WEB AUTO CONFIGURATION FOR N-TIER IN VM BASED DYNAMIC ENVIRONMENT BY REINFORCEMENT LEARNING APPROACH: A STUDY

Keyur Prajapati[1] and Dr. Dinesh J Prajapati[2]

[1]PhD Scholar, Department of Computer Engineering,
M.B. Patel Institute of Technology, New V.V. Nagar, India
[2]Associate Professor, Department of Information Technology,
A. D. Patel Institute of Technology, New V.V. Nagar, India

ABSTRACT

In Web system, configuration is the crucial part to achieve performance with service availability. Now in days, because of dynamics web traffic, virtualization is the key factor. How to handle required resources is a challenging task in virtual environment. Apply optimize configurations for different servers as per available resources is a tedious task to achieve high throughput with low latency.

In this paper we have described the studied methodology of machine learning, which will guide how optimize all the parameters with the best results in terms of web usability.

KEYWORDS

Virtualization, Resource Management, Reinforcement Learning, Q-Learning, SARSA.

1. INTRODUCTION

Web systems for Apache and Tomcat bases applications normally contain many parameters; their settings are crucial to systems performance and service availability. Manual configuration based on operator's experience is a lethargic and error-prone task. Current studies notify that more than 60% root causes of Internet service outages was due to false configuration caused by administrator mistakes. The configuration challenge is due to several reasons. First is the scalability of system and complexity which introduce more and more configurable parameters to a level beyond the capacity of an average-skilled operator. For example, both Apache server and Tomcat server have more than a hundred configurable parameters to set for different tuning environments. In a multi-component system, the interaction between the components makes performance tuning of the parameters even harder.

Past studies devoted to autonomic configuration of web systems. Most of them focused on performance parameters tuning for dynamic traffic in static environments. Their optimization approaches are very hard applicable to online setting of the parameters in VM-based dynamic platforms due to their different level complexity. There were a few control approaches targeted at online tuning in response to changing workload. They were largely limited to tuning of single Max Client parameter because of the inherent workload balancing complexity.

In this paper, we propose a reinforcement learning approach, namely RAC, for automatic configuration of multi-tier web systems in VM-based dynamic environments. Reinforcement

learning is a process of taking actions based on previous results. For a web system, it's possible configurations form a state space. We define actions as reconfiguration of the parameters. Reinforcement learning is intended to determine appropriate actions at every step to maximize the highest reward. Recent studies showed the feasibility of RL approaches in resource allocation power management [17] job scheduling in grid and self-optimizing memory controller and auto tuning config parameters. To best of our knowledge, the RAC approach should be the first one in the application of the RL principle to automatic configuration of web systems.

RL will help the system admin for configuring the files at once. Rest at all, it will apply its own algorithms based on past rewards and finally optimize the whole system at its own best shown in fig [1].

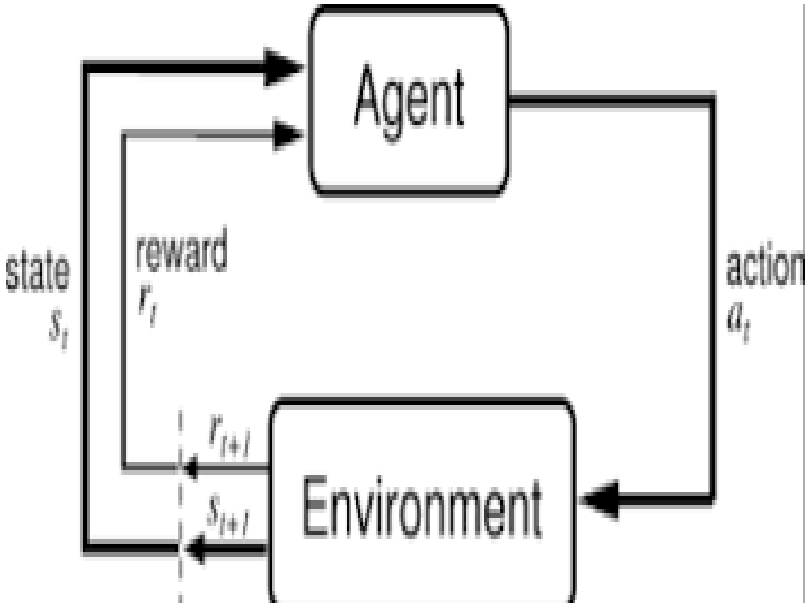

Fig 1 [18]

