Efficient Bandwidth Management for Load Balancing in Grid Computing

Mahdi S. Almhanna
Information Security Department, College of Information Technology, University of Babylon, Babylon, Iraq.
E-mail: mahdi.almhanna@uobabylon.edu.iq

Firas Sabah Al-Turaihi
Information Security Department, College of Information Technology, University of Babylon, Babylon, Iraq.

Received August 05, 2021; Accepted November 20, 2021
ISSN: 1735-188X
DOI: 10.14704/WEB/V19I1/WEB19017

Abstract

Incoming Information Technology (IT) services appear with cloud computing perspectives that provide users access to IT resources anytime, anywhere. These services should be good enough for the user with some advantages for the cloud service provider. To achieve this goal, you must face many challenges, load balancing is one of these challenges. The most convenient option for some functions does not mean that option is always a good choice to achieve the entire work all the time. Resource overload and bad traffic that can lead to time exhaustion should be avoided, this can be obtained through appropriate load balancing mechanisms. This paper offers a simple solution for choosing the preferred server to distribute functions based on minimum bandwidth consumption.

Keywords

Cloud, Bandwidth Management, Load Balancing, Grid Computing, Minimum Bandwidth Consumption.

Introduction

Grid computing is a set of machines connected to each other by a network that works together as a virtual powerful computer to do big jobs, (Anthony, 2016; Almuttairi et al, 2011; Almuttairi et al, 2010; Almuttairi et al, 2017 and Almuttairi et al, 2010), such as analyzing huge sets of data through the cloud, you can aggregate and use numerous computer networks for different periods of time and for specific purposes, you pay if needful only for what you utilize to save time and buy and deploy the necessary resources without the involvement of others. Also, by dividing the jobs on multiple devices, the processing time is greatly reduced to increase efficiency and reduce wasted resources, (Sakr et al, 2005; Almhanna, 2010 and Almuttairi et al, 2010).

Grid computing systems collect many resources for the purpose of creating a virtual computing repository that enables users to withdraw resources from this reservoir for a fee based on usage.

There are many resource allocation problems in grid computing that have been studied in several ways, including dynamic and classical (Almuttairi et al, 2010).

In this work, we deem one such issue that deals with concurrent requirements for computing potential and connection bandwidth so that the network is able to meet the changing and fluctuating requirements quickly and significantly for many users and more economically (Johnsson *et al*., 2005; Nathan *et al*., 2019; Almhanna, 2010 and Almuttairi *et al*., 2010).

If the traffic load is unbalanced, there is a significant possibility that the traffic will cause packet loss, congestion, and deterioration of the network quality of the service (Zeng et al, 2019). Current resource allocation mechanisms are focused on saving CPU, memory or number of connections and do not consider bandwidth as a significant barrier.

Many applications perform many operations that require different devices to communicate with each other and share their data continuously such as scientific simulations or real-time applications such as financial services all require large and sustainable data transfer and this requires guaranteed bandwidth at the application level. Many network systems have been developed, Condor (Litzkow, *et al*., 1988), Globus (Foster; Kesselman, 1997), and Legion (Chapin *et al*., 1999) are good examples of such systems, and however, so far we have to address many issues of resource allocation in systems. Resource allocation for network computing involves sharing of resources as suggested by Chun and Kohler (Chun-Culler, 2000), since all functions must have access to certain resources, all previous models do not deal with resource allocation with explicit bandwidth limitations. This search differs significantly from the others because the requests may have been allocated across many several servers if minimum bandwidth restrictions are met.

In this work, we will assume that there is a set of jobs, each of which requires some computing resource that needs some bandwidth and is profitable if selected. Where the Hungarian algorithm method was used for the purpose of communication between two nodes by using the least sufficient amount of bandwidth.

Load Balancing

It is professional artistry for computer networks to divide workload across numerous computers, network links, disk drives, CPUs, or any other resources (Afzal-Kavitha, 2019; R.M, 2010), to optimize resource usage, reduce response time, increase productivity, and avoid overload. Moving away from using a single component and instead using multiple components may result in an increase in reliability (Ammar, 2011). One of the major issues in grid computing is load balancing (Joshi-Kumari, 2016). It is an important mechanism designed to equitably distribute the load. (maybe not equally) among all servers within the network that are authorized by the service provider, because the purpose of this process is to avoid loading some servers too much, while others don't allocate loads or perhaps allocate too little, so that it does not fit the capacity of the server. Behind this idea is the idea of this research paper, where we worked on distributing requests between servers so that the least possible bandwidth packets are consumed.

