Fog Computing Through Public-Resource Computing and Storage

Saúl Alonso-Monsalve*, Félix García-Carballeira†, and Alejandro Calderón‡
Computer Science and Engineering Department
Carlos III University of Madrid
Leganés, Madrid, Spain
Email: *saul.alonso@uc3m.es, †felix.garcia@uc3m.es, ‡alejandro.calderon@uc3m.es

Abstract—**Fog computing is a model in which data and processing are concentrated on devices at the edge of the network, rather than almost entirely in the cloud. This new paradigm tries to solve the latency and bandwidth issues that the current cloud systems have to deal with, and also addresses the scalability problems caused by the exponential growth of the number of smart devices due to the success of the Internet of Things. In this paper we introduce the idea of using public-resource computing and storage techniques in order to process part of the workload of the current cloud systems so as to avoid saturating the cloud. This idea proposes the use of devices working as participants, which form a data center between the cloud service providers and the end-clients. A participant can be any type of device, from a traditional PC (Personal Computer), to a smartphone or tablet or even a smart TV. We have evaluated two different case studies by simulating the use of participating nodes in video-transfer applications. The results of the simulations demonstrate that public-resource computing and storage can be used to solve the latency and bandwidth issues that affect the current cloud systems. Therefore, our results represent a feasible solution for applications that process or store public data.**

Keywords: **Edge computing, Fog computing, Performance, Public-resource computing, Simulation.**

I. Introduction

Cloud computing is a paradigm through which everything that a computer system can offer is provided in the form of services over the Internet. It provides on-demand, pay-per-use, and highly scalable computing capabilities for services that enhance the user experience in a transparent way for the user. In recent years, cloud systems have provided computing and storage solutions to numerous problems in business, universities, data centers, and even in governments. The process of migrating data and applications to a cloud system, indeed, has become widespread. This 'cloudification' process allows both the users and the companies to have a centralized and simple vision of all the services they provide.

Furthermore, the Internet of Things (IoT) is a concept that refers to the digital interconnection of everyday objects with the Internet [1]. This idea was proposed by Kevin Ashton at the MIT Auto-ID Center (Massachusetts, Boston) in 1999 [2]. Currently, there are IoT applications in healthcare, agriculture, transportation, security, toys, etc. However, the number of devices connected to the Internet worldwide (and, therefore, accessing cloud services) is in continuous growth. In 2011, the company Cisco Systems predicted that there will be 50 billion devices with Internet access by 2020 [3], so most of the current cloud systems will not be able to handle the amounts of data generated by such a large number of devices.

The aim of moving the storage and computation tasks into the cloud is because the cloud has more power and capacity than any device the end-users own. This has been an efficient solution so far. However, with the exponential growth of IoT devices that access this kind of systems, networks will become a bottleneck due to bandwidth limitations, so the scientific community is searching for solutions to this problem. There are two alternatives: the first one consists on setting a limit on the number of users in the cloud, which would frustrate users and drastically halt the advance of this type of computing; the other alternative, which is gaining popularity, is to bring the computation and storage of the cloud systems to the edge of the network, instead of performing all these tasks in the cloud servers. This new model is called fog computing.

Fog computing (or simply "the fog") is a novel solution that proposes an extension of what we know as the cloud. Fog computing is a model in which data and processing are concentrated on devices at the edge of the network, rather than almost entirely in the cloud. In fact, Weisong Shi, from Wayne State University (Detroit, Michigan), affirmed a few months ago that "we are entering the post-cloud era" [4], referring to fog computing. Deploying part of the storage and computation at the edge of the network instead of in the totality of the cloud would allow all of our day-to-day devices to connect to the Internet. In addition, it would also improve the scalability of the systems that have applications that transmit large volumes of data over the network.

In this paper we introduce the idea of using public-resource computing and storage techniques in order to process and store part of the computation of the current cloud systems and therefore not saturate the cloud. These idea proposes the use of devices working as participants, which form a data center between the cloud service providers and the end-clients. As we will discuss throughout the document, a participant can be any type of device, from a traditional PC (Personal Computer), to a smartphone or tablet or even a smart TV. The use of participating nodes is an inexpensive way to move the processing and storage tasks from servers in the cloud into devices at the edge of the network, and is also a very accessible solution for the systems that use public content.

The evaluation carried out consists on a simulation of two different scenarios that use public-resource computing for video-transfer applications in order to perform the storage or processing of tasks at the edge of the network. On the one hand, we have simulated a video-download application using the participating devices as a storage cache. On the other hand, we have simulated a video-upload application where the content uploaded to the cloud must be filtered. We have measured the performance and compared it with the scenario in which all tasks are run in the cloud. The results show that the cloud systems become more scalable and efficient when applying the concept of fog computing by using processing participants.

