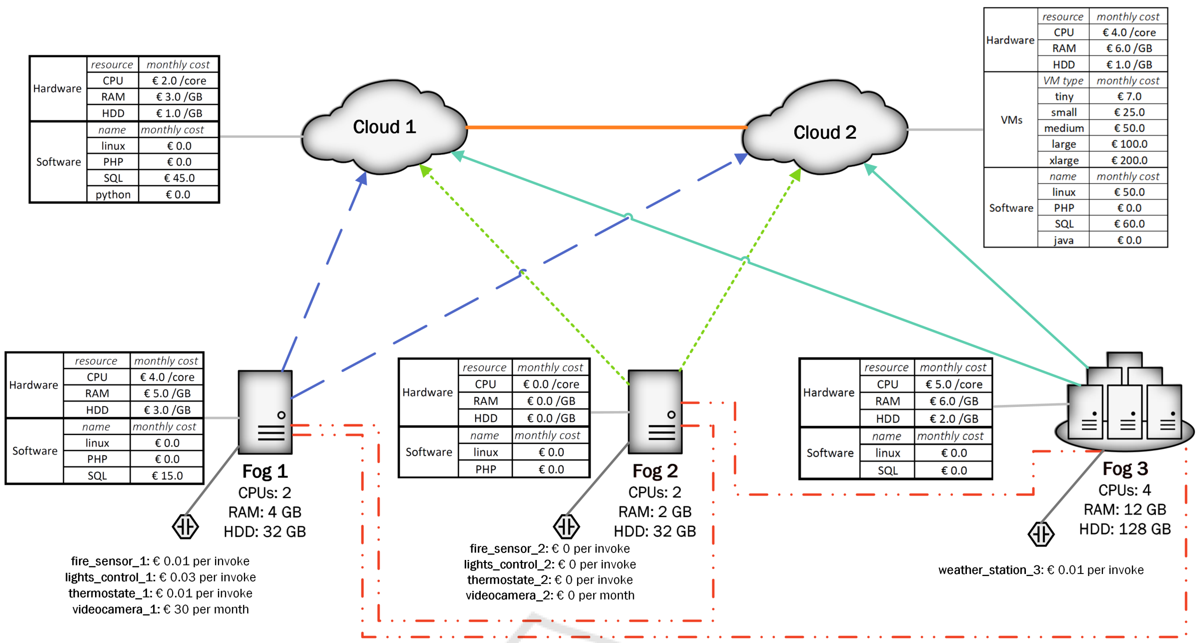

Figure 2: Fog infrastructure of the motivating example.

VM type (e.g., a *tiny* instance at Cloud 2 costs €7 per month), or to build on-demand instances by selecting the required number of cores and the needed amount of RAM and HDD to support a given component.

Fog nodes offer software capabilities, along with limited hardware resources (i.e., RAM, HDD, CPUs). Cloud nodes offer software capabilities, whilst hardware is considered unbounded assuming that one can always purchase extra or larger instances on-demand.

Finally, Table 2 lists the QoS profiles of the available communication links[4], which are represented as probability distributions based on real data[5], to account for variations in the QoS they provide. **Green** color links at Fog 2 initially feature a 3G Internet access. As per the current technical proposals (e.g., (Bonomi et al., 2014) and (OpenFog, 2016)), we assume Fog and Cloud nodes being able to access directly connected Things as well as Things at neighbouring nodes via a specific middleware layer (through the associated communication links).

Planning to sell the deployed solution for €1,500 a month, the system integrators set the limit of the monthly deployment cost at €850. Also, the customer requires the application to be compliant with the specified QoS requirements at least 98% of the time. Then, interesting questions for the system integrators before the first deployment of the application are, for instance:

Q1(a) — *Is there any eligible deployment of the application reaching the needed Things at Fog 1 and Fog 3, and meeting the financial (at most €850 per month) and QoS-assurance (at least 98% of the time) constraints mentioned above?*

Q1(b) — *Which eligible deployments minimise resource consumption in the Fog layer so to permit future deployment of services and sales of virtual instances to other customers?*

Suppose that with an extra monthly investment of €20, system integrators can exploit a 4G connection at Fog 2. Then:

Q2 — *Would there be any deployment that complies with all previous requirements and reduces financial cost and/or consumed Fog resources when upgrading from 3G to 4G at Fog 2?*

In Section 5, we will show how the new version of FogTorchΠ – extended with the cost model of Section

Table 2: QoS profiles of communication links.

