BIG, MEDIUM AND LITTLE (BML) SCHEDULING IN FOG ENVIRONMENT

Bashir Yusuf BICHI[1,+], Saif UI ISLAM[1], Anas Muazu KADEMI[3]

[1]COMSATS University, Department of Computer Science, Park Road, Islamabad, Pakistan

[2]Yasar University, Department of Computer Engineering, No 37-39 Bornova, Izmir, Turkey

bbichi2009@kustwudil@.edu.ng, saiflu2004@gmail.com, anas.kademi@yasar.edu.tr

Abstract

Fog computing has got great attntion due to its importance especially in Internet of Things (IoT) environment where computation at the edge of the network is most desired. Due to the geographical proximity of resources, Fog computing exhibits lower latency compared to cloud; however, inefficient resource allocation in Fog environment can result in higher delays and degraded performance. Hence, efficient resource scheduling in Fog computing is crucial to get true benefits of the cloud like services at the proximity of data generation sources. In this paper, a Big-Medium-Little (BML) scheduling technique is proposed to efficiently allocate Fog and Cloud resources to the incoming IoT jobs. Moreover, cooperative and non-cooperative Fog computing environments are also explored. Additionally, a thorough comparative study of existing scheduling techniques in Fog-cloud environment is also presented. The technique is rigorously evaluated and shows promising results in terms of makespan, energy consumption, latecny and throughput.

Keywords: Cloud node, Fog node, Max-Min, Min-Min, Big, Medium, Little, Task, Resource, Cooperative and Non-Cooperative Systems.

[+] This paper has been presented at the ICAT'20 (9th International Conference on Advanced Technologies) held in Istanbul (Turkey), August 10-12, 2020.

1. Introduction

In todays world, distributed computing is continuously drawing attention with the sole purpose of bringing computing resources closer to clients. The advancement of this idea leads to the introduction of grid computing which later advances to cloud computing, and now even closer, by what is called fog or edge computing. Cloud and Fog Computing are two terms that are almost the same, the main difference in between the two is that fog computing was introduces to bring computing resources closer to needing computing environment (i.e. Clients and IoT devices) for task processing. Bringing the two ecosystems (i.e. Cloud and Fog) together adds efficiency and reduces delay in processing task as compared to sending task to far distant environment (Cloud). Every day billions of tasks are generated, and each task needs to be processed in the shortest possible time. Therefore, the need for Fog has become a necessity to provide clients with efficient and timely task processing capabilities. According to [16] Fog computing has some outstanding advantages which include real-time processing of task, it also bridges (IoT) with the internet computing infrastructure. Another major advantage is that it reduces the latency and improves quality of service (QoS) of a server by bringing the computation services, storage and networking services closer to the edge [20]. These advantages give clients an environment to get access to various resources at earliest possible time and highly efficient manner. For Fog to attain efficiency it needs to collaborate with the cloud environment or other fog environments which is required to introduce the concept of load sharing between fog nodes or fog and cloud environment. Fog computing plays a vital role.

Toward reducing delay in providing service to the IoT [1] or clients in need of computing resources, for example mobile devices (such as those along the highways) demand high quality streaming via proxies and access points position along the highway and other places from a nearby Fog node [3], therefore Fog nodes must be equipped with such high-quality streaming capabilities to meet the client demand. Bringing the edge closer to the IoT nodes is an important milestone towards minimizing delay in processing a given request, as stated in [3]. Fog is the best solution because it is considered more efficient and easy to access. Though we have seen the advantages, the edge needs to balance the set of requests and tasks along its resources for effective use of such resources. According to [5][9], load balancing is necessary to enhance the overall performance of the distributed environment. Multiple load balancing techniques were introduced with the objective of giving each node a fair computing time. There are different load balancing algorithms proposed in [17] and [19], in the second algorithm the technique allows for requested data to move from one tier to another i.e. the IoE (Internet of Every thing) tier to Fog tier and to Cloud tier until the request finds a node that will process the data, while the former balances load using a technique known as graph re-partitioning. Having seen the importance of Fog towards providing efficient services, these data needs to be secured and free from intrusion, [18] [21] discusses some challenges in the link between the Fog and IoT, these challenges include authentication/access control where there are issues regarding the security of data along the Fog node, other issues include man-in-the middle attack. In the authentication issue the adversary which is often called malicious user may change their smart meter or spoof an IP address, while the man-in-the-middle attack may temper with the gateway services of the Fog environment. All these threats are not predominant, but are considered as a potential issue that may either disrupt services to the IoT or may increase the latency level in which the user may spend trying to get services.

1.1. Motivation

Our approach was to build a scheme based on Cloud level max-min, min-min scheduling scheme. The approach goes beyond the concept of Cloud by lowering d o w n to the Fog level by extending the mentioned schemes into what we call Big, Medium, and Little (BML) Scheduling in Fog Environment". Load balancing is very important when handling a heterogeneous environment that processes a request or task sent from a remote IoT to a given server, it also helps to enhance the performance of a system. The 'loadh balancer' receives request from IoT or any other device that send a request and tries to balance these series of request across different resources (VM) that are within the system [24]. Many task scheduling algorithms were introduced in the cloud environment with the aim of balancing the load across different resources of the server such as in [20] with the aim of obtaining better resource utilization, among these algorithms are the max-min and min-min algorithm. This paper proposes a scheduling algorithm that was based on the idea of the two mentioned algorithms, and applies these algorithms to the fog environment for efficient task processing. These algorithms will then be compared in terms of their make-span and utilization. As observed in the algorithms proposed in [2] [5] [6] [15] [12] [20] [25] in the cloud environment which proved very effective in improving the environment, but the algorithms were not tested in the Fog environment to ascertain its behaviors. Therefore, we rigorously evaluated the algorithms to investigate their behavior in fog environment, at the same time we compared these algorithms with our proposed scheme with the aim of observing how the algorithm works in term of its make-span and utilization rate. One of the main reason of fog environment is to bring computing resources closer to the needy environment so that those tasks in need of processing time will be processed quickly, and not to be deprived of a fair processing time. the scheme we proposed helps by providing a task with a moderate processing time as well as timely completion of the task. The proposed algorithms contribute immensely in the fog environment in the following:

1. Reducing make-span: The make-span determines the maximum time at which all the resources will complete executing a given task, therefore reducing the maximum make-span implies reducing the time a task will wait seeking computing time.
2. Higher Utilization rate: Making the system as busy as possible is another important issue, as cost of processing can yield to more profit to the providers.
3. Fair Processing Time: The algorithm also give task a fair processing time, i.e. a task is not deprived of and not given too much processing time as well, this can translate into user and providers concession.
4. Cooperation between Cloud to Fog and Fog to Fog environment for computing resources.