The rest of the paper is organized as follows: Section II presents the background and discusses the related work; Section III explains in detail the solution that we propose; in Section IV we analyze the performance of our solution; and finally, Section V concludes the paper and presents some future work.

II. Background and Related Work

Coulouris et al. in [5] introduce the seven main challenges in distributed systems: heterogeneity, openness, security, scalability, failure handling, concurrency, and transparency. For a large number of clients in a cloud-based system, scalability, security, and failure handling are probably the most notable challenges. Clients exchange data within services in the cloud, and when the number of clients (or data transmitted) increases, then the network might become a bottleneck, causing scalability problems.

In order to distribute the workload in a more balanced manner, there are two complementary possibilities: (a) moving part of the processing work to the client, and (b) moving part of the computing tasks to the interconnection devices. These two alternatives are inspired by active storage [6], where part of the processing tasks are moved to the storage elements. In cloud computing, this strategy can be applied by moving part of the workload from the core to the edge of the network (the fog).

A. Fog Computing

In 2014, Vaquero and Rodero-Merino, proposed the following definition of fog computing [7]: *"Fog computing is a scenario where a huge number of heterogeneous (wireless and sometimes autonomous) ubiquitous and decentralised devices communicate and potentially cooperate among them and with the network to perform storage and processing tasks without the intervention of third-parties."*

In addition to the definition presented above, Gupta et al. defined fog computing as a distributed computing paradigm that extends the services provided by the cloud to the edge of the network [8]. This idea can be seen represented in Figure 1, where a large number of devices of all types access cloud services. However, much of the processing is done near the edge of the network instead of entirely in the core, taking advantage of the large number of sources on the edge (powered by IoT) [9].

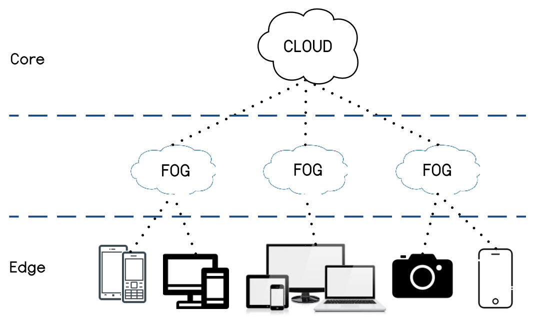


Fig. 1: A basic fog computing environment.

The main benefits of fog computing are:

- The latency is lower in the fog than in the cloud since the processing is done at the edge of the network instead of on servers at the core of the network. The data need much less hops in order to reach their destination than in cloud computing.
- Fog computing avoids scalability issues, as devices can access different servers that are close to the edge of the network. In addition, greater bandwidth can be achieved as users will use different links over the network rather than all accessing a cloud.

As commented in [10], mobile devices represent part of fog computing (smartTVs, autonomous-cars, etc. are also part of this type of computing). A current smartphone is able to handle almost all of the users' at-home and even many at-work-tasks without needing anything else. In the very near future, the smartphones/tablets will be almost exclusively the only computer devices that ordinary users will utilize. There is also much work that deals with mobile edge computing [11]–[13], which consists on placing processing at the edge of the cellular network. Besides, there is novel research that tries to take full advantage of edge processing and storing capabilities in radio access networks [14]. This paradigm is known as Fog-computing-based Radio Access Networks (F-RAN) and can be considered a type of fog computing, as it tries to extend cloud computing to the edge of the network. The fog is a very recent type of distributed computing, so there are not too many solutions yet. The studies in this topic are limited to describing this model, the important issues and the future challenges [15].

Furthermore, in [16] the authors describe how the interactions between the fog and the cloud can transform video applications and services. They propose different alternatives, but only focused on new wireless network architectures for video-caching services, and without performing an evaluation in this regard. Conversely, in [17] the authors present a fog computing simulator, called iFogSim, in order to measure the network latency and bandwidth and the energy consumption of the IoT and fog environments modeled. Nevertheless, iFogSim cannot be used to measure the performance of our alternative, since the simulator does not support simulations of network topologies with participating nodes.