Round Robin Load Balancing

To maintain a balanced work environment and for the purpose of distributing user requests to a number of different servers, round-robin load balancing is a suitable approach to doing such work. Where each request is routed to a different server in turns. The process is repeated several times and alternately until the completion of all requests. As a simple example, suppose an organization has three servers:

Server I, Server II, and Server III. The request distribution is as follows:

- The first demand is distributed on the server I.
- The second demand is distributed on the server II.
- The third demand is distributed on the server III.

One of the worst drawbacks of the round-robin algorithm in load balancing (Ghutke-Shrawankar, 2014) it assumes that the capabilities and characteristics of the servers are similar to deal with requests, in addition to the large consumption of time. Also, the distribution process is in a blind sequence without taking into account the size of the load on that server or the size of the file to be handled, for example, the first file goes exclusively to the first server, the second and third files to the second and third servers respectively, etc. regardless of the file size, server capacity, congestion in the path between clients and servers or the bandwidth in between.

Assignment Model

It is a particular situation of a transmission dilemma, so that requires different clients to be paired to different receivers so that, the total cost for a couple is minimized or maximized.

Structure of Assignment Problem

Table 1 Structure of Assignment Problem

				Server			
Results		1	2	………	j	………	n
	1	t_{11}	t_{12}	………	t_{1j}	………	t_{1n}
	2	t_{21}	t_{22}	………	t_{2j}	………	t_{2n}
	.	.	.				
	.	t_{i1}	t_{i2}		t_{ij}		t_{in}
	.	.	.				
	n	t_{n1}	t_{n2}		t_{nj}		t_{nn}

Where n indicates the number of requests or number of Clint (same number of servers) and t_{ij} the connecting cost between Clint *i* and server *j*.

1. Mathematical Formulation of the Assignment Problem

The standard allocation problem is to allocate some functions to an equal number of servers to achieve the goal of maximizing or minimizing costs. Each server is specifically assigned one function, and each function is specifically assigned one specific server to implement reducing the total cost of assigning servers to functions.

2. Mathematical form of the Assignment Problem is as follows (Bufardi, 2008, Taha, 2013 and Lee et al, 1983)

Let X_{ij} = the assignment of server *i* to job *j* such that:

$$X_{ij} = \begin{cases} 0, \text{ if the } i^{th} \text{ server is not assigned to } j^{th} \text{ job.} \\ 1, \text{ if the } i^{th} \text{ server is assigned to } j^{th} \text{ job.} \end{cases}$$

Then the form is given by:

$$Mninmize\ Z = \sum_{j=1}^{n} \sum_{i=1}^{n} Cij.\mathrm{Xij}$$

Subjected to constraint $\sum_{j=1}^{n}$ $, j = 1,2,3, \ldots\ldots n$ (one job for each server)

$\sum_{i=1}^{n}$ $, j = 1,2,3, \dots\dots n$ (one server for each job)
And X_{ij}= 0 or 1

Where for all i, j = 1, c_{ij} is the cost of assigning server i to job j, x_{ij} = 1 means that server i is assigned to job j and x_{ij} = 0 means that server i is not assigned to job j.

Also, in addition to reducing the cost of allocation to the least possible, the problem of allocation may address other important functions such as reducing the time of completion then it is called the problem of cost minimization assignment.

Sonia mentioned (Puri, 2008) that different methodologies have been proposed such as dual primary algorithms, simplicity-like procedure, cost management and forest algorithms, even also relaxation techniques to resolve the customization problem for the purpose of reducing the cost.

Further. It is known that the Hungarian method developed by Kuhn is a first practical way to solve SAP (Systems and Products Applications in Data Processing). Many improvement problems are of a multi-objective nature and rarely a single objective is sufficient to fully contain aspects of the problem.

As well as assignment problems, as they can also include multiple objectives.

One of the important objectives in the problem of assigning functions to the servers is to minimize the time of completion.

Hungarian Algorithm

The Hungarian algorithm (Kuhn, 2010) has four proceedings in which the first two proceedings are Implemented only once, while the remaining are reiterated until the optimal task is found. The input of the algorithm is a square matrix n by n with only positive numbers.

