Simulation Modelling Practice and Theory 101 (2020) 101966

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Sphere: Simulator of edge infrastructures for the optimization of performance and resources energy consumption

Damián Fernández-Cerero*,[a], Alejandro Fernández-Montes[a], F. Javier Ortega[a], Agnieszka Jakóbik[b], Adrian Widlak[b]

[a] Escuela Técnica Superior de Ingeniería Informática. Universidad de Sevilla, Spain
[b] Institute of Computer Science, Cracow University of Technology, Poland

ARTICLE INFO

Keywords:
Edge computing
Fog computing
Cloudlet computing
Cloud computing
Energy-aware scheduling

ABSTRACT

Edge computing constitutes a key paradigm to address the new requirements of areas such as smart cars, industry 4.0, and health care, where massive amounts of heterogeneous data from continuous geographically-distributed sources have to be processed and computed near real-time. To this end, new distributed infrastructures consisting on small computing clusters close to data sources, also known as *Cloudlets* have emerged. In order to evaluate the performance of these solutions we present *Sphere*, a simulation tool that enables researchers to establish various scenarios, including: (a) topology and orchestration model of the infrastructure; (b) incoming workload patterns; (c) resource-managing models; and (d) scheduling policies. Moreover, *Sphere* allows researchers to apply efficiency and performance policies both at infrastructure and cluster levels. The simulator presents the following benefits: (a) Evaluation of various orchestration models; (b) Analysis of resource-efficiency and performance strategies at Edge-infrastructure and cluster (Cloudlet/Cloud) level; (c) Execution of diverse workload generation patterns; (d) Evaluation of strategies for the infrastructure communication, as well as the impact on tasks completion time (makespan); and (e) Simulation of each cluster (Cloudlet/Cloud) independently, including resource-managing, scheduling and resource-efficiency models. Finally, we performed a deep evaluation based on realistic Edge-Computing use cases. The results of the experiments confirm that it is a performant and reliable tool for the analysis of orchestration, graph-resolving, energy-efficiency, resource-managing and scheduling strategies in Edge-computing environments.

1. Introduction

The spread of new computing paradigms, such as Internet of Things and Internet of Everything, as well as the consolidation of concepts such as Smart City [1], lead to new models of data production, processing and utilization.

The requirements of traditional offline and centralized Cloud-Computing scenarios, for instance, those imposed by Map-Reduce environments, are different from those present in the new data-processing models mentioned above. Current trends in areas such as industry 4.0, smart cars, health-care and security [2], usually have in common the massive amount of heterogeneous, geographically-distributed, and inter-connected data that are produced by thousands or even millions of different sources [3], which need to be

* Corresponding author.
E-mail addresses: damiancerero@us.es (D. Fernández-Cerero), afdez@us.es (A. Fernández-Montes), javierortega@us.es (F. Javier Ortega), ajakobik@pk.edu.pl (A. Jakóbik), adrian.widlak@pk.edu.pl (A. Widlak).

https://doi.org/10.1016/j.simpat.2019.101966
Received 15 May 2019; Received in revised form 1 July 2019; Accepted 2 August 2019
Available online 03 August 2019

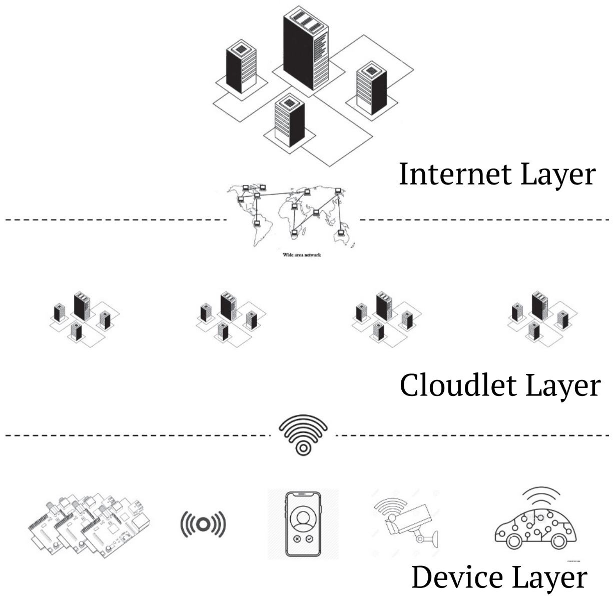

Fig. 1. Cloudlet-Computing Architecture.

securely processed near real-time [4]. Centralized Cloud-Computing infrastructures can hardly meet these requirements, mainly due to network congestion and high latency [5].

Edge-Computing consists of a set paradigms which have emerged to face the aforementioned challenges by placing computing resources near the data sources, in order to reduce the network latency and congestion generated by the offload of tasks of end devices. The most relevant Edge-Computing approaches include: a) Fog Computing; and (b) Cloudlet Computing. It should be borne in mind that Cloudlet Computing does not relate the concept of Cloudlet employed in some cloud simulators, e.g., Cloudsim. The latter refers to a task to be executed by a computing resource, whilst the former denotes each of the clusters located near the data sources, which are present in Cloudlet-Computing environments.

The goal of Fog computing is the minimisation of the latency by offloading latency-sensitive workload in edge nodes, including servers, routers and switches within the LAN, where the resulting data is temporarily stored as well. Once these latency-sensitive tasks are successfully finished, compute-intensive and offline workloads, as well as permanent data storage, are to be processed by centralized Cloud-Computing infrastructures.

Cloudlet Computing provides an alternative to Fog Computing by placing small computing clusters with wide bandwidth close to the data sources [6]. The formal definition of Cloudlet Computing is given by the National Institute of Standards and Technology in [7]. Cloudlet-Computing systems may be divided in three layers, as shown in Fig. 1:

1. the **Device Layer** is composed of data sources, such as physical sensors and mobile devices.
2. the **Cloudlet Layer** contains the network of Cloudlets that provide Virtual Machines to users. Each Cloudlet may contain the Virtual Device Representation (VDR), Virtual Service Representation (VSR), and local services.
3. the **Internet Layer** refers to the centralized Cloud-Computing infrastructure.

The aforementioned architecture differs from the simplicity of centralized Cloud-Computing architectures. Therefore, Cloud-Computing simulators, such as CloudSim [8], CloudSched [9], GreenCloud [10], and SCORE [11] present numerous limitations to simulate the complex, layered and inter-connected architectures of Cloudlet-Computing environments.

Edge-Computing simulators are essential tools for the development of distributed large-scale Edge-Computing solutions. When it comes to the design and optimisation of such systems we face various problems, from choosing the proper power policies to the configuration of the infrastructure and networking. Therefore, the proposed simulator must enable researchers to model and evaluate Edge-Computing use cases through Cloudlet-Computing infrastructures.

In this work, we present the following contributions:

- A novel open-source Cloudlet-Computing simulation tool called *Sphere* based on the SCORE simulator [11], which provides a framework for the simulation of Cloudlet Computing environments out of the box.
- An extensible toolkit to implement: (a) Edge Graph modelling and resolution models; (b) centralized, Distributed and Hybrid Cloudlet orchestration models; (c) Geographically-distributed and heterogeneous workload generators; (d)Inter-Cloudlet energy-efficiency models based on the shut-down of idle clusters; (e) Cluster resource-managing models; (f) Cluster energy-efficiency policies; g() An example of usage of the simulation tool and the analysis of its performance in a typical Cloudlet-Computing

Table 1
Fog and Edge simulators comparison, modeled attributes, data centers are denoted as DC.

FogNetSim+	iFogSim	FogTorchII	EgdeCloudSim	IOTSim	EmuFog	Fogbed
Target system:						
Fog	Fog	Fog	Edge (IoT)	Edge(IoT)	Fog	Fog
Infrastructure:						
Distributed DC	Cloud DC	Latency	Cloud DC	Cloud DC	Network links	Virtual nodes
Sensors	Sensors	Bandwidth	Network links	Latency	fog nodes	Switches
Fog nodes	Actuators		WLAN/LAN delay	Bandwidth	Routers	Instance API
Broker	Fog devices		Bandwidth			Network links
Network links	Network links					
Delay	Delay					
Handovers	Network					
Bandwidth	Energy cons.n					
Application:						
Fog network	data stream	Fog applic.s	Mobile Edge	IoT	Fog	Fog network
Bandwidth	Energy cons.					
Consumption:						
RAM/CPU	Power	RAM/storage	RAM/CPU	RAM/CPU/storage	Workload	RAM/CPU
	Allocation policies		Failure (mobility)			Bandwidth
	s					Workload
Mobility:						
Yes	No	No	Yes	No	No	No
Scalability:						
Yes	No	No	No	Yes (MapReduce)	No	No

scenario; and (h) A deep analysis on the performance of the proposed simulation tool.

The rest of the paper is organized as follows: Section 2 presents a comparison of the available Edge-Computing simulation tools. In Section 3, the architecture of *Sphere* is described. Experiment parametrization and results are described in Section 4. Section 5 presents an example simulation scenario and the results provided. Sphere performance is shown in Section 6. We finally summarize the paper, state the conclusions and discuss the future work in Section 7.