Dash Type	Profile	Latency	Download	Upload
	Satellite 14M	40 ms	98%: 10.5 Mbps 2%: 0 Mbps	98%: 4.5 Mbps 2%: 0 Mbps
	3G	54 ms	99.6%: 9.61 Mbps 0.4%: 0 Mbps	99.6%: 2.89 Mbps 0.4%: 0 Mbps
	4G	53 ms	99.3%: 22.67 Mbps 0.7%: 0 Mbps	99.4%: 16.97 Mbps 0.6%: 0 Mbps
	VDSL	60 ms	60 Mbps	6 Mbps
	Fibre	5 ms	1000 Mbps	1000 Mbps
	WLAN	15 ms	90%: 32 Mbps 10%: 16 Mbps	90%: 32 Mbps 10%: 16 Mbps

[4]Arrows on the links in Figure 2 indicate the upload direction.

[5]Satellite: https://www.eolo.it/, 3G/4G: https://www.agcom.it, VDSL: http://www.vodafone.it.

4 – can be exploited to obtain answers to all the above questions.

3 OVERVIEW OF FogTorchΠ

FogTorchΠ (Brogi et al., 2017) is an open-source Java prototype[6] that permits to describe IoT+Fog+Cloud infrastructures and applications (based on the model of (Brogi and Forti, 2017)) so to determine QoS- and context-aware application deployments. Before introducing the cost-aware extension to FogTorchΠ, we summarise its basic functioning. FogTorchΠ inputs:

1. an *IoT+Fog+Cloud infrastructure* I, with the specification of the Fog and Cloud nodes available for deployment (each with its hardware, software and IoT capabilities), and the probability distributions of the QoS (viz., latency, bandwidth) featured by the communication links interconnecting such nodes[7],
2. a *multi-component application* A, specifying all hardware (i.e., CPU, RAM, storage), software (i.e., OS, libraries, frameworks) and IoT requirements (i.e., which type of Things to exploit) of each component, and the QoS (i.e., latency and bandwidth) needed to support component-component and component-Thing interactions,
3. a *Things binding* ϑ, mapping each IoT requirement of an application component to an actual Thing in I, and
4. a *deployment policy* $\delta(\gamma)$, white-listing the nodes where component γ of A can be deployed[8] according to security or business-related constraints.

Based on such input, FogTorchΠ determines all the eligible deployments of the components of A to Cloud or Fog nodes in I. An *eligible deployment* Δ maps each component γ of A to a Cloud or Fog node n in I so that (1) $n \in \delta(\gamma)$ and it satisfies the processing requirements of γ, (2) hardware resources are enough to deploy *all* components of A mapped to n, (3) Things specified in ϑ are reachable, and (4) component-component and component-Thing interactions mapped to the same communication link do not exceed the available bandwidth and meet their latency requirements.

[6]Available at https://github.com/di-unipi-socc/FogTorchPI/tree/costmodel/.

[7]Actual implementations in Fog landscapes can exploit monitoring tools (e.g., (Breitbart et al., 2001), (Fatema et al., 2014)) to get updated information on the state of I.

[8]When δ is not specified for a component γ of A, γ can be deployed to any compatible node in I.

```
 1: procedure MONTECARLO(A, I, ϑ, δ, n)
 2:     D ← ∅                    ▷ dictionary of ⟨Δ, counter⟩
 3:     for n times do
 4:         I_s ← SAMPLELINKSQOS(I)
 5:         E ← FINDDEPLOYMENTS(A, I_s, ϑ, δ)
 6:         D ← UNIONUPDATE(D, E)
 7:     end for
 8:     for Δ ∈ keys(D) do
 9:         D[Δ] ← D[Δ]/n
10:     end for
11:     return D
12: end procedure
```

Figure 3: Pseudocode of the Monte Carlo simulation in FogTorchΠ.

FogTorchΠ employs the Monte Carlo method (Dunn and Shultis, 2011) to estimate the *QoS-assurance* of output deployments, by aggregating the eligible deployments obtained when varying the QoS of communication links (as per their input probability distributions). In addition, FogTorchΠ outputs the percentage of resources (RAM and HDD) consumed in the Fog layer (or in specified Fog nodes) after performing an eligible deployment.

Figure 3 shows the pseudocode of FogTorchΠ functioning. First, an empty dictionary D is created to contain key-value pairs $\langle\Delta, \texttt{counter}\rangle$, where the key ($\Delta$) represents an eligible deployment and the value (`counter`) keeps track of how many times Δ will be generated during the Monte Carlo simulation (line 2). Then, at the beginning of each run of the simulation, a state I_s of the infrastructure is sampled according to the probability distributions of the QoS of the communication links in I (line 4).

The function FINDDEPLOYMENTS(A, I_s, ϑ, δ) (line 5) employs an exhaustive (backtracking) search to determine the set E of eligible deployments Δ of A to I_s, i.e. deployments of A that satisfy all processing and QoS requirements in that particular state of the infrastructure. Each output deployment Δ also contains information about its Fog resource consumption, which is computed during the search. The objective of this step is to look for eligible deployments, whilst dynamically simulating changes in the underlying network conditions. At the end of each run, the set E of eligible deployments of A to I_s is used to update D. The function UNIONUPDATE(D, E) (line 6) updates D by adding deployments $\langle\Delta, 1\rangle$ discovered during the last run ($\Delta \in E \setminus keys(D)$) and by incrementing the `counter` of those deployments that had already been found in a previous run ($\Delta \in E \cap keys(D)$).