The rest of this paper is organized as follows; Task Scheduling algorithms was discussed in section 3, section 4 focuses on transmission and propagation delay in an environment that consist of cooperative and non-cooperative system of task processing between fog-to-fog and/or fog-to-cloud node directly. Section 5 discussed on the mathematical models that were used to derive the relationship between the edge not i.e.

fog and the upper node i.e. cloud. Lastly section six discussed on the simulated result, discussions, conclusion and remarks.

2. Related Works

Since the advancement of grid computing which yields to cloud, researchers have been putting efforts toward finding an efficient strategy in handling the ever-growing demand of resources, there are number of things being done. The research work is grouped into two context: scheduling scheme context and environmental context. In the scheduling context, our work was build based on max-min and min-min scheduling algorithms in cloud environment which aimed at providing an appropriate scheduling scheme for a set of task for the cloud resources. [2][5][6][12][15][20][25][26] work on the basic idea of max-min or min-min algorithm or both but in the context of cloud environment with the aim of providing appropriate scheduling scheme for the set of tasks in the cloud environment. [10][23] this leads to the algorithms that were used as motivation to the fog environment. The table (table 1) examines the different scheduling algorithms that were in one way or another related to our work on scheduling scheme for fog environment. [10] proposes a scheme that work by predicting the completion time of a task on dynamic and static mode of allocation to the available resources while [23] proposed another scheme known as multi-tenant Load Distributed Algorithm where task are allocated based on priorities. All the related works are examine and described in Table 1.

Table 1. Related work

Related work	**Problem/Environment**	**Technique used**	**Advantage**	**Limitation**	**Future work**	**Tool**
[2]	Comparison of different scheduling algorithm in Cloud Environment	Min-Min, Max-Min MCT, MET	Based on the compared criteria, Min-Min perfumes in terms of makespan, degree of in-balance and throughput.	Max-Min and Min-Min algorithm are suitable for small scale task scheduling Min-min also gives higher priority to smaller task while max-min give higher priority to	Improve min-min algorithm by optimizing cost for task scheduling	CloudSim

				larger tasks		
[5]	Resource management for Cloud Environment	Max-Min, Proposed Max-Min,	Produce lower makespan	Small scale task processing		MATLAB
[6]	Min-Max algorithm in Cloud Environment	Min-Min, Min-Max Max-Min	Improved the resource utilization	Despite pair tasks are processed at a time larger task may still be paired together	Price of resources Energy consumption	Java Programming
[10]	Priced Time petri net in Fog environment	Predict time cost and prices cost of a task	Improved efficiency of resource utilization	Prediction may not give accurate result, mean small task can have higher cost while larger one may have lower cost.	Extend resource allocation strategy, average completion time, fairness.	Dawn parallel machine and Linux cluster Extend
[12]	Cluster based Max-Min for Cloud Environment	Cluster based Max-Min algorithm, Improve Max-Min, Enhance Max-Min	Produce lower makespan than Improved Max-Min and Enhance Max-Min	Larger task gets to be processed first than the smaller once	K-means clustering, Fuzzy C-mean clustering	CloudSim
[15]	Improve Max-Min algorithm for Cloud Environment	Min-Min Max-Min Improved Max-Min	Improve utilization and performance of the system, larger tasks are handled by slower resource while smaller once are handled by	larger tasks are assigned to slower resource which all adds up to improving scalability, availability and stability of	Not mentioned	CloudSim
[20]	Modified Max-Min for Cloud Environment	Max-Min Modified Max-Min algorithm	Improve utilization and performance of resource	As in [15] larger tasks are assigned to slower resources	improve utilization, performance	CloudSim
[23]	Load Distribution in Fog Environment	mtLDAF	improved efficiency of resources. Task are sent to resource	improving load balance across Fog-Cloud layers	Not mentioned	Java

			based on priority			
[25]	User-Priority Guided Min-Min Scheduling Algorithm in Cloud Environment	LBIMM and PA-LBIMM	Reduce makespan and produce better performance	priority customers enjoys better services when compared to the general users. Deadline of a task, high heterogeneity of interconnected network, geographical location of task to improve PA-LBIMM	Not mentioned	MATLAB
[26]	Enhancing Load balance in Cloud Environment	LBMM and ELBMM	Better makespan and resource utilization	Larger task may occupy a resource for a long time.		CloudSim

3. Proposed Big, Medium and Little Scheduling

Presently there aree over 20 billion networks devices across the [4], and these devices will work effectively when connected via efficient and reliable host like Fog and Cloud ecosystem. Therefore, for efficient and better services a proper scheduling technique will be needed to assign different task to a given resource. Task schedulings are categorized into two [22]– static and dynamic. In static the task information is known prior to the scheduling, while in dynamic task are assigned to a given resource as they arrive i.e. without any prior knowledge of the arriving task. Scheduling algorithm for Fog environment was proposed in [10] where the user has the upper hand in choosing a resource from the group of pre-allocated resources autonomously,: this algorithm is called Price Time Petri Nets (PTPN). Scheduling algorithm is aimed at minimizing the completion time of a given task [10]. Another algorithm proposed for the Fog environment is the multi-tenant load distribution algorithm [23], the proposed load balancing algorithm considers two key parameters (delay and priority) when a task is sent to the fog environment,. The algorithm is supposed to minimizes delay and at the same time increases the utilization of resources. In this paper we proposed a load balancing algorithm for the Fog environment based on some existing schemes in the cloud computing environment. The idea of scheduling task was to make appropriate assignment of task to a resource (which could be virtual resource) in case of fog or cloud environment. Balancing the load on these nodes can help in minimizing delay and as well allow for higher utilization of such resource. The main purpose of Fog is to bring a processing server close to the user (i.e. IoT or client) this allows for a greater efficiency as stated in [3], therefore it will be even more efficient when these nodes cooperate with each other. Different scheduling algorithms were discussed in [2] each with the purpose of minimizing the makespan of a resource. Makespan is a measure of throughput among set of computing resources. The makespan is considered as a queue which hold request or task that need to be processed [15], therefore reducing the makespan time will be necessary to minimize the delay in which such task will wait until the time it will be executed. [2] [12] discussed a set of task scheduling algorithms in the cloud environment and these algorithms are the bases for our proposed scheme for Fog environmentThe algorithms are;

