A Performance Study of High-End Fog and Fog Cluster in iFogSim: Proceedings of the Computational Intelligence in Information Systems Conference (CIIS 2018)

Chapter *in* Advances in Intelligent Systems and Computing · January 2019

DOI: 10.1007/978-3-030-03302-6_8

Fatin Rahman
Universiti Teknologi Brunei

Thien Wan Au
Universiti Teknologi Brunei

S.H. Shah Newaz
Universiti Teknologi Brunei

A Performance Study of High-end Fog and Fog Cluster in iFogSim

Fatin Hamadah Rahman[1], Thien Wan Au[1], S.H. Shah Newaz[1,2], and Wida Susanty Hj. Suhaili[1]

[1] School of Computing and Informatics, Universiti Teknologi Brunei, Jalan Tungku Link, Gadong BE1410, Brunei Darussalam.
[2] KAIST Institute for Information Technology Convergence, 291 Daehak-ro, Yuseong-gu, Daejeon 34141, South Korea.
p20171005@student.utb.edu.bn
twan.au@utb.edu.bn
shah.newaz@utb.edu.bn
wida.suhaili@utb.edu.bn

Abstract. Fog computing is introduced to help leverage the processing burden in cloud as it can no longer sustain the ever-growing volume, velocity and variety of the IoT-generated data. Existing studies have shown that fog computing is able to reduce the latency and provide other benefits in the IoT-fog-cloud environment. While fog is heterogeneous since can be any device with computing, networking and storage ability, its scalability must be ensured. Currently, not many studies have been conducted to see the performance of fog in different scalability approaches i.e scaling up and scaling down. This paper provides a brief explanation on iFogSim, which is a Java-based program that allows modelling and simulation of fog computing environments. The iFogSim is used in this study to simulate the fog environment in scaling up and scaling out approaches running in five configuration settings. In the scaling out approach, it presents a cluster of fogs with similar specifications and the scaling up approach presents a high-end fog with greater capabilities than the fogs in the first approach. Our initial findings delineate that the scaling out approach gives a better result in reducing the cost of execution in cloud. In this paper, we provide an insightful discussion on the strength and weakness in these two approaches. This would open up new avenues of further research.

Keywords: Fog computing · Scalability · iFogSim.

1 Introduction

Executing domestic tasks are effortless nowadays from brewing coffee the moment a person wakes up to receiving real-time inventory updates. This is all made possible with the Internet of Things (IoT) that has positively affect people's lives globally. With the IoT growing every day, the Business Insider reported that by 2020, 34 billion devices will be connected to the Internet, 24 billion of which will be IoT devices [19]. Consequently, the need for data to be processed quickly also increases. While cloud computing contributes tremendously to the success of IoT, however, it can no longer sustain in processing the ever-growing IoT demand whilst meeting the stringent latency requirement. As a solution, fog computing is introduced to help leverage the burden in cloud by allowing the processing to be done locally [5]. To illustrate how a network bandwidth reduction can be achieved by implementing fog computing, let us consider a city surveillance scenario. If the city employs 5 megapixel IP surveillance cameras, streaming it from the cloud using the T1 connection with transmission rate of 1.54Mbps would consume a total of 3.9Tb data for one camera alone. If the city municipal were to deploy multiple advanced cameras, additional bandwidth would be required. Mitigating part of the tasks to fog not only reduces the burden in cloud, much of the processing can be done locally in the fog as well. From the various works such as in [10] and [18], it is obvious that implementing the fog layer in the IoT-cloud environment have brought a significant reduction in energy consumption, latency and network usage.

While the fog layer is heterogeneous in a way that a fog can be any device that has networking, computing and storage capabilities, ranging from servers, routers, set-top boxes to access points, the fog layer must also be able to dynamically scale depending on the network needs. Although there are studies that have looked into scalability in the fog [8] [16] [20], these studies did not specifically look into the scaling out (horizontal) and scaling up (vertical) approaches of the fog, and how these approaches would impact the cloud. Scaling up or scaling out has their own benefits and drawbacks. Thus, it is important to identify the appropriate scaling approach in order to maximize the overall performance without compromising other equally important factors. To date, there are no studies that have compared the scaling performance in the fog layer domain. Hence, the aim of this paper is to see the effect of implementing scaling up and scaling out approaches in fog and gauge the performance in terms of the cost of execution.