At the end of the simulation ($n \geq 100,000$), the QoS-assurance of each deployment $\Delta \in keys(D)$ is computed by dividing the `counter` associated to Δ by n (lines 8–10). Thus, the QoS-assurance is the percentage of runs a certain deployment Δ was found by

FINDDEPLOYMENTS(A, I_s, ϑ, δ). Such percentage estimates how likely Δ is to meet all QoS constraints of A, taking into account variations in the communication links as per historical behaviour of I. Finally, dictionary D is returned (line 11).

The next section introduces the cost model extension of FogTorchΠ, which permits estimating the monthly deployment cost of output deployments. We extended FINDDEPLOYMENTS(A, I_s, ϑ, δ) so to return each deployment Δ with its associated monthly cost[9] in addition to its Fog resource consumption.

4 COST MODEL

Our cost model extends to Fog computing previous efforts in Cloud VM cost modelling (Díaz et al., 2017), and includes software costs, and costs due to IoT (Niyato et al., 2016).

At any Cloud or Fog node n, our cost model considers that a hardware offering H can be either *default VMs* (Table 1) offered at a fixed monthly fee or *on-demand VMs* (built with an arbitrary amount of cores, RAM and HDD). Being R the set of resources considered when building on-demand VMs (viz., $R = \{CPU, RAM, HDD\}$), the estimated monthly cost for a hardware offering H at node n is

$$p(H,n) = \begin{cases} c(H,n) & \text{if } H \text{ is a default VM} \\ \sum_{\rho \in R} [H.\rho \times c(\rho,n)] & \text{if } H \text{ is an on-demand VM} \end{cases}$$

where $c(H,n)$ is the monthly cost of a default VM H at Fog or Cloud node n, whilst $H.\rho$ indicates the amount of resource $\rho \in R$ used by[10] the on-demand VM represented by H, and $c(\rho,n)$ is the unit monthly cost at n for resource ρ.

Analogously, for any given Cloud or Fog node n, a software offering S can be either a predetermined software bundle or an on-demand subset of the software capabilities available at n (each sold separately). The estimated monthly cost for S at node n is

$$p(S,n) = \begin{cases} c(S,n) & \text{if } S \text{ is a bundle} \\ \sum_{s \in S} c(s,n) & \text{if } S \text{ is on-demand} \end{cases}$$

where $c(S,n)$ is the price for the software bundle S at node n, and $c(s,n)$ is the monthly cost of a single software s at n.

Finally, in Sensing-as-a-Service (Perera, 2017) scenarios, a Thing offering T exploiting an actual Thing t can be offered at a monthly subscription fee or through a pay-per-invocation mechanism. Then, the cost for offering T at Thing t is

$$p(\mathrm{T},t) = \begin{cases} c(\mathrm{T},t) & \text{if T is subscription based} \\ \mathrm{T}.k \times c(t) & \text{if T is pay-per-invocation} \end{cases}$$

where $c(\mathrm{T},t)$ is the monthly subscription fee for T at t, while $\mathrm{T}.k$ is the number of monthly invocations expected over t and $c(t)$ is the cost per invocation at t (including Thing usage and/or data transfer costs).

In what follows, we assume that Δ is an eligible deployment for an application A to an infrastructure I, as introduced in Section 3. In addition, let $\gamma \in A$ be a component of the considered application A, and let $\gamma.\overline{\mathcal{H}}$, $\gamma.\overline{\Sigma}$ and $\gamma.\overline{\Theta}$ be its hardware, software and Things requirements, respectively. Overall, the expected monthly cost for a given deployment Δ can be first approximated by combining the previous pricing schemes as in:

$$cost(\Delta,\vartheta,A) = \sum_{\gamma \in A} \Big[p(\gamma.\overline{\mathcal{H}},\Delta(\gamma)) + p(\gamma.\overline{\Sigma},\Delta(\gamma)) + \sum_{r \in \gamma.\overline{\Theta}} p(r,\vartheta(r)) \Big]$$

Although this formula gives an estimate of the monthly cost for a given deployment, yet it does not feature a way to select the "best" offering to match the application requirements at the VM, software and IoT levels. Particularly, it may lead the choice always to on-demand and pay-per-invocation offerings when the application requirements do not match exactly default or bundled offerings, or when a Cloud provider does not offer a particular VM type (e.g., starting its offerings from *medium*). This can lead to overestimate the monthly deployment cost.

For instance, consider the infrastructure of Figure 2 and the hardware requirements of a component to be deployed to Cloud 2, specified as $R = \{\texttt{CPU}:1, \texttt{RAM}:1\texttt{GB}, \texttt{HDD}:20\texttt{GB}\}$. Since no exact matching between the requirement and an offering at Cloud 2 exists, this first cost model would select an on-demand instance, and estimate its cost of €30 [11]. However, Cloud 2 also provides a *small* instance that can satisfy the requirements at a (lower) cost of €25.