a. Max-min: this scheduling scheme always assign task with maximum expected completion time to a resource that give minimum completion time of that given. There are other schemes that are based on max-min algorithm which aimed at reducing the makespan, for instance improved max-min algorithm as implemented in [14] gives lower makespan when compared to the max-min algorithm. The Max-min Algorithm as given i n [22] select the task that requires long processing time and assign it to a resource while the smaller task wait till all the larger task are completed.
b. Min-Min Scheduling: in this scheme task with minimum completion time is always assigned to its corresponding resource. The scheme starts with a set of unscheduled tasks then it determines the minimum completion time for each task on all resources, the task is then assigned to the resources or machine that give the list completion time [8] [25].

There are many other task scheduling algorithms that were proposed, however we restricted ourself to the algorithms mentioned above since our proposed schemes only covers some unique characteristics of the Cloud schemes. All these algorithms are employed in the cloud environment and our focus is on fog node in relation to the cloud node, so we take the idea of the algorithms and use in fog environment. The algorithms reduce the maximum makespan of a given resources that is processing a set of tasks. The makespan is nothing but the measure of throughput of a resource. The pseudo code below in algorithm 1, algorithm 2, and, algorithm 3 depict the nature of our proposed Big, Medium and Little Scheduling scheme in Fog environment.

```
For all tasks submitted to the meta-task Ti
For all resource Rj
c(ij) = E(ij) + rj
while meta-task is not empty do
    Find n=number of all minimum values less than Tm
    Find set of all minimum values less than Tm
    Take Tmmod  n= Tn
    Assign task Tn to resource Rn, n <> m
    Remove task Tm form the meta-task
    Update rj for selected Rj
    Update C(ij) for all task
end
```

Algorithm 1: Medium Scheduling Algorithm

```
For all submitted tasks in meta-task Ti
For all resources Rj
C(ij) = E(ij) + rj
while meta-task is not empty do
  Find task Tk consumes maximum completion time.
  Assign Tk to the resource Rj which gives minimum execution time
  Remove Tk from meta-tasks set update rj for selected Rj       update
  C(ij) for all task
end
```

Algorithm 2: Big Scheduling Algorithm

```
For all submitted tasks in meta-task Ti For all resource Rj
C(ij) = E(ij) + rj
while meta-task is not empty do
  Find the task Tk consumes maximum completion time.
  Assign task Tk to the resource Rj with minimum execution time.
```

```
    Remove the task Tk from meta-tasks set
    Update $r_j$ for selected $R_j$ Update $C_{(ij)}$ for all task
end
```

Algorithm 3: Small Scheduling Algorithm

The algorithm allows for the resultant n modulo value to assign the task to the resources that give minimum completion time when compared to the maximum completion time. The advantage is to give the task a moderate time it may require in executing the given task at hand. In this paper we will investigate the makespan, the utilization rate of a resource and energy consumption of a given resource, the energy consumption is discussed in [7] and shown in equation (4), makespan and average resource utilization are two very important metrics when handling scheduling algorithms, as pointed out in [12] [13] [25] the makespan and average resource utilization are define using the following mathematical relations;

$$makespan = m_{ax}(C_{i,j}) \qquad (1)$$

$$Resourceusage(RU) = \frac{R_t}{Makespan} \qquad (2)$$

$$Averageutilization = \sum_{i=1}^{m} \frac{RU}{M} \qquad (3)$$

$$E_i = (P_{max} - P_{min}) * U_i + P_{min} \qquad (4)$$

Were P max is the peak power consumption and P min is the minimum power consumption, U is the utilization of a given resource. The graphs below describe how the three (Big, Medium, and Little) schemes works in the Fog environment. We consider a set of five randomly selected tasks and resource (T1=200, T2=250, T3=150, T4=300 and T5=100) in MI and three resources (R1=50, R2=100, and R3=40) in MIPS then the meta-task table is as follows;

The Big scheme works by taking the maximum among the set of tasks e.g. the maximum value in task 4 (T4) is 7.5. Therefore, we take 7mod2=1, 2 is the number of all the task on task 4 that are less than 7.5 i.e. 6,3, now if from the resulting value we take 6, but since our resulting value is 1 we take 2. We then go back to our initial meta-task and find in what resource 2 falls. finally that resource will be picked. In the example given, the resource R2 will be assigned the task. The graph below shows how each task is assign to a given resource based on the meta-task in Table II.