B. Public-Resource Computing and Storage

Public-Resource Computing (PRC) uses the resources of devices that belong to the general public to do scientific supercomputing. BOINC (Berkeley Open Infrastructure for Network Computing) [18] is an open-source platform for PRC, and it is the most widely used middleware system. It can also be used on mobile devices. In this kind of platforms, the BOINC application only computes when the device is plugged into a power source (AC or USB) and the battery is over 90% of charge, so it will not significantly reduce the battery life or the recharge time. Moreover, BOINC transfers data only when the devices are connected to a WiFi network. In addition, there are current studies that try to exploit this model by using smart TVs as participating[1] nodes [19], which shows that this type of computing can become part of the IoT world.

Since fog computing is an emergent model, very little work has been done to try to link PRC with fog computing. In 2013, Chandra et al. [20] introduced Nebula, a highly decentralized cloud that uses volunteer edge resources. However, this article simply presents an overview and does not make any relevant change. Two years later, Ghafarian et al. [21] proposed the combination of PRC and Cloud resources in order to enhance computational capabilities. As a weakness, this solution does not consist of fog computing, since it focuses on running workflows in a specified deadline and minimizing dependencies.

III. Solution Proposed

In this section we will describe in detail our proposed solution. As we have previously commented, the idea of fog computing is to use the capabilities of the devices at the edge of the network to perform storage and computation tasks. Generally, these devices can range from access points to servers at the edge of the network, and are used to reduce the network traffic and the latency between end-users and cloud servers. The solution we propose is to use any type of device with Internet access and located at the edge of the network in order to deploy fog computing applications. To do this, we have been inspired by the paradigm of Public-Resource Computing (PRC), in which users donate the computing and storage capacities of their devices to scientific projects.

Users who wish to participate in a Public-Resource Computing (PRC) project should download a specific software for the project they want to collaborate with. This client program periodically contacts project-operated servers over the Internet to request jobs and report the results of completed jobs. The computing resources that power PRC are shared with the owners of the client machines. Because the resources are volunteered, utmost care is taken to ensure that the PRC tasks do not obstruct the activities of each machine's owner; a task is suspended or terminated whenever the machine is in use by another person. As a result, PRC resources are volatile in the sense that any number of factors can prevent the task of a PRC application from being completed. These factors include mouse or keyboard activity, the execution of other user applications, machine reboots, or hardware failures. Moreover, PRC resources are heterogeneous, in the sense that they differ in operating systems, CPU speeds, network bandwidth and memory and disk sizes. To sum up, the PRC tasks are executed during the idle time of the participating machines.

[1]In this paper, we refer to the devices that contribute to do scientific supercomputing in public-resource computing as participating nodes.

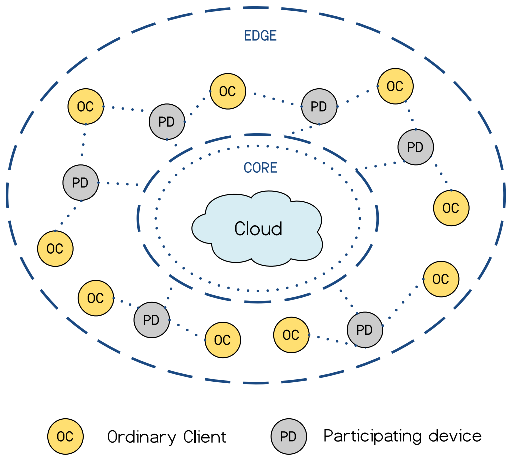


Fig. 2: Network topology of the proposed solution.

In our solution, users who want to collaborate on a cloud service should download a specific program in their devices (smartphones, tablets, laptops, PCs, or even smart TVs) for each service they want to collaborate with. The process is the following: a participating user downloads a specific software from a cloud service provider in order to participate in the service, so this action automatically subscribes the user to the system by adding the user to a list of collaborators, including its geographical location. In this way, the participating users form a data center between the cloud service provider and the end-users. Once this has been done, when an ordinary client wants to consume a cloud service, the client will download the list of participants, and it will contact the nearest participant located at the edge of the network. If that participant is not available, the client will access the next one on the sorted list, and so on (see Algorithm 1). This behavior is shown in Figure 2, where the Ordinary Clients (OC) access the Participating Devices (PD) at the network edge instead of accessing the cloud. Depending on the software downloaded, each participant will be responsible for processing (e.g. image or video filtering, streaming, pattern recognition, etc.) or storing tasks and content. In addition, depending on the type of application, the PD and the cloud must be synchronized.

It is possible that the users that want to collaborate on a cloud service may not have enough resources (bandwidth, CPU, RAM, disk storage), so each cloud provider must allow only users that satisfy the minimum requirements needed to participate in the service.

