An Approach to New Technical Solutions in Resource Allocation Based on Artificial Intelligence

Tung Nguyen Trong[1(✉)], Nguyen Hai Vinh Cuong[2], Tran-Vu Pham[3], Nguyen Ha Huy Cuong[4], and Bui Thanh Khiet[2]

[1] Dong A University, 33 Xo Viet Nghe Tinh Street, District Hai Chau, Da Nang City, Vietnam
tungqn@donga.edu.vn
[2] Thu Dau Mot University, Thủ Dầu Một, Binh Duong Province, Vietnam
cuongnhv@tdmu.edu.vn
[3] Faculty of Computer science and Engineering, Ho Chi Minh City University of Technology (HCMUT), 268 Ly Thuong Kiet Street, District 10, Ho Chi Minh City, Vietnam
ptvu@hcmut.edu.vn
[4] The University of Danang, 41 Le Duan Street, District Hai Chau, Da Nang City, Vietnam
nhhcuong@sdc.udn.vn

Abstract. Deadlock is a major problem for systems that allocate resources (AL). There are many solutions to the deadlock problem in distributed systems, the solutions are divided into the following three groups: deadlock-prevention, deadlock-avoidance, and deadlock-detection. AL and related deadlock prevention originate from the design and implementation of operating systems and distributed computing. In this article, we systematize research related to distributed systems, problems of AL, strategies in AL, and solutions to deal with deadlock situations in AL. We present deadlock avoidance algorithms, and deadlock prevention, in addition, we present a deadlock detection algorithm using a two-way search with running time complexity of the horizontal arc O(m1/2) when the edge (v,w) is added to the graph. Compare the two-way search algorithm with the improved algorithm, and finally the experimental results.

Keywords: Resource allocation · Heterogeneous · Deadlock detection · Deadlock Prevention · Virtual machine

1 Introduction

Cloud computing (CC), also known as virtual server computing, is a computing model that uses computer technology and develops based on the Internet. CC provides a high degree of flexibility in deploying, terminating, migrating, and scaling applications and services [1]. Cloud Data Center (CDC) - consists of a series of physical cloud servers connected through high-speed links that provide a variety of cloud computing services such as Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure

S. Nandan Mohanty et al. (Eds.): ICISML 2022, LNICST 471, pp. 325–334, 2023.
https://doi.org/10.1007/978-3-031-35081-8_27

as a Service (IaaS) [2, 3]. In addition, CDC provides a variety of customizable settings and additional features to meet the functional and non-functional requirements of end-users. Efficient management and use of cloud resources are of considerable importance to continuously meet application needs while ensuring Quality of Service (QoS) [4].

Ensuring smooth system protection one of the issues that need to be concerned is Deadlock. In a distributed system, when there exists at least one process that requires a resource to be held and is blocked indefinitely by another process, then the system will generate a deadlock. A set of processes that request resources held by other processes are called a deadlock [5]. In distributed systems, dealing with deadlock problems can be divided into three strategies: deadlock-prevention [6, 7], deadlock-avoidance [8–10], and deadlock-detection [11–15]. Deadlock is a major problem for systems that AL in a distributed system. The main problem in deadlock avoidance is checking the safe resource allocation status and checking whether there is a cycle after the allocation. Deadlock prevention is a sequence of actions that ensures the constraints between the processes in the system and the resources imposed on external actors. These constraints monitor to ensure that external actors do not send requests that cause deadlocks [16] (Fig. 1).

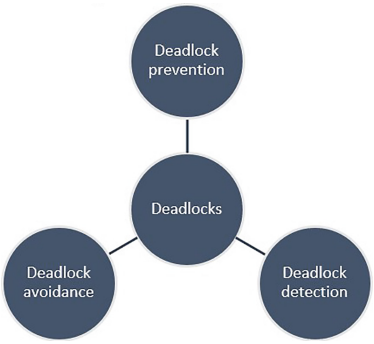

Fig. 1. Types of deadlocks in distributed systems

In the past, deadlock avoid has been proposed in [17] using load balancing algorithm to satisfy user requests. A heterogeneous server-based resource allocation algorithm that overcomes deadlock by harvesting available resources from previously provisioned users [18]. The scheduling algorithm AL for services on heterogeneous virtual server platforms, in order to improve the detection algorithm and prevent deadlocks. [19]. Wysk et al. [7] has developed a deadlock detection algorithm by finding optimal time intervals through integer programming. The algorithm uses constraints to ensure agents do not release resources unless specified. Classify the local and global deadlock detection problem in component-based systems as NP-hard [20]. The reinforcement learning method is proposed to find the solution to optimize the schedule in resource allocation, and the processes used in deadlock detection [21].