2. CHALLENGES IN WEB CONFIGURATION

2.1. Match Configuration to Workload

Based on dynamic environment, lot of configurations is needed to be set to achieve performance hike. To optimize web server and application server's config parameters, one agent (watcher) would be there, and it must take its own decision to apply result and dispatch them to achieve good result in terms of performance.

2.2. Match Configuration to Dynamic VM Environments

Now in days VM are quietly used to host web sites and web applications. Possibly more than one OS are to be hosted in VM and so we can manage resources in between (like storage and Memory).

Our main aim is how to manage resources in between two or more OS through VM. If we are talking about single OS in VM and it can communicate to host OS and manage resources but its toughest job to deal with OS and it is looking like to impossible task to play with OS internals as shown in fig [2].

• Proposed research work meets the requirement of dynamic web movement with large set of databases with stop-the-world hike.
• Architecture that enables auto config implementations for the context of web.
• Proposed work fit in short pauses with low latency and high throughput.
•Provide some of the performance metrics which will help others in testing these implementations in different environments.

3. REINFORCEMENT LEARNING APPROACH TO AUTO CONFIGURATION

3.1. Parameter Selection and Auto-configuration

In VM with existing execution of web, RL agent will decide parameters to be set and deploy it.

RL agent's crucial task is to decide the list of parameters is to be optimized. Because of interrelated configuration parameters, to load auto scalar is also the responsibility of RL agent.

3.2. RL-based Decision Making

Reinforcement learning is a process of learning through interactions with an external environment (or the web system in this paper). The reconfiguration process is typically formulated as a finite Markov decision process (MDP), [19] which consists of a set of states and several actions for each state.

State Space: For the online auto-configuration task, we define a state as possible system configuration. For the selective group of n parameters, we represent a state by a vector in the form as: si = (Para1, Para2, · · ·, Paran).

Action Set: We define three basic actions: increase, decrease, and keep associated with each parameter. We use a vector ai to represent an action on parameter i. Each element itself is a 3-element vector, indicating taken/not taken (1/0) of three actions. For example, the following notation represents an increase action on parameter i: an increase i = (· · ·, Parai(1, 0, 0), Paran(0, 0, 0)) immediate reward. The immediate reward should correctly reflect the system performance.

The immediate reward r at time interval t is defined as rt = SLA − perft, where SLA is a reference time predefined in-Service Level Agreement, and perf is measured response time. For a given SLA, a lower response time returns a positive reward [20] to the agent; otherwise, the agent will receive a negative penalty.

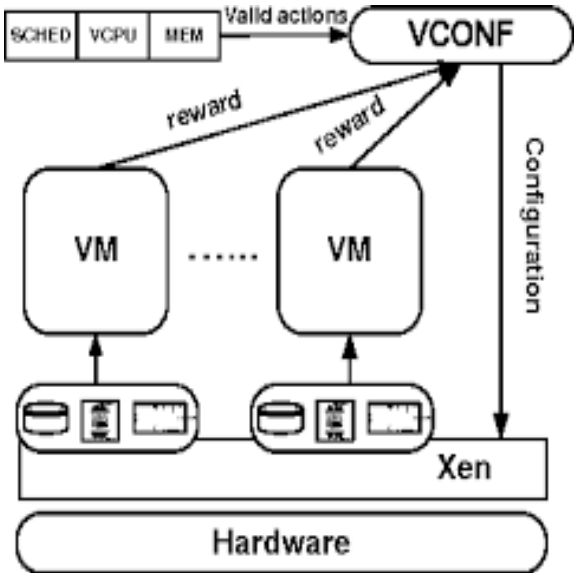

Fig 2 [16]

