IOP Conference Series: Materials Science and Engineering

Model of load balancing using reliable algorithm with multi-agent system

IOP Conf. Series: Materials Science and Engineering **190** (2017) 012033 doi:10.1088/1757-899X/190/1/012033

Model of load balancing using reliable algorithm with multi-agent system

M F Afriansyah[1], **M Somantri**[1], **and M A Riyadi**[1]

[1]Department of Electical Engineering, Diponegoro University, Semarang, 50275, Indonesia

[1]afriansyahfaizal@gmail.com

Abstract. Massive technology development is linear with the growth of internet users which increase network traffic activity. It also increases load of the system. The usage of reliable algorithm and mobile agent in distributed load balancing is a viable solution to handle the load issue on a large-scale system. Mobile agent works to collect resource information and can migrate according to given task. We propose reliable load balancing algorithm using least time first byte (LFB) combined with information from the mobile agent. In system overview, the methodology consisted of defining identification system, specification requirements, network topology and design system infrastructure. The simulation method for simulated system was using 1800 request for 10 s from the user to the server and taking the data for analysis. Software simulation was based on Apache Jmeter by observing response time and reliability of each server and then compared it with existing method. Results of performed simulation show that the LFB method with mobile agent can perform load balancing with efficient systems to all backend server without bottleneck, low risk of server overload, and reliable.

1. INTRODUCTION

Server becomes a very important thing to be aware of on the era of technology. Server become a services center of all users. The more users lead to the more complex of the server infrastructure. Development of server infrastructure must be able to accommodate a number of requests coming from users. Load of the server will be high in accordance with the number of user of the services provided by the server. High server load can affect the speed of process that will be slowing down and can make server stop working. The appropriate technology is needed to overcome this problem. Distributed systems are used in the infrastructure to distribute incoming requests from users to servers to be processed. In addition to distributed systems, the application of load balancing mechanism can be able to handle that problem reliably. Load balancing in distributed system has been proposed by a number of scientists [1]–[4]. Load balancing is necessary to improve the reliability of services with many users on a large scale. In distributed systems, load balancing is a method to share the load equally across all unit components (server) [5]. Load balancing is needed on a large scale process. The load balancing mechanism aims to distributing resources to each user's demand or request on the servers so that no node server is overloaded or remained idle. Load balancing focused on cloud computing environment using mobile agent has been analyzed by Singh [6]. Load balancing method using a multi-agent system can make reliable and efficient system.

Multi-agent system is a software-based distributed system using the agent with special abilities and autonomous (independent) on a network [3]. Software agents will continuously perform tasks that have been assigned by the user in a specific environment [7]. Agent has the ability or characteristic such as cooperative, independent, able to communicate between agents, and mobility [8]. Agents can move from one node to another node freely according to a given task.

Previously, many methods of developing load balancing using a mobile agent. Mobile agent method is an implementation of load balancing server dynamically using the mobile agent, the agent collects information for specific purposes in accordance with a task that has implemented. Dynamic load balancing in a distributed system is needed to improve performance and reliability of the system in large scale [14]. Distributed system is usually distributed as resource computing unit which is connected to the network to meet the needs of large-scale and high-performance [15].

Load balancing can be combined with a multi-agent system to automatically perform load sharing in accordance with the specified parameters and configuration. Multi-agent system is a group of an agent that cooperate to perform their task. Definition agent according to Lange is the active agent and can move towards to another node (computer), or exploring the network to collect information, data, or detect changes [16]. Agent has a life cycle in the environment, such as the creation, migration, and disposal. Multi-agent system purpose is to find server computing resource information using a mobile agent. In distributed systems, all computing units will look like single unit groups that cooperate or interact. Distributed system is carried out systematically and regularly to process and distribute the data or information. Load balancing in distributed systems shares or distribute the assigned tasks to different servers to keep workload to a server till given task ends. Load balancing in this research focuses on the algorithm that will be proposed to share the workload equally. Load balancing algorithm is important to determine obtained results. Incoming requests will go through the process of load balancing server until to the server to be processed in accordance with the desired flow algorithm.