2. Related work

The high complexity of Edge-Computing systems and the high costs of deployment of such infrastructures make simulation the main approach to develop, evaluate and analyze the behaviour of several aspects of these environments, including: (a) Cloudlet management and orchestration; (b) Workload distribution; (c) Scheduling strategies; (d) Consolidation and migration models; (e) Networking; and (f) Efficiency policies;.

In this section, we evaluate the most relevant Edge-Computing simulators for the implementation and evaluation of orchestration, resource managing, scheduling, and energy-efficiency techniques. A summary of the performed comparison is shown in Table 1

iFogSim [12] is a Fog-computing simulator focused on the measurement of workload and scheduling performance, network utilization and energy consumption of centralized clusters, edge devices, sensors, network links, data streams, and stream-processing applications, supporting also data dependency between applications and edges. Such applications may be deployed only in centralized data centers or in edge clusters. Nevertheless, as an extension of CloudSim [8], the scalability of iFogSim is limited, and the performance when large scenarios are considered is poor [13,14].

Myifogsim [15] is an extension of iFogSim that includes mobility support through migration of virtual machines between Cloudlets. Similarly, in [16], Naas et al. propose an extension of iFogSim that implements data-placement strategies in Fog and IoT scenarios.

IOTSim is an extension of CloudSim which includes storage and Big-Data-processing layers. In the storage layer, the network and storage delays are simulated for IoT applications. IOTSim is limited to the simulation of MapReduce workloads.

EdgeCloudSim extends CloudSim to provide a modular architecture that supports network models for WLAN and WAN, device mobility models, and simple workload generators. **PFogSim** [17] extends EdgeCloudSim to include different network, application, and orchestration models.

FogTorchII [18] is an extension of FogTorch [19], which is a Fog-Computing simulation tool. FogTorchII utilizes Monte Carlo simulations to implement variations in communication links used as inputs. QoS and resource-consumption results are provided as main outputs. Nonetheless, the scalability of FogTorchII is limited.

FogNetSim++ [20] extends OMNeT++ [21] to enable users to deeply model large fog networks. FogNetSim++ allows the inclusion of mobility models and custom scheduling policies. However, as an extension of OMNet++, it is focused on networking parameters, such as execution delay, packet error rate, handovers, and latency. Therefore, FogNetSim++ presents limitations to incorporate complex orchestration, resource managing, and energy-efficiency strategies. Similarly, **ECSim++** [22] extends the OMNeT++ simulator by employing the INET framework to implement Edge-Computing environments and communications, performance and energy consumption models.

RECAP [23,24] enables the modelling of complex and scenario-specific requirements to find optimal solutions for resource management in Edge-Computing systems. This project involves several simulation tools focused on networking areas, such as: (a) Network Function Virtualization (NFV); (b) Virtual Content Distribution Networks (vCDN); and (c) Smart cities. Notwithstanding, some simulation tools are difficult to manage and configure, and, in general, the simulators cannot be easily extended or modified.

The analysis performed show that the current simulation tools have, in general, the following drawbacks:

1. Poor performance of CloudSim-based solutions when large-scale scenarios are considered;
2. Network-only-oriented frameworks lack of the global and complex orchestration, resource-managing and energy-efficiency models needed to correctly evaluate green Edge-Computing environments;
3. Difficulty to modify or extend the simulators.

In this work, the *Sphere* simulation tool is proposed to overcome the aforementioned limitations. *Sphere* extends the previously developed SCORE and GAME-SCORE simulation tools [11,13], which, in turn, follow the design and simplifications described on the Google Omega simulator [25]. *Sphere* enables the simulation of large-scale Cloudlet-Computing scenarios in a performant way by focusing in resource-managing, scheduling and energy-efficiency policies while keeping the complexity of modelling the scenarios and network details low.

3. Sphere architecture

In this work, we followed the model proposed by Robinson in [26] in order to design the general architecture of the *Sphere* simulator, following this workflow:

1. Definition and initialization of the problem;
2. Determination of the modelling and general objectives;
3. Identification of the model inputs;
4. Identification of the model outputs; and
5. Determination of the model content and level of detail.

Based on this workflow and model, we present the resulting architecture of Sphere in Fig. 2

This architecture has led our software development, which is publicly available at Github: https://github.com/DamianUS/sphere. Sphere follows a hybrid simulation paradigm, merging the discrete event approach and multi-agent models for graph orchestrators and schedulers. The architecture has been divided in four main modules: two for higher-level edge-related responsibilities and two for lower-level cluster-related responsibilities. Based on these responsibilities we have identified the following tiers:

1. The **Edge Simulator** tier, which is the core simulation engine. It is composed by: (a) the edge simulation execution, (b) the *Cloudlet* models and (c) the workload generators. The *Edge simulation* module is in charge of leading the simulation process. It is

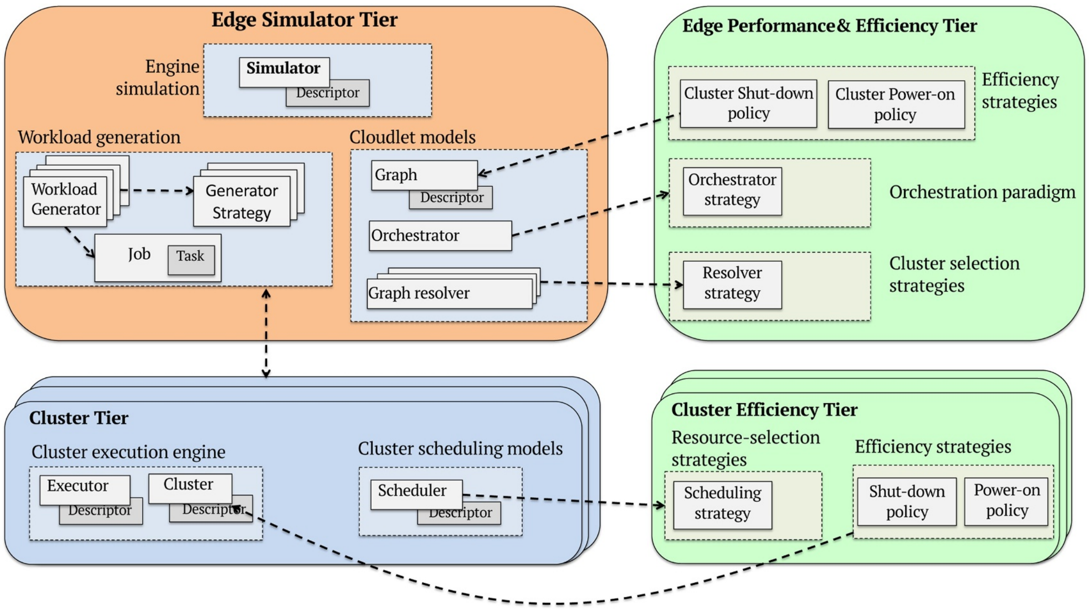

Fig. 2. *Sphere* general Aarchitecture.

responsible for the general set-up of each experiment and its execution as well. This module expects a set of descriptors to configure and run the experiments, as detailed in Section 4.1. *Cloudlet* models are represented by a graph, which determines the structure and connections between each cluster of the Edge infrastructure. Moreover, it includes a *Graph resolver* that determines the best path in the graph at a given time. We also have included an *Orchestrator*, which is responsible for the coordination between clusters in the edge infrastructure. Finally the *Workload-generation* module is in charge of the creation of jobs and tasks, and may follow various models. The Cloudlet-models module, including graph design, graph resolving and orchestration models, are explained in detail in Sections 3.1 and 3.2, respectively. Similarly, the *Workload-generation* module is described in depth in Section 3.3.

2. The **Edge-Performance and Efficiency** tier, which manages the performance and efficiency strategies at the higher Edge level and includes the policies for the shutting-down and powering-on of Edge clusters, as well as the definition of the graph-resolution strategy. These modules are described in detail in Section 3.4.
3. The **Cluster Simulation** tier handles the actual simulation engine, as well as the execution of the incoming workloads. As the Edge-infrastructure is composed by numerous clusters, this tier will be instantiated once per cluster, enabling each cluster to have a particular behaviour by applying different components, such as: resource manager, scheduling policy, and efficiency and performance policies at cluster level. Within this tier, we have identified two modules: The *Cluster scheduling models* module, which may employ various task-scheduling algorithms; and the main *Cluster execution engine*, which simulates the cluster execution and is responsible for the configuration of the cluster (number of machines and their characteristics, including memory and computing capacity) and for the tracking and update of the cluster status (e.g., the energy state of the machines, such as idle, off, on, and executing).
4. Finally, the **Cluster Efficiency** tier defines the scheduling strategy that makes the actual decision of task deployment. It is also responsible for the selection of the resource-managing model for each cluster. Centralized resource-managing strategies employed in industry commonly fall into the following models: monolithic, two-level and shared-state. More models may be implemented, such as fully-distributed and hybrid alternatives. This tier is also responsible for the modelling of the energy-efficiency strategies to be applied to the servers at a cluster level. This tier is described in Section 3.4.