Difference between cloud computing and grid computing in resource Allocation, batch scheduling problem [22]. Recent studies have mostly focused on application performance and resource variability for infrastructure. In this study, we explore solutions related to the deadlock problem in distributed systems, then propose an improved deadlock detection algorithm using a two-way search algorithm. Comparing the results with the previous algorithm, the evaluation shows that the algorithm has improved performance and efficiency in the system with heterogeneous resources.

The article has the following structure: in Sect. 2, we present related works; in Sect. 3, we present the existing models; in Sect. 4, we present approaches for improving the Two – way algorithm; in Sect. 5, we present our conclusions and suggestions for future work.

2 System Model Resource Allocation in Heterogeneous Distributed Platforms

Resource provisioning in cloud computing is divided into two main areas, which are strategy-based and parameter-based. Depending on the characteristics, different purposes to use different strategies Fig. 2.

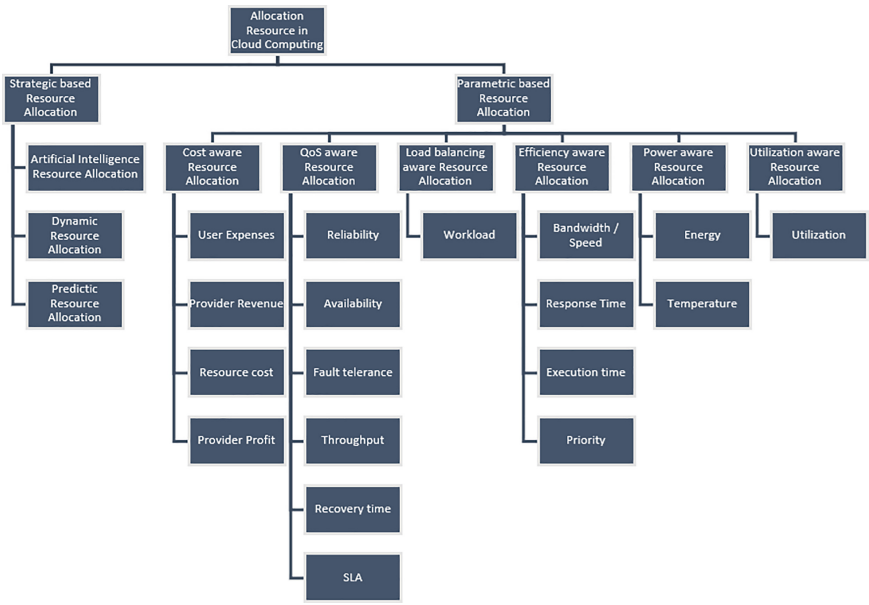

Fig. 2. Categorization of resource allocation in cloud computing

The advent of heterogeneous system computing presents many advantages over traditional distributed computing and heterogeneous computing which includes many different types of architectures. Cloud computing has a different resource provisioning model than scheduling, or grid computing [23–26].

2.1 The Application

A set of processes p1, p2,…pn transmit messages over the network, each using its processor, different memory, different physical clocks, and different means of communication. Out of order, messages may be duplicated or truncated, processors may fail, or network links may be broken, as defined in a heterogeneous distributed system. A directed graph consists of nodes and edges, in which processes are represented as nodes, the edges of the graph are considered as packets transmitted between nodes, so the distributed system will represent directed graphs.

The background charts for the grid platform. Figure 3 depicts eight processes, each of which is equivalent to an independent processor in memory. The author has modeled the heterogeneous set of resources and linked them as nodes and edges. Example 1 A example simple platform.