In this research, we have designed a load balancing mechanism using reliable algorithm with multi-agent on distributed system. This mobile agent monitors all the activities that occur in the network group. The agents migrate through the nodes in the same group to monitor congestion that may occur. With the integration of this method, the demand for the desired resource can fulfill as possible and full traffic of node can be identified more efficiently [9]. Load balancing system in this research uses an algorithm and java technology called JADE [10]–[13]. JADE programming is like aglets programming that use an agent to find information condition on the server, whether it is large load or still idle (normal). JADE is the successor of mobile agent programming aglets. The information will be sent back to the server and processed. This multi-agent system is cross platform so it can be used anywhere (any OS) based on java. The load balancing algorithm that shares the load is looking at the information on the condition of the server. Thus, there is a central server which controls the load into the server to avoid server overload or idle and monitor agent performance. The research purpose is to develop load balancing algorithm that can share the load equally and reliably on a large scale. Load balancing server can perform load balancing mechanism efficiently to all backend server, no bottleneck, low-risk overload server, and accelerate response time.

Overall, this report is divided into several sections. Section 2 discusses the research methodology. Section 3 discusses the result of this research. Section 4 is the conclusion of this research.

2. **RESEARCH METHODOLOGY**

3.1. System Overview

System load balancing in distributed environment has been simulated using the multi-agent system with proposed load balancing algorithm. All multi-agent system using JADE run with JDK 1.8 and load balancing software using nginx as a reverse proxy. Overall, the system consists of one physical server machine (server host) and the specification is 6 core processors AMD Phenom II X6 1055T, 16 GB RAM, and 3 T hard drive. The host server is divided into five virtual servers, three virtual servers running on Linux debian 3:16 operating system and 2 virtual servers running on Windows 2012

operating system standard. This virtual system software using proxmox software-based debian 3:16. The host server operating system running on debian Jessie (kernel 3.16).

The entire virtual machine connected over virtual LAN use bridge configuration in proxmox. In this scheme, load balancer server has two NIC, public and private. The public NIC connected directly to the user / users and the private NIC is used to interconnect with the backend servers. Figure 1 is a simple description of implemented topology of the system.

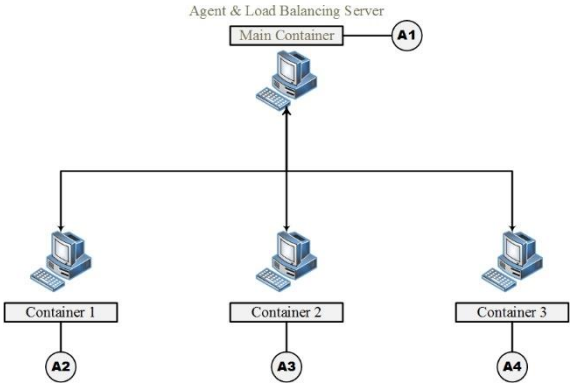

Figure 1. Topology of the load balancing system

Figure 1 shows that the load balancing system is divided into two parts: main container and the regular container. Main container served as a central of information and data from mobile agent, also as a load balancing server. The regular container is the backend server of a service provider. The main container will receive the request from users then be forwarded to the backend server for processing.

The first process from this method is to forward request from *n* request to backend server according to least time first-byte algorithm. Then main server sends a mobile agent to all backend servers. The mobile agent of main server is configured to ask for resource information from each backend server. Once the agent has reached the backend server, backend server will process requests from the main server agent. The resource information will be sent to the main server via mobile agent of the backend server. The mobile agent of the backend server contains information resource backend server and sent to the main server. Afterwards, the main server determines the condition of backend server load that can or cannot perform the service based on provided information by backend server agent. The main server inserts the results into the list of load balancing with the "normal" or "overload". The normal condition is a condition where backend server can perform the service and load server does not exceed the threshold. An overload condition is a condition which the backend server cannot perform the service because workload resource of backend server is high or exceeded the threshold. Proposed threshold in this research is measuring the load of CPU and memory. Threshold values greater than or equal to 90% of CPU load and memory load. The main server sends back mobile agent repeatedly to monitor the condition of backend server resources. Backend server temporary blocked from the list if backend server gets overload condition until back to normal conditions. Furthermore, the main server calculating the request with proposed LFB algorithm. The backend severs processing that request from the user and give back the result to the user through main server. This method of load balancing can share workload equally on all backend server that makes efficient service, keeping large request and has fast response time. Simulation has to performed by comparison of previous research algorithm with LFB algorithm.