The benefits of this architecture may be summarized as follows:

i Allows researchers to evaluate various orchestration models and their implications. These orchestration models may include centralized models, such as monolithic, two-level, shared state, as well as fully-distributed and hybrid approaches.
ii Efficiency and performance strategies can be mixed at two levels: the edge-infrastructure level and cluster level in an independent way.
iii Workload generation may follow the diverse patterns present in Edge-Computing scenarios.
iv The modules related to the graph modeller and resolver enable other researchers to implement various strategies to determine best paths for a cluster to reach its neighbors.
v Each cluster acts in an independent manner, including its own cluster configuration, resource-managing, scheduling and efficiency models.

3.1. Graph model

In this section we formalize the model employed by *Sphere* to represent the structure of the network of clusters as a weighted directed graph. Such model enables the application of graph-based algorithms to achieve the objectives, both in terms of performance and energy efficiency in Edge-Computing environments. Here we focus our attention on the ‘Graph modelling’ component inside the *Cloudlet models* part of the *Edge Simulator Tier* illustrated in Fig. 2.

Formally, our simulator models the Edge-Computing network as a directed weighted graph $G = (V, E)$, where V is the set of nodes which represents each cluster in the network and the data sources (sensors, user devices, etc.); and E denotes the set of edges representing the links between the clusters. This representation is shown in Fig. 3. We assume that we have an underlying network where all clusters are connected between them, directly or indirectly, so we are interested in a graph using just a subset of the edges: the direct ones, the ones that connect the clusters faster.

Given a node (cluster), the graph definition is completed by stating its attributes: the *load* of the cluster, generated according to a chosen probabilistic function:

- **Uniformly distributed**: for jobs that are arriving at uniform rate, of a uniform size and consuming the same amount of resources.
- **Exponential distribution**: the arrival time between jobs, the number of tasks of a job, and the duration of the tasks withing a job are sampled from exponential distributions.
- **Exponential built from a trace file**: both the duration and the number of tasks of all jobs are sampled from exponential distributions built from a trace file.
- **Empirical distribution**: generates workloads that reproduce the behaviour found in a given trace file.

Each cluster may be in one of the following four statuses: SWITCHED ON, SWITCHED OFF, POWERING-ON and SHUTTING-DOWN. Regarding the edges between two given nodes i and j, from the C available clusters, $i, j \in 1, 2, \ldots C$, we are interested in the *bandwidth* $b_{i,j}$, the *latency* $l_{i,j}$ and the *congestion* $c_{i,j}$ of the link between node i and j, which define the status of the network.

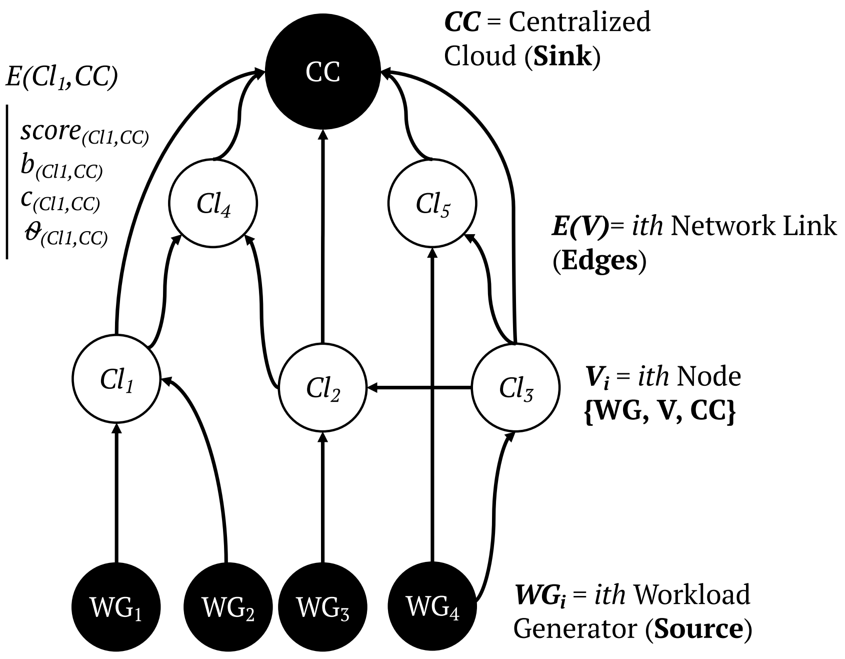
CC = Centralized Cloud (Sink)
E(Cl1,CC)
score(Cl1,CC)
b(Cl1,CC)
c(Cl1,CC)
θ(Cl1,CC)
CC
Cl4
Cl5
E(V)= ith Network Link (Edges)
Cl1
Cl2
Cl3
Vi = ith Node {WG, V, CC}
WG1
WG2
WG3
WG4
WGi = ith Workload Generator (Source)

Fig. 3. Example of a graph model: nodes representing clusters, and edges representing the connectivity between them, in terms of bandwidth, latency, and network congestion. The score of a network link is a metric that takes into account the network features together with the load of the destination clusters.

Therefore, the *score* $score_{i,j}$ indicates the availability of the network link between i and j. It is used as a metric that includes network related features (congestion, latency, etc.) as well as some cluster metrics (like the load of the destination cluster θ_j) that may affect the data transmission performance. The score of a network link between nodes i and j is computed as follows:

$$score_{i,j}(n, k) = \omega_b * b_{i,j}(n, k) + \omega_l * l_{i,j}(n, k) + \omega_c * c_{i,j}(n, k) + \omega_\theta * \theta_j(n, k) \quad (1)$$

where $n = 1, 2, ..., N_k$ is the edge from i to j for each time step $k = 1, 2, ..., K$. The weights ω_b, ω_l, ω_c, ω_θ may take values from the interval $(-\infty, +\infty)$. Such a definition allows the modelling of bidirectional cluster links, but the score in each direction may be different due to different cluster loads.

Sphere provides the ability to dynamically re-compute the attributes of the elements in the graph while the simulation is running and, therefore, these attributes are changing. Together with the graph model, *Sphere* includes a specific component that, given a graph model representing the network, applies different graph-based algorithms to obtain useful insights from it in terms of scheduling and energy efficiency. Focusing on this last aspect, *Sphere* provides different graph-based efficiency strategies in the 'Edge performance and Efficiency Tier':

- Energy Efficiency as a max-flow min-cut problem [27]: this well-known algorithm allows *Sphere* to obtain a list of candidate clusters to be switched-off by computing the set of edges with minimum sum of their weights, or the maximum flow of the network (see Fig. 4). In this case we have two options for the selection of candidates: we may consider those clusters with outgoing edges affected by the 'cut', or those that do not receive incoming links once the edges in the 'cut' are removed from the graph. This algorithm has been proven useful to face scheduling problems [28].
- Energy Efficiency as a shortest path problem [29]: by computing the shortest path from each data source to the main cluster, we can switch-off those clusters not present in any of those paths.
- A variation of the previous point consists in the computation of the *closeness* or the *betweenness* of each node in the graph, being candidates to be switched-off those with a lower value of the metric.

In terms of the performance of the Edge-computing network, *Sphere* can take advantage of the well-known PageRank algorithm [30] and its variants, by computing a personalized PageRank of nodes according to their loads and the capacities of the edges in the network, in order to select the best nodes to send the data to (see Fig. 5). Through the 'graph descriptor' of *Sphere*, it is possible to define node attributes intended to store the PageRank score or any other centrality measure that can be used in the 'resolver strategy' of the 'Edge Performance and Efficiency Tier'.

Finally, the modular architecture of the simulator enables the definition of different graph topologies through the implementation of various graph models. In this way, we can perform experiments with the above discussed graph structure or any other possible graph representation included as a graph 'descriptor' (see Fig. 2). In addition, it is possible to specify the equations that calculate the weights of nodes and edges in terms of the aforementioned attributes, as follows:

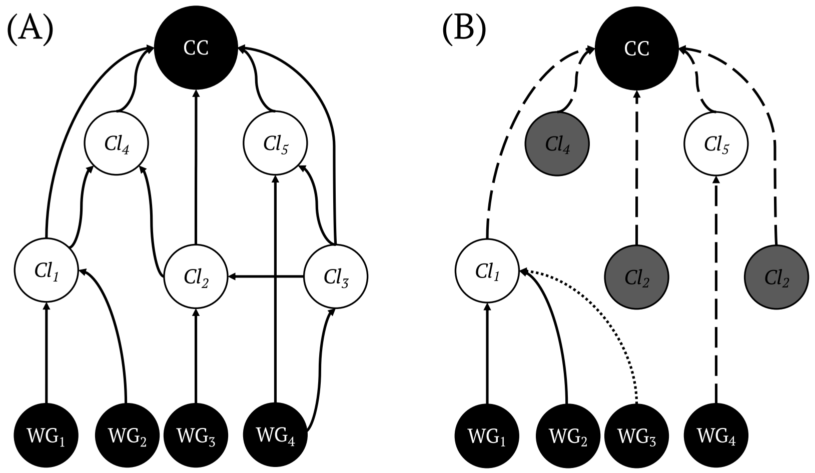

Fig. 4. Example of min cut (B) on a *Sphere* graph model (A): clusters with no incoming edges in the cut (slashed lines) can be considered as candidates to be switched off. The grey dotted lines represent alternative paths used by data sources to communicate with the working clusters.