4. EFFECT OF EXPLORATION

How to balance exploration and exploitation is one of the challenges in online RL algorithms. Insufficient explorations would result in suboptimal configurations while too much exploration would incur prohibitive performance degradation. Effect of the exploration rate in RAC performance is very important part. Two types of explorations were considered: in batch training and in online learning.

Existing Methodologies (Reinforcement Learning):

1) Model Based: In the context of Model based, agent hired existing policies which are available in the model. It is not able to create its own new policy but choose right one among the existing in the model.
2) Model Free: In this era, intelligent agent can create a new policy by its own new idea and deploy it to optimize the parameters for achieving high throughput and low latency.

Q -learning: Q-learning is an off-policy reinforcement learning algorithm that seeks to find the best action to take given the current state. It's considered off-policy because the q-learning function learns from actions that are outside the current policy, like taking random actions, and therefore a policy isn't needed. More specifically, q-learning seeks to learn a policy that maximizes the total reward.

When q-learning is performed we create what's called a *q-table* or matrix that follows the shape of [state, action] and we initialize our values to zero. We then update and store our *q-values* after an episode. This q-table becomes a reference table for our agent to select the best action based on the q-value.

An agent interacts with the environment in 1 of 2 ways. The first is to use the q-table as a reference and view all possible actions for a given state. The agent then selects the action based on the max value of those actions. This is known as ***exploiting*** since we use the information, we have available to us to decide.

The second way to act is to act randomly. This is called ***exploring***. Instead of selecting actions based on the max future reward we select an action at random. Acting randomly is important because it allows the agent to explore and discover new states that otherwise may not be selected during the exploitation process. You can balance exploration/exploitation using epsilon (ε) and setting the value of how often you want to explore vs exploit. Here's some rough code that will depend on how the state and action space are setup.

Learning Rate: lr or learning rate, often referred to as *alpha* or α, can simply be defined as how much you accept the new value vs the old value. Above we are taking the difference between new and old and then multiplying that value by the learning rate. This value then gets added to our previous q-value which essentially moves it in the direction of our latest update.

Gamma: gamma or γ is a discount factor. It's used to balance immediate and future reward. From our update rule above you can see that we apply the discount to the future reward. Typically, this value can range anywhere from 0.8 to 0.99.

Reward: reward is the value received after completing a certain action at a given state. A reward can happen at any given time step or only at the terminal time step.

Max: np.max () uses the numpy library and is taking the maximum of the future reward and applying it to the reward for the current state. What this does is impact the current action by the possible future reward. This is the beauty of q-learning. We're allocating future reward to current actions to help the agent select the highest return action at any given state.

SARSA: SARSA algorithm is a slight variation of the popular Q-Learning algorithm. For a learning agent in any Reinforcement Learning algorithm, it's policy can be of two types: -

1. **On Policy:** In this, the learning agent learns the value function according to the current action derived from the policy currently being used.
2. **Off Policy:** In this, the learning agent learns the value function according to the action derived from another policy.

Q-Learning technique is an **Off Policy** technique and uses the greedy approach to learn the Q-value. SARSA technique, on the other hand, is an **On Policy** and uses the action performed by the current policy to learn the Q-value.

This difference is visible in the difference of the update statements for each technique: -

1. **Q-Learning**

$$\pi^*(s) = \arg\max_\pi [r(s,a) + \gamma V^*(\delta(s,a))]$$

2. **SARSA**

$$Q(s,a)=Q(s,a)+\alpha[r+\gamma Q(s',a')-Q(s,a))]$$

Here, the update equation for SARSA depends on the current state, current action, reward obtained, next state and next action. This observation led to the naming of the learning technique as SARSA stands for **State Action Reward State Action** which symbolizes the tuple (s, a, r, s', a').