3.2. Simulation method

Simulation method is a method to verify a research by the system parameter. Performed simulation method is to obtain quantitative result or information about the system being simulated. The obtained result from this simulation is the average response time, throughput, error number, and a deviation of load balancing system. This simulation method is inspired from previous research by Choi [17] using

weight least Connection (WLC) algorithm to perform load balancing across 10 virtual servers with equal server resource, The number of requests simulated in this research was 800 request. WLC algorithm is a combination of weight and least connection algorithm. The least connections algorithm is selects the server with the least number of active connections to ensure that the load of the active requests is balanced on the backend servers. Fewest or least connection would be a priority of this algorithm. Weight algorithm is algorithm to determine backend servers with a higher weight value will receive a larger percentage of connections according to weight values at any one time. This algorithm will repeat back to the first server if a request of the user has reached the last server. The request will be processed equally by WLC algorithm with a weight configuration values that have been determined in according to request per server. Time duration that used to perform this research was 10 seconds. WLC algorithm can maintain the balance between the server and get fast response time.

Based on this method, WLC simulation method has chosen because research environment is similar to previous environment research. Simulation method of this research was using load balancing algorithm described earlier to find the value of response time, throughput, error, and deviation. LFB algorithm was combined with the mobile agent to keep the request from users to the system or server. Agent was set by scheduling to get information resource from each server. This algorithm was compared with WLC algorithm according to simulation method using 800 requests for 10 seconds. The second simulation did a comparison with the WLC algorithm using an 1800 request for 10 seconds. Benchmark application that used in this simulation is Apache JMeter.

In this scheme, each backend server embedded mobile agent ready to provide resource information from each backend server. The main server is the center of gathering information from each backend server. The main server creates a table that contains information resource information from each backend server. This information is combined with load balancing algorithm. The value of resource information is CPU load, memory load and the number of connections. The number of inbound connections is linear with CPU and memory load. Based on that, obtained formula is used to measure the load of a server as follows:

$$load = \frac{\%CPU.\%memory}{N_{request}} \tag{1}$$

The results of that load are then used as a point that determines backend servers are in normal or overload condition. After determining the condition of each backend server, then that result will be combined with the proposed load balancing algorithm. Servers with high load value will be not active or delist from load balancing list until the load value decrease and back to normal.

The response time is a value of the data from this research. Response time value obtained from server response time to the user. The value of the response time can be used as a variable that will be entered into the equation. Average value equation can be determined by using response time variable. The average value equation of response time is as follows

$$Average = \frac{\sum_{i=1}^{n} xi}{n_{request}} \tag{2}$$

Variable *n* is the number of requests that will be simulated. Variable x is the value of the response time every request. The average value obtained from dividing the amount of response time with a number of requests in accordance with Equation (2). Another equation can be generated by using response time and the number of request variable from user. Median, variance, and standard deviation equations are determined using these formulas:

$$Variance = \frac{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}{n\,(n-1)} \tag{3}$$

$$Std.\,Deviation = \sqrt{variance} \tag{4}$$

$$Throughput = \frac{n}{\Sigma t} \quad (5)$$

The value of variant is used to determine how much distribution value of response time against average. Standard deviation shown in Equation (4) is a value that indicates the level variation of a set of data. Backend server group consists of three servers as described previously. Throughput value of each server in the Equation (5) is obtained from total incoming requests divided by total response time. The value of the resulting error in testing to be performed. There is also have error value to be analysis. This error value is the number of requests that fail or not processed on the server. Information about the server load will remain on the main server as log information. Log information to maintain if there are miss or error in one backend server or agent.