The structure of the paper is as follows: Section 2 describes the background study of the fog computing paradigm and scalability. Section 3 elaborates on iFogSim and the experiment configurations. Section 4 shows the results and discussions. Finally, conclusions and future works are presented in section 5.

2 Related Work

2.1 Fog Computing Paradigm

It is indisputable that research interest in the fog computing domain has been getting attention, with possible appealing use cases. The OpenFog consortium has laid out several use cases that includes a fog-based drone delivery services [14] and smart cities [15]. Additionally, the existence of fog computing is applicable to a wide variety of applications such as smart grids, wireless sensor and actuator networks (WSAN), decentralized smart building control (DSBC), IoT and Cyber-physical systems (CPSs), and connected vehicles [4] [17]. Currently, there are only a handful simulators capable of simulating a fog environment, one of which is iFogSim [7] , a Java-based program that is an extension to CloudSim. Various existing studies have used iFogSim to explore the capabilities of fog. The authors in [9] have extended the iFogSim to provide mobility support through migration of virtual machines between cloudlets. Along with their proposed migration policy, MyiFogSim can be used to analyze the policy impact on application quality of service. On the other hand, authors in [12] proposed a latency-aware application module management policy for fog environment that meets the diverse service delivery latency and amount of data signals to be processed in per unit time for different applications. An energy-aware allocation strategy for placing application modules (tasks) on fog devices is proposed by authors in [10]. Meanwhile, the authors in [18] presented a Module Mapping Algorithm for efficient utilization of resources in the network infrastructure by efficiently deploying Application Modules in Fog-Cloud Infrastructure for IoT based applications. Despite the studies conducted in fog, it is apparent that fog's scalability has not been widely explored.

Scalability is defined as the computer system, network or application ability to handle increasing amount of work, both in terms of processing power as well as storage resources [16]. A system is said to be scalable if the performance improves after additional resources are added and it can be scaled in two ways i.e. scaling up (vertically) or scaling out (horizontally) [1]. Scaling up adds resources to a single node in a system such as adding storage, processors or memory to a single computer while scaling out adds more nodes to a system, such as adding a new computer to a distributed software application. However, both approaches have their own advantages and disadvantages as elaborated in Table 1. Preference as to whether the use of scaling up or scaling out is well-suited for a system is context-dependent. For instance, scaling out approach is preferred by Google where they have demonstrated that 15,000 commodity-class PCs with fault-tolerant software is more cost-effective than a comparable system built out of a smaller number of high-end servers [3]. Data processing on the other hand, such that of Hadoop MapRduce are often better served by a scale-up server than a scale-out cluster [2].

3 Simulation

In this section, the scalability in fog is tested using iFogSim as it allows the flexibility of configuring the fog devices' specifications. To gauge the performance between the two approaches, the cloud execution cost metric is used.

3.1 iFogSim

The classes in iFogSim are annotated in a way that users without prior knowledge of CloudSim can easily define the infrastructure, service placement and resource allocation policies for Fog computing. The main java classes is depicted in Figure 1 which comprise of FogDevice, Controller, ModuleMapping, ModulePlacementMapping, ModukePlacement, Application, AppEdge, AppLoop and

Table 1: Advantages and disadvantages of scaling up and scaling out.

Scale Up	**Advantages:** 1. Less physical space and cooling is needed. 2. Easier maintenance. **Disadvantages:** 1. In some cases, upgrading some resources could be expensive. 2. Less availability and risk having a single point of failure. Hence, increasing chances of server downtime and affecting the Service Level Agreement.
Scale Out	**Advantages:** 1. Provides redundancy thus increasing the availability of the system. **Disadvantages:** 1. Troubleshooting and maintenance can be challenging and time-consuming. 2. It would require more space and cooling.