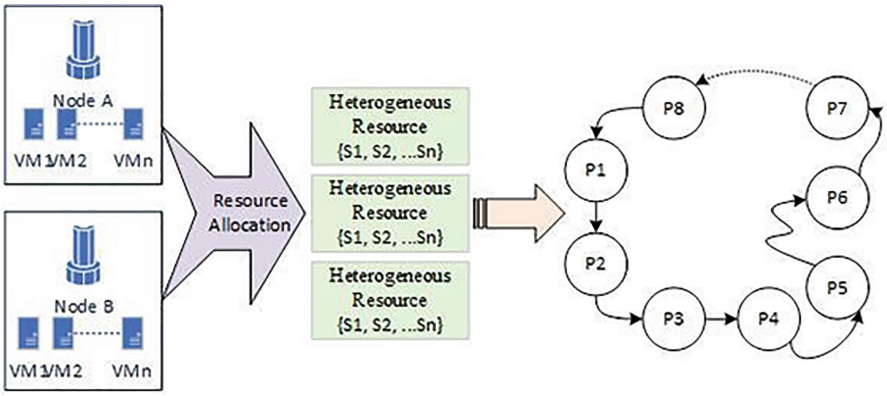

Fig. 3. A example simple platform

The resource allocation process, a process is always in one of two states: running or blocked. The process that is provided with the required resources will execute and stay running. Conversely, a process that requests a resource that is being held by another process will be in a waiting state.

2.2 Wait–For–Graph (WFG)

For distributed systems, we use a directed graph to model the state of the system, called a waiting-for-graph (WFG) [24]. In a WFG, Processors are represented as Nodes of the graph, and edges are represented as resource requests, for example, edge E1 is directed to E2, if E1 is waiting for resource E2, waiting for E2 to release the resource. If the directed edges E form a cycle, the processes occupy the resources, there is a cycled WFG graph, and the system will be locked.

2.3 The Algorithms Distributed System Resources Detection

Step 1. Build an agent whose task is to list the states N1, N2, ….Nm with the property of reachability. In other words, from an agent point A can represent N1, N2, ….Nm, so that all other agents know and can access.

Step 2. The Breadth-First-Search (BFS) is created with the nodes below:

{N1}, {N2},…,{Nm}, {N1, N2},…, {Nm − 1, Nm},…, {N2,…, Nm}, {N1, N2,…, Nm}

Step 3. Each unique node is checked, based on BFS, and following these steps:

a) All resource-equivalent prerequisites in the node are set to "true".

b) Resource requests in the list of nodes are the cause of the infinite loop, calculate the cost generated and return nodes if there is a path.

Algorithm execution results show occupied resources, these are resources held by other processes. These resources will be the input list to run the deadlock-detection algorithm.

1. Mitchell and Merritt's deadlock detection algorithm

Mitchell and Merritt's algorithms belong to the group of edge-chasing algorithms where the probe is sent upwards directly on the edge of the WFG. The algorithm appears in four steps as follows:

- Blocked
- Active
- Transmit
- Detect

The algorithm works by assigning labels to processes, a private label, and a public label. The sequence of the algorithm is as follows: first, a public label is assigned to all processes and initialized to the initial value. Then each process's label is assigned to the corresponding value. When a process P requests a process Q, both processes will be assigned a label greater than that of P and referred to initially. This is the blocking step, at which point P is considered blocked. While process P is blocked, process Q is waiting. Meanwhile, if the above public label on process Q is greater than P, then process P puts the public label on process Q. This is the transition step. In the system, if there is a waiting cycle M, the maximum transition is M - 1 before deadlock is detected.

2. Algorithm to Avoid deadlock

To avoid deadlock, we assign a list of groups called "bad groups" to all active agents and name it "To-Avoid-List". In this list, the elements have their sublists. Thus, the list contains the states of the active agents of the deadlock elements detected by the system in the previous steps. In the event of an environmental impact, agents actively check the ToAvoid-List to ensure that the system does not perform a bad transition. When the request process is sent to an active agent, it predicts its next persistent state. The agent's state prediction system actively checks the ToAvoid List, to find if the state is in the sublist. If the alert state is present, a query is sent to the other agents active in the sublist. If the current state of all agents matches the states in the sublist, the end system enters a deadlock state. Therefore, agents actively prevent requests from performing tasks, thereby avoiding deadlocks.