3. **Result and discussion**

The result of the simulation was compared between the methods of previous research using WLC algorithm [17] with LFB research algorithm. Assessment in this simulation is the value of the response time, average response time, std. deviation, throughput, and an error occurred. The input task was generated by sending 800 requests to the server for 10 seconds with 3 backend server. The values taken from simulation result were compared and analyzed. Results of the response time of load balancing using WLC and LFB are illustrated in Figures 2 and 3.

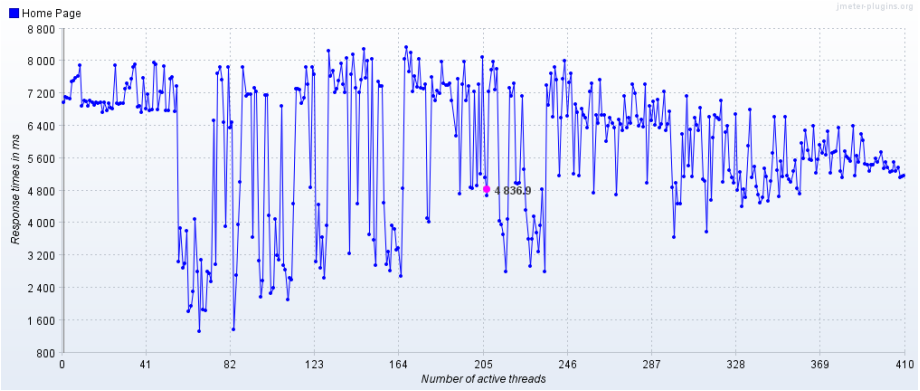

Figure 2. Response time of WLC method for 800 request

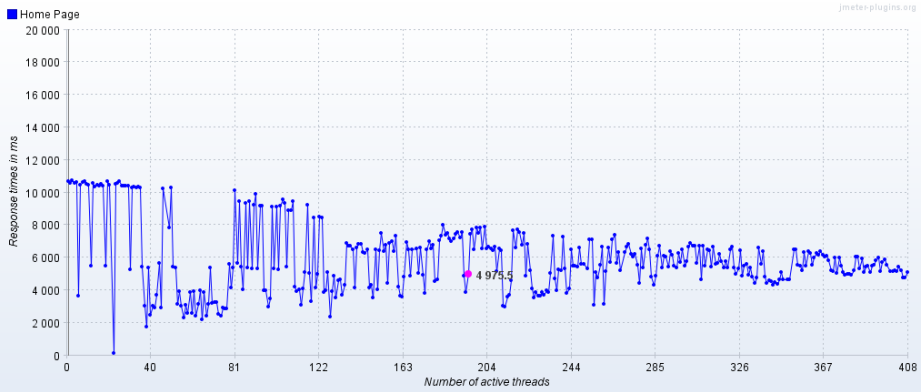

Figure 3. Response time of LFB method for 800 request

Figure 2 describes the response time simulation of WLC with 800 user request for 10 seconds, while Figure 3 describes the response time of LFB with same scheme. WLC method provides faster response time than LFB. WLC response time is less than 8800 ms, 4836 of average response time, 2435.76 of std.deviation, 47.7 of throughput and 0 % error. LFB response time is less than 1100 ms, 4975 of average response time, 3056.35 of std.deviation, 38.7 of throughput and 0 % error. In the first

of simulation with 800 requests, WLC is better than the LFB according to the result. The next result is with doing the second simulation. The purpose of second simulation was to see the limits of each server backend and comparison of the limit of each server using WLC and LFB method. The second simulation was divided in two stages: the first stage by giving 600 requests directly to each server and the second stage by sending 1800 request using WLC and LFB method for load balancing. The limit of each server is using 600 requests based on experiments to measure the limit. The error value as the point to be analysis is whether the server has passed the limits and the request become down. Preliminary results of the second simulation for the first stage will be illustrated in Figures 4, 5 and 6.