AppModule. The Controller, ModuleMapping, ModulePlacementMapping and ModukePlacement belong to the same package. Meanwhile Application, AppEdge, AppLoop and AppModule belong to another package. In order to run the program in the iFogSim, another java class that is not shown in the diagram named Simulation is created. The details of the classes are further explained in Table 2 [7]. Nevertheless, iFogSim is packaged with two application module placement strategies; cloud-only placement and edge-ward placement as elaborated below [11]:

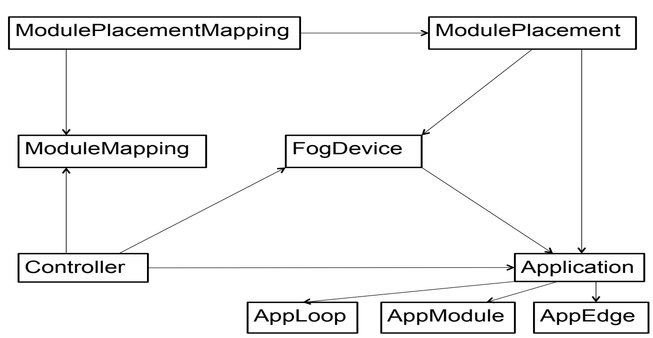

Fig. 1: Main iFogSim java classes and their relationships.

A. Cloud-only placement: The cloud-only placement strategy is based on the traditional cloud- based implementation of applications where all modules of an application run in data centers. The sense-process-actuate loop in such

Table 2: iFogSim's Java classes.

Java Class	**Remarks**
FogDevice	The hardware characteristics of Fog devices and their connections to other Fog devices, sensors and actuators are specified in this class. The major attributes of the fog device include accessible memory, processor, storage size, uplink and downlink bandwidths. The methods in this class define how the resources of a fog device are scheduled between application modules running on it and how modules are deployed and decommissioned on them.
Controller	The Controller object launches the AppModules on their assigned Fog devices following the placement information provided by Module Mapping object and periodically manages the resources of Fog devices.
ModuleMapping	This class maps the node name to the module name.
ModulePlacementMapping	Provides the placement mapping information.
ModulePlacement	Contains the abstract placement policy that needs to be extended for integrating new policies.
Application	Represents an application in the Distributed Dataflow Model.
AppEdge	denotes the data-dependency between a pair of application modules and represents a directed edge in the application model.
AppLoop	An additional class, used for specifying the process-control loops of interest to the user.
AppModule	Represents the processing elements of fog applications.
Simulation	This class is the simulation setup that runs the whole program. It determines the fog attribute values as well as managing other application-related processes such as adding modules to the application model and connecting application modules in the application model.

applications are implemented by having sensors transmitting sensed data to the cloud where it is processed and actuators are informed if action is required.

B. Edge-ward placement: Edge-ward placement strategy is inclined towards the deployment of application modules close to the edge of the network. However, devices close to the edge of the network may not be computationally powerful enough to host all operators of the application. In such a situation, the strategy iterates on fog devices towards cloud and tries to place remaining operators on alternative devices.

3.2 Scenarios for Performance Comparison

To investigate the fog's scalability, both scaling out and scaling up approaches are simulated in a smart surveillance environment. Both of the approaches have similar settings in the cloud, proxy and end devices except the fogs. It is worth noting that the proxy here is the Internet Service Provider proxy. Additionally, the fogs operate on three areas where each area has four cameras. These cameras run the same application on both approaches where the application modules include object detector, motion detector, object tracker and a user interface. Scenario 1 represents the scaling out approach where it shows clusters of fogs with similar specifications and scenario 2 presents the scaling up approach consisting of high-end fogs of greater capabilities than the fogs in the scaling out approach. These are illustrated in Figure 2.