This solution aims to exploit the greater computing power and storage of the current devices, something unthinkable

Algorithm 1 Ordinary client consumes a service.

```
function CONSUME(service_A)
    if $service\_A.participants\_list$ == NULL then
        SEND $request$ to service_A.provider
        $message$ = RECEIVE from service_A.provider
        $service\_A.participants\_list = message.participants\_list$
    end if
    $service\_A.participants\_list$.SORT_BY_NEAREST_LOCATION
    for each $addr$ in $service\_A.participants\_list$ do
        if $addr$ is active then
            CONSUME_SERVICE
            $break$
        end if
    end for
end function
```

years ago. Today and even more in the very near future, we are all going to have multiple devices connected to the internet at home, including the television, the refrigerator, etc. so we can take advantage of the time that these devices are idle. In addition, as in current BOINC projects, it is necessary to create a system of incentives so that users can see their collaboration rewarded. BOINC projects, grant credit to users for the amount of computational work they have contributed to the system. It is just a measure of how much work a computer has done and it does not have monetary value [22].

This solution can be applied to various cloud systems that use public content, since the data will be manipulated by untrusted users. BOINC provides a form of redundant computing for their projects in which each computation is performed on multiple clients [23], the results are compared, and are accepted only when a 'consensus' is reached. If the number of errors for a task exceeds a limit, new workunits must be sent to the clients. Our solution follows the same procedure. The system administrators of each service must define parameters such as the quorum, the maximum number of errors allowed, etc.

Our solution can be applied to two key case studies, as we will see in the next section: applications where users download public content (data-intensive) or where users send large computational tasks to be executed in the cloud (computationally intensive).

IV. Evaluation and Discussion

The goal of this section is to show the performance (servers and network load, and throughput) that would result when intermediate participating nodes are used in video-transfer systems as a form of fog computing. In terms of implementation, we have used ComBoS [24], a simulator created by the authors, as a starting point. We have also carried out all the simulations using the MSG API of SimGrid [25].

The two case studies presented in this section simulate a video transfer system. The scenario used in the simulations is formed by a cluster hosted in a typical cloud, and a variable number of Ordinary Clients (OC) that access the cloud through the network (see Figure 3). The networks that connect the OC with the servers are fixed in all simulations. The cloud connection network has a bandwidth of 10 Gbps and each server has a power of 10 GigaFLOPS. The total number of OC varies in each simulation. The simulated clients are devices (PCs, laptops, mobile devices, etc.), and their power is not relevant since these clients do not perform any computation in the simulations. The latency of each of the packets that an OC sends to the cloud over the network is 50 ms. Finally, every execution in this section has simulated 100 hours.

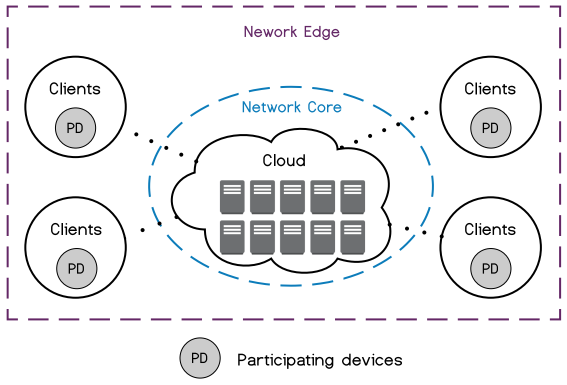


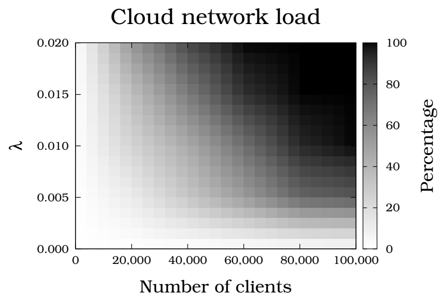
Fig. 3: Simulated scenario (video-transfer cloud system).

In the two case studies presented below, we use Participating Devices (PD) in order to test our solution. These PD are located at the edge of the network together with the OC. The latency of each of the packets that an OC sends to a PD is 1 ms (they are both in the network edge). In order to model the power of the PD, we have used the CPU power traces of the client hosts that make up the participating nodes of three different BOINC projects (SETI@home, Einstein@home and LHC@home) [26]–[28]. We have not used any other traces. In order to model the availability and unavailability of the PD, we used the results obtained in [29]. This research analyzed about 230,000 hosts' availability traces obtained from the SETI@home project. According to this paper, 21% of the hosts exhibit truly random availability intervals, and it also measured the goodness of fit of the resulting distributions using standard probability-probability (PP) plots.