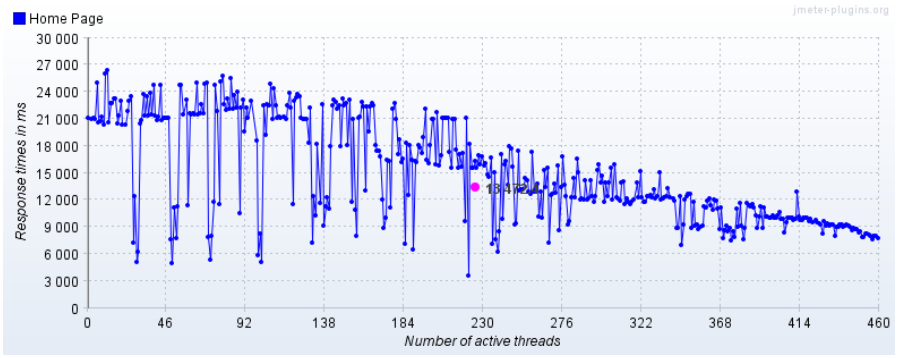

Figure 4. Response time of backend server 1 for 600 request

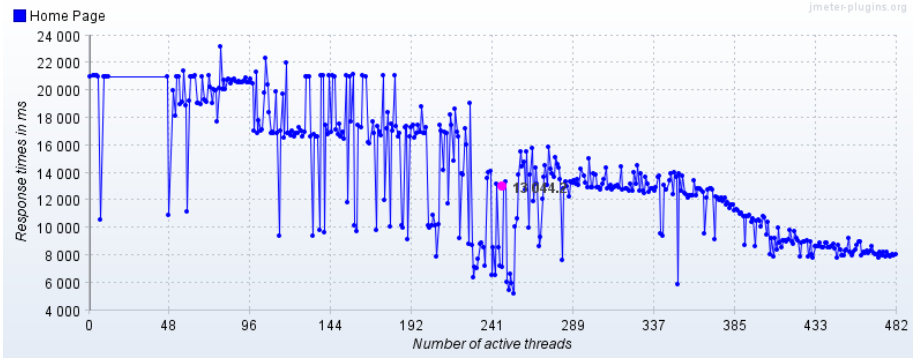

Figure 5. Response time of backend server 2 for 600 request

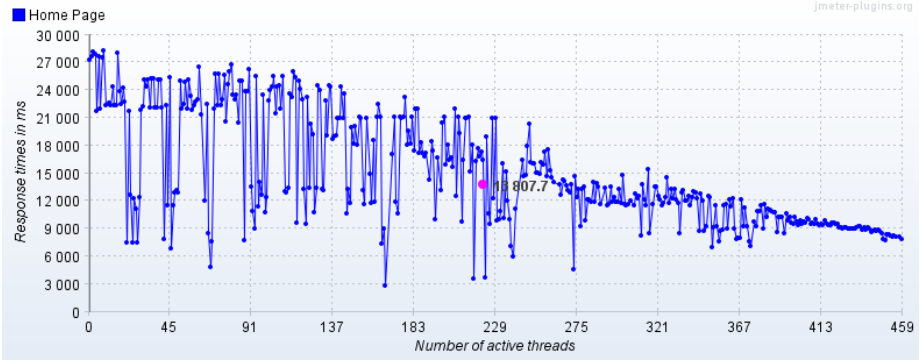

Figure 6. Response time of backend server 3 for 600 request

Figure 4 describes the response time of backend server 1 with 600 requests for 10 seconds. Backend server 1 provides the response time within less than 27,000 ms, average response time of 13,472 ms, std.dev 7165.6 ms, throughput 19.4 and 6.5 % error. Figure 5 describes the response time of backend server 2 with 600 requests for 10 seconds. Backend server 2 provide the response time within less than 24,000 ms, average response time of 13044 ms, std.dev 5703.96 ms, throughput 20.8 and 8.17 % error. Figure 6 describes the response time of backend server 3 with 600 requests for 10

seconds. Backend server 3 provide the response time within less than 30,000 ms, average response time 13807 ms, std.dev 7692.48 ms, throughput of 18.9 and 5.67 % error. Error occurred on all backend servers, identified by the ability limit for backend server on simulation of this research. Backend server 2 has the highest error of 8.17%, followed by backend server 1 at 6.50% and backend server 3 at 5.67%. The total error of the backend server is 20.34%. In second stage of the second simulation comparison between the methods WLC with LFB was performed. The task of simulation was by sending 1800 request simultaneously for 10 seconds. The results of second stage of the second simulation can be described in Figure 7 and 8.