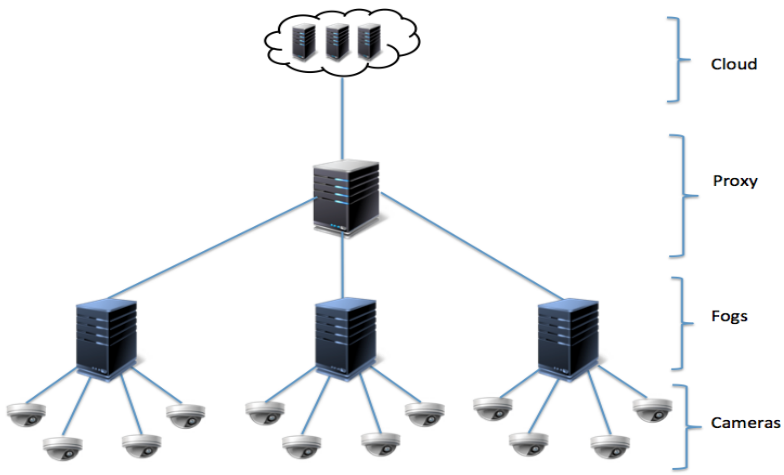

Fig. 2: Generic fog deployment scenario.

3.3 Parameter Configurations

Table 3 shows the default configurations for the cloud, proxy server, fog, and end devices in terms of million instructions per second (MIPS), RAM in gigabyte, bandwidths, hierarchy level, rate per MIPS, and busy and idle powers. There are five configurations for both approaches, named Config 1, Config 2, Config 3, Config 4 and Config 5. In the scaling out approach, each area would have one, two, three, four and five fogs in Config 1, Config 2, Config 3, Config 4 and Config 5, respectively. In scaling out approach, one high-end fog is allocated in all five configurations. However, the high-end fog's specifications are increased to match the total specifications of the fog clusters in the scaling out approach. For instance, when the total RAM is 4GB (from one fog of 4GB RAM), 8GB (from two fogs of 4GB RAM), 12GB (from three fogs of 4GB RAM), 16GB (from four fogs of 4GB RAM) and 20GB (from five fogs of 4GB RAM) for Config 1, Config 2, Config 3, Config 4 and Config 5 in scaling out approach, the RAM of high-end fog in scaling up approach is similarly set to 4GB, 8GB, 12GB, 16GB and 20GB in Config 1, Config 2, Config 3, Config 4 and Config 5. The latencies between the source and destination in our simulation setup are also using the iFogSim's default values where the cloud and proxy connection has a latency of 100ms, proxy to fog and fog to the end devices both have 2ms latency.

Table 3: Default entity configurations in iFogSim.

Attribute	**Cloud**	**Proxy**	**Fog**	**Camera**
MIPS	44800	2800	2800	1500
RAM (GB)	40	4	4	2
Uplink Bandwidth	100	10000	10000	10000
Downlink Bandwidth	10000	10000	10000	10000
Hierarchy Level	0	1	2	3
Rate per MIPS	0.01	0	0	0

4 Results and Discussions

As the accumulated resources of the fogs increases, the cost of execution decreases in both scaling out and scaling up approaches. Our initial results depicted in Figure 3 has shown that the cloud execution cost is reduced from 61778 to 58891 for the scaling out approach, and from 62476 to 59090 for the scaling up approach. The reductions in both approaches is accredited from the shift of application or task processing from the cloud to the fog layer. Although the scaling up approach has exhibited a higher cloud execution cost compared to scaling out approach, the differences are insignificant. Table 4 shows that using the scaling up approach only made less than %2 increase of execution cost in all of the configurations. Nonetheless, both approaches have their own advantages and disadvantages as mentioned in Section 2 that still have to be considered.

Table 4: Percentage increase of cost execution using the high-end fog.