5. REVIEW OF CLOUD AUTO SCALING BASED ON RL TECHNIQUES

A well-known algorithm based on Dynamic Programming. On the other hand, there are the so-called Model-free techniques, which allow an agent to obtain a proper policy in online mode without requiring a perfect model of the environment. In other words, the policy is learned and improved over time in a process of continuous interaction with the environment. The surveyed papers found in the Model-free category use Q-learning and SARSA

RL-based strategy to schedule migrations of VMs, considering the current use of network resources in a Cloud. The idea is to learn to determine the most appropriate time to migrate a group of VMs from an overloaded physical machine to an underloaded one. The proposal aims to reduce the saturation of network resources during rush hours, as well as to reduce the migration time of the involved VMs.

Reductions of 59%, 40%, and 47% in the number of failed tasks, the total execution time, and the task execution time, respectively, were observed. The approach also reduces the use of CPU and memory by 22% and 20%, respectively.

A decentralized RL-based technique for responding to volatile and complex arrival of tasks through a set of simple states and actions. The technique is implemented within a distributed architecture that cannot only scale up quickly to meet rising demand but also scale down by shutting down excess servers to save costs. The states consist of two types of attributes: system state and application state. System state reflects the level of utilization of resources of a server such as CPU, and the application state represents the performance of each application hosted on the server in terms of metrics such as its response time.

6. LITERATURE SURVEY

In [1], proposed technique is RoR (Reinforcement on Reinforcement), in which two intelligent agents take place. Outer agent is responsible to hire the resources by applying its own procedures. Generated output by outer agent is passed to inner agent and it is responsible to make optimize configurations which will achieve high QoS (Quality of services). Future extended work is worried about the tuning cost of algorithm selection. Because of whatever the input given to inner agent (prepared by outer agent); it must find the policy algorithm to achieve better deployment.

In [2], proposed technique is multi agent controllers which will be able to find the optimal path through switch migration. It will be helpful to find the correct path though network optimization. One more agent will be responsible to find correct estimation of resources and will be hired through network. Whatever the resource it has now, based on it, last agent will configure server's parameter auto configuration (increase, keep as it is, decrease). Authors found still there is a great scope of optimization in terms of NP Completeness in the context of resource sharing.

In [3], proposed technique is MULTISCALER, mainly composed of three different levels working closely with each other to achieve an optimal resource allocation. One agent is composition of resource calculation, hired resources and optimize configuration. In virtualization there is a fast movement are going on for resource hungriness. For the same authors found there is a problem in the context of contention among allocated VM's.

In [4], proposed technique is containerization. Authors have compared Containers with VMs to host and scale a multi-tier web application. Their experimental evaluation shows a significant reduction of 46.48% to 70.23% in a total number of rejected requests for using Containers to host

and auto-scale a multi-tier web application. Future work is extended by inviting containerized application in virtual environment. Once you allowed these applications in the context of virtual environment, contention problems will be arising and because of applications overload resource sharing will be disturbed and not able to get optimized configurations for different servers.

In [5], proposed technique is reinforcement learning. An agent is there on web server and continuously watching the incoming requests. Once it finds more no of requests then it must identify one mechanism which will be helpful to achieve necessary resources. Among the availability of the latest resources, it is also responsible to auto configure services based on new allocated resource. Future work is, because of RL agent's random exploration based on policy, throughput and latency issues will be there.

In [6], proposed technique is reinforcement learning for utilized to recommend the optimal allocation policy. Evaluation results show that the proposed mechanism allocates the cloud infrastructure resources with Power Usage Effectiveness (PUE) value in an efficient range from 1.79 to 1.96 having reasonable resource utilization (above 50%) for the cloud data center. Still there is a scope for future extended work is optimize the policy for resource scheduling. In the era of web traffic still there is gap to find and execute the policy which will give the expected resource and based on it, apply optimal configurations for servers on the fly.

