International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 03 | Mar 2019 www.irjet.net p-ISSN: 2395-0072

Load Balancing and Crash Management in IoT Environment

Md. Arfath Azeez[1], D.T. Supreeth Rao[2], Chinmay Kini[3]

[1,2,3]*Eight Semester, Dept. of Ise, The National Institute of Engineering, Mysore*

Abstract - *The standard IoT architecture or system that is present today connects the clients (users) to the various devices in a uniform manner using a socket connection in a network. After the clients are connected to the devices, the devices are further connected to the data centers nearest to the user's vicinity which collect all the data sent from the users and send the data to the users if the user requests it. The main intention of this paper to solve the issue of crash management encountered in the traditional IoT architecture. This paper implements the MQTT cloud-based protocol to access the various server's information and provides publish/subscribe mechanisms to the clients connected to the severs. The proposed system includes an application delivery controller, an application delivery controller (ADC) is a network component that manages and optimizes how client machines connect to server for processing or data center to store / read information between the devices and the data center. The devices and the data centers are connected to this distributor (ADC) via a network. Here, if there is any increase in load, the distributor (ADC) handles this by sending the data to a data center which is idle, if there are any crashes or if the data center handling the requests goes down, the distributor (ADC) then would re-route all the data to the other data centers until it recovers. This provides an alternative way to handle the unexpected network failures or crashes so that the whole system does not go down.*

***Key Words*: IoT, Crash management, Load balancing, cloud architecture, MQTT protocol.**

1. INTRODUCTION

The main objective of this paper is to provide load balancing, uninterrupted connectivity and crash management to all the clients (users) connected to various devices in an IoT environment, even if there is an increase in load, network failure or any crashes encountered between the client's devices and the data centers. To facilitate the communication between the various devices we use the concept of cloud computing, which is implemented by the cloud services provided by Amazon's EC2. In this paper we introduce an additional component which provides an interface between the users and the data centers, called the "Application Delivery controller" which implements the performance counter algorithm, it extracts load information about the servers and broadcasts to the Application delivery Controller (ADC). ADC also manages the load balancing part of the IoT environment by keeping tabs on the various servers which are distributed widely by continuously monitoring the servers by assessing the data sent to them in the cloud. It uses the MQTT protocol, which is a publish/subscribe based messaging protocol. The users can publish some information to the cloud using it or the users can subscribe for a particular service through it. Application Delivery Controller (ADC) monitors all the data centers by assessing their availability (RAM), accessibility and load (CPU). If any data center goes down due to network failure or if it gets overloaded the Application Delivery Controller redirects all the requests or service required to other data centers.

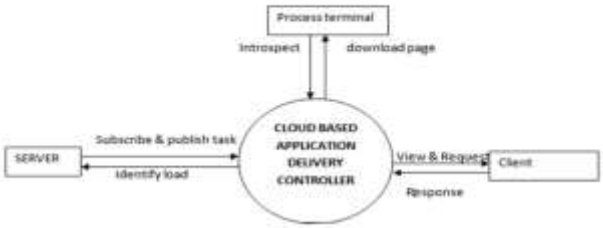

Fig-1: Working of application delivery controller

2. RELATED WORK

a) Klaithem Al Nuaimi, Nader Mohamed, Mariam Al Nuaimi and Jameela Al- Jaroodi 2012, proposed survey of the current load balancing algorithms developed specifically to suit the Cloud Computing environments.

b) Martin Randles, David Lamb, A. Taleb-Bendiab, 2010 Investigates three possible distributed solutions proposed for load balancing; approaches inspired by Honeybee Foraging Behavior, Biased Random Sampling and Active Clustering.

c) Sidra Aslam, MunM Ali Shah, 2015, focused and provide a comprehensive overview of interactive load balancing algorithms in cloud computing. Each algorithm addresses different problems from different aspects and provides different solutions. Some limitations of existing algorithms are performance issue, larger processing time, starvation and limited to the environment where load variations are few etc.

d) Reena Panwar, Prof. Dr. Bhawna Mallick, 2015**,** included some algorithms of load balancing algorithms in cloud computing which is analyzed on a specific environment of virtual machine. In this paper, they proposed a Dynamic Load Management algorithm which will distributes the load uniformly at the servers by considering the current status of

all the available virtual machines intelligently and later response time of this algorithm is compared with the existing VMAssign Algorithm.