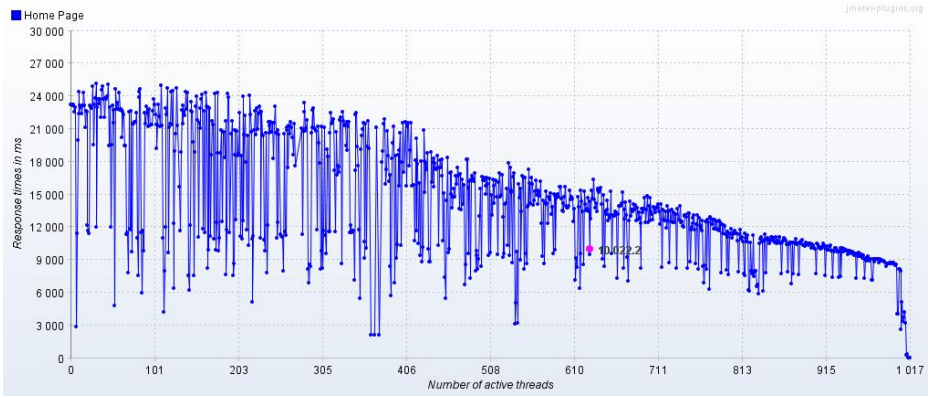

Figure 7. Response time of WLC method for 1800 request

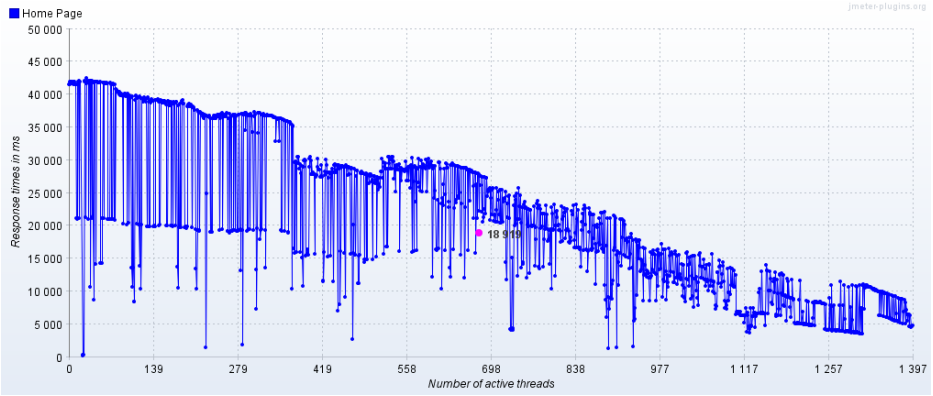

Figure 8. Response time of LFB method for 1800 request

Figure 7 describes the response time of the 1800 request on WLC method for 10 seconds. The result shows response time using methods WLC is less than 27,000 ms, 10,022 ms of average response time, std.dev 8213.64 ms, throughput of 61 and 21.11 % error. Figure 8 describes the response time of the 1800 request on LFB method for 10 seconds. The result shows response time using LFB method is greater than the WLC method, which is less than 45,000 ms, average response time of 18,918 ms, std.dev 13440.64 ms, throughput of 35 and 0 % error. However, the LFB method was more resilient in dealing with a large number of requests. WLC method was only capable of holding within 78.99% from the 1800 request. The LFB method could withstand 100% from the 1800 request although it sacrificed in response time, throughput and others. Based on this second stage of the second simulation showed that the LFB method combined with the mobile agent is more powerful and reliable than the WLC method.

4. Conclusion

System load balancing becomes important in maintaining the reliability of a service. Innovations in making a load-balancing algorithm must be improved. Innovation was created to address the problems that continue to evolve in a distributed system. In this research, load balancing algorithms that can

improve reliability and efficient with the mobile agent has been discussed. The proposed algorithms are the least time of first byte (LFB) combined with the mobile agent. LFB algorithm combines the least connection with checks on the time the first byte of data. The mobile agent complements LF algorithm to monitor the state of the resource of each server backend that does not reduce the quality of their service. Based on the results, LFB load balancing method is more reliable and robust compared with WLC method of processing 1800 requests. The WLC method is only capable of processing 78.99% of 1800 while the method of LFB can process 100% from the 1800 request. LFB method is expected to be a reliable load balancing solution in addressing a number of requests on a large scale. However, the response time and system throughput on the results of simulation requires improvement. Improved response time and system throughput would make the system work much better than before.