Config	**1**	**2**	**3**	**4**	**5**
Execution cost (%)	+1.10	+0.58	+1.80	+1.1	+0.34

Furthermore, although scaling out has proven to be beneficial in various ways, there are issues in implementing the scaling out approach that are unique in the fog domain as elaborated as follows:

- Availability: Scaling out in a highly distributed and heterogeneous fog environment means that availability will be a challenge. Availability can be affected by failure of hardware, software and internet connectivity. As fog facility can also be included within the already deployed equipment such as in routers, switches, and optical network units [13] [6] where they are closer

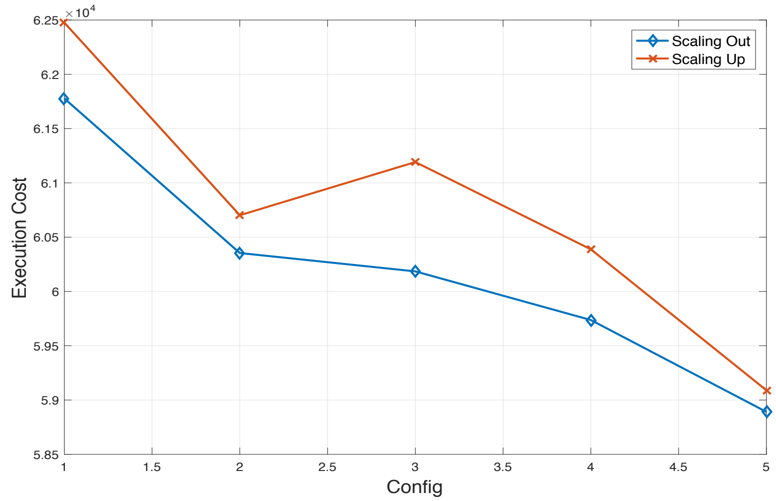

Fig. 3: Execution cost comparison between scaling out and scaling up.

to the end users, hence availability must be guaranteed. Moreover, the fog has a better mobility support as compared to the cloud. Integrating mobile fogs would create a dynamically changing topology, thus affecting the availability. Other factors that would affect the availability include physical threats such as human-made actions either deliberate or unintentional, and natural disasters. Therefore to increase availability, apart from having redundant power supply, multihoming approach can be applied with each of the fogs under both approaches. However, this can be achieved at the price of increasing the cost.

- Security: Ensuring homogeneous security in a heterogeneous fog environment using the scaling out approach would be a necessity. The various kinds of nodes that are added to scale up the network would often have different security requirements and platforms used. This would give rise to incompatibility problems. For each deployment location, it is imperative to understand all kinds of security threats that would occur in many angles as well. Some of the threats include network intrusion, information leakage by attackers during routing, and denial of service. Furthermore, authorization issues would arise as fog resources would be shared by various users. Without proper security measures, the network will be vulnerable to myriad of threats and become easily compromised.
- Network bandwidth: While shifting the processing from cloud to fog would reduce the network bandwidth usage, scaling out the fog domain would imply

the addition of more nodes to the fog. Hence, careful planning is required as this approach could introduce network communication overhead as well.

5 Conclusions and Future Work

This paper has demonstrated the different implementations of fog towards the cloud's execution cost. The scaling out presents a cluster of fogs with lower specifications and the scaling up presents a high-end fog. The results show that implementing fogs using the scaling out approach performs slightly better compared to fogs in scaling up approach. Due to time constraints and the limitations of iFogSim, other performance metrics such as throughput, latency and packet delivery ratio cannot be extracted. However, iFogSim can be further modified to provide these metrics to give better assessment of the fog. Nonetheless, this study has shed some insightful directions to work on in the future:

- Load-balancing policy: Since only one application runs in this study, more applications can be added to see how the overall performance will be affected. Hence, load balancing policy needs to be created and applied.
- Mobility: Unlike cloud, fogs support mobility. While mobile fogs add more complexity to the environment, applying mobility allows us to understand how task allocation takes place in a dynamic environment.
- Dynamic clustering: Currently, the iFogSim only allows static fog clusters that have similar attribute values, future work will incorporate dynamic fog clustering where different cluster would have different values.
- Security: Considering that security solutions made for cloud may not suit well in fog, in addition to fogs' mobile capability, fogs are vulnerable in many ways. Although grouping the fogs helps to reduce the overhead and ease the management, it does not guarantee the security of the fog as a whole. Hence, there is a need to evaluate the trustworthiness of the fog layer to ensure that they are able to perform up to expectation.