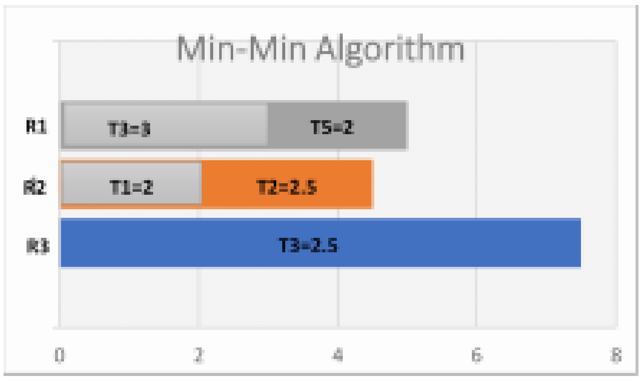

Fig. 1. Gantt chart of Little Scheme

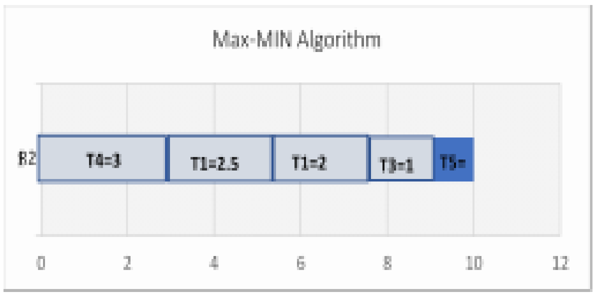

Fig. 2. Gantt chart of Big Scheme

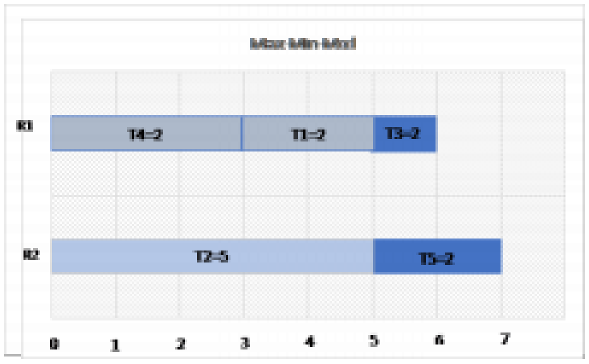

Fig. 3. Gantt chart of Medium Scheme

Based on the above Gantt chart in the figure (fig.1), (fig.2), and (fig.3), the little scheme produced a makespan of 7.5; Big scheme produced a makespan of 10 seconds and our medium scheme produced a makespan of 7 second. Hence, from the results we conclude that the medium scheme gives less makespan and at the same time gives each task a fair amount of time that the task may need to fully utilize the resource.

Table 2. Meta-Task

	R1	R2	R3
T1	4	2	5
T2	5	2.5	6.25
T3	3	1.5	3.75
T4	6	3	7.5
T5	2	1	2.5

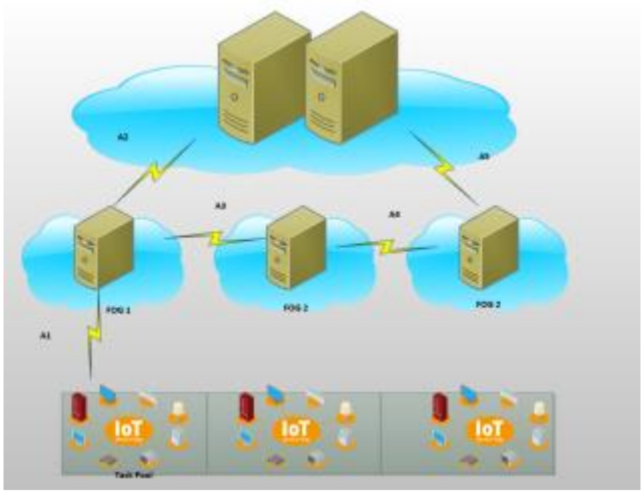

Fig. 4. -A1-A3-A4-A5: Cooperative Systems. A1-A2: Non-cooperative.

4. Cooperative and Non-Cooperative Systems

The main purpose of Fog is to bring computing resources closer to the IoT, or the client, in this section we presented what is called cooperative and Non-Cooperative system mode of operation, this scenario is module as a set $N_c = (F_1, F_2, .. F_n, C)$ for the Cooperative Systems and $N_n = (F, C)$ for the Non-Cooperative Systems. In Cooperative system the Fog environment get to collaborate with its neighboring Fog node for computing resources. The conceived idea is presented in the Figure (fig.4). with the purpose of allowing the fog to contact its neighboring fog whenever it needs external computing resources. For a set of n tasks equation (5) and (6) show how these set of tasks will be processed at each layer level.

In Cooperative system the Fog environment get to collaborate with its neighboring Fog node for computing resources, for instance if we have n number of task to be process in h Fog environment and potentially the cloud environment, these tasks will be sent from the pool of resources as shown in the figure above (fig.4). tThe system model below shows how all the tasks will be process across the cooperative fog environment.

$$T(n) = \begin{cases} \sum_{i=1,\ \ j=1}^{i=4, j=x} F_i(j), & For\ 1 \le i \ge 4,\ 1 \le j \ge x \\ \\ C_l(j), & For\ x < j \le n \end{cases} \tag{5}$$

The conceived idea presented in the figure above (fig.4-A1- A3-A4-A5) allows the edge to send task to its neighboring fog whenever it needs external computing resources. The flow chart below explains further how the task will be handle. If a node has C capacity, that node will select task to its maximum capacity and sent the rest to its neighboring Fog node until the task if there is any reached the Cloud environment as describe in the flow chart in the figure (fig.5).

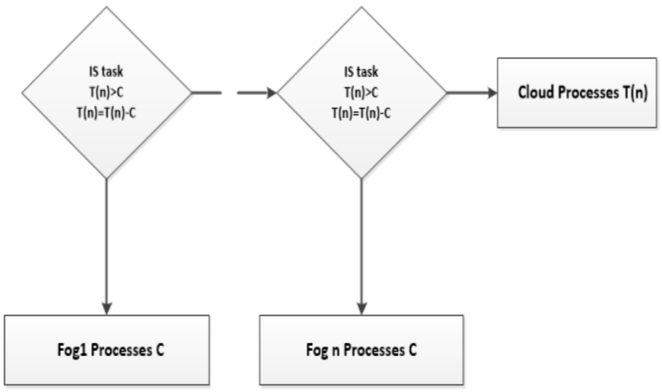

Fig. 5. Cooperative Systems

The Non-Cooperative Systems is the second approach where fog node processes the requested task and possibly sends some to the cloud environment as shown in equation (6). The idea of fog collaboration with its neighboring fog node for computing resources was discussed in [1] but with a slightly different approach.

$$T(n) = \begin{cases} F_l(j), & For\ 1 \le i \ge x \\ \\ C_l(j) & For\ x < j \le n \end{cases} \tag{6}$$

The figure (fig.2-A1-A2) below depicts the non-cooperative systems, fog environment will select tasks according to its capacity and send the rest are send to the cloud environment as indicated in the task flow in the figure (fig6).