Since larger VM types always satisfy smaller hardware requirements, bundled software offerings may satisfy multiple software requirements at a lower price, and subscription-based Thing offerings can be

[9] Cost computation is performed *on-the-fly* during the search step, envisioning the possibility to exploit cost as a heuristic to lead the search algorithm towards a best candidate deployment.

[10] Bounded by the maximum amount purchasable at any chosen Cloud or Fog node.

[11] €30 = 1 CPU x €4/core + 1 GB RAM x €6/GB + 20 GB HDD x €1/GB

more or less convenient depending on the number of invocations on a given Thing, some policy must be used to choose the "best" offerings for each software, hardware and Thing requirement of an application component. In what follows, we refine our cost model to also account for this fact.

A *requirement-to-offering matching policy* $p_m(r,n)$ matches hardware or software requirements r of a component ($r \in \{\gamma.\overline{\mathcal{H}}, \gamma.\overline{\Sigma}\}$) to the estimated monthly cost of the offering that will support them at Cloud or Fog node n, and a Thing requirement $r \in \gamma.\overline{\Theta}$ to the estimated monthly cost of the offering that will support r at Thing t.

Overall, this refined version of the cost model permits to estimate the monthly cost of Δ including a cost-aware matching between application requirements and infrastructure offering (for hardware, software and IoT), chosen as per p_m. Hence:

$$cost(\Delta, \vartheta, A) = \sum_{\gamma \in A} \left[p_m(\gamma.\overline{\mathcal{H}}, \Delta(\gamma)) \; + \; p_m(\gamma.\overline{\Sigma}, \Delta(\gamma)) \; + \sum_{r \in \gamma.\overline{\Theta}} p_m(r, \vartheta(r)) \right]$$

The new version of FogTorchΠ – extended with the cost model – exploits a *best-fit lowest-cost* policy for choosing hardware, software and Thing offerings. Indeed, it selects the cheapest between the first default VM (from *tiny* to *xlarge*) that can support $\gamma.\overline{\mathcal{H}}$ at node n and the on-demand offering built as per $\gamma.\overline{\mathcal{H}}$. Likewise, software requirements in $\gamma.\overline{\Sigma}$ are matched with the cheapest compatible version available at n, and Thing per invocation offer is compared to monthly subscription so to select the cheapest[12].

Formally, the cost model used by the new version of FogTorchΠ can be expressed as:

$$p_m(\overline{\mathcal{H}}, n) = \min\{p(H,n)\}$$
$$\forall\, H \in \{\text{default VMs, on-demand VM}\} \wedge H \models \overline{\mathcal{H}}$$

$$p_m(\overline{\Sigma}, n) = \min\{p(S,n)\}$$
$$\forall\, S \in \{\text{on-demand, bundle}\} \wedge S \models \overline{\Sigma}$$

$$p_m(r,t) = \min\{p(T,t)\}$$
$$\forall\, T \in \{\text{subscription, pay-per-invocation}\} \wedge T \models r$$

where $O \models R$ reads as offering O *satisfies requirements* R.

It is worth noting that the proposed cost model separates the cost of purchasing VMs from the cost of purchasing software. This choice keeps the modelling general enough to include both IaaS and PaaS Cloud offerings. Furthermore, even if we referred to VMs as the only deployment unit for application components, the model can be easily extended so to include other types of virtual instances (e.g., containers).

[12] Other policies are also possible such as, for instance, selecting the largest offering that can accommodate a component, or always increasing the component's requirements by some percentage (e.g., 10%) before selecting the matching.

5 MOTIVATING EXAMPLE (CONTINUED)

We now present the results of running the new version of FogTorchΠ over the smart building example of Section 2 and to get answers for the questions of the system integrators. FogTorchΠ outputs the eligible deployments (as per Section 3) along with their estimated QoS-assurance, Fog resource consumption and monthly cost (as per Section 4).

Table 3: Eligible deployments generated by FogTorchΠ for **Q1** and **Q2**[13].