In [7], proposed technique is reinforcement learning. It's focus on to achieve good latency with respect to increasing no. of user's requests. Authors discussed both approach (vertical scaling and horizontal scaling) for resource sharing and optimize configurations. They have discussed SARSA (State – Action – Reward – State – Action) mechanism for the same and compare with Q – learning mechanism. There is a still scope to decrease initial learning time for SARSA algorithm as future scope.

In [8], proposed technique is reinforcement learning with respect to decentralization mechanism. Authors have found this schema is quite good in terms of failure for getting resource from another context. Auto scaling co-ordination and cost minimization are the future scope.

In [9], proposed technique is reinforcement learning. Authors discussed about to configure virtual machines settings as per the demand (user requests). They have tested mechanism on Microsoft's Azure platform. They have found RL approach is quite valuable when to control settings of virtual machines with respect to user's throughput. They conclude that RL approach is valuable while auto configure VM's parameters as well as azure's parameters settings. Future scope is extended while no. of virtual machine is increased and control the deal with hypervisor.

In [10], proposed technique is MAPE loop, which is monitor, analyze, plan, and execute policy. They applied Q – learning (Off policy) to hired resources form another context. Intelligent agent is also responsible for applying optimized control settings for different server's parameters. They have also discussed how Q – learning is fast as compared to other policies to take a decision for resource management and applied desired configuration settings. In future work they have defined Integration of the proposed approach with admission control strategies and extension of planning phase of the control MAPE loop.

In [11], proposed techniques are URI Space Partitioning and URI Distributions. They put more focused on application access patterns and distributions to characterize the workload. As per user's demand on specific application context, they have found this mechanism is quite useful to survive as a best response time. Applications accessed with respect to no. of user's request highly depends on its own configuration settings. Future work is to maintain the better characterized the workload and accordingly applied parameters size changing on the fly

In [12], proposed technique is fuzzy Q – learning to control resources. Authors have defined RL based controller to identify expected resource management. They have discussed a RL controller is also capable to predict future outcomes based on fuzzy techniques. And it would be very helpful manage resources at early stage to reduce the user's latency. Future work is Comparison with a variety of model-based techniques, integrate both vertical and horizontal scaling, Addressing interference aware auto scaling.

In [13], proposed technique is MAPE phases, which controls monitoring, analyzing, planning and execution. Their whole approach among the virtualization allocation. RL agent controls resources through virtual environment and responsible for new virtual machine allocation or place VM in quarantine. All VM will be controlled by RL agent in the context of resource management. Future work may be to keep all working in VM which are in quarantine without affection by any new resources.

In [14], proposed technique is Microbenchmark, Memory Intensive benchmarks, Local Memory Mutualization, Global Memory Mutualization. Authors defined memory sharing concepts with cloud environment. They have group memories as per their load. Labialized memories have its own meanings and deserve for specific tasks, and they will not be available for any future load. Future work will lead to predictive maintenance of memory and remote memory performance degradation.

In [15], proposed technique is Dynamic huge page-based memory balancing system (HPMBS). They discussed about the management of huge pages in memory with the execution of application. Future scope extended by, to create memory pool to keep and manage huge pages (in terms of memory).

In [16], proposed technique is Virtualization and Virtual machine consolidation overhead. They have defined to create a needed resource collection and give them fly at the top, which will be helpful to manage resources dynamically. Future scope remains to handle dynamic environment taking an account in virtualization overhead.

In [17], proposed technique is soft resource allocation strategies (novel model) on n-tier system performance. By using this model, you can have control over the n-tier architecture across the no. of request from existing users in the web context. Future scope is still mystery to handle the existing model in high scalability.