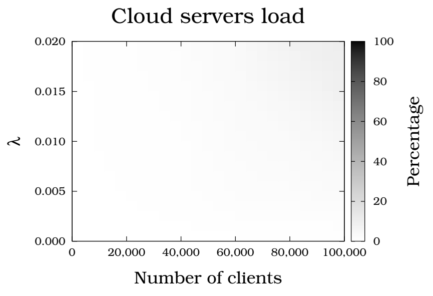
TABLE I: Case studies parameters.

Parameter	Value (it may depend on the simulation)
Simulation time	100 hours.
Number of OC	0, 20000, 40000, 60000, 80000, 100000.
Number of PD	0, 4000, 8000, 12000, 16000, 20000.
PD availability	Weibull ($shape = 0.393$, $scale = 2.964$).
PD unavailability	Log-normal ($\mu = -0.586$, $\sigma = 2.844$).
PD power	CPU power traces of BOINC projects.
Cloud bandwidth	10 Gbps.
Latency between OC and cloud	50 ms.
Latency between OC and PD	1 ms.

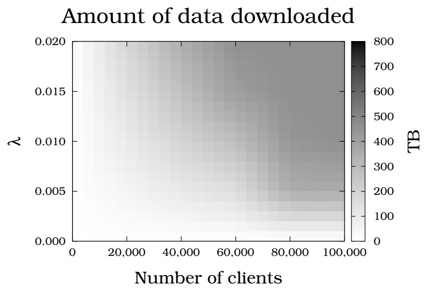
The commented parameters that are shared by all the simulations are shown in Table I. The number of PD corresponds to 20% of the total number of OC in each simulation. In order to account for the randomness of the simulations and to deem the results reliable, each simulation result presented in this section



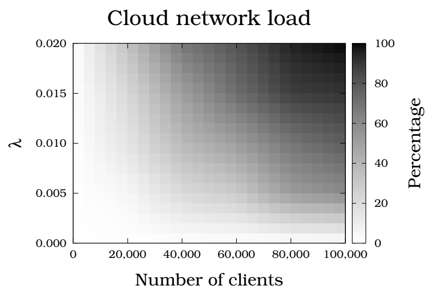
(a) Load of the cloud network (average percentage) in the original scenario.



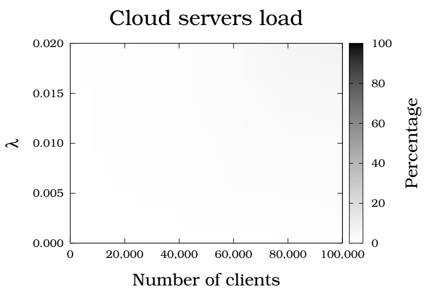
(b) Load of the cloud servers (considering I/O) in the original scenario.



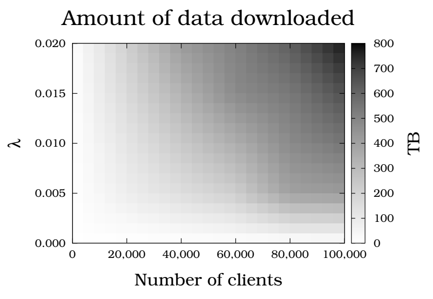
(c) Amount of video-data downloaded by the end-clients in the original scenario.



(d) Load of the cloud network (average percentage) in our alternative using PN.



(e) Load of the cloud servers (considering I/O) in our alternative using using PN.



(f) Amount of video-data downloaded by the end-clients in our alternative using PN.

Fig. 4: Case study 1 (video downloading). A comparison between the performance of the simulation of the original environment (graphs a, b, and c) and our alternative using PN (graphs d, e, and f) that correspond to 20% of the total number of clients.

is based on the average of 30 runs. For a 95% confidence interval, the error is less than $\pm$ 2% for all values.

A. *Case Study 1: Video-download*

In this first case study we analyze the performance of a video download application, where the clients download the contents from the cloud servers according to the scenario presented in Figure 3. In addition, in this section we will compare the performance of this system with the performance that the system would get with a set of PD working as a storage cache.

