Open Access
Baghdad Science Journal **Vol.16(1)2019**

DOI: http://dx.doi.org/10.21123/bsj.2019.16.1.0130

Developing Load Balancing for IoT - Cloud Computing Based on Advanced Firefly and Weighted Round Robin Algorithms

Marwa M. Abed *Manal F. Younis*[*]

Received 13/8/2018, Accepted 22/11/2018, Published 11/3/2019

Abstract:

The evolution of the Internet of things (IoT) led to connect billions of heterogeneous physical devices together to improve the quality of human life by collecting data from their environment. However, there is a need to store huge data in big storage and high computational capabilities. Cloud computing can be used to store big data. The data of IoT devices is transferred using two types of protocols: Message Queuing Telemetry Transport (MQTT) and Hypertext Transfer Protocol (HTTP). This paper aims to make a high performance and more reliable system through efficient use of resources. Thus, load balancing in cloud computing is used to dynamically distribute the workload across nodes to avoid overloading any individual resource, by combining two types of algorithms: dynamic algorithm (adaptive firefly) and static algorithm (weighted round robin). The results show improvement in resource utilization, increased productivity, and reduced response time.

Key words: Cloud computing, Firefly algorithm, IoT, Load balancing.

Introduction:

Healthcare systems are growing rapidly and getting improved in the recent years. This growing has promoted a high increase in data transmission volume; therefore, a demand for effective techniques to process and manage data has appeared (1). The benefits of this improvement are: 1) diagnosis and 2) prevention of the disease from early stage by using wearable sensors. This is performed by providing data for doctor and patient anywhere and anytime. Healthcare systems that rely on advanced technologies such as Internet of things (IoT) and cloud computing have huge numbers of users. The enormous number of users perform frequently data request from servers; thus, there are numerous overloads on the cloud's VM. Consequently, there is an urgent demand for techniques and algorithms to overcome the overload problem. To handle this problem, load balancing algorithms and transfer protocols are used (2).

IoT

IoT is a concept of a communication among billions of physical devices that are connected to the internet around the world via smart things from anywhere, anyhow, and anytime. There are many advantages of IoT in our lives, which can help persons by connecting many sensors and actuators.

Department Computer, College of Engineering, University of Baghdad, Baghdad, Iraq
[*]Corresponding author: manalfyounis@gmail.com

IoT applications produce large volumes of data and multiple computational processing; thus, they need a big data center for storing; this leads to integrate cloud computing with IoT in order to provide computing infrastructure with cost effective (3).

Cloud computing

Cloud computing is a popular way to access online computing resources and user requirement in a low-cost manner. Cloud computing provides different services to users according to its requirement such as pay per use and on-demand service. These services are achieved by supporting the virtualization way which shows virtualized data centers.

Basically, cloud computing works on 5-4-3 principles consisting of the five essential characteristics, four deployment models, and three service offering models. The five essential characteristics are (4):

1. On-demand self-service: Service's provider provides automatically a server, network storage, and other capabilities of computing to the consumer.
2. Broad network access: Capabilities can be provided without taking care about the client platform either thin or thick, by standard mechanisms through the network.
3. Elastic resource pooling: Client can get dynamic virtual or real resources according to his

specification and can redefine it on demand. These resources are pooled to give the service to several consumers by using the multi-tenant model.

4. Rapid elasticity: Elastically services provisioned automatically to scale out and scale in quickly.
5. The measured service: Is transparent between the client and the service provider, that the cloud offered by control and reports the status of the service's type such as storage, processing, etc.

The four deployment models of the principles are (5):

1. The public cloud: Available to everyone, which offers different types of services such as storage, applications, and VMs. IBM's Blue-Cloud and Google App Engine are examples of public cloud.
2. The private cloud: Cloud provides infrastructure, discrete and secure environment for the consumers in a specific organization that may be managed by the organization itself and/or a third party.
3. The community: Infrastructure is provided to the consumer from multiple organizations. These organizations can manage the cloud or by a third party.
4. Hybrid cloud: Combination of two or more types of cloud (private, public, or community) this makes the entities have the protection of the private and community cloud in addition of connecting with each other or with the word.

The last principle is the three service offering models that explain the pyramid of the service as the following (6):

1. Software as a Service (SaaS): It is an apportionment model where the software is put up by the seller hardware and can be used by the client via the Internet. The service-oriented architecture (SOA) and the web services depend on the SaaS as an implicit technology. In this service, the payment is based on the time or amount the service demand. The SaaS is such as Yahoo Mail, Dropbox and Gmail.
2. Platform as a Service (PaaS): The developer uses this service to build and test their applications by using various languages and tools. The PaaS offer all levels of creating a program without a need to software. Salesforce, Heroku, AWS Elastic Beanstalk, and Microsoft Azure are examples of PaaS services.
3. Infrastructure as a Service (IaaS): The consumers can manage their deployed applications in addition to the storage and the operating systems with restricted of control on the components of the network. IaaS includes the Amazon EC2, Windows Azure, and Google Compute Engine.

Virtualization

Virtualization is a technique utilized to make a single physical infrastructure act as a multiple logical infrastructure or resources. Virtualization takes many forms such as memory, processor, input/output (I/O), network, operating system (OS), data, and applications (7). It can virtualize the server to reduce the requirement of physical resources. In server virtualization, servers can host more than one virtual server simultaneously, which allows the users to reduce the number of servers to be reserved for various purposes and thus increases the resources utilization (8). Servers are required to process large number of concurrent requests in order to handle millions of IoT devices and users' messages. These requests or messages are handled with load balancing (9). In cloud computing the requests for the services should balance the virtual resources. Load balancing reduces the workload and increases the utilization of the resources, respectively (5).