e) K. Venkata Subba Reddy, J. Srinivas, A. Abdul Moiz Qyser, 2014**,** have proposed dynamic hierarchical load balancing service architecture for cloud data centers. Objective of the work is to enhance the load balancing performance from the virtual component of cloud datacenter and also proposes a generic performance evaluation matrix to evaluate the performance of cloud datacenters. The various parameters are based on availability, computational speed, storage and redundancy. This work improves the performance of load balancing algorithm using virtual migration policy and evaluated the validity of virtual machine and physical host using performance evaluation matrix.

f) S M S Suntharam, 2013**,** has proposed load balancing by Maxi-Min Algorithm in private cloud environment. The objective is to use max-min algorithm in Cloudsim to show how to balance the load across the different storage nodes in the private cloud, which reduce the make span and data traffic. Max-Min algorithm is also used to reduce idle time and so efficient in mapping the load across the nodes. The parameters considered are VM processing power, bandwidth, memory. The result obtained is algorithm consumes less time in storing a job in node and decreases problem of deadlock in cloud environment. Thus, algorithm attains high sufficiency and scheduling efficiency to all jobs in the private cloud. In future work, considered to improve the complexity and fault tolerance.

g) Kousik Dasgupta, Brototi Mandal, Paramartha Dutta, Jyotsna Kumar Mondal, Santanu Dam, 2013, have proposed a novel load balancing strategy using Genetic Algorithm (GA). Objective of the work is to balance the load of the cloud infrastructure while trying minimizing the make span of a given tasks set using the mechanism of natural selection strategy and performance of the algorithm is compared with Round Robin, FCFS and local search algorithm. The parameters considered are average response time, processing power and performance. The result obtained by genetic novel load balancing algorithm, it outperforms the existing algorithms and achieve the system performance by maximum resource utilization. It also guarantees the quality of service required. In future work, single point of crossover and variation in selection strategy can be considered.

3. MODULES

The modules implemented in this paper are briefly described below.

A. Performance counter algorithm to extract load information.

Performance counter algorithm is used to extract load information about the servers and broadcast to the application delivery controller.

The Pseudo code for the above is shown below:

- Total percent CPU processor time ("Processor", "% Processor Time")
- Percent committed memory ("Memory", "% Committed Bytes in Use")
- Percent page file usage ("Paging File", "% Usage")
- Network adapter throughput.

B. Development of a cloud based MQTT Protocol

The MQTT protocol which is a publish/subscribe messaging protocol is implemented in a cloud environment to submit each machine load to the application delivery controller (ADC), which is a better alternative to the socket communication used in the network. There are many advantages of this protocol, to quote a few:

- Lightweight
- Reduced Network Traffic
- Less expensive

C. Implementation of Application Delivery Controller:

An application delivery controller (ADC) is a network component that manages and optimizes how client machines connect to web and enterprise application servers. In general, it is a hardware device or a software program that manages or directs the flow of data between two entities (client and server). An ADC essentially functions as a load balancer, optimizing end-user performance, reliability, data center resource use and security for enterprise applications. But ADCs also perform other functions, like application acceleration, caching, compression, traffic shaping, content switching, multiplexing and application security.

D. Client implementation for sending request and acknowledge after processing:

The clients (users) are given functionalities to either request for a service they require or publish their own information to the servers via the MQTT protocol which acts as an interface between the users and the servers. Finally, when the user completes his task, the server acknowledges it by sending a unique id called the “machine id” to the user which can be used by the users as a reference to connect to the servers to either send new requests or complete their tasks in the future.

4. SYSTEM DESIGN

In this section we introduce our proposed system design that aims at solving the problem of load balancing and crash management. The system is designed based on the Master-Slave architecture. A brief description about each component is explained below:

A. System Components

The proposed system design consists the following components:

1) Clients (Users): Many clients (remote processors) request and receive services from a centralized server (host computer) or publish some information to the server. The client computer sends request to the Application delivery controller which acts as a "master" over the network connection, the requests are then processed and delivered to the client directly by the server which acts as a "slave". These requests can include application access, storage, file sharing, printer access etc. Client's computers provide an interface to allow a computer user to request services of the server and to display the results the server returns. Clients are often situated at workstations or on personal computers. For example, a client computer can be running an application program for entering patient information while the server computer is running another program that manages the database in which the information is permanently stored. Many clients can access the server's information simultaneously, and, at the same time, a client computer can perform other tasks, such as sending e-mail. The requests from the client could include HTTP request for page download or fetch address using Google API.