Dep. ID	IoTController	DataStorage	Dashboard
Δ1	Fog 2	Fog 3	Cloud 2
Δ2	Fog 2	Fog 3	Cloud 1
Δ3	Fog 3	Fog 3	Cloud 1
Δ4	Fog 2	Fog 3	Fog 1
Δ5	Fog 1	Fog 3	Cloud 1
Δ6	Fog 3	Fog 3	Cloud 2
Δ7	Fog 3	Fog 3	Fog 2
Δ8	Fog 3	Fog 3	Fog 1
Δ9	Fog 1	Fog 3	Cloud 2
Δ10	Fog 1	Fog 3	Fog 2
Δ11	Fog 1	Fog 3	Fog 1
Δ12	Fog 2	Cloud 2	Fog 1
Δ13	Fog 2	Cloud 2	Cloud 1
Δ14	Fog 2	Cloud 2	Cloud 2
Δ15	Fog 2	Cloud 1	Cloud 2
Δ16	Fog 2	Cloud 1	Cloud 1
Δ17	Fog 2	Cloud 1	Fog 1

For question **Q1(a)**, the new version of FogTorchΠ outputs eleven eligible deployments ($\Delta 1$ — $\Delta 11$ in Table 3), determined as described in Section 3.
It is worth recalling that we envision remote access to Things connected to Fog nodes from other Cloud and Fog nodes. In fact, some output deployments map components to nodes that do not direclty connect to all the required Things. For instance, in the case of $\Delta 1$, IoTController is deployed to Fog 2 but the required Things (fire_sensor_1, light_control_1, thermostate_1, video_camera_1, weather_station_3) are attached to Fog 1 and Fog 3, still being reachable with suitable latency and bandwidth.

Figure 4 only shows the five output deployments that satisfy the QoS and budget constraints imposed

[13] Results and Python code to generate 3D plots as in Figures 4 and 5 are available at: https://github.com/di-unipi-socc/FogTorchPI/tree/costmodel/results/SMARTBUILDIN G18/.

by the system integrators. Δ3, Δ4, Δ7 and Δ10 all feature 100% QoS-assurance. Among them, Δ7 is the cheapest in terms of cost, consuming as much Fog resources as Δ4 and Δ10, although more with respect to Δ3. On the other hand, Δ2, still showing QoS-assurance above 98% and consuming as much Fog resources as Δ3, can be a good compromise at the cheapest monthly cost of €800 (what answers question **Q1(b)**).

Finally, to answer question **Q2**, we change the Internet access at Fog 2 from 3G to 4G. As mentioned in Section 2, this increases the monthly expenses by €20. Running FogTorchΠ now reveals six new eligible deployments (Δ12 — Δ17) in addition to the previous output. Among those, only Δ16 turns out to meet also the QoS and budget constraints that the system integrators require (Figure 5). Interestingly, Δ16 costs €70 less than the best candidate for **Q1(b)** (Δ2), whilst sensibly reducing Fog resource consumption. Hence, overall, the change from 3G to 4G would lead to an estimated monthly saving of €50, enacting Δ16 instead of Δ2.

The final choice for a particular deployment is left to the system integrators, leaving them the freedom to select the "best" trade-off among QoS-assurance, resource consumption and cost. Indeed, the analysis of application specific requirements (along with data on infrastructure behaviour) can lead decision towards different segmentations of an application from the IoT to the Cloud. Conversely to multi-objective optimisation techniques (Gao et al., 2013), we follow a human-driven approach – aided by predictive tools like FogTorchΠ – to determine the best trade-off among metrics that describe likely run time behaviour of a deployment and make it possible to evaluate changes in the infrastructure (or in the application) before their actual implementation (*what-if* analyses (Rizzi, 2009)).

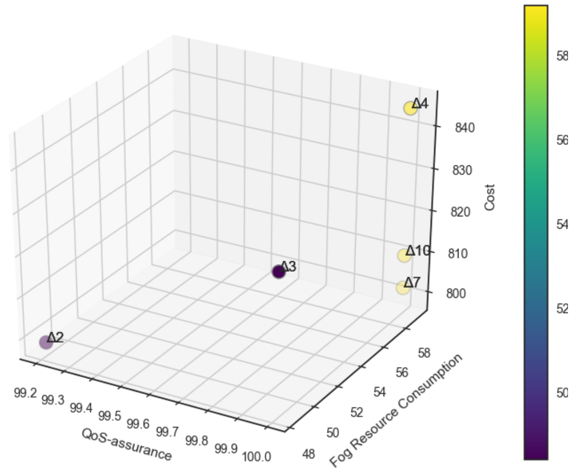

Figure 4: Results for **Q1(a)** and **Q1(b)**. Colormap refers to Fog resource consumption.

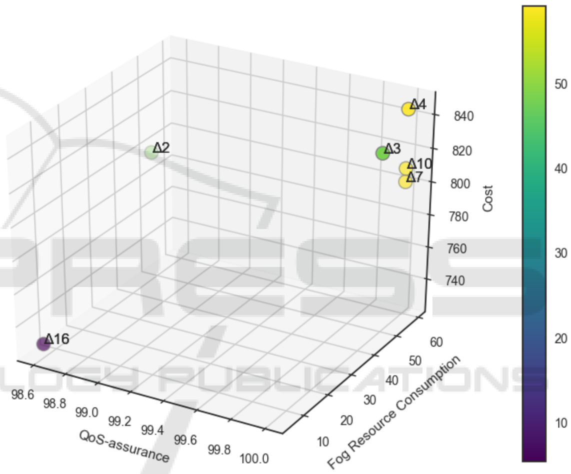

Figure 5: Results for **Q2**. Colormap refers to Fog resource consumption.