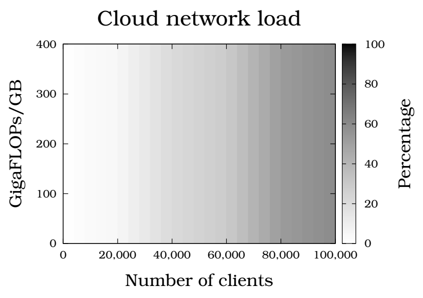
In this case study, when an OC wants to watch a video, the OC downloads it from the cloud over the network. However, in our alternative, the client asks the cloud for the list of PD that have the video, and downloads it from the nearest PD. If no PD has the media, the client requests the nearest available PD to download the video file. Therefore, the next time an OC wants to download that same video, this client can download it from a PD, located at the edge of the network, taking advantage of the data locality of the media files. In this way, the PD form a cache between the cloud and the OC.

In our simulations, the frequency at which clients download videos follows an exponential statistical distribution [30], with a lambda (λ) variable that differs in each simulation (values from λ=0.005 to λ=0.02). In addition, as commented in [31], the YouTube media popularity follows the Zip-f law (with β = 0.56), so it is the distribution that determines which videos are downloaded in our simulations.

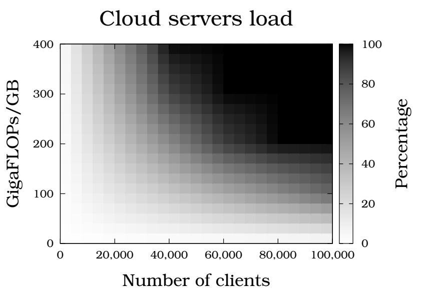
Figure 4 shows the results of our simulations. In each graph, the X-axis indicates the total number of clients in the system, and the Y-axis indicates the frequency at which these clients access the system. In particular, graphs 4a, 4b, and 4c show the results of the original scenario; and graphs 4d, 4e, and 4f show the results of the scenario that uses intermediate PD. As can be seen in Figures 4a and 4b, the original system has a network saturation problem when many clients download content very frequently ($\approx$ 70,000 clients and $\lambda \approx 0.15$), while the servers in the cloud system have no overhead problems. Nevertheless, with our alternative, we have managed to reduce the load of the network (graph 4d, while the cloud servers remain the same (graph 4e). In addition, we get a better system performance (see graphs 4c and 4f), since the amount of data downloaded is much greater (almost twice for 100,000 clients and λ=0.02) than in the original scenario. This is because the PD are much closer to the OC than the servers, so the access is faster. Besides, OC download the most popular videos several times from the PD instead of accessing the cloud, which greatly reduces traffic in the cloud network.

B. *Case Study 2: Video-filtering*

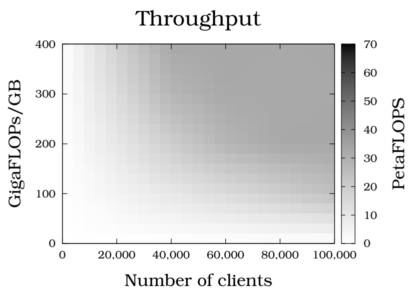
In this second case study we analyze again the performance of a video transfer application, but in this case we want to evaluate an application in which the clients upload media so that the cloud has to filter these data. This case study uses the same scenario as the first one (Figure 3).



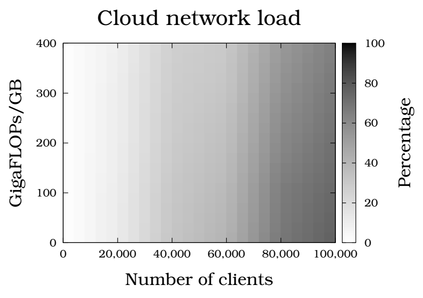
(a) Load of the cloud network (average percentage) in the original scenario.



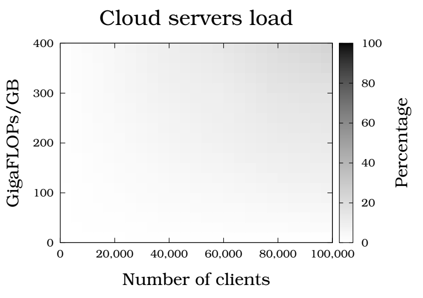
(b) Load of the cloud servers (considering I/O) in the original scenario.



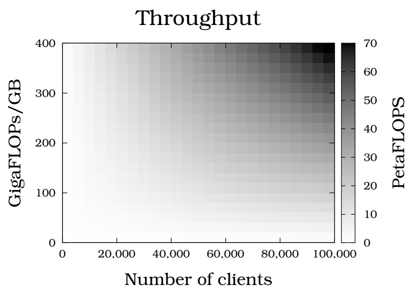
(c) Throughput in PetaFLOPS of video-data processed in the original scenario.



(d) Load of the cloud network (average percentage) in our alternative using PN.