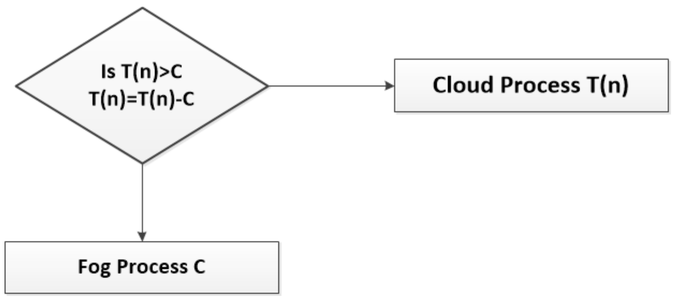

Fig. 6. Non- Cooperative Systems

5. Discussion and Mathematical Formulation

For the environment showed in figure (Fig.4), some metF rics are considered which include the propagation delay and transmission delay between each node. As earlier mentioned, the simulation environment is assumed to be cooperative and non-cooperative. The link from the task pool to the nearest Fog node is L distance and another L distance to the next nearest Fog, the Last nearest node is also a distance apart from the Cloud node by another L distances. Therefore, the transmission and propagation delay of each link needs to be computed. $T_d^{F(i-1)F_i}$, and $T_p^{F(i-1)F_i}$: are the Transmission and Propagation delay between Fog node i and $i-1$ respectively. L_f and L_c: is the average latency at Fog and Cloud node respectively as shown in equation (7) and (8) respectively. *AvrgTh*: is the average throughput of the link as shown in equation (9), DF_i: Average processing delay by a resource at Fog node i as shown in equation (10), D_C: Average processing delay as shown in equation (11) by a resource at Cloud node. T_{pro}: Average processing time at Fog or Cloud Node.

$$L_f = \sum_{1=1}^{n} T_d^{F(i-1)F_i} + \sum_{1=1}^{n} T_p^{F(i-1)F_i} \qquad (7)$$

$$T(n) = L_f + T_p(F_iC) + T_d(F_iC) \qquad (8)$$

$$AvrTh = \frac{averagefilesize}{T_{T^*}(2T_p)} \qquad (9)$$

$$DF_i = L_f + T_{pro} \qquad (10)$$

$$D_C = L_c + T_{pro} \qquad (11)$$

3. Simulation and Results

We evaluate our proposed scheme by randomly sending set of tasks to the Fog environment in a uniform distribution pattern from the task pool. wWe assumed the pool to send 500, 1000, 1500, 2000, 2500, and 3000 set of tasks. The set of tasks size ranges from 50KB to 2000KB, The Fog nodes were assumed to be IEEE 802.11a/g which have link rate of 100 Mbps each and that of the Cloud environment is assumed to be 1Gbps. And at each simulation we assumed the nodes to have the following processing capacity 100, 200, 300, 400, 500, and 600 tasks at a time. we assumed five resources each with computing power as follows; R1=200, R2=150, R3=100, R4=245, and R5=270, the simulation environment is MATLAB R2014b. The results of our simulations are given as follows;

- Makespan: As explain earlier makespan is a measure of throughput of all the resources of a given node. The graph in figure (fig.7) shows the makespan at each level of execution for the set of tasks that were sent from the task pool to the fog environment for the three algorithms i.e. the Big algorithm, the Medium algorithm and the Little algorithm for the fog environment. Based on the graph in figure (fig.7), Medium algorithm obtained less makespan when executing a given task as compare to the other algorithms.

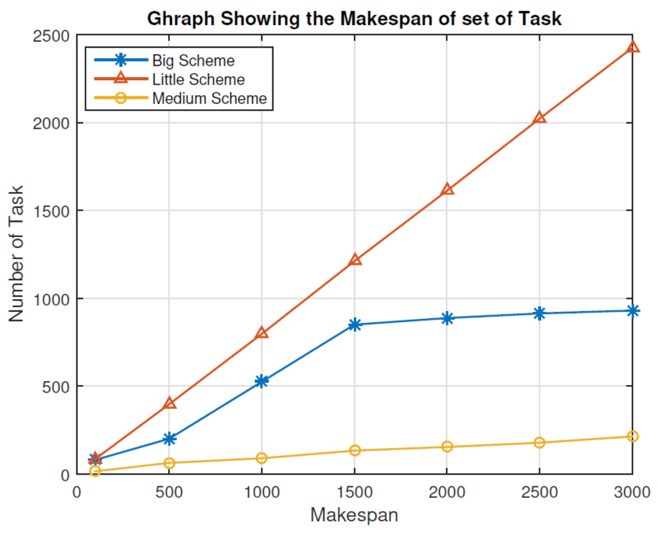

Fig. 7. Graph of Makespan for the three algorithms

- Utilization: The Utilization rate from equation (3) shows the level at which computing resources are utilized throughout the period at which tasks are processed, the figure (fig.8) shows the graph of utilization of the three algorithms. From the result obtained in fig.8 the utilization rate of medium algorithm is greater than the other two. Based on the figure (fig 8) can conclude that max-min-mod algorithm make used of almost all the available resources within, the system which make the environment fully utilized unlike in other algorithm as stated in [6] that big algorithm has a very low utilization rate.

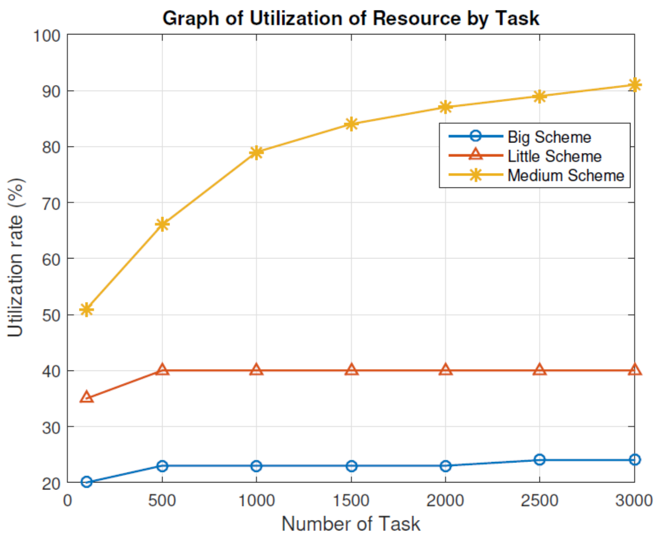

Fig. 8. Utilization rate of the three Algorithm

- Energy Consumption: In the simulation and based on equation (4) we used some predefined values that shows Load consumption and power values (in watt) that the resources (Virtual) will contribute in the overall power consumption of the system, the server predefined values are from Hp ProLiant G4 86 server. The figure (fig.9) below shows that energy consumption contributed by the resources for our three algorithms.