Proceeding 1: subtract the lowest value for each row and from each value in that row.
Proceeding 2: subtract the lowest value for each column, and each value in this column.
Proceeding 3: Use as minimal as possible horizontal and vertical lines to pass all zeros in the resulting array. If less than n Line is needed, go to the next proceeding otherwise the algorithm is stopped and the solution exists.
Proceeding 4: look for the less value does not pass by the lines which were in the second proceeding above, and then subtract this value from all the values that are not covered by

the lines and add to all the values that are located at the intersection of horizontal and vertical lines.

Proposed Work

Let C_i denoted to the client i, S_j denoted to the server j, F_j denoted to the file j and i, j =1......n, B_{ij}= bandwidth capacity between client i and server j, n= the number of clients/files= number of the servers.

The clients' requests to upload/download n number of different files F, where all the servers are in deferent location, the proxy determines which servers will be handled and which is equal to the number of files to be downloaded, Calculate the value of the bandwidth (Sarr et al, 2006 ; Johnsson et al, 2005) between the client and the server as shown in the Table 2, apply the Hungarian algorithm to get the lowest value of the bandwidth by selecting a specific server for each particular file. Finally, the result will determine how to distribute files to servers based on consumption of the lowest possible bandwidth.

Table 2 Bandwidth Structure between *the Client and the Server*

Clients / Server	S_1	S_2	....	S_j	....	S_n
C_1	B_{11}	B_{12}	....	B_{1j}	....	B_{1n}
C_2	B_{21}	B_{22}	....	B_{2j}	....	B_{2n}
.	.	.	....	.	....	.
.	B_{i1}	B_{i2}	....	B_{ij}	....	B_{in}
.	.	.	....	.	....	.
C_n	B_{n1}	B_{n2}	....	B_{nj}	....	B_{nn}

Case Study Example

Assume you currently have three deferent clients' requests to upload/ download three file F1, F2, and F3 respectively, to/from three deferent servers S1, S2, and S3 in deferent location. Bagdad, India and Russia respectively, the table below (Table 3) shows the bandwidth capacity between the clients and servers:

Table 3 Bandwidth Capacity between the Clients and Servers

	Server 1	Server 2	Server 3
Clint 1 / F1	4000 Mbps	4000 Mbps	3500 Mbps
Clint 2 / F2	4000 Mbps	6000 Mbps	3500 Mbps
Clint 3 / F3	2000 Mbps	4000 Mbps	2500 Mbps

The question: where would you upload/download each file in order so that the bandwidth should be minimize?

Appling proceeding 1 of Hungarian algorithm, the result as sown in Table (4).

Table 4 Subtract the Lowest Value from Each Row

	Server 1	Server 2	Server 3
Clint 1 / F1	500 Mbps	500 Mbps	0
Clint 2 / F2	500 Mbps	2500 Mbps	0
Clint 3 / F3	0	2000 Mbps	500 Mbps

Applying Proceeding 2 of the Hungarian algorithm, the result appears as shown in Table (5).

In this proceeding we can cover all zeros in 3 lines, which are the same dimensions as the matrix, so the algorithm stops.

Table 5 Subtract the Lowest Value from Each Column

	Server 1	Server 2	Server 3
Clint 1 / F1	500 Mbps	0	0
Clint 2 / F2	500 Mbps	2000 Mbps	0
Clint 3 / F13	0	1500 Mbps	500 Mbps

Table 6 Assign Files to the Corresponding Servers

	Server 1	Server 2	Server 3
Clint 1 / F1		Clint 1	
Clint 2 / F2			Clint 2
Clint 3 / F3	Clint 3		

After Appling the Hungarian algorithm, we find the optimal solution for uploading/ downloading files is: F1 of clint1 from server 2, F2 of clint2 from server 3 and F3 of clint3 from server1. Where the amount of bandwidth consumed will be 9500 Mbps, which represents the lowest value of the amount of bandwidth to implement the mentioned orders.