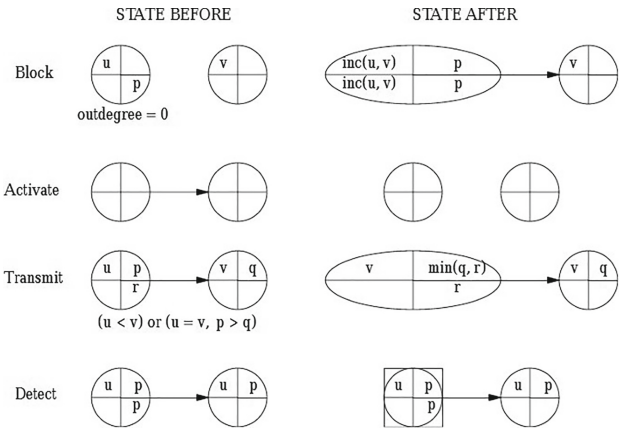

Fig. 4. The deadlock detection model for WFG

3. Deadlock detection for resource allocation using Two - way search in heterogeneous distributed platforms

In this study, we propose an algorithm for deadlock detection, saving the properties of a directed graph with n vertices, when a new node is added to the graph by a two-way search algorithm. The minimum time for the algorithm to detect deadlocks takes O(m1/2), the time limit for adding m edges to the directed graph. The results show that when the edge detection algorithm (v, m) always exists a path consisting of edges connecting vertices from *m* to *v*.

2.4 Algorithmic Approach to Deadlock Detection

The two-way purple search algorithm finds the cycles and then rearranges the order of the vertices if there is no cycle in the graph. The efficiency of a two-way search algorithm depends on the number of edges visited during cycle detection in the graph.

Fig. 5. Deadlock Detection Algorithms In Distributed System

C_j^{cpu}, C_j^{ram}: Maximum available capacity of CPU, RAM class IaaS provides j
r_j^{cpu}, r_j^{ram}: The new resource value of IaaS provides j (Fig. 5).

3 Our Algorithm to Prevention Deadlock

Consider the process Pi requesting resources, to solve deadlock prevention problems. Where P8 and P2 (Fig. 4) both processes request r1 and r2. Preventing babies from shutting down will help provide the best resources. Efficient resource scheduling prevents deadlocks.

Therefore, the proposed algorithm acts as the engine that provides the necessary content about the process situations whether each is in progress, missing or waiting, and this fact can be expressed by graph through the dependency graph presented in this paper. So, when the process waits - for a broken dependency, the corresponding content will be restored immediately from the system, if not a deadlock will occur.

Set {r1, r2,…, rn} is the set of available resources, processes P access resources R. Each process can access only a subset. Processes are required to follow the rule

- Any session, any process can call *re_resource(rk)* only when it has received all the resources *rj* it needs, that is rj < rk.
- Since process p1 at the moment owns the resource *Ra* and is waiting for the resource *Rb*, it calls *re_resource(Ra)* first and then *re_resource(Rb).*
- When process *p2* owns resource *Xb* and is waiting for the resource to come out, it first calls *re_resource(Rb)* and then it *re_resource(Rb).*

In heterogeneous distributed platforms, we propose to prevent deadlock algorithm includes the following steps (Fig. 6):

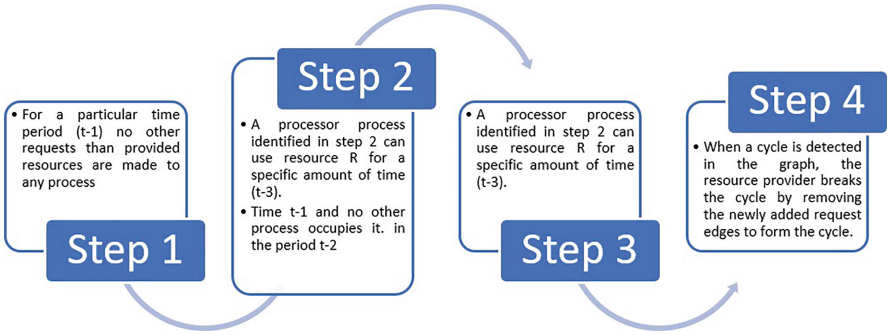

Fig. 6. Algorithm to prevention deadlock

4 Experiments and Results

In this study, we use a two-way improved cloud computing resource allocation method, experimental results have been verified on CloudSim, the platform is an open-source platform, and the research team has used Java language to implement programming classes

implementing algorithms [19]. Experimental results show that the two-way improved algorithm has many advantages Table 1.