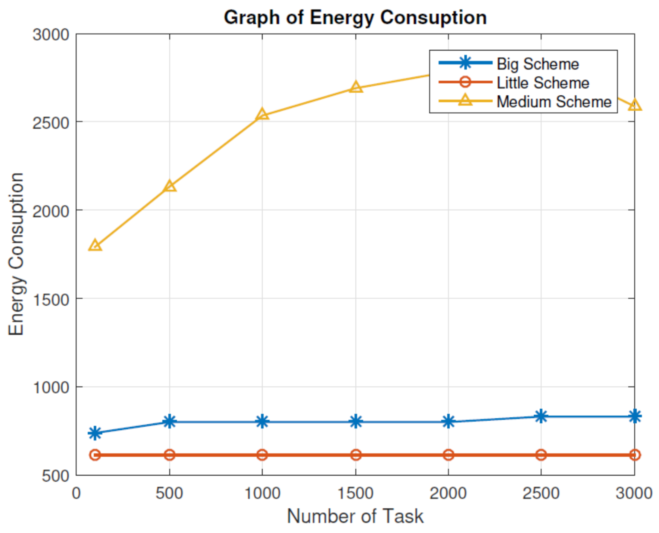

Fig. 9. Energy Consumption of the three Algorithm

The graph of energy consumption in the Fig. 9 indicates that medium scheme algorithm contributes more to the energy consumption of entire system. This is because the medium algorithm has the tendency of using more resources in some case as compared to the big and little algorithms.

A. COOPERATIVE SYSTEMS AND NON- COOPERATIVE SYSTEMS

As stated that the simulation is based on two systems i.e. cooperative, where a given fog node collaborate with its neighboring node for computing resources and non-cooperative were fog node collaborate with the cloud directly. The scenario is that if a fog node received set of task from the pool, it picks tasks to its capacity and offloads the remaining to the nearby fog nod or cloud. we investigate the relationship between the fog nodes and cloud node in terms of the latency of the link which involves the transmission and propagation delay of each link. We also consider the links directly and ignore other routing devices that may be found between the links, as we are interested in node to node communication only. The results below are based on the medium algorithm for task scheduling, the link rate of each fog nod is considered to be 100 Mbps and 1 Gpbs for the Cloud environment. wWe also assumed a distance between the pool of task to the first fog not to be 10Km, and other fog environment are separated by 30KM from each other, and the Cloud environment is at 500Km from the last fog environment.

1. *Average Processing Delay*

Figure fig.10 shows the average processing time of set of tasks by a given resources in all the Fog environments and that of the cloud environment. The nodes process these set of tasks almost concurrently, this collaboration between the nodes give all the task a fair time in which it will be processed completely, and another issue is that the task independently processes by each Fog node.

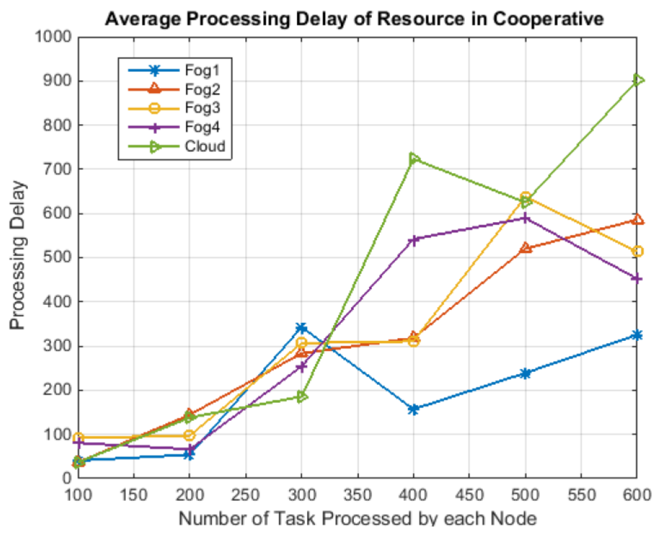

Fig. 10. Average Processing Delay

In the Non-cooperative systems, the Fog node sends task directly to the Cloud environment for processing,in the figure (Fg.11) the Fog sends remaining task directly to the cloud environment, and from our results the time that the tasks are processed in somewhat less as when compared to the time the task will wait at the Fog environments seeking for processing. Therefore, we can directly assume that in this case Non-Cooperative System works more efficiently and with less time consumption than the Cooperative Systems.

2. Latency and Throughput: Cooperative and Non-Cooperative

As stated the latency of the link is the time a task takes before it gets to the processing stage, while the utilization of the link is the rate at which the link to a given node is been utilized. Based on the result of our simulation in the figure (Fig.12), the latency at Cooperative system is higher when compared with that of the Non-Cooperative Systems. However, the graph also indicated the difference between the two environment is negligible but for a large stream of tasks the Cooperative System can be considered more suitable than the the Non-Cooperative Systems.