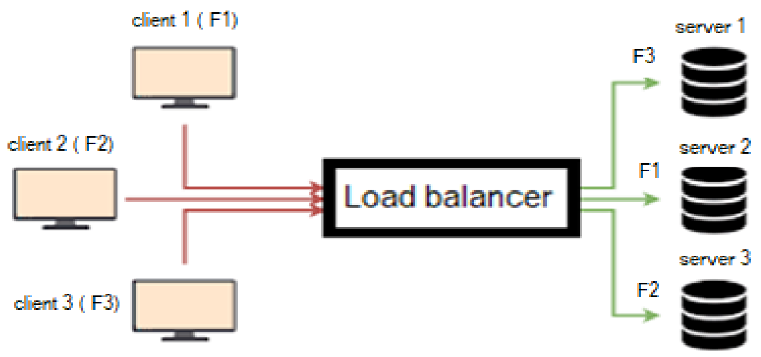
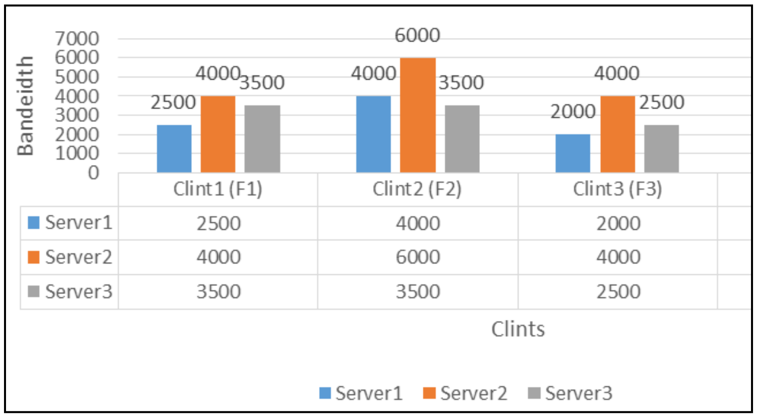


Figure 1 Send Files to Corresponding Servers via Load Balancer



	Clint1 (F1)	Clint2 (F2)	Clint3 (F3)
Server1	2500	4000	2000
Server2	4000	6000	4000
Server3	3500	3500	2500

Figure 2 Bandwidth between files and servers

Bandwidth value between each client and server.