Load Balancing

Cloud load balancing (CLB) is used to distribute the load across all the nodes in the data center. This distribution is performed by transferring the heavily loaded nodes to low loaded nodes to achieve effective resource utilization. CLB ensures that no node is overloaded; thus, the system performance is improved. Efficient load balancing algorithm supports: implementing failover, enhancing response time, enabling scalability, and avoiding bottlenecks (10) (11).

The load balancing algorithms is classified into two types (12):

1. **Static algorithm:** This type of algorithm is based on the prior information predefine all the nodes and their properties. Static algorithm does not consider the current status of the node.
2. **Dynamic algorithm:** This type of algorithm is based on the current system information according to the changes in the state of nodes. The implementation of dynamic schemes is expensive and very complex; however, it balances the load effectively.

The contribution of this paper is to implement two types of algorithms dynamic and static. The Adaptive Firefly Algorithm (AFA) is firstly used to dynamically calculate the capability of virtual machines (VMs). To perform virtual machine scheduling over the data centers for solving load balancing problem in cloud computing, AFA is integrated with Weighted Round Robin

Algorithm(WRRA). Then it can take the benefits of the static and the dynamic at the same time.

Related Works

Due to the fact that cloud resources are shared among millions of users tasks, scheduling these tasks to cloud computing environment requires load balancing. There are different techniques which have been proposed for load balancing in cloud computing; some of them are discussed as below:

Hou et al. (9) present IoT cloud and several applications which are based on it such as smart buildings, smart home/office, intelligent transportation, and smart healthcare. They use two types of load balancer: 1) HTTP load balancer used WRRA, and 2) MQTT load balancer used least connections algorithm load balance, which selects the least number of connections as the target server. It is good to separate the protocols of IoT and HTTP traffic but using static load balance algorithm make it unsuitable when the load changes.

Makasarwala and Hazari (5) explore the deployment of service in cloud computing and the virtualization means with its effect on the cloud in terms of load balance. A genetic algorithm is presented with the dependence of the time to decide the priority of requests as a solution of load balance. This algorithm can handle large size search. In addition, it can solve any type of problems; however, the benefit of its solution gets down when the problem size raises. In addition, the result of this algorithm is changed even if implemented in the same environment.

Gao and Wu (13) explain the cloud computing and the importance of load balance. It presents the dynamic strategies like forward-backward ant mechanism and max-min rules. The implementation of pheromone initialization and pheromone update has been discussed. It is important to decrease the time that consumers take in the search for a node, where this can be by understanding the generation of the ant.

An autonomous agent-based load-balancing algorithm has been presented in (14) which is considered a dynamic load balance. It works by searching for suitable VM, which is selected by the nearest to the threshold value. Load agent starts search for a VM from other data centers. Keeping information of VM earlier reduces service time.

A fuzzy-based method is presented in (15) to classify the VM in the cloud and sorting the task to decrease the consumption of the energy and increase the fault tolerance.

The main aim of makespan is minimization of the Honeybee algorithm presented in (16). It is a dynamic algorithm which is suitable for heterogeneity environment like cloud computing. Cloudsim and its workflow emulations are used to discover the difference in nature between the dependence and independence tasks. However, cloudsim is attained to settle the condition in a small distance of time and easier to design, fuzzy logic is considered hard to make a model and consumes more fine-tuning.

Xavier et al. (17) reduce the makespan using the chaotic social spider algorithm. This word is based on selecting the best appropriate VM for the task of the user. It emulates by cloudsim tooltik. The result shows advancement of the algorithm against other swarm intelligence algorithms such as ABD, PSO, and GA.

Proposed Method

To handle a large number of concurrent users' requests, the virtualization hardware resources in the amazon cloud environment that have been implemented. A server in the IoT cloud can be implemented as VM, also known as instance. This VM uses different components to implement the IoT cloud as shown in Fig.1.

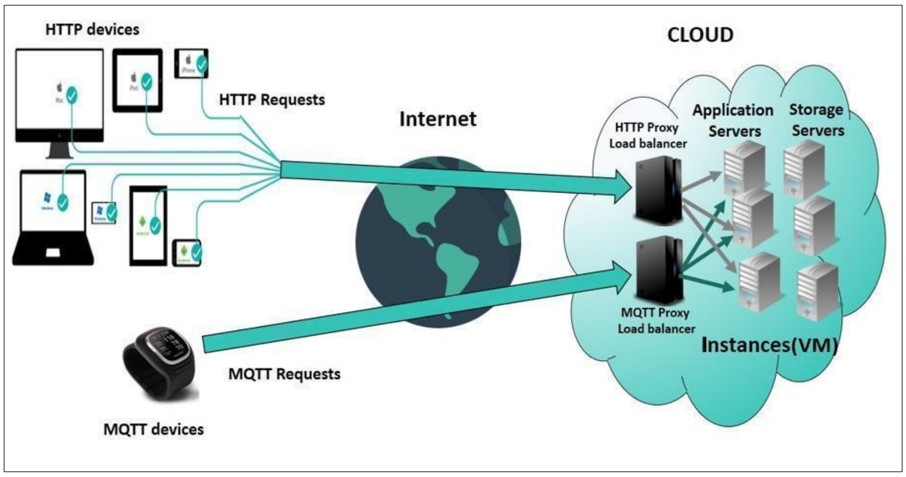


Figure 1. Illustration of the cloud implementation

The VM components are explained as follows:

1. Application server: Cloud provides HTTP and MQTT servers, which in turn provides services for the end-users. These servers are configured as Virtual Machines (VMs), where these VMs run independently on the same physical machine. Servers are developed using Meteor, which is a JavaScript platform is used for rapid prototyping and produces cross-platform (Android, iOS, Web) code, written by Node.js. The HTTP servers react with the client using HTTP protocol based on Request-response cycle, which is used to establish web and mobile applications. HTTP server takes the client request as web then sends the response to the client after processing. Generally, the client request is in three methods, first the GET method to gain a resource, second the POST method to send information from client to server, and third the DELETE to remove a specific resource from the HTTP server. MQTT server is based on a broker for publishing/subscribing message, which is employed the MQTT protocol to publish/subscribe communication. Note that the MQTT messages are considered lightweight. Messages published from the client are subscribed by MQTT servers and inserted in the DB. These messages contain the sensor data which is small packet size.

2. Broker: is considered a part of the MQTT server. All messages are received and published to all subscribers by broker. When multiple brokers are used on a cluster, the haproxy is utilized to distribute MQTT data between these brokers.

3. Database: The proposed IOT cloud utilizes Redis DB using key-value to store the data in-memory database. Redis DB is an open source code, high speed, and NoSQL. In addition, Redis DB increases the reliability because multiple nodes and clustering are used. The data is distributed among Redis DB nodes using a cluster. Therefore, the work is continuous whenever one or more Redis nodes are failed.

4. Proxy Load Balancer: Load Balancer manages cloud resources and decides the distribution of task among the VMs at run time, whenever an idle or least loaded VM is discovered by utilizing the resources. Many algorithms have been proposed to enhance load balancer, such as static WRRA. This algorithm type fairly distributes the load among all VMs. WRR algorithm is considered inappropriate for cloud computing because there is no previous acknowledgment of the process running time. This leads to unfair load distribution among the nodes (18). The dynamic load balance algorithm is considered more effective in cloud computing environment than static algorithm. Firefly algorithm is considered the most appropriate dynamic based on the comparison performed in (19). The main disadvantage of dynamic load-balancing algorithm is the run-time overhead. This is because decision-making to distribute the workload among processors to select processes increases the communication delay which is related to the job. Therefore, in this paper, we propose a new developed load balancing algorithm to reduce the communication delay. The proposed algorithm modified the Adaptive Firefly algorithm by utilizing the WRR to overcome the problem of the dynamic cloud computing environment (20).

In this paper, a healthcare application is implemented to reduce the effort of people, and to insure the status checking of patient. This is performed all the time and in real-time by detecting and monitoring the patient's heartbeat by the doctor and the patient himself. The healthcare web application consists of client-side which provides the login, registration, and all information required. The information includes stuff, doctor, patient, and patient’s accompanist. Furthermore, the application provides chatting with the client and displaying real-time data. The client-side sends information using HTTP protocols to the server side that is located in the cloud which works on managing and saving these data. The real-time data is received from patients' heartbeats sensor, which is connected to the omega controller. The omega sends data by MQTT protocol to the cloud proxy to select one of the available brokers. These brokers receive and publish the information to the MQTT server to be inserted in the Redis database. This information is required by the clients (doctor, patient and patient's accompanist) whose send a request from their own client-side application. Those requests are transferred via HTTP protocols to the WRRA proxy. Finally, a suitable server is selected to receive the requests. This is performed based on the way server sorted via firefly algorithm as detailed in Fig.2.

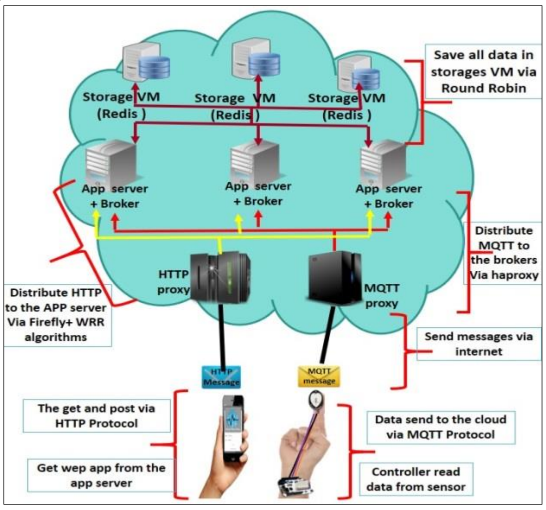

Storage VM (Redis)
Storage VM (Redis)
Storage VM (Redis)
Save all data in storages VM via Round Robin
App server + Broker
App server + Broker
App server + Broker
Distribute MQTT to the brokers Via haproxy
HTTP proxy
MQTT proxy
Distribute HTTP to the APP server Via Firefly+ WRR algorithms
Send messages via internet
HTTP Message
MQTT message
The get and post via HTTP Protocol
Get wep app from the app server
Data send to the cloud via MQTT Protocol
Controller read data from sensor

Figure 2. Architecture of IoT cloud computing system

The task scheduling is occurred as follows; there are four instances (VMs), one for proxy and the others are application servers (AS). The value of all CPUs, memories, and the latency between the proxy and the application servers are computed. Based on these parameters the servers are sorted for each time period. The URLs of sorted servers are inserted in the database that placed inside the proxy server. The proxy acquire these values (URLs) from the database based on the WRRA. The best server gets a higher weight than the second server and the second server will be a higher weight than the last one.