2) Application Delivery Controller (Master): The Application Delivery Controller is a data center architecture that distributes network traffic evenly across a group of servers. It acts as the "traffic cop" sitting in front of the servers and routing client requests across all servers capable of fulfilling those requests in a manner that maximizes speed and capacity utilization and ensures that no server is overworked, which could degrade performance. If a single server goes down, the load balancing server redirects traffic to the remaining online servers. When a new server is added to the server group, the load balancing server automatically starts to send requests to it. The main goal of load balancing server is to distribute client requests or network load efficiently across multiple servers and also ensures high availability and reliability by sending requests only to servers that are online. The Application Delivery Controller (ADC) collects memory usage, process counts and CPU load, and based on the requests of the client, it redirects the requests to the servers with the least load. It also provides crash management. Suppose if the server crashes due to network failure or if it is subjected to any kind of attacks, it provides uninterrupted connectivity by redirecting the requests of the client to an alternative server which has minimum load. This acts as a master where it monitors all its slaves (servers).

3) Servers (Slaves): Servers are considered as slaves in this architecture where it is controlled by master (Application delivery controller). Servers wait for requests to arrive from clients and then respond to them. Ideally, a server provides a standardized transparent interface to clients so that clients need not be aware of the specifics of the system (i.e. the hardware and software) that is providing the service. Here the server hosts, delivers and manages most of the resources and services to be consumed by the client. The server acts as the producer whereas the client acts as a consumer. Also, the server provides high-end, computing-intensive services to the client on demand. A server computer can manage several clients simultaneously whereas one client can be connected to several servers at a time, each providing a different set of services. All the communication between client and server in our system designed is taking place in a cloud environment (Amazon EC2) instance where mosquito broker is preinstalled which uses MQTT protocol to respond directly to each client machine.

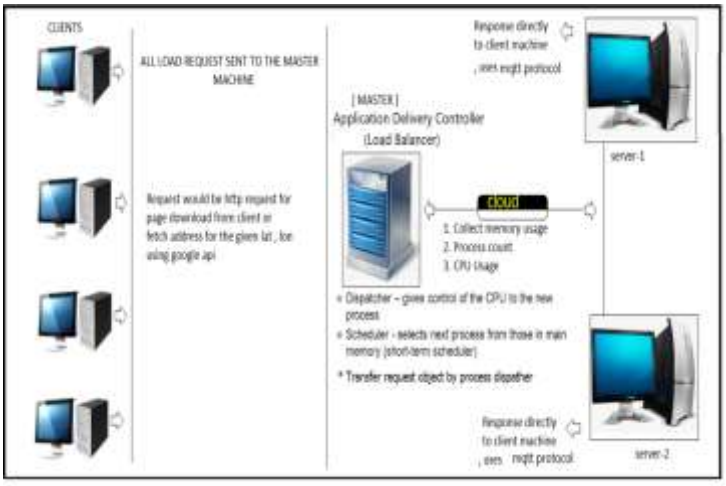

Fig-2: Overview of the system designed

5. IMPLEMETATION AND EXPERIMENTAL RESULTS

The system designed is implemented using various technologies. This design is carried out on Microsoft's Visual Studio and the core language being used is C#. Window's applications are created for each component and are appropriately linked to the cloud (Amazon EC2). The cloud has mosquito broker installed on it, which provides the desired MQTT protocol to help the communication between the two entities (clients and servers).