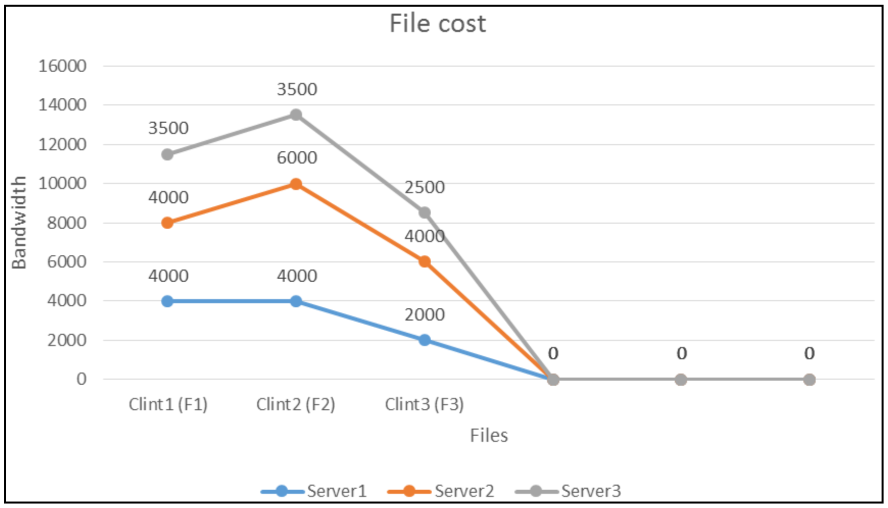


Figure 3 Bandwidth Value Cost Scheme between Clients and Servers

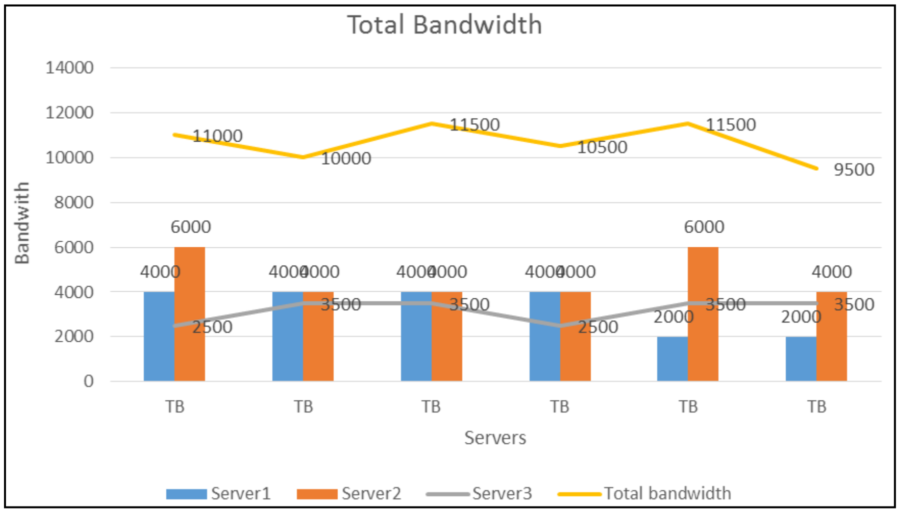
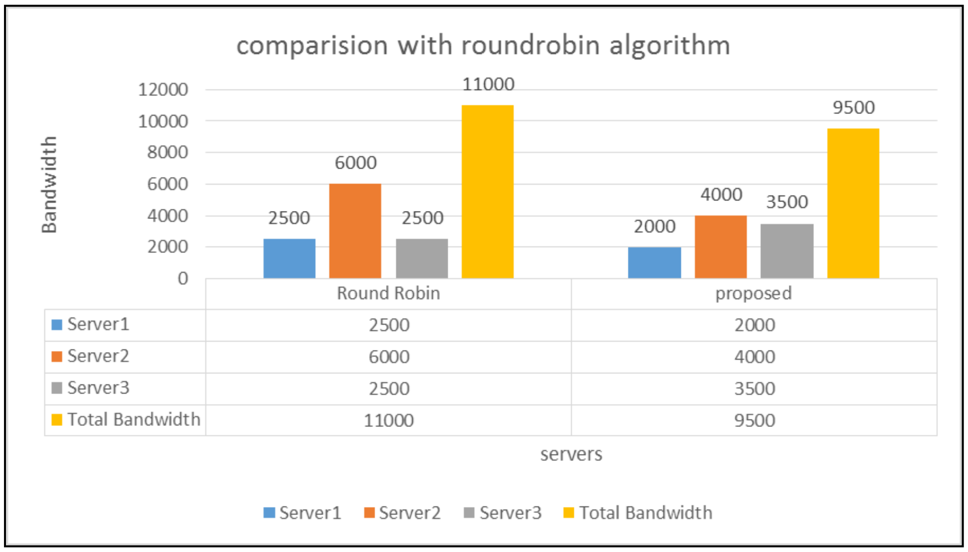


Figure 4 The Graph Shows the Bandwidth Value of All Possibilities If Each File is Distributed on One Different Server

From the above figure (Figure 4} we can see that the amount of bandwidth lies between two values, the highest value is 11000 and the smallest is 9500.



	Round Robin	proposed
🟦 Server1	2500	2000
🟧 Server2	6000	4000
⬜ Server3	2500	3500
🟨 Total Bandwidth	11000	9500

Figure 5 Comparison of the Proposed Method and the Round Robin Algorithm

Conclusion

In the case of the traditional round Robin algorithm, each file moves to a specific server, where the first file is requested from the first server, the second file and third one is requested from the second and third server, respectively, and so on. This method will consume more bandwidth about (11,000 Mbps), according to the example. The proposed method uses about 9500 Mbps, with a difference of 1500 Mbps. This difference in bandwidth consumption came as a result of optimal routing of requests to the appropriate servers. As a result, this will lead to a decrease in cost. In this way, sending the request to the non-convenient server is avoided so that the amount of bandwidth available is not suitable (higher) for executing that request, and the result for the total work, this may be lead to disconnection, or increase the time to fulfill those requests.

References

Anthony, R. (2015). *Systems programming: designing and developing distributed applications.* Morgan Kaufmann.

Almuttairi, R.M., Wankar, R., Negi, A., & Rao, C.R. (2011). Enhanced data replication broker. *In International Workshop on Multi-disciplinary Trends in Artificial Intelligence, Springer, Berlin, Heidelberg,* 286-297.

Almuttairi, R.M., Wankar, R., Negi, A., & Rao, C.R. (2010). Intelligent replica selection strategy for data grid. *In GCA 2010: proceedings of the international conference on grid computing & applications (Las Vegas NV),* 95-101.