At the same duration the sorted servers will be inserted to the database. Then, the proxy will read the new values and reweight his servers' URLs. The firefly algorithm continuously checks the servers' status and sorts them. At the same time the WRRA frequently reweights the servers based on firefly results. This parallel operation will lead to minimize the delay time as illustrated in Fig 3.

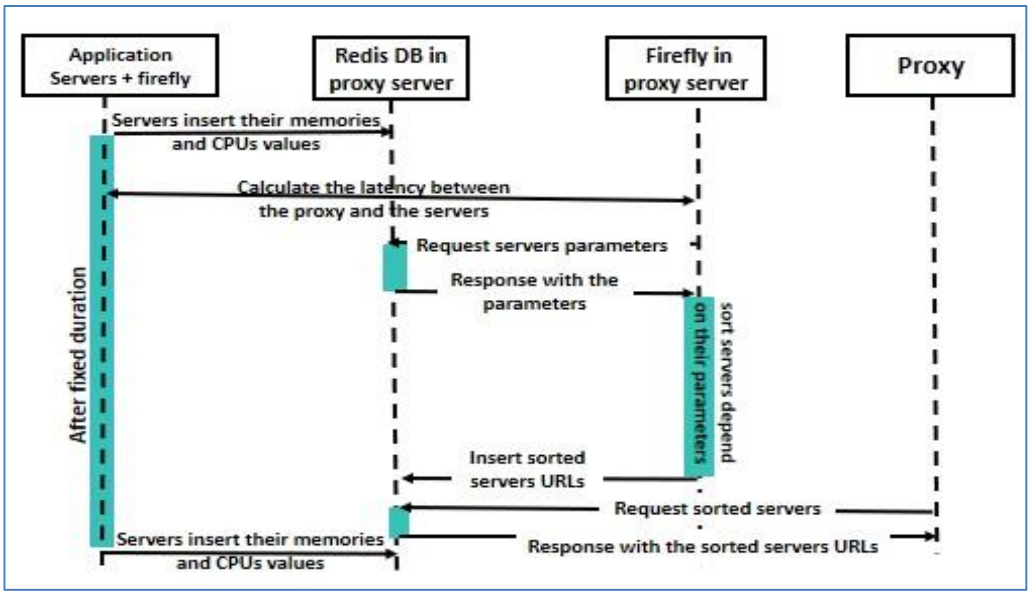

Application Servers + firefly
Redis DB in proxy server
Firefly in proxy server
Proxy
Servers insert their memories and CPUs values
Calculate the latency between the proxy and the servers
Request servers parameters
Response with the parameters
sort servers depend on their parameters
After fixed duration
Insert sorted servers URLs
Request sorted servers
Servers insert their memories and CPUs values
Response with the sorted servers URLs

Figure 3. The sequence diagram of load balancer procedure

The firefly algorithm takes into consideration the status of resources to select the optimal VM for running the task. The best servers checking is occurred every few milliseconds.

Scheduling and checking are represented by adding three parameters to the firefly algorithm; these parameters are CPU, memory, and latency. To represent these parameters between proxy and the servers, the probability range of the scheduling is divided into 10 values from (0.1 – 1). For each one, by giving the highest probability to CPU, then to memory, and finally the latency. The prioritization calculation is illustrated in Table 1:

Table 1. Firefly priority scheduling

CPU Ratio	Memory Ratio	Latency Ratio	Probability	Sorting Index
1	1	1	1	1
1	1	0.9	0.99	2
1	1	0.8	0.98	3
1	1	0.7	0.97	4
1	0.9	1	0.96	5
1	1	0.6	0.96	6
.				
.				
.				
0.1	0.1	0.2	0.11	999
0.1	0.1	0.1	0.1	1000

The Adaptive Firefly algorithm code is inserted in two locations. The first location is in the application servers (AS) which is responsible for collecting data from the server itself. The other location is in the proxy server in order not to waste time when proxy asks for the firefly decision; the job of this part is comparing the data received from the AS. Then sorting them is based on the information of scheduling table and save the data in Redis database, which it also located in proxy server. This is used by WRRA as a proxy. The AFA difference from the standard algorithm by using the scheduling table is illustrated in Table 1. In addition, AFA includes two parts working in parallel as show in the following pseudo-code:

Pseudo code of Adaptive firefly (AFF) algorithm

```
Begin
    Input: objective function f (x);
    Output: the sorted best solution;
    Initialize the population p: X = x1, x2, ..... xn ;
  /* This part is located in the application servers sending their status frequently*/

    Time interval do/* For each firefly lighting part1*/
        Calculate the objective function f (xi) for firefly position xi of firefly i;
        Light intensity Ii at xi is determined by f (xi);
    End

  /*This part is located in proxy server to check and sort the application servers */

    Time interval do
        Calculate the latency Li between congregation C and lighting fireflies
        For i=1 to No_of_fireflies do
            For j=1 to No_of_fireflies do
                If Ii>Ij then
                    Exchange firefly population;
                Else if Ii=Ij then
                    If Li> Lj then
                        Exchange fireflies population
                    End
                End
            End
        End
    End
    Output the best sorted fireflies in the population;
End
```

Results and Discussion:

In order to evaluate the proposed load balance algorithm in IoT cloud environment, various experiments are implemented. The implementation is performed by applying different conditions on the HTTP and MQTT proxy VMs separately. The performance of the HTTP proxy is measured by generating many users to connect between them and the HTTP server. The client requests a web page using the GET methods, and then if the request reaches to the HTTP servers successfully; it should respond with the web page. On the other hand, if the client does not get the response, this request is considered as a failure.