(e) Load of the cloud servers (considering I/O) in our alternative using using PN.



(f) Throughput in PetaFLOPS of video-data processed in our alternative using PN.

Fig. 5: Case study 2 (video processing). A comparison between the performance of the simulation of the original environment (graphs a, b, and c) and our alternative using PN (graphs d, e, and f) that correspond to 20% of the total number of clients.

In this case study, when an OC wants to upload a video and apply a filter on it, the OC sends the whole video file to the cloud over the network. However, in our alternative, the client asks the cloud for the list of PD, and sends it to the nearest PD. Then, the PD applies the filter an uploads the file to the cloud. The goal of using intermediate nodes is to reduce the use of cloud servers and thus improve the system throughput.

To ensure that the result of the filtering performed by an untrusted PD is correct, each OC sends the same video file to two different PD. Once the filtering is done, each PD sends to the cloud the result of applying a hash function to the filtered video. If these two values match, the server gives approval and one of the two devices (randomly) sends the filtered video to the cloud. This allows the system to reduce the network traffic.

Figure 5 shows the results of our simulations. In each graph, the X-axis indicates the total number of clients in the system, and the Y-axis indicates the number of floating-point operations needed to process a particular amount of data. Specifically, the Y-axis indicates the number of Giga Floating-point operations (GigaFLOPs) required to process each GB of data. In particular, graphs 5a, 5b, and 5c show the results of the original scenario; and graphs 5d, 5e, and 5f show the results of the scenario that uses intermediate PD. As can be seen in Figures 5a and 5b, the original system has a saturation problem in the cloud servers when many clients upload content to the cloud, and the cloud takes a long time to process these data ($\approx$ 60,000 clients and $\approx$ 250 GigaFLOPs/GB), while there are no network overhead problems. Nevertheless, with our alternative, we have managed to reduce the load on the cloud servers (graph 5e, while the network remains almost the same (graph 5d). This happens because almost all of the computation is now performed in the PD, rather than entirely in the cloud. In addition, we get a better system performance (see graphs 5c and 5f), since the amount of media processed is much greater (more than twice for 100,000 clients and 400 GigaFLOPs/GB of processing) than in the original scenario. This is because the saturation of the cloud servers in the original scenario prevented the servers from executing more tasks, whereas with our alternative this computation is transferred to the PD, allowing for a greater scalability and throughput of the system.

V. Conclusions and Future Work

This paper has presented a specific form of deploying fog computing by using participating devices at the edge of the network. A participating device can be any type of device, from a traditional PC, to a smartphone or even a smart TV. We have evaluated the performance of our alternative in two case studies by simulating different scenarios of video applications. The results of the simulations yield two main conclusions:

- For media download systems, the participating devices can form an intermediate cache that reduces the load on the cloud network and also improves the performance of these systems because of data locality.
- For media upload and filtering systems, the participating devices form a data center between the end-clients and the cloud, which reduces drastically the load on the cloud

servers, and therefore allows for more users in the system (more scalability), increasing the throughput.

As future work, we want to analyze more case studies and simulate new scenarios in order to measure the energy consumption of the machines involved in this new model. Besides, we want to deploy this solution in real environments. We are also working on the security matters that fog computing needs in order to use nodes to store and process private content.

Acknowledgment

This work has been partially supported by the Spanish MINISTERIO DE ECONOMÍA Y COMPETITIVIDAD under the project grant TIN2016-79637-P TOWARDS UNIFICATION OF HPC AND BIG DATA PARADIGMS.

References

[1] M. Conner, "Sensors empower the "Internet of Things," *WorldCat*, pp. 32–38, May 2010.

[2] K. Ashton, "That 'internet of things' thing," *RFID Journal*, vol. 22, no. 7, pp. 97–114, Jun. 2009.

[3] D. Evans, "The internet of things: How the next evolution of the internet is changing everything," *Whitepaper, CISCO Internet Business Solutions Group (IBSG)*, vol. 1, pp. 1–11, Apr. 2011.

[4] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, May 2016.

[5] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, *Distributed Systems: Concepts and Design*, 5th ed. USA: Addison-Wesley Publishing Company, 2011.

[6] E. Riedel, G. A. Gibson, and C. Faloutsos, "Active storage for large-scale data mining and multimedia," in *Proceedings of the 24rd International Conference on Very Large Data Bases*, ser. VLDB '98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 62–73.

[7] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27–32, Oct. 2014.

[8] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in internet of things, edge and fog computing environments," *CoRR*, vol. abs/1606.02007, 2016.