Almhanna, M.S. (2017). Minimizing replica idle time. *In Annual Conference on New Trends in Information & Communications Technology Applications (NTICT),* 128-131.

Almuttairi, R.M., Wankar, R., Negi, A., Chillarige, R.R., & Almahna, M.S. (2010). New replica selection technique for binding replica sites in data grids. *In 1st International Conference on Energy, Power and Control (EPC-IQ),* 187-194.

Almuttairi, R.M Almuttairi, R.M., Wankar, R., Negi, A., & Rao, C.R. (2010). Replica selection in data grids using preconditioning of decision attributes by k-means clustering (K-RSDG). *In Second Vaagdevi International Conference on Information Technology for Real World Problems*, 18-23.

Almuttairi, R.M., Wankar, R., Negi, A., & Rao, C.R. (2010). Smart replica selection for data grids using rough set approximations (RSDG). *In International Conference on Computational Intelligence and Communication Networks,* 466-471.

Abbas, S.A., & Almhanna, M.S. (2021). Distributed Denial of Service Attacks Detection System by Machine Learning Based on Dimensionality Reduction. *In Journal of Physics: Conference Series, 1804*(1).

Ammar, R.A., Sarhan, A.M., & Ragab, H.A.M. (2011). Achieving the workload balance of the clusters. *In IEEE International Symposium on Signal Processing and Information Technology (ISSPIT),* 086-092.

Afzal, S., & Kavitha, G. (2019). Load balancing in cloud computing–A hierarchical taxonomical classification. *Journal of Cloud Computing, 8*(1), 1-24.

Bufardi, A., 2008. On the efficiency of feasible solutions of a multicriteria assignment problem. The Open Operational Research Journal, 2(1).

Chapin, S.J., Katramatos, D., Karpovich, J., & Grimshaw, A.S. (1999). The legion resource management system. *In Workshop on Job Scheduling Strategies for Parallel Processing, Springer, Berlin, Heidelberg,* 162-178.

Chun, B.N., & Culler, D.E. (2000). *Market-based proportional resource sharing for clusters,* Berkeley: Computer Science Division, University of California, 1-19.

Foster, I., & Kesselman, C. (1997). Globus: A metacomputing infrastructure toolkit. The *International Journal of Supercomputer Applications and High Performance Computing, 11*(2), 115-128.

Ghutke, B., & Shrawankar, U. (2014). Pros and cons of load balancing algorithms for cloud computing. *In International Conference on Information Systems and Computer Networks (ISCON),* 123-127.

Joshi, S., & Kumari, U. (2016). Load balancing in cloud computing: Challenges & issues. *In 2nd International Conference on Contemporary Computing and Informatics (IC3I),* 120-125.

Johnsson, A., Melander, B., & Björkman, M. (2005). Bandwidth measurement in wireless networks. *In IFIP Annual Mediterranean Ad Hoc Networking Workshop, Springer, Boston, MA,* 89-98.

Kuhn, H.W. (1955). The Hungarian method for the assignment problem. *Naval research logistics quarterly, 2*(1-2), 83-97.

Litzkow, M.J., Livny, M., & Mutka, M.W. (1987). Condor-a hunter of idle workstations. *University of Wisconsin-Madison Department of Computer Sciences.*

Lee, S.M., & Schniederjans, M.J. (1983). A multicriteria assignment problem: A goal programming approach. *Interfaces, 13*(4), 75-81.

Nathan, B.T., Mukherjee, D., Paul, K.J.S., Pal, A., & Peter, M. (2019). Efficient Bandwidth Management of ISP by Load Balancing and Link Bundling. *In International Conference on Intelligent Computing and Control Systems (ICCS),* 1288-1292.

Puri, M.C. (2008). Two-stage time minimizing assignment problem. *Omega, 36*(5), 730-740.

Sarr, C., Chaudet, C., Chelius, G., & Guérin Lassous, I. (2006). A node-based available bandwidth evaluation in IEEE 802.11 ad hoc networks. *The International Journal of Parallel, Emergent and Distributed Systems, 21*(6), 423-440.

Taha, H.A. (2013). *Operations research: an introduction.* Pearson Education India.

Zeng, X., Wang, D., Han, S., Yao, W., Wang, Z., & Chen, R. (2019). An effective load balance using link bandwidth for SDN-Based data centers. *In International Conference on Artificial Intelligence and Security,* 256-265.