Throughput and response time are used to evaluate the performance of this system in both HTTP and MQTT sides. Throughput is the number of requests that a server can serve per second. The greater the value of throughput is, the better the system performance becomes. **Response time is the total time from the moment that a client sends a request until the time that a request has done.** The small value of response time is, the better the system performance becomes.

In Fig. 4. (a), we compare the throughput of the HTTP proxy between two algorithms (AFF combined with WRR and AFF alone) according to the number of users.

Overall, it is noticeable that the throughput of the (AFF+WRR) is always more than AFF alone during the entire period. The maximum difference between both algorithms occurs at 12000 users, as the combined algorithms reach its peak at more than 275 req/sec and AFF alone value a lit bit more than 100 requests per second.

At the beginning, both algorithms experienced exponential increase since the number of the user is growing up and it can be handled, but this increase stops at nearly 6k users with throughput approximately 75 req/sec in AFF. After that, the throughput started to be more stable with the minor rise to exceed 100 req/sec at 12 thousand users because the ability of proxy to serve more user is decreased. Finally, the throughput fluctuates put and down until it reached the maximum value for AFF at nearly below 150 req/sec at 20000 users when the ability of the proxy to process more request stops.

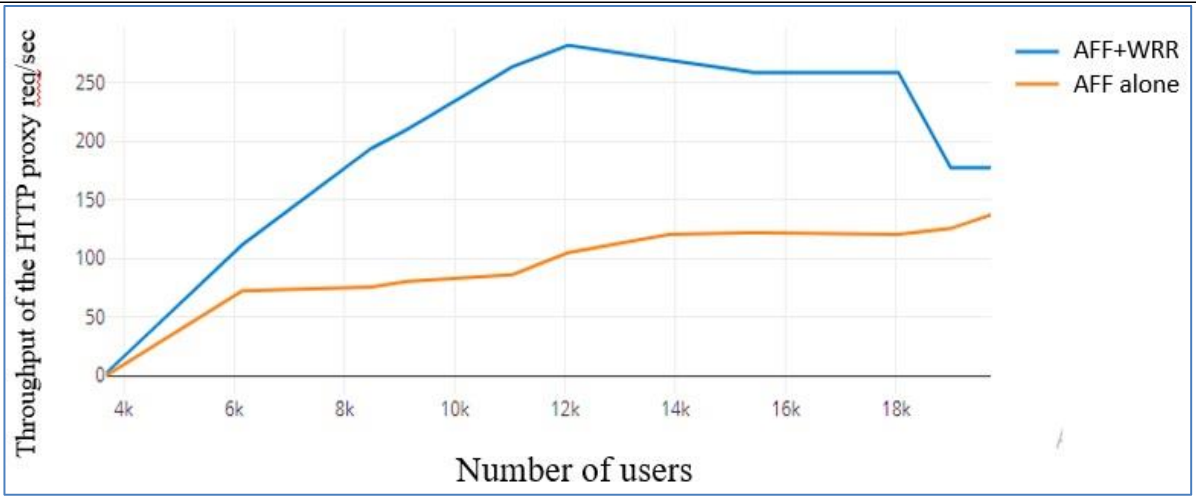
For AFF and WRR, the throughput continues rising from the initial value to reach its maximum value for a specific number of users that can be served which is 12000 users. Then it is slightly decreased to approximately 250 before reaching 16k users where the number of users reached to the highest capability value of the system. During the period between nearly 15000 and 18000 users, the throughput remains constant at almost 250 req/sec. Finally, the throughput suddenly drops to nearly 175 req/sec at 20k users because it overrides the limitation.

The average response time comparison of adaptive Firefly algorithm and its combination with weighted round robin is displayed in Fig.4(b)