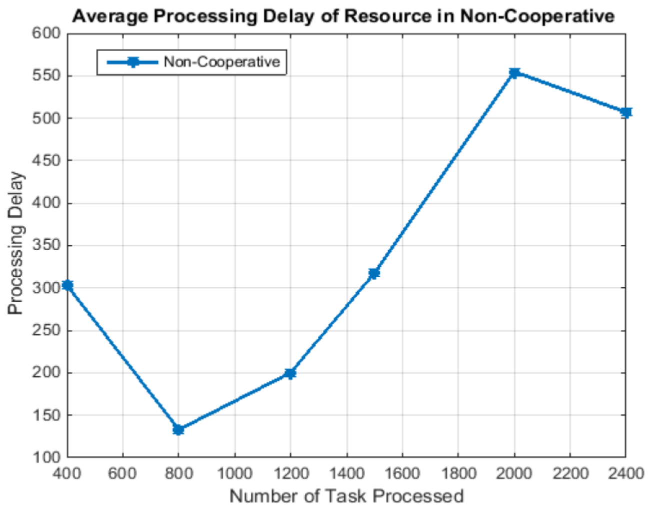

Fig. 11. Average Processing Delay: Non-Cooperative System

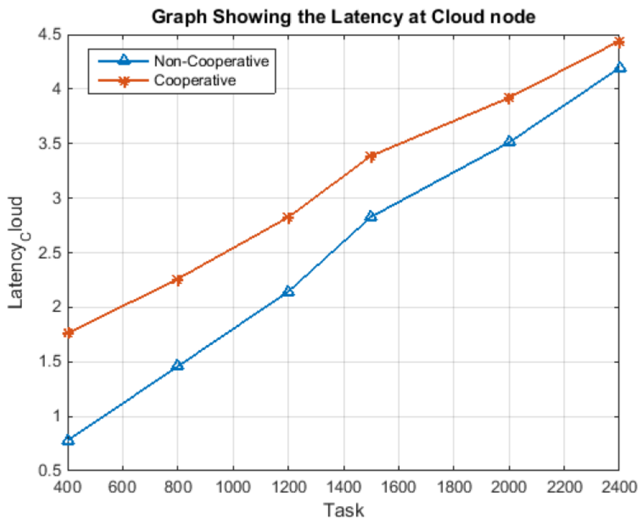

Fig. 12. Latency: Cooperative and Non-Cooperative Environment

The throughput of the link is given below in figure (Fig.13) and based on the results we find that the level at which the Cooperative System is utilized is far higher than that of the Non-Cooperative Systems. Maximum Utilization of a System is the interest of service providers, therefore the Cooperative Systems in this regard is far better than the Non-Cooperative Systems.

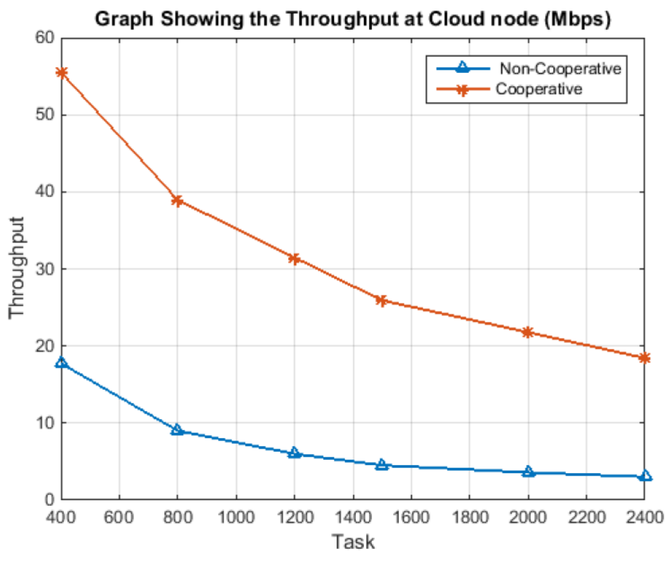

Fig. 13. Throughput: Cooperative and Non-Cooperative Environment

6. Conclusion

Efficiency in executing a task is the state of the art for cloud service providers and due to the ever-growing pool of tasks that is always getting larger and larger by the day, it becomes necessary for the service providers to look for efficient techniques to handle these tasks. Because of the geographical proximity of resources, Fog computing exhibits lower latency compared to cloud computing and inefficient resource allocation in Fog environment can result in higher delays and degraded performance. Efficient resource scheduling in Fog computing is crucial to get true benefits of the cloud like services at the proximity of data generation sources. The Big-Medium-Little (BML) scheduling technique efficiently allocate Fog and Cloud resources to the incoming IoT jobs. Our proposed scheme together with the idea of Cooperative and Non-Cooperative Systems timed the ever-growing pool of tasks by collaboration between the Fog and Cloud environments. The technique as evaluated, shows an improved result in terms of makespan, energy consumption, latency and throughput: an improved max-min and min-min scheduling algorithms in cloud environment which provide an appropriate scheduling scheme for a set of tasks for the cloud resources. In essence, the efficiency can be achieved by putting a very efficient scheme that requires less time to complete a given task, and helps in providing fast and real-time task executing within a limited amount of time.

References

[1] Ashkan Yousefpour, Genya Ishigaki, and Jason P. Jue. Fog Computing: Towards Minimizing Delay in the Internet of Things, 2017 IEEE 1st International Conference on Edge Computing, Honolulu, HI, USA, Jun 2017,pp 17-24, DOI: 10.1109/IEEE.EDGE.2017.12.

[2] Syed Hamid Hussain Madni, Muhammad Shafie Abd Latiff, Mohammed Abdullahi, Shafii Muhammad Abdulhamid, and Mohammed Joda Usman. Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment,*PLoS ONE* 12(5), Aug 2016. https://doi.org/10.1371/journal.pone.0176321.

[3] Mohammad Aazam and Eui-Nam Huh. Fog Computing: The cloud IoT/IoE middleware paradigm, *IEEE Potentials*, Volume:35, Issue:3, pp 40-44, DOI: 10.1109/MPOT.2015.2456213.

[4] Lin Gu, Deze Zeng, Song Guo, Ahmed Barnawi, Yong Xiang. "Cost Efficient Resource Management in Fog Computing Supported Medical Cyber-Physical System," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no.1, pp. 108-119,Jan-March 2017, doi:10.1109/TETC.2015.2508382

[5] Bashir Yusuf Bichi, Tuncay Ercan, and Anas Muazu Kademi. An Efficient Resource Management in Cloud Computing, *International Conference on Advanced Technology and Sciences, 3th International Conference, ICAT16 Konya, Turkey* Sept 01-03, 2016 Proceedings pp.1-6.

[6] Zhou Zhou and Hu Zhigang. Task Scheduling Algorithm based on Greedy Strategy in Cloud Computing,*The Open Cybernetics and Systemics Journal*, Sep 2014, pp 8-11.