In [18], proposed technique is Opportunistic resource sharing, Dynamic network resource demand predicting algorithm with GSO-INC-RBF based on GSO and RBF incremental design. Authors have defined how to use the best resources as per given time by using Radial Based Function technique in the technology of edge computing. Research can be extended to find optimum stability in RBF

In [19], proposed technique is Direct paging—Para-virtualized MMU virtualization and Hypervisor memory allocation model. Authors forced on to create a MMU (Memory Manager Unit) to survive good memory allocations to running web applications. Future scope can be extending by darkness, duality, and dynamism and diversity variation indifferent environment in terms of memory management.

In [20], proposed technique is gScale, a scalable and practical open-source GPU virtualization. They discussed about how to set gScale parameters which can accelerate FPGA with a Configurable IOMMU. Security concerns for GPUs and how to accelerate gScale parameters would be the future work.

In [21], proposed technique is the state-of-the-art optimization techniques according to memory migration. Authors discussed about the different migration techniques along with memory and research can be extended to optimize the migration process

In [22], proposed technique is Risk Modeling and threat evaluation as per the security concerns. They explore the model of Risk and evaluation of threat with different techniques of attacking. Future scope can be redefined with to provide security solutions at different stacks of virtualization.

In [23], proposed technique is three-layered model architecture for resource sharing in 5G networks. They have defined the architecture which provides good response time in different wireless network. Future scope can be extended with authorize access at different level.

In [24], proposed technique is C-RAN resources are allocated to competing mobile operators through auction mechanisms. Authors described Centralized Radio Accessed Network mechanism to mitigate resource demand and deliver optimized resources through virtual environment. Future work may be extended in improvements design of C-RAN with delay-tolerant.

In [25], proposed technique is - Predictable shared cache management framework for multi-core real-time virtualization. They have defined Hypervisor-level techniques, vLLC and vColoring mechanism to optimize the shared cached.

In [26], proposed technique is CFEngine which provides automated configuration management of large networked systems. Authors talked about this technique could be applied to different devices without worry of different environments. Future scope in this era is to provide elastic resource management with on demand scalability.

7. CONCLUSION

Virtualization is the key now in days to mitigate the resources with their high cost. I found still there is a great scope to optimize the resources and deploy configurations for server's parameter(s) with their own scalability in virtualization. This study described available solutions and techniques to optimize the resources in virtual environment but found there is a scope to redefine web server and application server's configurations parameters as per demand of resources are changed.

REFERENCES

[1] Dong, Linsen, et al. "Intelligent Trainer for Dyna-Style Model-Based Deep Reinforcement Learning." *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[2] Sun, Penghao, et al. "MARVEL: Enabling controller load balancing in software-defined networks with multi-agent reinforcement learning." *Computer Networks* 177 (2020): 107230.

[3] Al-Dulaimy, Auday, et al. "MULTISCALER: A Multi-Loop Auto-Scaling Approach for Cloud-Based Applications." *IEEE Transactions on Cloud Computing* (2020).