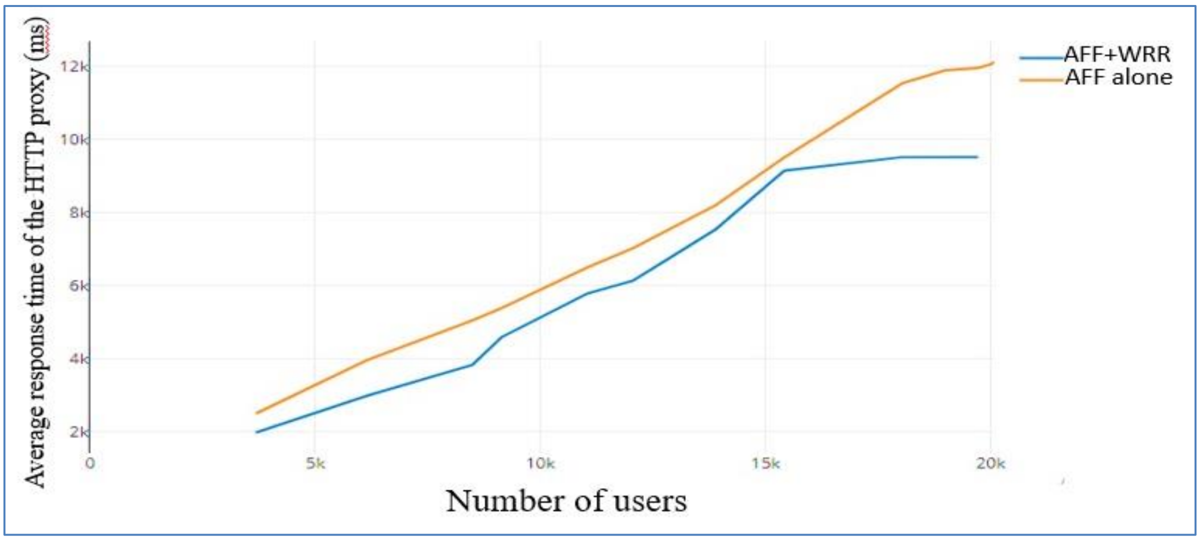
It is noticeable that in entire period the average response time of the algorithms' combinations is always lower than AFF. In addition, the difference of them increases when 15k users is reached, where the maximum peak of the AFF plus WRR is about 9000 ms and the AFF alone is about 12000ms.

At the start, the average response time of both algorithms increase gradually as the number of requests increases. On the other hand, the increase in the response time is saturated at nearly 16k users with average response time of ~10,000 ms in AFF. The average response time, after 16k users, slowly increases until it reaches 12kms for 20k users. Hence the system reaches the maximum allowable number of users.

For the combination of algorithms, the relation between number of users and average response time is linear until the number of users reaches14k. Because the response time depends on serving the request, the load of the proxy increases and thus it cannot redirect more than a specific number of requests and the rest of requests wait. In this case, the number of requests is between 14k and 20k. The solution of these limitations is to use more proxies or more cores.