6 RELATED WORK

With respect to the Cloud paradigm, the Fog introduces new problems, mainly due to its pervasive geo-distribution and heterogeneity, need for connection-awareness, dynamicity and support to interactions with the IoT, that were not taken into account by previous works (Varshney and Simmhan, 2017) (Wen et al., 2017) (Arcangeli et al., 2015). Particularly, some efforts in Cloud computing considered non-functional requirements (e.g., (Nathuji et al., 2010), (Cucinotta and Anastasi, 2011),(Rimal et al., 2011) and (Durao et al., 2014)) or uncertainty of execution (as in Fog nodes) and security risks among interactive and interdependent components (Wen et al., 2016). Only recently, (Wang et al., 2016) has linked services and networks QoS by proposing a QoS- and connection-aware Cloud service composition approach to satisfy end-to-end QoS requirements in the Cloud.

To the best of our knowledge, few approaches have been proposed so far to specifically model Fog infrastructures and applications, as well as to determine and compare eligible deployments for an application to an Fog infrastructure under different metrics. (Sarkar and Misra, 2016) aims at evaluating service latency and energy consumption of the new Fog paradigm applied to the IoT, as compared to traditional Cloud scenarios. The model of (Sarkar and Misra, 2016), however, deals only with the behaviour of software already deployed over Fog infrastructures and simulates it mathematically.

iFogSim (Gupta et al., 2017) is among the most promising prototypes to simulate resource management and scheduling policies applicable to Fog environments with respect to their impact on latency, energy consumption and operational cost. The focus of iFogSim model is mainly on stream-processing applications and hierarchical tree-like infrastructures, to be mapped either Cloud-only or Edge-ward so to compare results.

Building on top of iFogSim, (Bittencourt et al., 2017) compares different task scheduling policies, taking into account user mobility, optimal Fog resource utilisation and response time. (Tärneberg et al., 2017) presents a distributed approach to cost-effective application placement, at varying workload conditions, with the objective of optimising operational cost across the entire infrastructure. Apropos, (Shekhar et al., 2017) introduces a hierarchy-based technique to dynamically manage and migrate applications between Cloud and Fog nodes. They exploit message passing among local and global node managers to guarantee QoS and cost constraints are met. Similarly, (Skarlat et al., 2017) leverages the concept of Fog colonies (Skarlat et al., 2016) for scheduling tasks to Fog infrastructures, whilst minimising response times. (Aazam et al., 2016) provides a first methodology for probabilistic record-based resource estimation to mitigate resource underutilisation, to enhance the QoS of provisioned IoT services.

All the aforementioned approaches focus on monolithic or DAG application topologies, and do not take into account QoS for the interactions with the IoT, nor historical data about Fog infrastructure or deployment behaviour. Our approach permits instead to express arbitrary multi-component application topologies, as the one of the illustrated example. Furthermore, the attempts to explicitly target and support with predictive methodologies the decision-making process to deploy IoT applications to the Fog were very limited, and none of them considered matching of application components to the best virtual instance (Virtual Machine or container), depending on expressed preferences (e.g., cost or energy targets) in this work.

Pricing models for the Cloud are quite established (e.g., (Díaz et al., 2017), (Niyato et al., 2016) and references therein) but they do not account for costs generated by the exploitation of IoT devices. Cloud pricing models are generally divided into two types, pay per use scheme and subscription-based. In (Díaz et al., 2017), based on given user workload requirements, a Cloud broker chooses a best VM instance(s) among several cloud providers. The total cost of deployment is calculated considering hardware requirements such as number of CPU cores, VM types, time duration, type of instance (reserved or pre-emptible), etc.

On the other hand, IoT providers normally process the sensory data coming from the IoT devices and sell the processed information as value added service to the users. (Niyato et al., 2016) shows how they can also act as brokers, acquiring data from different owners and selling bundles. The authors of (Niyato et al., 2016) also consider the fact that different IoT providers can federate their services and create new offers for their end-users. Such end-users are then empowered to estimate the total cost of using IoT services by comparing pay-per-use and subscription-based offers, depending upon their data demand. In Fog scenario, however, there is a need to compute IoT costs at a finer level, also accounting for data-transfer costs (i.e., event-based). More recently, (Markus et al., 2017) propose a cost model for IoT+Cloud scenario. Considering parameters such as the type and number of sensors, number of data request and uptime of VM, their cost model can estimate the cost of running an application over a certain period of time.

Other recent studies tackle akin challenges from an infrastructural perspective either focusing on scalable algorithms for QoS-aware placement of micro-data centres (Selimi et al., 2017), on optimal placement of data and storage nodes that ensures low latencies and maximum throughput, optimising costs (Naas et al., 2017), or on the exploitation of genetic algorithms to place intelligent access points at the edge of the network (Majd et al., 2017).