Table 1. Comparison of the optimal time of our algorithm to Two-way algorithm

Cloudlet ID	Data center ID	VM ID	TWO-WAY			TWO-WAY-IMPROVED			Improved
			Start	End	Time	Start	End	Time	(%)
0	1	1	0.1	100.1	100	0.1	60.1	60	25%
1	2	2	0.1	110.1	110	0.1	70.1	70	22%
2	3	3	0.1	132.1	132	0.1	70.1	70	31%
3	3	4	0.1	145.1	145	0.1	90.1	90	23%
4	1	5	0.1	147.1	147	0.1	100.1	100	19%
5	2	6	0.1	145.1	145	0.1	104.1	104	16%
6	1	7	0.1	152.1	152	0.1	104.1	104	19%
7	2	8	0.1	153.1	153	0.1	110.1	110	16%
8	2	9	0.1	163.1	163	0.1	105.1	105	22%
9	2	10	0.1	168.1	168	0.1	111.1	11	20%
10	3	11	0.1	170.1	170	0.1	110.1	110	21%

Comparing the experimental results between the two algorithms, in most cases, the improved algorithm's execution time is better than the lowest 16% and the highest 31% (Fig. 7).

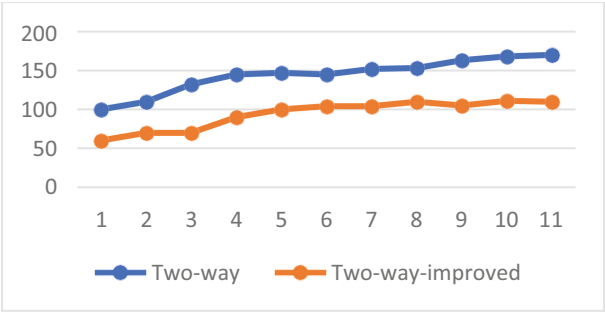

Fig. 7. Comparison the optimal time of algorithms

5 Conclusion

We learned about deadlock-related issues and introduced deadlock detection, deadlock prevention, and approaches in distributed system resource provisioning.

We proposed a deadlock detection algorithm approach implemented for resource allocation across heterogeneous cloud computing platforms. The experimental results in time complexity O(min(m1/2), which in real cases has improved approximately orders of magnitude. In the study, an improved algorithm for providing resources at the IaaS infrastructure layer on a distributed platform was proposed not homogeneous. Based on an improved two-way search algorithm (Two-Way).

From the experience of this study, we found that the application of appropriate scheduling algorithms will bring optimal performance for cloud computing resources, contributing to the prevention of deadlocks. In the next study, we will study to extend the distributed resource provisioning for virtual server systems. Simultaneously, continue to expand simulation and experimental methods.

Acknowledgment. This research is supported and funded by Thu Dau Mot University under Grant Number DT.22.1-010. I am grateful to all of those with whom I have had the pleasure to work during this and other related projects. I would also like to extend our thanks to The University of Da Nang and Dong A university for providing the opportunity to work with experienced professionals in the fields of machine learning and cloud computing.

References

1. Wu, J., Guo, S., Li, J., Zeng, D.: Big data meet green challenges: greening big data. IEEE Syst. J. **10**(3), 873–887 (2016). https://doi.org/10.1109/JSYST.2016.2550538
2. Banerjee, P., et al.: Everything as a service: powering the new information economy. Computer (Long Beach Calif.) **44**(3), 36–43 (2011). https://doi.org/10.1109/MC.2011.67
3. Rajkumar, M., Mariyadoss, M.K., Kandan, M.: Strategies for Resource Allocation in Cloud Computing: A Review Test Data Compression Methods: A review View project Hadoop Map Reduce View project Strategies for Resource Allocation in Cloud Computing: A Review. https://www.researchgate.net/publication/313160586
4. Kumar, N., Chilamkurti, N., Zeadally, S., Jeong, Y.S.: Achieving quality of service (QoS) using resource allocation and adaptive scheduling in cloud computing with grid support. Comput. J. **57**(2), 281–290 (2014). https://doi.org/10.1093/comjnl/bxt024
5. Khanna, D., Patel, T.P.: Deadlocks avoidance in Cloud computing using enhanced load balancing approach. IJRAR-Int. J. Res. Anal. Rev. (2018). http://ijrar.com/
6. Chao, D.Y., Chen, T.Y., Chen, J.T., Wu, K.C.: A best deadlock control for S3PMR to reach all states. Asian J. Control **14**(1) (2012). https://doi.org/10.1002/asjc.370
7. Leung, Y.T., Sheen, G.-J.: Resolving deadlocks in flexible manufacturing cells (1993)
8. Yu, T.H., Chao, D.Y.: Constructing the closed-form solution of control-related states real-time information for insufficient k-th order systems with one non-sharing resource to realize dynamic modeling large TNCS systems of petri nets. J. Chin. Inst. Eng. Trans. Chin. Inst. Eng. Ser. A **42**(3) (2019). https://doi.org/10.1080/02533839.2018.1562991
9. Roszkowska, E.: Coordination of cyclic motion processes in free-ranging multiple mobile robot systems. In: Bożejko, W., Bocewicz, G. (eds.) Modelling and Performance Analysis of Cyclic Systems. SSDC, vol. 241, pp. 87–104. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-27652-2_5
10. Shanu, S., Sastry, H.G., Marriboyina, V.: Optimal solution approach on large scale data to avoid deadlocks in resource allocations. Mater. Today: Proc. **47**, 7162–7166 (2020). https://doi.org/10.1016/j.matpr.2021.06.357