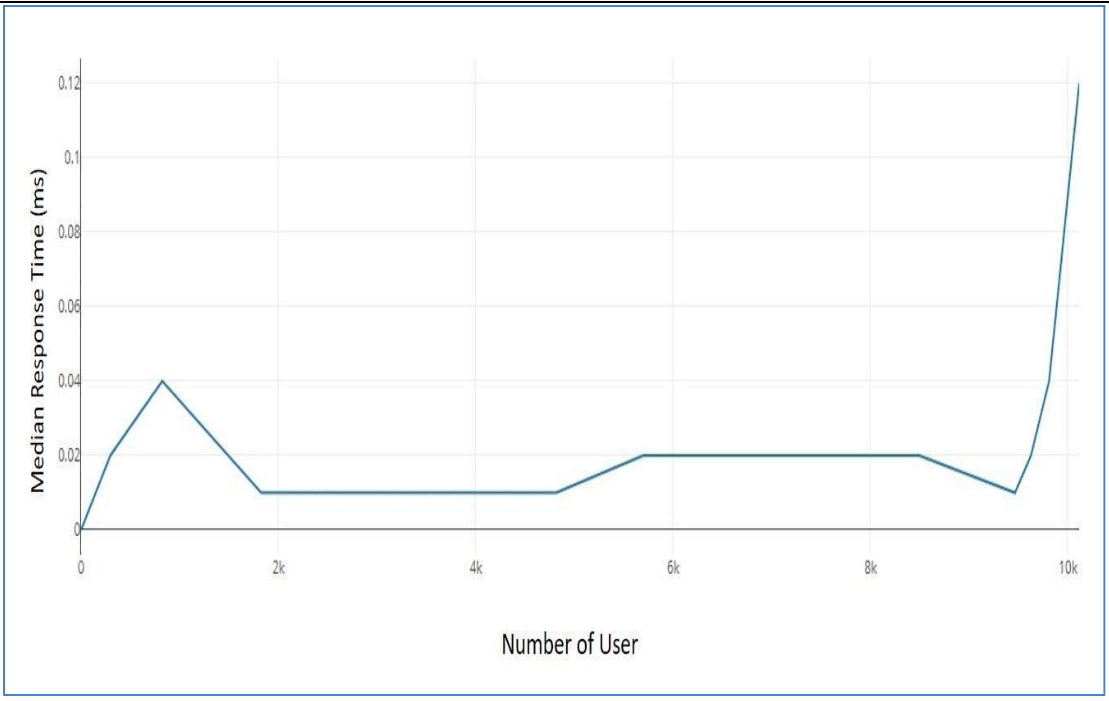
(a) Throughput of the HTTP proxy



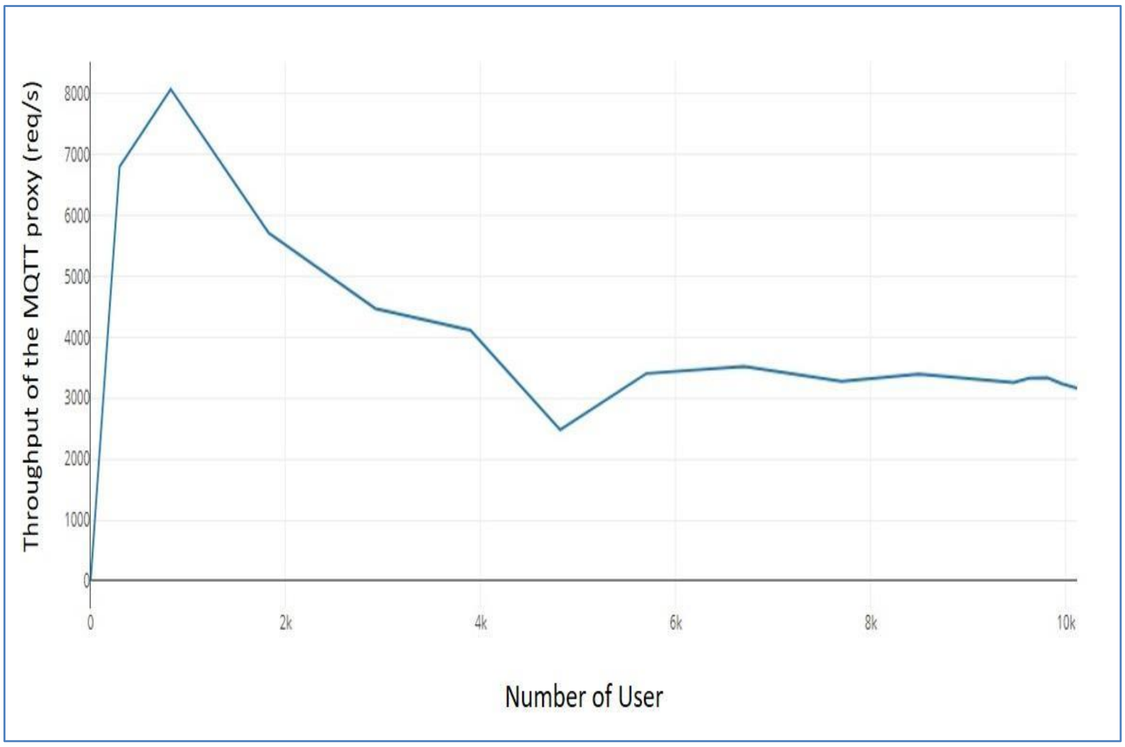
(b) Average response time of the HTTP proxy

Figure 4. HTTP proxy performance comparison of the combination algorithms and AFF alone

To estimate the performance of the MQTT proxy, 10k users is generated. Each user publishes a topic in a message based on the number of messages. These messages that are published and the delay of transmission are used to determine median response time and the throughput. Fig. 5(a) shows the median response time of the MQTT server increasing from 0.01 to 0.12 ms in the interval 9k and 10k users. Fig. 5 (b) shows the throughput of MQTT server. When the number of users is rising to more than 4000 the throughput is decreasing to less than 3500 requests per second because the limitation of the proxy then cannot handle all users.



a. Median Response time of the MQTT proxy



b. Throughput of the MQTT server

Figure 5. Performance of the MQTT server

Conclusion:

The development of IoT technology has led to a huge amount of information because there are millions of peoples connecting with each other by using different physical devices, then cloud computing is very important paradigm. The aim of this paper is to solve the load-balancing in the cloud-computing environment by using Adaptive Firefly Algorithm (AFA) based on the Round Robin (RR) to get high performance.

The proposed method minimizes the response time and maximizes the CPU utilization by distributing the workload between different VMs with considering availability and load of each VM. Whenever a virtual machine (VM) is loaded with multiple tasks, these tasks should be eliminated and sent to less loaded VMs in the same data center.

This research uses MQTT protocol to send the data from the sensors to the server at cloud computing and HTTP protocol utilized to send the data of the patient to the cloud computing. In addition, two different kinds of load balancing algorithms are used, the first is dynamic (Firefly) and the other is static (weighted round robin) to gain both types' benefits. Furthermore, the result shows the influence on the average response time and the throughput by the huge growing amount of user in the two sides, the HTTP and the MQTT sides. In the future, the proposed method can be developed by:

1. Updating the mechanism for the firefly algorithm in order to reduce the searching time for candidate nodes. Furthermore, we will check how to use other intelligent algorithms into this work in order to improve system performance and efficiency.
2. The system could include multiple kinds of sensors and other types of protocols.
3. Priority can be added to the message depending on its important information.
4. Finally adding the privacy and security to the whole system.

Conflicts of Interest: None.

References:

1. Abdulhussain S, Ramli A, Saripan M, Mahmmod B, Al-Haddad S, Jassim W. Methods and Challenges in Shot Boundary Detection: A Review. Entropy. 2018;20(4):214. doi:10.3390/e20040214.
2. Ahmed S, Saqib M, Adil M, Ali T, Ishtiaq A. Integration of Cloud Computing with Internet of Things and Wireless Body Area Network for Effective HealthCare. 2017 (ISWSN) [Internet]. 2017 Nov; DOI: 10.1109/ISWSN.2017.8250019.
3. Al-Joboury IM, Al-Hemiary EH. F2CDM: Internet of Things for Healthcare Network Based Fog-to-Cloud and Data-in-Motion Using MQTT Protocol. International Symposium on Ubiquitous Networking UNet 2017: Ubiquitous Networking [Internet]. 2017 Nov;10542:368-379. Available from: DOI: 10.1007/978-3-319-68179-5_32.
4. Ahmad MO, Khan RZ. Load Balancing Tools and Techniques in Cloud Computing: A Systematic Review. Advances in Computer and Computational Sciences (ACCS). 2017 Sept;554:181-195. DOI:10.1007/978-981-10-3773-3_18.
5. Makasarwala HA, Hazari P. Using Genetic Algorithm for Load Balancing in Cloud Computing. 2016 8th (ECAI) [Internet]. 2016 Jul [cited 2017 Feb 23];ESS-50-ESS-54. DOI:10.1109/ECAI.2016.7861166.
6. Kumar SM, Chakravarthi P, Jagadeesh B, Prakash SM. Load Balancing in Cloud Computing. IJET. 2018; 7(1.1):306-310.
7. Chandrasekaran K. Essentials of cloud computing. 9th ed. CRC Press, Boca Raton London NewYork: Chapman and Hall; 2015.7, Virtualization; p. 161-181.
8. Khan MA, Paplinski A, Khan AM, Murshed M, Buyya R. Dynamic Virtual Machine Consolidation Algorithm for Energy-Efficient Cloud Resource Management: A Review. Sustainable Cloud and Energy Services[Internet]. 2017 Sep [cited 2013];135-165.DOI:10.1007/978-3-319-62238-5_6.
9. Hou L, Zhao S, Xiong X, Zheng K, Chatzimisios P, Hossain SM, et al. Internet of Things Cloud: Architecture and Implementation. Communications Magazine (CM). 2016 Dec;54(12):32-39. DOI: 10.1109/MCOM.2016.1600398CM.
10. Makkar G, Kau PD. A Review of Load Balancing in Cloud Computing. IJARCSSE. 2015;5(4):594-597.
11. Gavathri G, Latha R. Implementing a Fault Tolerance Enabled Load Balancing Algorithm in the Cloud Computing Environment. IJEDR. 2017;5(1):249-256.
12. Karthika K, Kanakambal K, Balasubramaniam R. Load Balancing Algorithm Review's in Cloud Environment. IJERGS. 2015 Jun;3(3):661-667.
13. Gao R, Wu J. Dynamic Load Balancing Strategy for Cloud Computing with Ant Colony Optimization. Future Internet [Internet]. 2015 Sep [cited 2015 Nov 26]; 7 (4):465-483. DOI:10.3390/fi7040465.
14. Bhole R, Singh HJ, Khamkar P, Joshi P, Bendbhar R. Load Balancing in Cloud Computing Using Autonomous Agents. IJIR. 2017;3(3):237-239.
15. Moghtadaeipour A, Tavoli R. A New Approach to Improve Load Balancing for Increasing Fault Tolerance and Decreasing Energy Consumption in Cloud Computing. 2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI). 2015 Nov [cited 2016 Mar 21]; DOI:10.1109/KBEI.2015.7436178.
16. Gopinath G, Vasudevan SK. A Novel Improved Honey Bee Based Load Balancing Technique in Cloud Computing Environment. AJIT. 2016;15(9):1425-1430.
17. Xavier VMA, Annadurai S. Chaotic Social Spider Algorithm for Load Balance Aware Task Scheduling in Cloud Computing. Cluster Computing. 2018 Jan;1–11. DOI:10.1007/s10586-018-1823-x.
18. Aditya A, Chatterjee U, Gupta S. A Comparative Study of Different Static and Dynamic Load Balancing Algorithm in Cloud Computing with Special Emphasis on Time Factor. International Journal of Current Engineering and Technology (IJCET), 2015 Jun; 5(3):1898-1907.
19. Rajeshkannan R, Aramudhan M. Comparative Study of Load Balancing Algorithms in Cloud Computing Environment. Indian Journal of Science and Technology (IJST), 2016 May; 9(20). DOI:10.17485/ijst/2016/v9i20/85866.
20. Singh A.B, Bhat S, Raju R, D'Souza R. Survey on Various load Balancing Techniques in Cloud Computing. Advances in Computing (AC). 2017; 7(2):28-34.

تطوير موازنة الاحمال لأنترنت الأشياء - الحوسبة السحابية اعتمادا على خوارزميات اليراعة المتقدمة والدورية المرجحة

مروة محمد عبد **منال فاضل يونس**

قسم الحاسابات، كلية الهندسة، جامعة بغداد، بغداد، العراق.

الخلاصة:

أدى التطور في إنترنت الأشياء (IoT) إلى ربط البلايين من الأجهزة المادية غير المتجانسة معاً لتحسين نوعية الحياة البشرية، من خلال جمع البيانات من بيئتهم. يجب تخزين هذه البيانات الهائلة التي تم تجميعها في سعة تخزين كبيرة بالإضافة إلى قدرات حاسوبية عالية، التي توفيرها الحوسبة السحابية. يتم نقل بيانات أجهزة IoT باستخدام نوعين من البروتوكولات. نقل الرسائل في قائمة انتظار النقل (MQTT) وTransfer Protocol (HTTP) Hypertext. يهدف هذا البحث لتحسين أداء النظام وزيادة الموثوقية من خلال الاستخدام الفعال للموارد. من خلال، استخدام موازنة التحميل في الحوسبة السحابية لتوزيع عبء العمل ديناميكيًا عبر العقد لتجنب زيادة التحميل على أي مورد فردي. من خلال الجمع بين نوعين من الخوارزميات: الديناميكية خوارزمية (اليراعة المتقدمة Advanced Firefly Algorithm) والخوارزمية الثابتة (Weighted Round Robin Algorithm). وأظهرت النتيجة تحسن في استخدام الموارد وزيادة الإنتاجية وتقليل وقت وقت الاستجابة.

الكلمات المفتاحية: الحوسبة السحابية، خوارزمية اليراعة، إنترنت الأشياء، موازنة الحمل.