[9] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, *Fog Computing: A Platform for Internet of Things and Analytics*. Cham: Springer International Publishing, 2014, pp. 169–186.

[10] E. Borcoci, "Fog computing, mobile edge computing, cloudlets - which one?" in *11th International Conference on Systems and Networks Communications (ICSNC 2016)*, ser. ICSNC '16. Rome, Italy: IARIA, Aug. 2016.

[11] M. T. Beck and M. Maier, "Mobile edge computing: Challenges for future virtual network embedding algorithms," in *8th International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP 2014)*. Rome, Italy: IARIA, Aug. 2014, pp. 65–70.

[12] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *Proceedings of the 10th International Conference on Intelligent Systems and Control (ISCO 2016)*, Coimbatore, India, Jan. 2016.

[13] F. Rodrigo-Duro, J. Garcia-Blas, D. Higuero, O. Perez, and J. Carretero, "Cosmic: A hierarchical cloudlet-based storage architecture for mobile clouds," *Simulation Modelling Practice and Theory*, vol. 50, pp. 3 – 19, 2015, special Issue on Resource Management in Mobile Clouds.

[14] S. Yan, M. Peng, and W. Wang, "User access mode selection in fog computing based radio access networks," *CoRR*, vol. abs/1602.00766, 2016.

[15] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Workshop on Mobile Big Data*, ser. Mobidata '15. Hangzhou, China: ACM, 2015, pp. 37–42.

[16] J. Zhu, D. S. Chan, M. S. Prabhu, P. Natarajan, H. Hu, and F. Bonomi, "Improving video performance with edge servers in the fog computing architecture," in *Proceedings of the 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*, ser. SOSE '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 320–323.

[17] P. García-López, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric Computing: Vision and Challenges," *ACM SIGCOMM Computer Communication Review*, vol. 5, no. 45, pp. 37–42, 2015.

[18] D. P. Anderson, "BOINC: A System for Public-Resource Computing and Storage," in *5th IEEE/ACM International Workshop on Grid Computing*, 2004, pp. 4–10.

[19] R. Nakanishi, "A study on utilization of TV sets in volunteer computing," in *ITE Tech. Rep.*, ser. BCT '16, vol. 40, no. 23, Hokkaido, Japan, Jul. 2016, pp. 17–20.

[20] A. Chandra, J. Weissman, and B. Heintz, "Decentralized edge clouds," *IEEE Internet Computing*, vol. 17, no. 5, pp. 70–73, Sep. 2013.

[21] T. Ghafarian and B. Javadi, "Cloud-aware data intensive workflow scheduling on volunteer computing systems," *Future Generation Computer Systems*, vol. 51, no. C, pp. 87–97, Oct. 2015.

[22] D. P. Anderson, "Local scheduling for volunteer computing," in *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*. IEEE, 2007, pp. 1–8.

[23] "Creating BOINC Projects," Last visited, May 2016. [Online]. Available: https://boinc.berkeley.edu/boinc.pdf

[24] S. Alonso-Monsalve, F. García-Carballeira, and A. Calderón, "Analyzing the performance of volunteer computing for data intensive applications," in *14th International Conference on High Performance Computing & Simulation (HPCS 2016)*. Innsbruck, Austria: IEEE, Jul. 2016, pp. 597–604.

[25] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter, "Versatile, scalable, and accurate simulation of distributed applications and platforms," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2899–2917, Jun. 2014.

[26] S. project, "CPU performance of SETI@home client devices," Last visited, Mar. 2016. [Online]. Available: http://setiathome.berkeley.edu/cpu_list.php

[27] E. project, "CPU performance of EINSTEIN@home volunteer computers," Last visited, Mar. 2016. [Online]. Available: https://www.einsteinathome.org/cpu_list.php

[28] L. project, "CPU performance of LHC@home volunteer computers," Last visited, Mar. 2016. [Online]. Available: http://lhcathomeclassic.cern.ch/sixtrack/cpu_list.php

[29] B. Javadi, D. Kondo, J.-M. Vincent, and D. P. Anderson, "Discovering statistical models of availability in large distributed systems: An empirical study of seti@home," *Parallel and Distributed Systems, IEEE Transactions*, vol. 22, pp. 1896–1903, 2011.

[30] S. Jin and A. Bestavros, *Generating Internet Streaming Media Objects and Workloads*. Boston, MA: Springer US, 2005, pp. 177–196.

[31] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Youtube traffic characterization: A view from the edge," in *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '07. San Diego, California, USA: ACM, 2007, pp. 15–28.