11. Rout, K.K., Mishra, D.P., Salkuti, S.R.: Deadlock detection in distributed system. Indonesian J. Electr. Eng. Comput. Sci. **24**(3) (2021). https://doi.org/10.11591/ijeecs.v24.i3.pp1596-1603
12. Lu, F., Cui, M., Bao, Y., Zeng, Q., Duan, H.: Deadlock detection method based on Petri net mining of program trajectory. Jisuanji Jicheng Zhizao Xitong/Comput. Integr. Manuf. Syst. CIMS, **27**(9) (2021). https://doi.org/10.13196/j.cims.2021.09.014
13. Knapp, E.: Deadlock detection in distributed databases. ACM Comput. Surv. **19**(4) (1987). https://doi.org/10.1145/45075.46163
14. Lu, F., Tao, R., Du, Y., Zeng, Q., Bao, Y.: Deadlock detection-oriented unfolding of unbounded Petri nets. Inf. Sci. (N Y) **497** (2019). https://doi.org/10.1016/j.ins.2019.05.021
15. Rout, K.K., Mishra, D.P., Salkuti, S.R.: Deadlock detection in distributed system. Indonesian J. Electr. Eng. Comput. Sci. **24**(3), 1596–1603 (2021). https://doi.org/10.11591/ijeecs.v24.i3.pp1596-1603
16. Li, Z., Shpitalni, M.: Smart deadlock prevention policy for flexible manufacturing systems using Petri nets. IET Control Theory Appl. **3**(3) (2009). https://doi.org/10.1049/iet-cta:20070399
17. Wu, C.W., Lee, K.J., Su, A.P.: A hybrid multicast routing approach with enhanced methods for mesh-based Networks-on-Chip. IEEE Trans. Comput. **67**(9) (2018). https://doi.org/10.1109/TC.2018.2813394
18. Nguyen, H.H.C., Doan, V.T.: Avoid Deadlock Resource Allocation (ADRA) model V VM-out-of-N PM. Int. J. Innov. Technol. Interdisc. Sci. **2**(1), 98–107 (2019). www.IJITIS.org, https://doi.org/10.15157/IJITIS.2019.2.1.98-107
19. Huy, H., Nguyen, C., Le, S., Le, V.S.: Detection and Avoidance Deadlock for Resource Allocation in Heterogeneous Distributed Platforms (2015). www.ijcst.org
20. Minnameier, C.: Local and global deadlock-detection in component-based systems are NP-hard. Inf. Process. Lett. **103**(3), 105–111 (2007). https://doi.org/10.1016/j.ipl.2007.02.016
21. Chen, M., Rabelo, L.: Deadlock-detection via reinforcement learning. Ind. Eng. Manage. **6**(03) (2017). https://doi.org/10.4172/2169-0316.1000215
22. Huy, H., et al.: Technical solutions to resources allocation for distributed virtual machine systems (2015). http://sites.google.com/site/ijcsis/
23. Cu`ơng, N.H.H., Sơn, L.V.: 'Kỹ Thuật Cung Cấp Tài Nguyên Cho L´ớp Hạ Tầng (IAAS) Technical Resources Provision for Infrastructure (2014). http://www.cs.huji.ac.il/
24. Kshemkalyani, A.D., Singhal, M.: Distributed Computing: Principles, Algorithms, and Systems (2008)
25. Kumar, K., Feng, J., Nimmagadda, Y., Lu, Y.H.: Resource allocation for real-time tasks using cloud computing (2011). https://doi.org/10.1109/ICCCN.2011.6006077
26. Professor, D.: A Survey on Resource Allocation Strategies in Cloud Computing (2012). www.ijacsa.thesai.org