5. References

[1] J. Cao, Y. Sun, X. Wang, and S. K. Das, "Scalable load balancing on distributed web servers using mobile agents," *J. Parallel Distrib. Comput.*, vol. 63, no. 10, pp. 996–1005, 2003.
[2] Q. Long, J. Lin, and Z. Sun, "Agent scheduling model for adaptive dynamic load balancing in agent-based distributed simulations," *Simul. Model. Pract. Theory*, vol. 19, no. 4, pp. 1021–1034, 2011.
[3] M. A. Metawei, S. A. Ghoneim, S. M. Haggag, and S. M. Nassar, "Load balancing in distributed multi-agent computing systems," *Ain Shams Eng. J.*, vol. 3, no. 3, pp. 237–249, 2012.
[4] A. Yaseen, H. Ji, and Y. Li, "A load-balancing workload distribution scheme for three-body interaction computation on Graphics Processing Units (GPU)," *J. Parallel Distrib. Comput.*, vol. 87, pp. 91–101, 2016.
[5] O. Rihawi, Y. Secq, and P. Mathieu, "Load-Balancing for Large Scale Situated Agent-Based Simulations," *Procedia - Procedia Comput. Sci.*, vol. 51, pp. 90–99, 2015.
[6] A. Singh, D. Junejab, and M. Malhotraa, "Autonomous Agent Based Load Balancing Algorithm in Cloud Computing," *Int. Conf. Adv. Comput. Technol. Appl. (ICACTA- 2015)*, vol. 45, no. Procedia Computer Science, pp. 832–841, 2015.
[7] P. Braun and W. Rossak, *Mobile Agents: Basic Concepts, Mobility Models, and the Tracy Toolkit*. 2004.
[8] V. Baousis, S. Hadjiefthymiades, G. Alyfantis, and L. Merakos, "Autonomous mobile agent routing for efficient server resource allocation," *J. Syst. Softw.*, vol. 82, no. 5, pp. 891–906, 2009.
[9] X.-J. Shen *et al.*, "Achieving dynamic load balancing through mobile agents in small world P2P networks," *Comput. Networks*, vol. 75, pp. 134–148, 2014.
[10] F. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems with JADE*. 2007.
[11] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa, "JADE: A software framework for developing multi-agent applications. Lessons learned," *Inf. Softw. Technol.*, vol. 50, no. 1–2, pp. 10–21, 2008.
[12] A. V. Sandita and C. I. Popirlan, "Developing A Multi-Agent System in JADE for Information Management in Educational Competence Domains," *Procedia Econ. Financ.*, vol. 23, no. October 2014, pp. 478–486, 2015.
[13] C. V. Trappey, A. J. C. Trappey, C. J. Huang, and C. C. Ku, "The design of a JADE-based autonomous workflow management system for collaborative SoC design," *Expert Syst. Appl.*, vol. 36, no. 2 PART 2, pp. 2659–2669, 2009.
[14] R. Z. Khan and J. Ali, "Classification of Task Partitioning and Load Balancing Strategies in Distributed Parallel Computing Systems," *Int. J. Comput. Appl.*, vol. 60, no. 17, pp. 48–53, 2012.
[15] Y. Jiang, "A Survey of Task Allocation and Load Balancing in Distributed Systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 9219, no. c, pp. 1–1, 2015.
[16] D. Lange and M. Oshima, "Mobile agents with Java: the Aglet API," *World Wide Web*, pp. 1–18, 1998.
[17] D. Choi, K. S. Chung, and J. Shon, "An Improvement on the Weighted Least-Connection Scheduling Algorithm for Load Balancing in Web Cluster Systems," *T.-h. Kim al. GDC/CA 2010*, vol. CCIS 121, pp. 127–134, 2010.