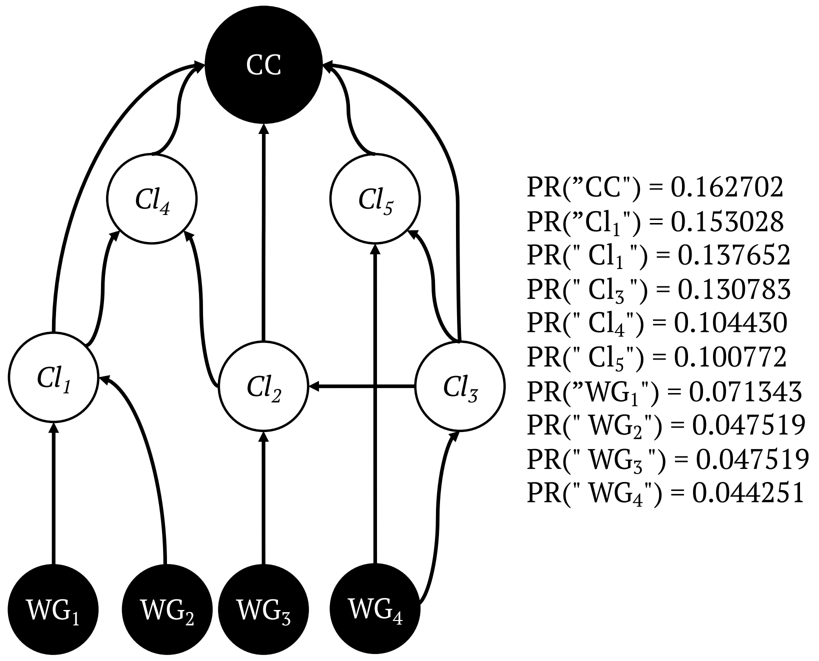

Fig. 5. Example of a PageRank computation in order to obtain the best clusters to send the data to.

$$relevance_i^{apriori} = F(load_i,\ status_i,\ \ldots) \tag{2}$$

$$weight_{ij} = W(b_{ij},\ l_{ij},\ c_{ij},\ load_j,\ status_j,\ \ldots) \tag{3}$$

where F: $\mathbb{R}^* \to \mathbb{R}$ and W: $\mathbb{R}^* \to \mathbb{R}$, are given functions of i and $j \in V$ and $[i, j] \in E$ is the edge from node i to node j.

According to this model, various graph-based algorithms based on max-flow min-cut, shortest-path and Pagerank may be implemented in the simulator.

3.2. Orchestrator model

The orchestration models of the Cloudlets (edge clusters) may follow three general paradigms: centralized, hybrid and distributed orchestration models. When a centralized strategy is employed, all the decisions are coordinated by a central agent, including the business-logic, state of the edge infrastructure (stored as a graph) and the graph resolver.

When a hybrid model is selected, each cluster has the responsibility of deciding where to deploy their dynamically-generated workloads. Within this approach, *Sphere* can be adapted to run at least these solutions:

1. Each node keep a copy of the graph, so the orchestrator is in charge of updating their information periodically. In some cases this solution can lead to nodes having an outdated copy of the graph state, which may result in sub-optimal decisions, or even in deploying tasks on shut-down nodes. In this case, the node is forced to query the orchestrator for a new fresh copy of the graph and to make a new scheduling decision.
2. There is a centralized copy of the graph maintained by the orchestrator. In this case, each node queries a centralized copy of the graph to the orchestrator, and makes a scheduling decision based on that copy. In this case, the orchestrator has to wait for the decision of this node before offering the graph to other nodes. Thus, the graph is locked while a node is making decisions. This approach prevents nodes from making decisions over an outdated copy of the graph state, but may cause bottlenecks if the frequency of decisions is very high or the connection between the nodes and the orchestrator has high latency or is congested.

Both, centralized and hybrid approaches, need a specific node, usually the cloud node, to run the orchestration logic.

The advantages of the centralized model includes the easiness of the implementation and deployment, as well as the decrease of network traffic between clusters and the centralized agent that performs the orchestration logic. The advantages of hybrid approaches include the possibility of each cluster running a different graph resolver strategy that could suit better to the particular cluster requirements, as well as the minimization of bottlenecks due to the execution of the graph-resolving algorithm.

Sphere currently offers centralized orchestration, but it supports the implementation of hybrid approaches and fully distributed approaches, which are still in development.

3.3. Workload model

In this paper, we extend the workload model presented in SCORE in order to include the workload characteristics and generation patterns present in Internet of Things and Edge-Computing paradigms. Therefore, the atomic work unit to be executed in a particular machine is called Task, $Task_n$, which compose Jobs, Job_m, which, in turn, compose Workloads $Workload_p$:

$$Workload_p = [Job_1^p, Job_2^p, \ldots, Job_{J^p}^p] \tag{4}$$

for $p = 1, 2, ., P$ and

$$Job_m^p = [Task_1^{p,m}, Task_2^{p,m}, \ldots, Task_{N^{p,m}}^{p,m}] \tag{5}$$

for $m = 1, 2, \ldots, M$ where $P, N^{p,m}, J^p, M$ denotes the number of elements concatenated in each vector. Each workload belongs to one of the following categories:

- **Application type**. There are two main groups of application types:
  a) **Batch jobs**, which perform a given set of instructions and then finish. MapReduce jobs are an example of a *Batch* job; and
  b) **Service jobs**, which represent long-running jobs with no determined end. Long-running services, such as databases, streaming engines, and web servers are representative Service jobs. The duration/makespan of Service jobs are not usually affected by bandwidth and computation limitations, since they are usually killed by edge infrastructure operators.
- **Resource usage**. We consider two kinds of jobs according to which resources are under pressure when they are deployed:
  a) **Data-intensive jobs**. These jobs are composed of tasks which lightly process large amounts of data. The tasks of these jobs require large amounts of bandwidth and data, but low computing capacity.
  b) **Compute-intensive jobs**. These jobs are composed of tasks which perform complex computations. Therefore, the tasks require high computing capacity, while they do not require long data transmissions.
- **Deployment requirements**. Two types of jobs are considered in this work depending on where jobs can be deployed:
  a) **Cloud-only jobs**. These jobs has permanent storage or computing requirements. Only centralized Cloud-Computing infrastructures can process such jobs.
  b) **Flexible jobs**. These jobs may be executed by any node, since no special requirements apply to them.
  . It should be borne in mind that the completion time of the tasks may be negatively affected by network/computing limitations.

The total workload:

$$[Workload_1, Workload_2, \ldots, Workload_P] \tag{6}$$

to be processed by the Graph $G = (V, E)$, is generated by a set of Workload Generators WG_g, $g \in \{1, 2 \ldots G\}$. Each Workload Generator WG_g usually represents a geographic area where workload patterns and users are similar. Each Workload Generator employs a finite number of Generation strategies $s_g^1, s_g^2, \ldots, s_g^{S^g}$, such as statistical distributions, to simulate the arrival of users.

In this work, we implement two main types of workload generation strategies:

- **Static strategies**: These generation strategies usually employ statistical distributions, such as Exponential and Weibull distributions to generate all the Jobs to be processed during the simulation time, as well as their attributes. Therefore, static strategies have no direct impact on the runtime environment and are usually used to represent the workload generated by final users.
- **Dynamic strategies**: These generation strategies take into account the processed workload in a particular window time and the

whole graph state in order to generate new jobs. Cloudlets (edge clusters) employ these strategies to generate aggregation workloads, which must be stored and processed in centralized Cloud infrastructures.

The following job attributes are taken into account in this work for all kind of workloads:

- Geographical position, the "latitude" ϕ and The "longitude" λ which may fall within the Graph and workload generator borders;
- Inter-arrival time, t_n, representing the time elapsed between two consecutive jobs: job number n and job number $n + 1$;
- Number of tasks $N^{p,m}$ of the job m in the workload p;
- Tasks duration $d([Task_1^{p,m}, Task_2^{p,m}, ..., Task_{N^{p,m}}^{p,m}])$ for this job, which includes a makespan-degradation model due to poor network/computing performance. We implemented new module responsible for the computation of the tasks duration considering the graph $G = (V, E)$, and the cluster state (representing the state of the servers), for each cluster. Additionally, the task initial duration $d_0([Task_1^{p,m}, Task_2^{p,m}, ..., Task_{N^{p,m}}^{p,m}])$ has been added;
- CPU, memory, and bandwidth requirements of each task.

.

In this work, we follow the Edge Computing and Internet of Things literature trends presented in [31], where IDC estimates that approximately 40% of the data will be analyzed on edge nodes and clusters (cloudlets). Synthetic workloads following these industry trends are generated for each simulation run. Deeper explanations on the synthetic workload generation patterns can be found in [32].

3.4. Energy and performance-awareness model

Sphere has been designed to be aware of energy consumption and cluster performance from its roots. The general architecture includes two different tiers to achieve this goal: a general Edge performance and efficiency tier and a cluster tier that includes efficiency and performance strategies at cluster level. This higher level Edge tier makes two main decisions:

(a) to shut down or to power on an Edge cluster (b) graph resolution, which determines the best Edge cluster for a job to be deployed.

The former decisions are performed taking into account the efficiency strategies and may affect performance as well. Efficiency strategies may be: (a) Deterministic strategies, such as *Always switch off* if possible; and (b) Probabilistic strategies, which employ statistical distributions. It is important to notice that a cluster can not be shut down if some tasks are currently deployed on its resources.

Cluster candidates to be switched off can be selected by executing one of the aforementioned graph-based algorithms listed in Section 3.1.

The latter decisions made by the *Graph resolver* have an important impact on the workload that a cluster must process at a given moment. Graph-resolving strategies may range from fully occupying clusters (so other clusters may be left idle to shut them down) to more performant and less energy-efficient solutions, where the workloads are spread among all nodes.

The lower-level Cluster-Efficiency tier is in charge of determining whether a machine has to be shut down, left in idle state, or powered on. These decisions are made by the efficiency strategies at cluster level. Similarly to Edge-level decisions, these choices will affect the internal cluster performance: typically the lower the number of available resources, the worse the performance achieved. Finally, it is important to notice that other decisions may also impact on the general efficiency and performance of a cluster, such as those made by the resource-managing and scheduling strategies.

The computation of the energy consumed by the edge infrastructure is the result of the sum of the energy consumed by each cluster. Similarly, the energy consumed by a cluster is computed based on the energy consumed by its servers, which, in turn, depends on the resource energy states: ON (executing a task), IDLE (waiting for a task to be deployed), SWITCHED-OFF, POWERING-ON, and SHUTTING-DOWN.

4. Experimentation inputs and outputs

4.1. Experiment design parameters

In Appendix A, the key parameters of the simulator used in experiments are presented.

4.2. Experimentation output results

The results provided by *Sphere* extend those provided by SCORE. Appendix B presents the key performance indicators for each workload. In the same way, Appendix C presents the key performance parameters for each computing cluster.

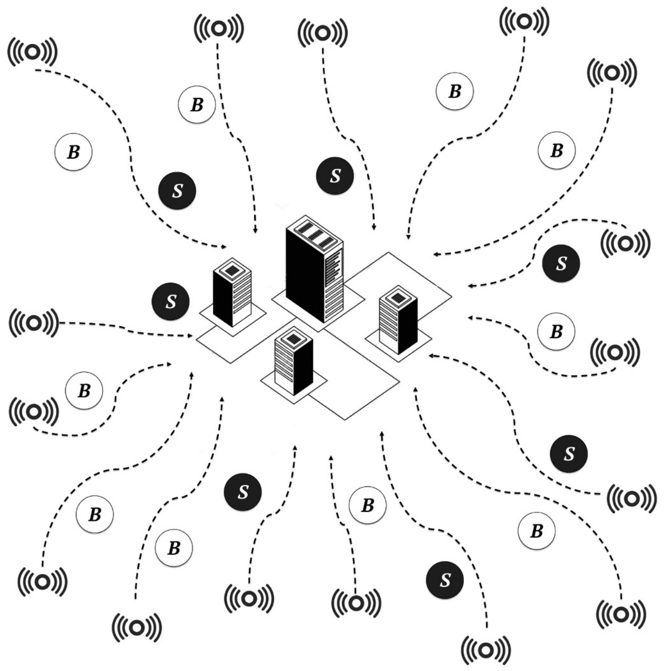

Fig. 6. Cloud-only environment designed to evaluate the limitations of centralized architectures when Internet of Things workloads are under consideration. In this scenario, multiple sensors generate data-intensive and compute-intensive jobs, which are deployed to a single 500-machines data center, which may impose poor network performance.

5. Experimentation example

5.1. Experimentation environment

In this Section, we propose an example consisting on the following four simple Internet of Things scenarios where the proposed simulation tool can be analyzed and evaluated, and the results compared to those provided by centralized Cloud-Computing infrastructures.

1. **Cloud-only scenario**. In this use case, the workload is processed by a centralized Cloud-Computing data center composed of 500 high-computing-capacity servers, as shown in Fig. 6. 4 workload generators, one for each quarter of the area, equipped with static strategies based on Exponential distributions are employed in this scenario. This use case is modelled to show the limitations of centralized environments.
2. **Cloudlets-Low Computing power scenario**. The workload is processed by 4 Cloudlets composed of 500 low-computing nodes each, such as Single-Board-Computers (SBCs) or similar, as shown in Fig. 7. This scenario is modelled to present the limitations in terms of computing power of isolated Cloudlets. 4 workload generators, one for each quarter of the area, equipped with static strategies based on Exponential distributions are employed in this scenario.
3. **Collaborative High-Performance Cloudlets + centralized cloud scenario**. This environment is comprised of 4 Cloudlets, which are composed of 60 Cloud servers each, and the aforementioned centralized Cloud data center. The mix of data-intensive and computing-intensive workloads is generated following current trends in order to be distributed between Cloudlets ($\sim$ 40% of workload) and the centralized Cloud ($\sim$ 60% of workload). 8 workload generators are employed in this scenario: four of them equipped with static strategies based on Exponential distributions - one for each quarter of the area-, and four additional workload generators - one for each Cloudlet -, which equip dynamic strategies to generate approximately 10% of the processed workload and send it to the centralized cloud.
4. **Collaborative Low-Performance Cloudlets + centralized cloud scenario**. This environment replaces each High-Performance Cloudlet composed of 60 Cloud Servers used in Scenario 3 by a Low-Performance Cloudlet composed of 500 Low-Computing-Capacity nodes, such as those presented in Scenario 2. 8 workload generators are employed in this scenario: four of them equipped with static strategies based on Exponential distributions - one for each quarter of the area-, and four additional workload generators, one for each Cloudlet, which equip dynamic strategies to generate approximately 10% of the processed workload and send it to the centralized cloud. Fig. 8 presents this collaborative and low-cost scenario.

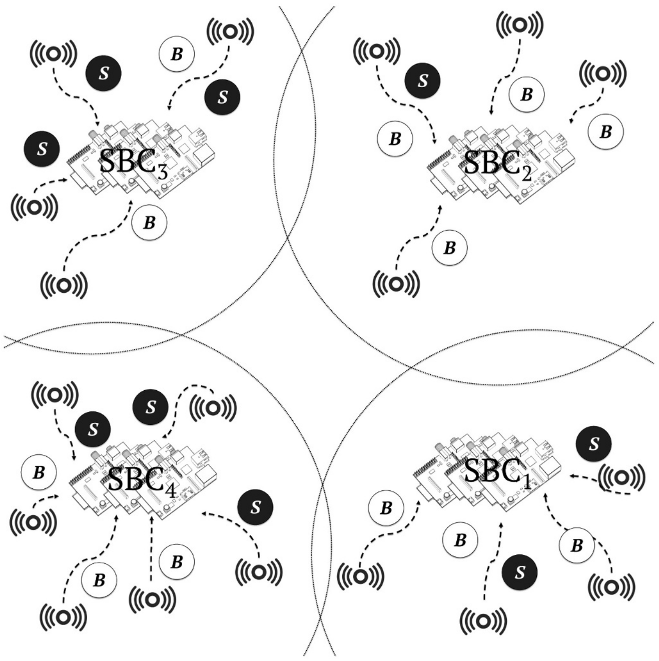

Fig. 7. Cloudlets-Low Computing power environment designed to analyze the computing limitations of low-computing isolated Clouds when compute-intensive Internet of Things workloads are under consideration. In this scenario, multiple sensors generate data-intensive and compute-intensive workloads which are distributed among the Cloudlets uniformly according to the location. Each of the 4 cloudlets is composed of 500 low-computing-capacity nodes, such as SBCs.

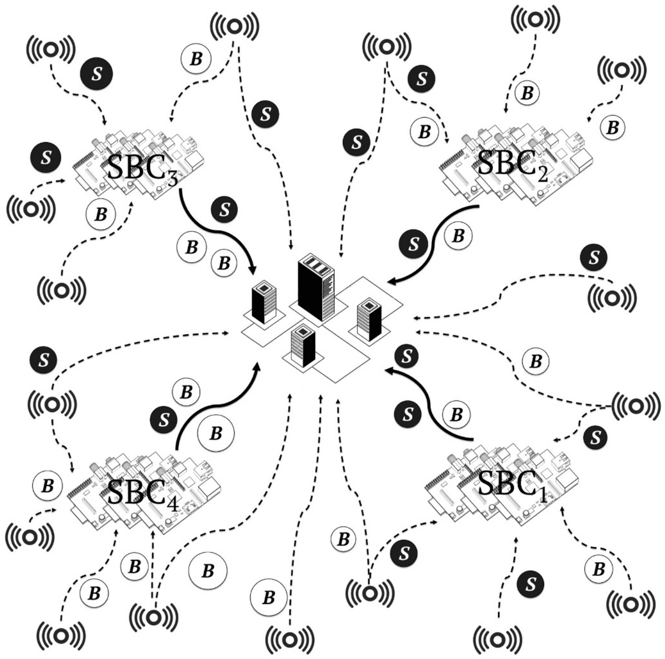

Fig. 8. Collaborative Cloud-Cloudlet environment. Data-intensive and computing-intensive workloads are generated following current trends in order to be distributed between Cloudlets (~ 40% of workload) and the centralized Cloud (~ 60% of workload). First, the real-time workloads are processed by nearby Cloudlets and then, more intensive computational jobs and the data to be permanently stored are sent to the centralized Cloud infrastructure. In this scenario, the workloads, which are uniformly distributed based on geographical position, may suffer from the low computing capacity of nearby Cloudlets and from poor network performance of the centralized data center.

Table 2
Performance, energy consumption and costs results for each scenario for data-intensive workloads. t_j^{net} represents the network delay, while t_j^{first} and t_j^{full} represent the time a job spends in queue until its first task and its last task is scheduled, respectively. B means Batch jobs while S means Service jobs.

Scenario	t_j^{net}	CPU	$t_j^{first}(s)$		$t_j^{full}(s)$		Makespan (s)		MWh
	(ms)	occ. (%)	B	S	B	S	B	S	cons.
Isolated									
Cloud	50	20.80	5.70	5.56	5.70	5.56	99.21	2,003.19	9.73
Cloud	150	21.00	5.61	5.33	5.61	5.33	114.66	1,985.85	9.75
Cloud	300	21.11	6.26	5.33	6.26	5.33	142.27	1,980.65	9.76
HE-Cloudlets	10	22.46	0.91	0.77	3.21	2.46	88.24	2,058.18	4.77
SBC-Cloudlets	10	18.26	0.87	0.77	1.85	1.46	95.90	1,996.77	1.03
Collaborative									
Cloud +	50	19.18	1.71	1.38	2.10	1.82	90.62	1,979.64	14.47
HE-Cloudlets	10	17.52							
Cloud +	150	19.22	1.72	1.39	2.11	1.59	90.62	1,979.64	14.48
HE-Cloudlets	10	17.70							
Cloud +	300	19.18	1.73	1.41	2.12	1.60	90.62	1,979.64	14.48
HE-Cloudlets	10	18.45							
Cloud +	50	19.24	1.69	1.46	1.74	1.95	93.62	2,019.96	10.72
SBC-Cloudlets	10	14.11							
Cloud +	150	19.19	1.70	1.47	1.75	1.72	93.62	2,019.96	10.72
SBC-Cloudlets	10	14.68							
Cloud +	300	19.23	1.72	1.49	1.76	1.74	93.62	2,019.96	10.73
SBC-Cloudlets	10	15.13							

5.2. Experimentation example results

We simulated each simulation scenario with the following realistic parameters:

- A hybrid orchestration model and a Page-Rank-based graph resolver are employed.
- The time the scheduling algorithm needs to initially schedule a job, t_{sch}^{j}, is set to 150ms and the time required by the scheduling algorithm to make scheduling decisions for each task, $\sum_{c=1}^{m} t_{sch}^{cj}$, is set to 100ms. Network latency, t_{net}^{j}, is set to 10ms for Cloudlets, while a range of values was considered for this parameter in Cloud facilities.
- Tasks number and duration were set equally for both experiment sets, while the processing power of SBCs is set to 20% of that of Cloud servers.
- The power consumed by Cloud machines when executing tasks, P_{busy}^{i}, was set to 110W, and the power consumed in idle state, P_{idle}^{i}, to 50 W per Cloud CPU respectively. P_{busy}^{i} was set to 15W and P_{idle}^{i} to 8W per SBC.

More than 6500 Batch and 600 Service jobs were deployed during the seven-days operation period for each simulation experiment. Both Cloud and Cloudlet clusters employed a centralized monolithic scheduler and started with 20% of resource utilization, according to industry patterns [33].

Each simulation has been run five times during 7 days of operation time, and an ANOVA test with trust level $p < .05$ was performed. The average values for all KPIs are shown in Tables 2 and 3 for data and compute-intensive environments, respectively.

These results represent a good example where the benefits and limitations of Edge Computing and Cloudlet architectures are clear. In general, when the workload considered are composed of a vast majority of data-intensive and not-compute-intensive jobs, a Cloudlet-only architecture may perform sufficiently well. On the other hand, centralized architectures show clear signals of network and even scheduling congestion, which may lead to poor overall performance.

6. Sphere simulation performance

In this Section, we perform a set of simulations to show the performance and resource consumption of the developed simulation tool.

The performance of Sphere depends mainly on the following parameters:

- **Graph size**, representing the number of clusters which compose the Edge-Computing infrastructure. The graph size has a direct impact on the graph-resolver performance and memory consumption.
- **Servers per cluster**. The higher the number of servers in each cluster, the less performant the intra-cluster scheduler and more memory is consumed.
- **Number of jobs**. Large-scale scenarios execute thousands or million of tasks per day. Each job and their tasks need to be

Table 3
Performance, energy consumption and costs results for each scenario for compute-intensive workloads. t_j^{net} represents the network delay, while t_j^{first} and t_j^{full} represent the time a job spends in queue until its first task and its last task is scheduled, respectively. B means Batch jobs while S means Service jobs. To compute acquisition costs, we took average Xeon server and Raspberry Pi 3 prices, 2500 and 35 dollars, respectively.

Scenario	t_j^{net}	**CPU**	$t_j^{first}(s)$		$t_j^{full}(s)$		**Makespan (s)**		**MWh**
	(ms)	**occ. (%)**	**B**	**S**	**B**	**S**	**B**	**S**	**cons.**
Isolated									
Cloud	50	20.82	5.19	5.18	5.19	5.18	98.61	2,008,41	9.72
Cloud	150	20.82	5.23	5.22	5.23	5.22	98.61	2,008.41	9.72
Cloud	300	20.82	5.29	5.28	5.29	5.28	98.60	2,008.41	9.72
HE-Cloudlets	10	22.32	0.88	0.85	2.96	3.32	91.56	2,051.33	4.73
SBC-Cloudlets	10	34.37	3.02	3.23	62.05	69.94	575.97	2,066.43	1.67
Collaborative									
Cloud +	50	19.21	1.75	1.42	2.09	1.64	90.62	1,979.64	14.47
HE-Cloudlets	10	17.42							
Cloud +	150	19.27	1.76	1.43	2.10	1.65	90.62	1,979.64	14.48
HE-Cloudlets	10	17.65							
Cloud +	300	19.28	1.78	1.44	2.11	1.67	90.62	1,979.64	14.48
HE-Cloudlets	10	18.07							
Cloud +	50	19.24	1.81	1.43	6.41	6.37	241.46	2,002.00	10.84
SBC-Cloudlets	10	21.32							
Cloud +	150	19.29	1.82	1.44	6.42	6.38	241.45	2,002.00	10.84
SBC-Cloudlets	10	21.34							
Cloud +	300	19.34	1.84	1.46	6.44	6.39	241.46	2,002.00	10.85
SBC-Cloudlets	10	21.42							

scheduled and deployed. It should be borne in mind that, in this Section, we consider each job to be composed, in average, of 165 tasks.

- **Power-off policy**. Every time a task is finished, a power-off decision is made on whether the machine that was executing that particular task should be shut-down or not. Complex shut-down decisions may lead to severe performance impact.

The following parametrization is employed as representative: (a) 4, 16, and 32 clusters compose the Edge-Computing infrastructure; (b) each cluster is composed of 100 and 500 homogeneous servers; (c) 10,000, 50,000, and 100,000 jobs, composed of ~ 1,500,000, ~ 7,500,000, and ~ 15,000,000 tasks, respectively, as the incoming workload; and (d) three representative power-off policies: never shut down any machine, always shut down all the machines if possible, and a shut-down policy based on the Gamma distribution. Each experiment is run five times in a 2012 iMac, equipped with an 4-cores Intel Core i5 at 2,7 GHz (Turbo Boost up to 3,2 GHz), with 6 MB of level 3 cache, 16GB of RAM @ 1.600 MHz and an SSD. Only one CPU thread is made available for the simulation process.

Table 4 presents the average of the performance results of the simulation tool and its memory consumption for the set of representative scenarios.

It can be noticed that the number of jobs increases proportionally the simulation time, but has a minor impact in memory consumption. Furthermore, complex shut-down policies, such as the Gamma policy, which makes shut-down decisions based on the Gamma distribution, have a notable impact both in terms of simulation speed and memory consumption. The number of servers per cluster has a lesser impact than the aforementioned parameters.

In general, the memory consumption is stable, and even scenarios consisting of million of tasks (more than 15,000,000 tasks for 100,000 jobs) can be executed in a short period of time (~ 3 min) due to the simplifications of the simulation model employed.

7. Conclusions and future work

In this paper, we presented a novel Edge-Computing simulator as an extension of the SCORE simulation tool: a performant simulator focused on the evaluation of resource-managing, scheduling and energy-efficiency models in large-scale data centers. The architecture of SCORE has been extended with the addition of the following modules: (a) The creation of graph-based Cloudlet networks; (b) The creation of an easily extensible orchestration module, including out-of-the box orchestrators based on the implemented cluster resource-managing models; (c) The creation of an easily extensible graph-resolution module, including out-of-the box resolvers, such as the PageRank variation; (d) The extension of the workload-generation module to support dynamic strategies and multiple parallel workload generators; (e) The extension of the workload characteristics to include geographic position, resource-intensity and deployment requirements; and (f) The development of makespan-degradation models according to the cluster computing capacity and network performance.

This simulation tool can overcome the limitations of CloudSim-based simulation tools and broaden the use cases of networking-focused simulation tools based on OmNet+ + thanks to the simple model used for networking and individual tasks.

Notwithstanding, Sphere lacks of important features which are to be developed in the future, such as:

Table 4
Sphere simulation performance results in terms of simulation time and memory consumption. Graph size denotes the number of clusters composing the Edge-Computing infrastructure. It should be borne in mind that each job is composed of 165 tasks in average.

Graph Size	Servers per cluster	# **Jobs (10ˆ3)**	Power-off policy	RAM (GB)	Time (s)
4	100	10	Never off	2.61	5.29
4	100	50	Never off	2.74	18.07
4	100	100	Never off	2.92	33.37
4	500	10	Never off	2.65	6.93
4	500	50	Never off	2.78	25.93
4	500	100	Never off	2.96	43.96
4	100	10	Always off	2.62	5.16
4	100	50	Always off	2.75	24.83
4	100	100	Always off	2.93	44.33
4	500	10	Always off	2.67	11.05
4	500	50	Always off	2.79	57.73
4	500	100	Always off	2.97	90.91
4	100	10	Gamma	2.76	20.17
4	100	50	Gamma	2.85	61.54
4	100	100	Gamma	3.12	121.61
4	500	10	Gamma	2.82	20.44
4	500	50	Gamma	2.89	96.14
4	500	100	Gamma	3.17	191.13
16	100	10	Never off	2.82	6.76
16	100	50	Never off	2.96	24.80
16	100	100	Never off	3.15	32.70
16	500	10	Never off	2.86	8.45
16	500	50	Never off	3.00	27.12
16	500	100	Never off	3.20	47.42
16	100	10	Always off	2.83	7.88
16	100	50	Always off	2.97	29.62
16	100	100	Always off	3.16	45.17
16	500	10	Always off	2.88	16.26
16	500	50	Always off	3.01	57.16
16	500	100	Always off	3.21	118.47
16	100	10	Gamma	2.98	27.17
16	100	50	Gamma	3.08	66.36
16	100	100	Gamma	3.37	184.02
16	500	10	Gamma	3.05	99.13
16	500	50	Gamma	3.12	97.45
16	500	100	Gamma	3.42	198.79
32	100	10	Never off	3.08	7.38
32	100	50	Never off	3.23	29.52
32	100	100	Never off	3.45	38.97
32	500	10	Never off	3.13	10.40
32	500	50	Never off	3.28	30.45
32	500	100	Never off	3.49	46.62
32	100	10	Always off	3.09	11.17
32	100	50	Always off	3.25	24.84
32	100	100	Always off	3.46	52.46
32	500	10	Always off	3.15	24.41
32	500	50	Always off	3.29	59.25
32	500	100	Always off	3.50	176.09
32	100	10	Gamma	3.26	38.86
32	100	50	Gamma	3.36	68.41
32	100	100	Gamma	3.68	249.69
32	500	10	Gamma	3.33	24.48
32	500	50	Gamma	3.41	97.79
32	500	100	Gamma	3.74	206.71

- Addition of mobility modes.
- Ease of use and extendability of the implemented code.
- Visual interface to model the Cloudlet Graph and parameters.
- Addition of a real-time visualizer of the power and performance state of the clusters and machines.
- Incorporation of support for dynamic mobility workloads.
- More detailed network and the related performance-impact models.
- Workload migration models.

Acknowledgement

This work has been funded by the Spanish Ministerio de Ciencia, Innovacin y Universidades through the project Energy Efficiency and Performance of Data Centers by Smart Virtualization and Deep Learning Event Detection (RTI2018-098062-A-I00).

Appendix A. Configurable experiment parameters

Table 5
Configurable experiment parameters.

Parameter	Description	Values
Graph	Parameters related to the edge infrastructure that must be fixed for all experiments	
#**Nodes**	Function to compute the relevance or centrality of each node in terms of its load, status, connectivity in the network, etc.	[1–∞)
#**Edges**	Function to compute the capacity or weight of the edge in terms of the network parameters (latency, congestion, and bandwidth), the load, and status of the connected nodes.	[0–∞)
Orchestration models	Orchestration strategies (centralized, distributed)	Array
Graph performance	Parameters related to performance iterated in order to create all experiment variations	
Graph-resolver algorithm time	Time spent by the resolver in order to compute the best candidate for deploying a job. This simulates the performance of the graph resolver algorithm	[0.001–∞)
Resolver algorithms	The resolver algorithms to be run, such as: PageRank variation	Array
Graph energy	Parameters related to energy iterated in order to create all experiment variations	
Node Shut-down policies	The shut-down algorithm to be executed at graph level, such as *max-flow min-cut*	Array
Node Power-on policies	The power-on policies to be run at graph level	Array
Cluster	Parameters related to each node of the graph Each node is independently instantiated	
#**Machines**	Data-center size	[1–∞)
#**Cores**	Number of CPU cores for every machine	[1–∞)
RAM	Amount of RAM in GB for every machine	[0.1–∞)
Heterogeneity	Flag to decide whether node machines are heterogeneous	Boolean
Machine performance profile	Describe the performance of every machine in the node. The lower this value, the more performant the server is	Array, size: number of machines [0.01–∞)
Machine security profile	Describe the security of every machine in the node. The higher this value, the more secure the server is [34]	Array, size: number of machines [1–5]
Machine energy profile	Energy consumption of every machine in the node. The lower this value, the more energy-efficient the server is	Array, size: number of machines [0.01–∞)
Power-on time	Time required to boot a server (seconds)	[0.1–∞)
Shut-down time	Time needed to hibernate a server (seconds)	[0.1–∞)
Cluster Performance	Parameters related to performance iterated in order to create all experiment variations	
Per-job algorithm time	Time spent (seconds) by the scheduler to make a job-level scheduling decision, representing the performance of the scheduling algorithm	Array [0.001–∞)
Per-task algorithm time	Time spent (in seconds) by the scheduler to make a task-level scheduling decision, representing the performance of the scheduling algorithm	Array [0.001–∞)
Blacklist	The percentage of machines not to be used	Array [0.0–∞)
Inter-arrival	Inter-arrival time generated for all jobs, replacing it with a fixed time instead	Array [0.001–∞)
Cluster Energy	Parameters related to performance iterated in order to create all experiment variations	
Shut-down policies	The shut-down policies to be run, such as: *Always power off, Exponential,* and *Gamma*	Array
Power-on	The power-on policies to be run	Array

(*continued on next page*)

Table 5 (*continued*)

Parameter	Description	Values
Scheduling	The scheduling strategies to be run	Array
Sorting	These strategies are used by greedy scheduling strategies to sort the candidate servers for their later selection	Array
Specific	Parameters used by specific schedulers	
Schedulers assigned to workloads	Mapping that describes how many and which schedulers are assigned to which workload type (Batch / Service) used by non-monolithic schedulers. Each row adds a new scheduler to serve a workload	Map [Scheduler name -> Workload name]
Conflict mode	Strategy used by parallel schedulers/orchestrators to decide whether a *commit* results in a conflict	resource-fit sequence-number
Transaction mode	Decision used by parallel schedulers/orchestrators to be applied when a *commit* creates a conflict	all-or-nothing incremental

Appendix B. Key results related to the performance of each workload

Table 6
Key results related to the performance of each workload.

Parameter	Description	Values
Queue time - first task	The time jobs wait for their first task to be scheduled (seconds).	[0.0–∞)
Queue time - all tasks	The time jobs wait for all their tasks to be scheduled (seconds).	[0–∞)
Makespan	The time jobs need to complete their execution (seconds).	[0–∞)
Timed-out jobs	Number of jobs left unscheduled after 100 unsuccessful scheduling tries for a given job or 1000 tries for any given task in a job.	[0.0–∞)
Job scheduling operations	Number of job scheduling attempts needed to fully schedule jobs.	[0–∞)
Task scheduling operations	Number of tasks scheduling attempts needed to fully schedule jobs.	[0–∞)

Appendix C. Key results related to the performance of each cluster

Table 7
Key results related to the performance of each cluster.