References

1. Agrawal, D., El Abbadi, A., Das, S., Elmore, A.J.: Database scalability, elasticity, and autonomy in the cloud. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 6587 LNCS(PART 1), 2–15 (2011)
2. Appuswamy, R., Gkantsidis, C., Narayanan, D., Hodson, O., Rowstron, A.: Scale-up vs scale-out for Hadoop. pp. 1–13 (2013), http://dl.acm.org/citation.cfm?doid=2523616.2523629
3. Barroso, L., Dean, J., Holzle, U.: Web Search for a Planet: The Google Cluster Architecture. IEEE Micro 23(2), 22–28 (2003), http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1196112
4. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog Computing and Its Role in the Internet of Things. Proceedings of the first edition of the MCC workshop on Mobile cloud computing pp. 13–16 (2012)

5. Cisco: Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are pp. 1–6 (2015), http://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf
6. Dastjerdi, A.V., Gupta, H., Calheiros, R.N., Ghosh, S.K., Buyya, R.: Fog Computing: Principles, architectures, and applications. In: Internet of Things: Principles and Paradigms (2016)
7. Gupta, H., Dastjerdi, A.V., Ghosh, S.K., Buyya, R.: iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing Environments pp. 1–22 (2016), http://arxiv.org/abs/1606.02007
8. Hong, K., Lillethun, D.: Mobile Fog: A Programming Model for Large-Scale Applications on the Internet of Things. pp. 15–20 (2013), http://dl.acm.org/citation.cfm?id=2491270
9. Lopes, M.M., Capretz, M.A.M., Bittencourt, L.F.: MyiFogSim : A Simulator for Virtual Machine Migration in Fog Computing. In: Proceedings of the 10th International Conference on Utility and Cloud Computing. pp. 47–52 (2017)
10. Mahmoud, M.M., Rodrigues, J.J., Saleem, K., Al-Muhtadi, J., Kumar, N., Korotaev, V.: Towards energy-aware fog-enabled cloud of things for healthcare. Computers and Electrical Engineering 67, 58–69 (2018), https://doi.org/10.1016/j.compeleceng.2018.02.047
11. Mahmud, R., Buyya, R.: Modelling and Simulation of Fog and Edge Computing Environments using iFogSim Toolkit. In: Fog and Edge Computing: Principles and Paradigms, chap. 17, pp. 1–35. Wiley Press (2018)
12. Mahmud, R., Ramamohanarao, K., Buyya, R.: Latency-aware Application Module Management for Fog Computing Environments. ACM Transactions on Embedded Computing Systems 9(4), 1–21 (2017)
13. Newaz, S.S., Lee, G.M., Uddin, M.R., Mohammed, A.F.Y., Choi, J.K., et al.: Towards realizing the importance of placing fog computing facilities at the central office of a pon. In: Proceedings of the 19th International Conference on Advanced Communication Technology (ICACT). pp. 152–157 (2017)
14. OpenFog Consortium: Out of the Fog: Use Case Scenarios pp. 0–9 (2016)
15. OpenFog Consortium: Out of the Fog: Use Case Scenarios pp. 1–8 (2017)
16. Skala, K., Davidovic, D., Afgan, E., Sojat, Z.: Scalable Distributed Computing Hierarchy : Cloud , Fog and Dew Computing. Open Journal of Cloud Computing 2(1), 16–24 (2015)
17. Stojmenovic, I., Wen, S.: The Fog Computing Paradigm: Scenarios and Security Issues. vol. 2, pp. 1–8 (2014), https://fedcsis.org/proceedings/2014/drp/503.html
18. Taneja, M., Davy, A.: Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm. pp. 1222–1228 (2017)
19. Westbase Technology: Fog Computing vs Cloud Computing : What's the difference ? (2017), http://www.westbaseuk.com/news/fog-computing-vs-cloud-computing-whats-the-difference/
20. Ye, D., Wu, M., Tang, S., Yu, R.: Scalable Fog Computing with Service Offloading in Bus Networks. In: Proceeding of the 2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud). pp. 247–251 (2016), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7545926