To the best of our knowledge, our attempt to model costs in the Fog scenario is the first that extends Cloud pricing schemes to the Fog layer and integrates them with costs that are typical of IoT deployments.

7 CONCLUDING REMARKS

In this paper, we presented a novel cost model to estimate multi-component application deployment cost to IoT+Fog+Cloud infrastructures. The model considers various cost parameters (hardware, software and IoT), extending Cloud computing cost models to the Fog computing paradigm, whilst taking into account costs associated to the usage of IoT devices and services.

We included it in the FogTorchΠ prototype to show how it can assist IT experts in deciding how to distribute a multi-component application over a given infrastructures in a QoS-, context-, and cost-aware manner. We envision the possibility of exploiting such a cost model to drive design of billing of new services

offered by small and medium enterprises in the Fog marketplace. Indeed, supporting deployment decision in the Fog requires comparing a multitude of offerings where providers are able to deploy their applications to their infrastructure integrated with the Cloud, with the IoT, with federated Fog devices as well as with user-managed devices.
We see three main directions for future work:

- Exploiting a multitude of highly distributed nodes, Fog computing is likely to consume more energy with respect to the Cloud. Application deployments should also consider energy-related issues so to guarantee reliable service provisioning and longer deployment lifetime when exploiting battery powered IoT devices or Fog nodes. Hence, we aim at further extending our contribution to consider energy consumption as a characterising metric for eligible deployments.
- Monte Carlo simulation is in general computationally expensive but it can be efficiently parallelised and optimised. Furthermore, FogTorchΠ exploits exponential search algorithms. Apropos, another direction for future work is to parallelise the simulation and tame the complexity of FogTorchΠ algorithms to scale better over large infrastructures, by leading search with improved heuristics and by approximating metrics estimation.
- Currently, Fog computing lacks medium to large scale test-bed deployments (i.e., infrastructure and applications) to test devised approaches. Last, but not least, we intend to contribute further in engineering FogTorchΠ and to assess validity of the prototype over an experimental lifelike test-bed that is currently at study.

REFERENCES

Aazam, M., St-Hilaire, M., Lung, C. H., and Lambadaris, I. (2016). MeFoRE: QoE based resource estimation at Fog to enhance QoS in IoT. In *2016 23rd International Conference on Telecommunications (ICT)*, pages 1–5.

Arcangeli, J.-P., Boujbel, R., and Leriche, S. (2015). Automatic deployment of distributed software systems: Definitions and state of the art. *Journal of Systems and Software*, 103:198–218.

Bittencourt, L. F., Diaz-Montes, J., Buyya, R., Rana, O. F., and Parashar, M. (2017). Mobility-aware Application Scheduling in Fog Computing. *IEEE Cloud Computing*, 4(2):26–35.

Bonomi, F., Milito, R., Natarajan, P., and Zhu, J. (2014). Fog computing: A Platform for Internet of Things and Analytics. In *Big Data and Internet of Things: A Roadmap for Smart Environments*, pages 169–186.

Breitbart, Y., Chan, C.-Y., Garofalakis, M., Rastogi, R., and Silberschatz, A. (2001). Efficiently monitoring bandwidth and latency in IP networks. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 933–942. IEEE.

Brogi, A. and Forti, S. (2017). QoS-aware Deployment of IoT Applications Through the Fog. *IEEE Internet of Things Journal*, 4(5):1185–1192.

Brogi, A., Forti, S., and Ibrahim, A. (2017). How to best deploy your Fog applications, probably. In Rana, O., Buyya, R., and Anjum, A., editors, *Proceedings of 1st IEEE International Conference on Fog and Edge Computing (ICFEC), Madrid*, pages 105–114.

Chiang, M. and Zhang, T. (2016). Fog and IoT: An overview of research opportunities. *IEEE Internet of Things Journal*, 3(6):854–864.

Cucinotta, T. and Anastasi, G. F. (2011). A heuristic for optimum allocation of real-time service workflows. In *Service-Oriented Computing and Applications (SOCA), 2011 IEEE Int. Conf. on*, pages 1–4. IEEE.

Dastjerdi, A. V. and Buyya, R. (2016). Fog Computing: Helping the Internet of Things Realize its Potential. *Computer*, 49(8):112–116.

Díaz, J. L., Entrialgo, J., García, M., García, J., and García, D. F. (2017). Optimal allocation of virtual machines in multi-cloud environments with reserved and on-demand pricing. *Future Generation Computer Systems*, 71:129–144.

Dunn, W. L. and Shultis, J. K. (2011). *Exploring Monte Carlo Methods*. Elsevier.

Durao, F., Carvalho, J. F. S., Fonseka, A., and Garcia, V. C. (2014). A systematic review on cloud computing. *The Journal of Supercomputing*, 68(3):1321–1346.