[4] Abdullah, Muhammad, Waheed Iqbal, and Faisal Bukhari. "Containers vs virtual machines for auto-scaling multi-tier applications under dynamically increasing workloads." *International Conference on Intelligent Technologies and Applications*. Springer, Singapore, 2018.
[5] Jin, Yue, et al. "Resource management of cloud-enabled systems using model-free reinforcement learning." *Annals of Telecommunications* 74.9 (2019): 625-636.
[6] Thein, Thandar, et al. "Reinforcement learning based methodology for energy-efficient resource allocation in cloud data centers." *Journal of King Saud University-Computer and Information Sciences* 32.10 (2020): 1127-1139.
[7] Benifa, JV Bibal, and D. Dejey. "Rlpas: Reinforcement learning-based proactive auto-scaler for resource provisioning in cloud environment." *Mobile Networks and Applications* 24.4 (2019): 1348-1363.
[8] Nouri, Seyed Mohammad Reza, et al. "Autonomic decentralized elasticity based on a reinforcement learning controller for cloud applications." *Future Generation Computer Systems* 94 (2019): 765-780.
[9] Zykov, Sergey V., et al. "Applicative-based automatic configuration management for virtual machines." *Procedia Computer Science* 126 (2018): 1771-1778.
[10] Ghobaei-Arani, Mostafa, Sam Jabbehdari, and Mohammad Ali Pourmina. "An autonomic resource provisioning approach for service-based cloud applications: A hybrid approach." *Future Generation Computer Systems* 78 (2018): 191-210.
[11] Iqbal, Waheed, Abdelkarim Erradi, and Arif Mahmood. "Dynamic workload patterns prediction for proactive auto-scaling of web applications." *Journal of Network and Computer Applications* 124 (2018): 94-107.
[12] Ibidunmoye, Olumuyiwa, et al. "Adaptive service performance control using cooperative fuzzy reinforcement learning in virtualized environments." *Proceedings of the10th International Conference on Utility and Cloud Computing*. 2017.
[13] Aslanpour, Mohammad Sadegh, Mostafa Ghobaei-Arani, and Adel Nadjaran Toosi. "Auto-scaling web applications in clouds: A cost-aware approach." *Journal of Network and Computer Applications* 95 (2017): 26-41.
[14] Aslanpour, Mohammad Sadegh, Mostafa Ghobaei-Arani, and Adel Nadjaran Toosi. "Auto-scaling web applications in clouds: A cost-aware approach." *Journal of Network and Computer Applications* 95 (2017): 26-41.
[15] Sha, Sai, et al. "Huge page friendly virtualized memory management." *Journal of Computer Science and Technology* 35 (2020): 433-452.
[16] Bermejo, Belen, and Carlos Juiz. "Virtual machine consolidation: a systematic review of its overhead influencing factors." *The Journal of Supercomputing* 76.1 (2020): 324-361.
[17] Wang, Qingyang, et al. "Optimizing N-Tier Application Scalability in the Cloud: A Study of Soft Resource Allocation." *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)* 4.2 (2019): 1-27.
[18] Xiao, Xiancui, Xiangwei Zheng, and Tian Jie. "Dynamic resource allocation algorithm of virtual networks in edge computing networks." *Personal and Ubiquitous Computing* 25.3 (2021): 571-586.
[19] Mishra, Debadatta, and Purushottam Kulkarni. "A survey of memory management techniques in virtualized systems." *Computer Science Review* 29 (2018): 56-73.
[20] Vogel, Pirmin, Andrea Marongiu, and Luca Benini. "Exploring shared virtual memory for FPGA accelerators with a configurable IOMMU." *IEEE Transactions on Computers* 68.4 (2018): 510-525.
[21] Noshy, Mostafa, Abdelhameed Ibrahim, and Hesham Arafat Ali. "Optimization of live virtual machine migration in cloud computing: A survey and future directions." *Journal of Network and Computer Applications* 110 (2018): 1-10.
[22] Asvija, B., Rajagopal Eswari, and M. B. Bijoy. "Security in hardware assisted virtualization for cloud computing—State of the art issues and challenges." *Computer Networks* 151 (2019): 68-92.
[23] Kliks, A., Musznicki, B., Kowalik, K., & Kryszkiewicz, P. (2018). Perspectives for resource sharing in 5G networks. *Telecommunication Systems*, *68*(4), 605-619.
[24] Gu, Sijia, et al. "Virtualized resource sharing in cloud radio access networks through truthful mechanisms." *IEEE Transactions on Communications* 65.3 (2016): 1105-1118.
[25] Kim, Hyoseung, and Ragunathan Rajkumar. "Predictable shared cache management for multi-core real-time virtualization." *ACM Transactions on Embedded Computing Systems (TECS)* 17.1 (2017): 1-27.
[26] Armstrong, Django, et al. "Contextualization: dynamic configuration of virtual machines." *Journal of Cloud Computing* 4.1 (2015): 1-15.