[7] Young Choon Lee Albert Y. Zomaya. Energy efficient utilization of resources in cloud computing systems, *Journal of Supercomputing* (2012) 60:pp 268280, Springer Science+Business Media, LLC 2010, DOI 10.1007/s11227-010-0421-3.

[8] Bo Li, Yijian Pei1, Hao Wu1, and Bin Shen. Heuristics to allocate high-performance cloudlets for computation offloading in mobile ad hoc clouds, J Supercomput DOI 10.1007/s11227-015-1425-9 Springer Science+Business Media New York 2015.

[9] A Saif ul Islam, Jean-Marc Pierson, and Nadeem Javaid. A Novel Utilization-aware Energy Consumption Model for Content Distribution Networks, *International Journal of Web and Grid Services* June 2016. DOI: 10.1504/IJWGS.2017.085146.

[10] Lina Ni, Jinquan Zhang, Changjun Jiang, Chungang Yan and Kan Yu. Resource Allocation Strategy in Fog Computing Based on Priced Timed Petri Nets, *IEEE Internet of Things Journal* Volume: 4, Issue: 5, Oct. 2017, pp1216 - 1228, DOI: 10.1109/JIOT.2017.2709814.

[11] Xie Z, Shao X, Xin Y *(2016) A Scheduling Algorithm for Cloud Computing System Based on the Driver of Dynamic Essential Path.* PLoS ONE 11(8):Aug 2016, e0159932. doi:10.1371/journal.pone.0159932

[12] Zonayed Ahmed, Adnan Ferdous Ashrafi, and Maliha Mahbub. Clustering based Max-Min Scheduling in Cloud Environment, *(IJACSA) International Journal of Advanced Computer Science and Applications* Vol. 8, No. 9, 2017.

[13] Yiqiu Fang, Fei Wang, and Junwei Ge. A Task Scheduling Algorithm Based on Load Balancing in Cloud Computing, *2010 International Conference on Web Information Systems and Mining (WISM 2010) Oct 2324, 2010, Sanya, China* pp. 271277, 2010. Springer-Verlag Berlin Heidelberg 2010.

[14] S. DEVIPRIYA, and C. RAMESH. IMPROVED MAX-MIN HEURISTIC MODEL FOR TASK SCHEDULING IN CLOUD, 2013 International Conference on Green Computing, Communication and Conservation of Energy (ICGCE) Chennai, India Dec 2013 pp. 883- 888 EEE 2013.

[15] Shubham Mittal and Avita Katal. An Optimized Task Scheduling Algorithm in Cloud Computing, Advanced Computing (IACC), 2016 IEEE 6th International Conference, Feb. 2016 Bhimavaram, India, pp. 197-202, DOI: 10.1109/IACC.2016.45.

[16] Songqing Chen, Tao Zhang, and Weisong Shi. Fog Computing, IEEE Internet Computing Volume: 21, Issue: 2, Mar 2017, pp 4-6, DOI:10.1109/MIC.2017.39.

[17] SONG Ningning, GONG Chao, AN Xingshuo, and ZHAN Qiang. Fog Computing Dynamic Load Balancing Mechanism Based on Graph Repartitioning, China Communications Volume: 13, Issue: 3, Mar 2016.pp 156 - 164, DOI: 10.1109/CC.2016.7445510, IEEE Apr 2016.

[18] Ivan Stojmenovic, and Sheng Wen. The Fog Computing Paradigm: Scenarios and Security Issues, Proceedings of the 2014 Federated Conference on Computer Science and Information Systems pp1-8, DOI:10.15439/2014F503.

[19] Rajwinder Kaur, and Pawan Luthra. Load Balancing in Cloud System using Max Min and Min Min Algorithm, National Conference on Emerging Trends in Computer Technology (NCETCT-2014) International Journal of Computer Applications (0975 8887).

[20] Kang Kai, Wang Cong, and Luo Tao Fog computing for vehicular Ad-hoc networks: paradigms, scenarios, and issues, The Journal of China Universities of Posts and Telecommunications Volume 23, Issue 2, April 2016, pp 56-65, 96. https://doi.org/10.1016/S1005-8885(16)60021-3.

[21] Arwa Alrawais, Abdulrahman Alhothaily, Chunqiang Hu, and Xiuzhen Cheng, Fog Computing for the Internet of Things: Security and Privacy Issues, IEEE Internet Computing Volume: 21, Issue: 2, Mar.-Apr. 2017, pp 34-42. DOI:10.1109/MIC.2017.37.

[22] Neeta Patil,and Deepak Aeloor. "A Review - Different Scheduling Algorithms in Cloud Computing Environment," Intelligent Systems and Control (ISCO) 2017 11th International Conference Jan. 2017, Coimbatore, India. DOI:10.1109/ISCO.2017.7855977.

[23] Euclides C. Pinto Neto, Gustavo Callou, and Fernando Aires, An Algorithm to Optimise the Load Distribution of Fog Environments, Systems, Man, and Cybernetics (SMC), 2017 IEEE International Conference, Oct 5-8, 2017 Banff Center, Banff, Canada, pp1292-1297. DOI:10.1109/SMC.2017.8122791.

[24] Einollah Jafarnejad Ghomi, Amir Masoud Rahmani, and Nooruldeen Nasih Qade. Load-balancing algorithms in cloud computing: A survey, Journal of Network and Computer Applications 88 (2017), pp 5071, journal homepage: www.elsevier.com/locate/jnca.

[25] Huankai Chen, Frank Wang, Na Helian, and Gbola Akanmu Userpriority guided Min-Min scheduling algorithm for load balancing in cloud omputing, National Conference on Parallel Computing Technologies, PARCOMPTECH 2013 Publisher: IEEE, DOI: 10.1109/Par- CompTech.2013.6621389.

[26] Gaurang Patel Rutvik Mehta Upendra Bhoi Enhanced Load Balanced Min-min

Algorithm for Static Meta Task Scheduling in Cloud Computing, 3rd International Conference on Resent Trends in Computing 57(2015), pp 545-553, https://doi.org/10.1016/j.procs.2015.07.385.