Fatema, K., Emeakaroha, V. C., Healy, P. D., Morrison, J. P., and Lynn, T. (2014). A survey of Cloud monitoring tools: Taxonomy, capabilities and objectives. *Journal of Parallel and Distributed Computing*, 74(10):2918–2933.

Gao, Y., Guan, H., Qi, Z., Hou, Y., and Liu, L. (2013). A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. *Journal of Computer and System Sciences*, 79(8):1230–1242.

Gupta, H., Dastjerdi, A. V., Ghosh, S. K., and Buyya, R. (2017). iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing Environments. *Software: Practice and Experience*, 47(9):1275–1296.

Iorga, M. et al. (2017). The NIST Definition of Fog Computing (draft SP 800-191).

Majd, A., Sahebi, G., Daneshtalab, M., Plosila, J., and Tenhunen, H. (2017). Hierarchal Placement of Smart Mobile Access Points in Wireless Sensor Networks Using Fog Computing. In *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pages 176–180.

Markus, A., Kertesz, A., and Kecskemeti, G. (2017). Cost-

Aware Iot Extension of DISSECT-CF. *Future Internet*, 9(3).

Naas, I., Raipin, P., Boukhobza, J., and Lemarchand, L. (2017). iFogStor: an IoT Data Placement Strategy for Fog Infrastructure. In Rana, O., Buyya, R., and Anjum, A., editors, *Proceedings of 1st IEEE International Conference on Fog and Edge Computing, Madrid*. In press.

Nathuji, R., Kansal, A., and Ghaffarkhah, A. (2010). Q-Clouds: Managing Performance Interference Effects for QoS-Aware Clouds. Association for Computing Machinery, Inc.

Newman, S. (2015). *Building microservices: designing fine-grained systems*. " O'Reilly Media, Inc.".

Niyato, D., Hoang, D. T., Luong, N. C., Wang, P., Kim, D. I., and Han, Z. (2016). Smart data pricing models for the internet of things: a bundling strategy approach. *IEEE Network*, 30(2):18–25.

OpenFog (2016). OpenFog Reference Architecture.

Perera, C. (2017). *Sensing as a Service for Internet of Things: A Roadmap*. Lulu. com.

Rimal, B. P., Jukan, A., Katsaros, D., and Goeleven, Y. (2011). Architectural Requirements for Cloud Computing Systems: An Enterprise Cloud Approach. *Journal of Grid Computing*, 9(1):3–26.

Rizzi, S. (2009). What-if analysis. In *Encyclopedia of Database Systems*, pages 3525–3529. Springer.

Sarkar, S. and Misra, S. (2016). Theoretical modelling of fog computing: a green computing paradigm to support IoT applications. *IET Networks*, 5(2):23–29.

Selimi, M., Cerdà-Alabern, L., Sánchez-Artigas, M., Freitag, F., and Veiga, L. (2017). Practical Service Placement Approach for Microservices Architecture.

Shekhar, S., Chhokra, A., Bhattacharjee, A., Aupy, G., and Gokhale, A. (2017). INDICES: Exploiting Edge Resources for Performance-aware Cloud-hosted Services. In Rana, O., Buyya, R., and Anjum, A., editors, *Proceedings of 1st IEEE International Conference on Fog and Edge Computing, Madrid*.

Skarlat, O., Nardelli, M., Schulte, S., and Dustdar, S. (2017). Towards QoS-aware fog service placement. In Rana, O., Buyya, R., and Anjum, A., editors, *Proceedings of 1st IEEE International Conference on Fog and Edge Computing, Madrid*. In press.

Skarlat, O., Schulte, S., Borkowski, M., and Leitner, P. (2016). Resource Provisioning for IoT Services in the Fog. In *SOCA*, pages 32–39. IEEE.

Tärneberg, W., Papadopoulos Vittorio, A., Mehta, A., Tordsson, J., and Kihl, M. (2017). Distributed Approach to the Holistic Resource Management of a Mobile Cloud Network. In Rana, O., Buyya, R., and Anjum, A., editors, *Proceedings of 1st IEEE International Conference on Fog and Edge Computing, Madrid*.

Varshney, P. and Simmhan, Y. (2017). Demystifying Fog Computing: Characterizing Architectures, Applications and Abstractions. In *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*, pages 115–124.

Wang, S., Zhou, A., Yang, F., and Chang, R. N. (2016). Towards Network-Aware Service Composition in the Cloud. *IEEE Transactions on Cloud Computing*.

Wen, Z., Cala, J., Watson, P., and Romanovsky, A. (2016). Cost effective, reliable and secure workflow deployment over federated clouds. *IEEE Transactions on Services Computing*, PP(99):1–1.

Wen, Z., Yang, R., Garraghan, P., Lin, T., Xu, J., and Rovatsos, M. (2017). Fog Orchestration for Internet of Things Services. *IEEE Internet Computing*, 21(2):16–24.