Parameter	**Description**	**Values**
# Jobs	Number of jobs processed for each workload type	[0–∞)
CPU utilization	Percentage of computing resources occupied	[0.0–100.0]
RAM utilization	Percentage of memory resources occupied	[0.0–100.0]
Scheduler utilization	Percentage of time the scheduler is working	[0.0–100.0]
Energy-efficiency	Indicators regarding the cluster energy and resource efficiency.	
Energy consumed	Total cluster energy consumption (in kWh)	[0.0–∞)
Energy saved vs. current system	Total energy saved by applying energy-efficiency policies compared to the same scenario with no energy-efficiency policies applied (in kWh).	[0.0–∞)
Shut-downs	Number of shut-down operations.	[0–∞)
Idle resources	Percentage of resources operating in an *Idle* state on average.	[0.0–100.0]
KWh saved per shut-down operation	This represents the energy saved against the number of shut-downs performed. It shows the stress suffered by hardware.	[0.0–∞)

References

[1] P. Neirotti, A. De Marco, A.C. Cagliano, G. Mangano, F. Scorrano, Current trends in smart city initiatives: some stylised facts, Cities 38 (2014) 25–36.
[2] Ángel Jesús Varela-Vaca, L. Parody, R.M. Gasca, M.T. Gómez-López, Automatic verification and diagnosis of security risk assessments in business process models, IEEE Access 7 (2019) 26448–26465, https://doi.org/10.1109/ACCESS.2019.2901408.
[3] Z. Ji, I. Ganchev, M. O'Droma, L. Zhao, X. Zhang, A cloud-based car parking middleware for IoT-based smart cities: design and implementation, Sensors 14 (12) (2014) 22372–22393.
[4] C. Bockermann, A survey of the stream processing landscape, Lehrstuhl fork unstliche Intelligenz Technische Universit. at Dortmund (2014).
[5] B. Zhang, N. Mor, J. Kolb, D.S. Chan, K. Lutz, E. Allman, J. Wawrzynek, E. Lee, J. Kubiatowicz, The cloud is not enough: saving IoT from the cloud, 7th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 15), (2015).
[6] M. Satyanarayanan, P. Bahl, R. Caceres, N. Davies, The case for VM-based cloudlets in mobile computing, IEEE Pervas. Comput. (4) (2009) 14–23.
[7] C. Mahmoudi, F. Mourlin, A. Battou, Formal definition of edge computing: an emphasis on mobile cloud and IoT composition, 2018 Third International Conference on Fog and Mobile Edge Computing (FMEC), (2018), pp. 34–42, https://doi.org/10.1109/FMEC.2018.8364042.
[8] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A. De Rose, R. Buyya, Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software 41 (1) (2011) 23–50.
[9] W. Tian, Y. Zhao, M. Xu, Y. Zhong, X. Sun, A toolkit for modeling and simulation of real-time virtual machine allocation in a cloud data center, IEEE Trans. Autom. Sci.Eng. 12 (1) (2015) 153–161.
[10] D. Kliazovich, P. Bouvry, S.U. Khan, Greencloud: a packet-level simulator of energy-aware cloud computing data centers, J. Supercomput. 62 (3) (2012) 1263–1283.
[11] D. Fernández-Cerero, A. Fernández-Montes, A. Jaköbik, J. Kołodziej, M. Toro, Score: simulator for cloud optimization of resources and energy consumption, Simul. Model. Pract. Theory 82 (2018) 160–173, https://doi.org/10.1016/j.simpat.2018.01.004. http://www.sciencedirect.com/science/article/pii/S1569190X18300030
[12] H. Gupta, A. Vahid-Dastjerdi, S.K. Ghosh, R. Buyya, ifogsim: a toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments, Software 47 (9) (2017) 1275–1296, https://doi.org/10.1002/spe.2509.
[13] D. Fernández-Cerero, A. Jakóbik, A. Fernández-Montes, J. Kołodziej, Game-score: game-based energy-aware cloud scheduler and simulator for computational clouds, Simul. Model. Pract. Theory 93 (2019) 3–20.
[14] D. Fernández-Cerero, A. Jakóbik, D. Grzonka, J. Kołodziej, A. Fernández-Montes, Security supportive energy-aware scheduling and energy policies for cloud environments, J. Parallel Distrib. Comput. 119 (2018) 191–202.
[15] M.M. Lopes, W.A. Higashino, M.A. Capretz, L.F. Bittencourt, Myifogsim: a simulator for virtual machine migration in fog computing, Companion Proceedings of the10th International Conference on Utility and Cloud Computing, ACM, 2017, pp. 47–52.
[16] M.I. Naas, J. Boukhobza, P.R. Parvedy, L. Lemarchand, An extension to ifogsim to enable the design of data placement strategies, 2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC), IEEE, 2018, pp. 1–8.
[17] Q. Wang, Pfogsim: A Simulator for Evaluating Dynamic and Layered Fog Computing Environments (2019).
[18] A. Brogi, S. Forti, A. Ibrahim, How to best deploy your fog applications, probably, 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC), (2017), pp. 105–114, https://doi.org/10.1109/ICFEC.2017.8.
[19] A. Brogi, S. Forti, QoS-aware deployment of IoT applications through the fog, IEEE Internet Things J. 4 (5) (2017) 1185–1192, https://doi.org/10.1109/JIOT.2017.2701408.
[20] T. Qayyum, A.W. Malik, M.A. Khan Khattak, O. Khalid, S.U. Khan, Fognetsim + +: a toolkit for modeling and simulation of distributed fog environment, IEEE Access 6 (2018) 63570–63583, https://doi.org/10.1109/ACCESS.2018.2877696.
[21] A. Varga, Omnet + +, Modeling and Tools for Network Simulation, Springer, 2010, pp. 35–59.
[22] T. Nguyen, E. Huh, Ecsim + +: an inet-based simulation tool for modeling and control in edge cloud computing, 2018 IEEE International Conference on Edge Computing (EDGE), (2018), pp. 80–86, https://doi.org/10.1109/EDGE.2018.00018.
[23] C.K. Filelis-Papadopoulos, K.M. Giannoutakis, G.A. Gravvanis, P.T. Endo, D. Tzovaras, S. Svorobej, T. Lynn, Simulating large VCDN networks: a parallel approach, Simul. Model. Pract. Theory 92 (2019) 100–114.
[24] J. Byrne, S. Svorobej, A. Gourinovitch, D.M. Elango, P. Liston, P.J. Byrne, T. Lynn, Recap simulator: simulation of cloud/edge/fog computing scenarios, 2017 Winter Simulation Conference (WSC), (2017), pp. 4568–4569, https://doi.org/10.1109/WSC.2017.8248208.
[25] M. Schwarzkopf, A. Konwinski, M. Abd-El-Malek, J. Wilkes, Omega: flexible, scalable schedulers for large compute clusters, Proceedings of the 8th ACM European Conference on Computer Systems, ACM, 2013, pp. 351–364.
[26] S. Robinson, Conceptual modelling for simulation part II: a framework for conceptual modelling, J. Oper. Res. Soc. 59 (3) (2008) 291–304.
[27] G.B. Dantzig, D.R. Fulkerson, On the Max Flow Min Cut Theorem of Networks, (1955).
[28] I. Gog, M. Schwarzkopf, A. Gleave, R.N. Watson, S. Hand, Firmament: fast, centralized cluster scheduling at scale, 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16), (2016), pp. 99–115.
[29] E.W. Dijkstra, A note on two problems in connexion with graphs, Numer. Math. 1 (1) (1959) 269–271, https://doi.org/10.1007/BF01386390.
[30] L. Page, S. Brin, R. Motwani, T. Winograd, The Pagerank Citation Ranking: Bringing Order to the Web, 1998.
[31] C. MacGillivray, V. Turner, R. Clarke, J. Feblowitz, K. Knickle, L. Lamy, M. Xiang, A. Siviero, M. Cansfield, IDC future scape: worldwide internet of things 2017 predictions, IDC Web Conference, (2016).
[32] D. Fernández-Cerero, A. Fernández-Montes, J.A. Ortega, Energy policies for data-center monolithic schedulers, Expert Syst. Appl. 110 (2018) 170–181, https://doi.org/10.1016/j.eswa.2018.06.007. http://www.sciencedirect.com/science/article/pii/S0957417418303531
[33] O.A. Abdul-Rahman, K. Aida, Towards understanding the usage behavior of Google cloud users: the mice and elephants phenomenon, IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Singapore, (2014), pp. 272–277.
[34] A. Jakobik, D. Grzonka, F. Palmieri, Non-deterministic security driven meta scheduler for distributed cloud organizations, Simul. Model. Pract. Theory (2017), https://doi.org/10.1016/j.simpat.2016.10.011. (available online 4 November 2016)