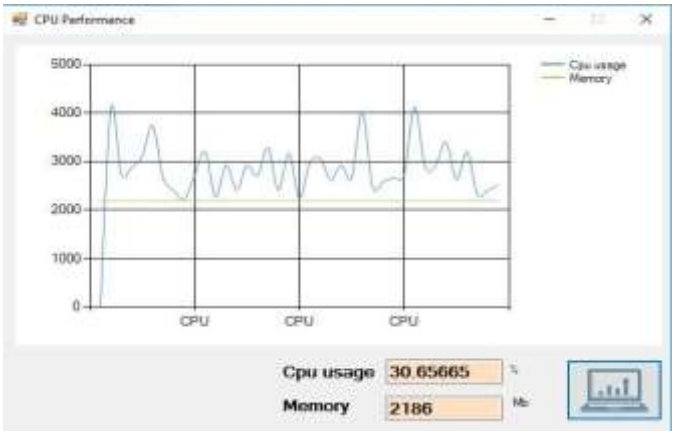

Fig-3: The Slave System

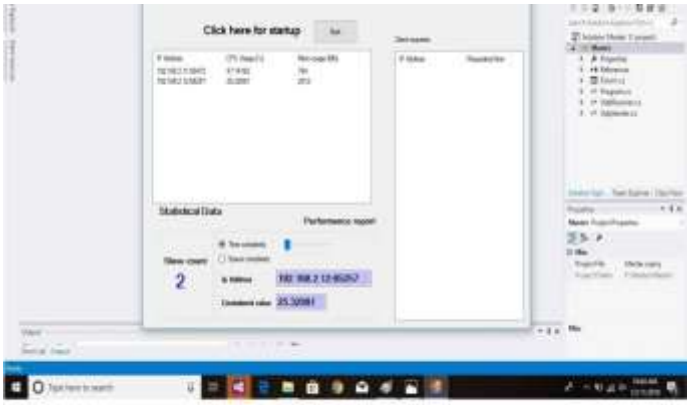

Fig-4: The Master System

6. CONCLUSION

The main motive of this paper, is to implement or solve the problem of crash management and provide load balancing in an IOT Environment, this is achieved simply by adding an additional component to the network called the Application delivery controller(ADC), whose task as described earlier is to find out which particular Server in the network is free to service the request of the client, providing this feature will greatly Enhance the performance/efficiency of a particular IOT environment, since Application delivery Controller (ADC), makes sure that the clients request will always be served immaterial of how many servers are down/overloaded in the network , this particular concept can be applied to any particular scenario to extract the best possible results from it. However, this concept in future can be made more efficient by creating interface for the client System using both wired and wireless technologies to provide seamless connectivity in IOT Environments.

REFERENCES

[1] Sandeep Sharma, Sarabhijit Singh, and Meenakshi Sharma "Performance Analysis of load balancing algorithms", World Academy of Science, Engineering and Technology International Journal of Civil and Environmental Engineering Vol:2, No:2, 2008.

[2] Klaithem Al Nuaimi, Nader Mohamed, Mariam Al Nuaimi and Jameela Al-Jaroodi "A Survey of Load Balancing in Cloud Computing:Challenges and Algorithms", IEEE Second Symposium on Network Cloud Computing and Applications,2012.

[3] Raza Abbas Haidri, C. P. Katti, P C Saxena," A Load Balancing Strategy for Cloud Computing Environment", International Conference on Signal Propagation and Computer Technology (ICSPCT) 2012.

[4] S M S Suntharam," Load Balancing by Max-Min Algorithm in Private Cloud Environment", International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064 Index Copernicus Value (2013): 6.14.

[5] Kousik Dasgupta, Brototi Mandal, Paramartha Dutta, Jyotsna Kumar Mondal, Santanu Dam," A Genetic Algorithm (GA) based Load Balancing Strategy for Cloud Computing", International Conference on Computational Intelligence: Modeling Techniques and Applications (CIMTA) 2013.

[6] Tingting Wang, ZhaobinLiu, Yi Chen, Yujie Xu," Load Balancing Task Scheduling based on Genetic Algorithm in Cloud Computing", IEEE 12th International Conference on Dependable, Autonomic and Secure Computing,2014.

[7] K. Venkata Subba Reddy, J. Srinivas, A. Abdul Moiz Qyser," A Dynamic Hierarchical Load Balancing Service Architecture for Cloud Data Centre Virtual Machine Migration", S. C. Satapathy et al. (eds.), Smart Intelligent Computing and Applications, Smart Innovation, Systems and Technologies 104, 2014.

[8] Sachin Uttam Kadam and Sandip U Mane," A Genetic-Local Search Algorithm Approach for Resource Constrained Project Scheduling Problem", International Conference on Computing Communication Control and Automation, 2015.

[9] Medhat Tawfeek, Ashraf El-Sisi, Arabi Keshk and Fawzy Torkey," Cloud Task Scheduling Based on Ant Colony Optimization", The International Arab Journal of Information Technology, Vol. 12, No. 2, March 2015. [10] Subasish Mohapatra, Ishan Aryendu, Anshuman Panda, Aswini Kumar Padhi," A Modern Approach For Load Balancing Using Forest Optimization Algorithm", Proceedings of the Second International Conference on Computing Methodologies and Communication (ICCMC 2018) IEEE Conference Record # 42656; IEEE Xplore ISBN:978-1-5386-3452-3 2018.

[10] Er. Pooja Yadav, Er. Ankur Mittal, Dr. Hemanth Yadav, "IoT: Challenges and Issues in Indian Perspective", 2018 3rd International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU).