International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 9, No.6 (Nov-2020)

http://dx.doi.org/10.12785/ijcds/0906021

Spot Hopping: Increasing Reliability and Reducing Cost

Ali Jassim Hasan[1] and Mustafa Hammad [1]

[1]College of IT, University of Bahrain

Received 13 Apr. 2020, Revised 1 Aug. 2020, Accepted 24 Aug. 2020, Published 1 Nov. 2020

Abstract: Cost reduction is one of the attractive features offered by the cloud. Spot leasing is one way to reduce the cost even more. Spot leasing is done by leasing the unused excess instances with low price. Spot instances are facing risks that minimize their availability and reliability. Risks including instances reclaiming and dynamic price changing. Minimizing the risks associated with the spot leasing is going to help in increasing the utilization of the spot instances, which in turn is going to attract more users. In this paper, a framework has been proposed to mitigate the instances reclaiming risk while reducing the leasing cost as possible. This is done by monitoring many markets and hopping between instances. The proposed framework has been evaluated through simulating using actual data for EC2 spot price history collected from Amazon web services. The proposed framework has scored 74% and 64% of cost reduction compared with on-demand and spot leasing schemes accordingly.

Keywords: Cloud, Spot Instances, Cost Reduction, Migration, Hopping.

1. INTRODUCTION

Cloud computing has become one of the major IT industries today. With its sophisticated paradigm, cloud computing allows us to have several benefits such as on-demand leasing, reduced the total cost of ownership, globalization and more. Such benefits attract the customers to use the cloud instead of using the traditional deployment models. Cloud computing was designed on the bases of leasing the required IT services from an external service provider instead of purchasing and maintaining the hole IT infrastructure. This business model helps the customers focusing on their business activities by dealing with IT as services without considering the physical implementation and maintaining issues. The IT services are offered by companies called a cloud service provider CSP. Cloud service provider company is responsible for hosting and maintaining the IT gears and hardware equipment while making them available to the customers whenever needed. The cloud service provider can be a data centre that hosts physical IT equipment, or IT can be as multiple datacenters spread globally and connected to form availability zones and regions.

Hosting and maintain the physical resources need to have high management level. Enabling essential cloud characteristics such as multi-tendency, elasticity and on-demand leasing require to keep extra resources in the datacenters. Keeping such extra resources costs the cloud provider. Many cloud providers such as AWS allow renting these extra recourses with reduced price to cover their running expenses. Those resources have many different names depending on the cloud provider. Amazon calls these spot instances, while Google and Microsoft call theories as preemptable virtual machines. Spot leasing considered as the lowest leasing schemes compared with the reserved and on-demand leasing schemes. With the spot leasing, the customers sacrifice the service availability by trading it with lower leasing prices. Since such leasing scheme has a high number of customers, the prices of the resources keep changing dynamically according to the supply and demand fluctuations. Such leasing scheme became possible with certain conditions. The First condition allows the cloud provider to reclaim the resources at any moment to rent it to the other customers and maintain the promised on-demand and elasticity features. Such a condition creates a risk for the customers who plan to use the spot leasing to run an uninterrupted process. When the cloud provider needs to reclaim the leased resources, it will give a short notice period for the customers to back up their running processes before it proceeds with reclaiming by force. Another condition that the spot leasing has subjected to price changing based on the availability of the unused resources along with the number of leasing requests or what is called market supply and demand.

Although Spot leasing considered good for resources utilization and cost reduction for both the cloud provider and for the customers, it comes paired with the price changing and interruption issues. Creating a process that

E-mail: Ajasim@uob.edu.bh, mhammad@uob.edu.bh

can monitor the cloud environment and adapt accordingly will help in avoiding those issues. Having such a cloud monitoring process can be used to increase the spot resources reliability and therefor increases the spot resources utilization percentage. Moreover, it can reduce the risk of application interruptions and revocation. Furthermore, combining features such as low leasing cost from spot scheme with high availability feature from the on-demand scheme can help in attracting more customers toward using the cloud. Harvesting the features of both leasing schemes can be done by hopping between them. The hopping ability can be used to increase the leasing costs and to mitigate the reclaiming risks.

2. LITERATURE REVIEW

Cloud computing is considered a promising technology Increasing the Utilization of excess resources is considered as a challenge that needs to be solved. Zhang et al. [1] present a dynamic allocation mechanism for AWS spot resources based on choosing the best suitable hardware that matches the application requirement, which can help to reduce the total cost of leasing including the power combustion. Work in [2] presents a cost-aware provisioning system that works on reserving the best suitable instance that matches the application needs. The proposed system shows promising results. The results were including a total cost reduction by 24%, reducing in the transmission time between instance, and showing the opportunities where the cost can be traded to minimize the execution time. Menache et al. [3] present an algorithm that can be used to dynamically allocation proper resources for batch jobs. The algorithm is taking the price of the on-demand and spot market resources as input factors before deciding to go with which option. Xin et al [4]. Proposes a cloud scheduler that can reduce the total cost of resources leasing. Moreover, the proposed scheduler has successfully maintained a web server up and running with no interruption using the spot instance. Work in [5] presents another mechanism to monitor the history of spot market prices and help in choosing the cheapest.

There have been several attempts to migrate between different cloud resources. Works in [4], [6], [7] show the results of studying the effects of VM migration between multiple instances and proposed solutions to minimizes those effects. On the other hand, Work in [8] studies the instance migration based on minimizing exchanges traffic and increase network performance. In [9], a proposed solution to monitor the network traffic exchanged between the leased services was introduced. The proposed mechanize tries to minimize the network latency between the services. The proposed system works on migrating those services that need to communicate with each other into close locations. The proposed system has been simulated, and results show that inter could traffic has been reduced by 25% to 60%. Shastri and Irwin in [10], [11] present a prototype resource container that keeps monitoring the market price. that container can hop between instances based on the price changing. Moreover, the proposed prototype has been implemented. The implementation shows that the prototype can reduce the cost of provisioning. Another novel cloud federation system has been proposed in [12]. That system monitors the market price changing and when a new service is needed, the proposed system is going to select the cheapest option provided. Moreover, with any price-changing detected, the system dynamically rearranges the location of the services based on the lowest price available. The proposed system has been simulated and the results show that the system can reduce the provisioning price using multiple cloud providers. Lee and Son in [13] present DeepSpotCloud, a framework that was designed to use the Graphical processing unit (GPU) power for deep learning while trying to minimize the cost. The proposed framework was implemented in AWS spot instances. The framework monitors the market price for any changes, then tries to minimize the cost by hopping to the cheapest instances even if it were found in another region. Another framework called spotweb has been proposed by Ali-Eldin et al in [14]. The proposed framework was designed to run a sensitive web server on spot instances while trying to maintain the quality of service. Moreover, the work presents a transient algorithm that predicts spot cost and use the predicted results to determine the needed migrations that could offer high service quality with low latency.

Spot leasing cost can be affected by the location of the leased resources. The price of a certain instance might be different from one zone to another, and between a region to another. Ekwe and Barker in [15] conducted an analysis to show the effects of the location over the deployment cost. the study includes several spot instances types from every zone in all AWS regions. The work shows that resource location plays a major role in calculating the final leasing cost. furthermore, the work presents that the spot resources can be used to achieve low cost and to reduce the termination risk.

Comparing the proposed framework with others The prototype presented in [10] affected by any cost chancing and instance resource utilization, even if that change was for a short period. That is going to increase the number of hops taken by the virtual machine. If compared to our proposed framework, the migration is happening on an hourly basis. This is going to filter any spikes that might affect the leasing costs. if the cost changing has lasted for more than one hour, then the proposed framework is going to consider it and going to do the hopping if needed. Moreover, the proposed framework simulation was showing the effects of hopping to multiple regions which were not shown the other work. Another point was that the results from the work in [10] were from hopping between multiple zones in one region, while our simulation includes two regions to show the difference of cross-region hopping. The work proposed in [12] presents a federation framework. that framework was designed to serve a cluster of instances. one of the instances represents

the main node that has the framework implemented on, while the others were being controlled by the main node. The main node was following the market changings and based on that it controls the other nodes. Comparing that work to the proposed framework in this paper, our proposed framework was designed to be running on a single instance which make it independent, that gives the instance the ability to perform the migration based on its migration policy, without waiting for the controlling commands for another instance. Having this feature (independent) can minimize the risk of having a single point of failure that might exist when using a single unit to control others. The work presented in [13] was focusing on utilizing the instances that were equipped with GPU, while for our framework the focus was on the CPU and RAM.

The proposed work in this paper has three variable parameters compare to the others. Those parameters can be used to adjust the framework decisions. The parameters including the hopping range, the instance types, the upper and lower utilization limits. The proposed work is considered as an extension over previously presented works in [16], [17].

3. BACKGROUND

Cloud computing has many definitions. according to NIST [18], Cloud computing can be defined as an on-demand pool of IT resources than can be accessed from anywhere using the internet, while having the minimum management efforts required. The could technology offers many IT resources, Such as computing servers, networking devices, storages, and applications.

Cloud computing technology has introduced a new paradigm that changes the way of dealing with IT. That paradigm came with five many characteristics including an on-demand acquiring. This characteristic allows the cloud user to provision and releases cloud resources based on his needs, whenever needed, with minimum human intervention required. Another characteristic is the ability to access the resources using the internet from anywhere using any internet client platform. A third characteristic is the pool of resources. With many different pools around the world, the user can acquire the resources from the pool that is located near to him or his targeted customers. Having a pool of resources with multitenancy feature allows the users to reuse the resources after being released by the current user. A fourth characteristic is the elasticity. Such characteristic helps in reducing the cost and management efforts that are needed to be done by the user. The elasticity allows to expand and shrink the resources automatically to match the application demands. The Last characteristic is the ability to monitor and measure cloud resources consumption, which plays a major role in the accounting and billing process. Combining those characteristics has created the cloud technology paradigm.

There are many advantages for using cloud computing. Advantages including cost-saving, quick implementation, high availability, Fault tolerance, on-demand self-service and many more. All those advantages are considered as strong motivators for the organizations and users toward using the cloud. That, in turn, helps to create new business models. Instead of purchasing the IT resources and pays for it as an upfront payment -which might require a big capital- changed into leasing the resources based on the needs of the organizations and pay only for what was used. According to Gartner in [19], The cloud market share in 2018 was around 182.4 billion dollars, and grows to by 17.5% to reach 214.3 billion dollars by 2019, and expected to reach 331 billion dollars by 2022. On the other hand, using the cloud comes associated with some disadvantages, such as the requiring of having high-speed internet, security, privacy, requiring for skilled cloud users, vendor lock-in and more.

The cloud providers offer many services. Those services come in different models such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS) and many more. Each service model defines the resources management level and actions that must be done by both the cloud service provider and the cloud user. In [IaaS], the cloud provider responsible only for delivering the underlying infrastructure, while the user is responsible for configuring the operating system, the environment and other service applications. In [PaaS] service model, the cloud provider handles more responsibility, which is managing the underlying infrastructure, the virtual machine, and the operating system. The user receives the ready virtual machine that got the OS installed, along with any required access credentials. Then the customer accesses the operating system and build-up the environment that matches his service needs. Finally, using [SaaS] service model, most of the managing operation will be under the cloud service provider responsibility, leaving the customer with the minimum configurations required for the service layer.

The cloud business plan encourages toward leasing services based on customer needs. On the other hand, traditional IT infrastructure users must invest a large capital at an early phase before proceeding in their production plan. This includes purchasing extra resources to support business growth. If the customer selects to go with the cloud leasing scheme, then the customer will pay only for what has been used, and when there is a demand for extra resources, it can be leased instantly. Cloud service providers offer different schemes for leasing cloud resources. Each cloud provider had a different naming for the offered schemes. AWS had the reserved, on-demand and spot leasing schemes. The following subsections explain each leasing scheme in more details.

A. Reserved leasing scheme

The reserved leasing scheme was designed for those customers who plan to lease the resources over a long period, or for applications that need to be running for many continuous hours. That is because reserve leasing requires to sign a contract that could last for one or three years. AWS attracts its customers to lease based on the

reserve scheme by giving them a discount rate on the leasing price that could reach up to 75% [20]. It is not recommended to lease using the reserved scheme for the customers who were having applications with variable resources requirements. in such case, AWS recommends using the on-demand or spot leasing schemes. Reserved leasing scheme guarantees that the user receives his reserved resources whenever needed. AWS gives priority for the reserved scheme users over other schemes users. AWS also prefers on-demand users over spot users.

B. On-demand leasing scheme

On-demand leasing plan was created to be suitable for the applications that need to be running for unpredictable, uninterrupted periods. The on-demand scheme is flexible. Based on the user needs, the resources can be leased when needed, and release them after finishing. No commitment or contract needed to be signed by the user. Furthermore, no discount offered for the on-demand scheme, which makes it the most expensive compared with the others.

C. Spot leasing scheme

Spot leasing was designed to offer excess and unused resources for leasing with reduced cost. The main benefit of the spot leasing was in its cheap leasing costs. Spot leasing cost has a high discount rate that could reach up to 90% over the on-demand cost for the same resources. This has been done to cover the operational expenses of those resources Instead of losing by keeping them running without any benefit. The spot leasing scheme could be used for the applications that are only feasible at low computing price and could handle being interrupted. The spot leasing costs were subjected to changing due to market supply and demand. Higher demand increases the spot leasing rate. The following subsections explain more about spot leasing associated risks.

D. Spot leasing associated risks

The cloud service providers keep extra resources in their data centers intentionally. Such resources were required to offer some characteristics such as the elasticity, scalability, and on-demand resources. When all the resources in the CSP datacenter got occupied, and a new request got received from an on-demand user, the CSP reclaims the resources from the spot users and reassign those resources to the on-demand users, as the resources were meant to serve them from the first place. The spot scheme was only introduced to increase the utilization of the cloud providers excess resources and to minimize the operational costs of the CSP. That is why the on-demand users are always having higher priority over the spot users when it came to competing over the same resources. Based on that, the spot instances were subjected to get interrupted, reclaim and reassign at any moment.

1) The risk of interruption and reclaiming

The CSP recommends using spot leasing scheme for the processes that can handle being interrupted. The spot users were given three options represents process interruption behaviors. The three interruption behaviors were hibernating, stopping, or terminating the spot instance [21]. The hibernating option will suspend the process activities and will save everything on an image, including the data located within the RAM. The second interruption behavior was "stopping", where the interrupted instance will be forced to shut down and the data stored in the disk (elastic block store) will be reserved, the main difference between the stop and the hibernate was that the data located in the RAM will be erased in the spot interruption behavior, while it won't be erased in the hibernate. Comparing the hibernate and stop interruption behavior options with the terminate option, both hibernate and stop options can be started again once the required resources got available, while the terminate option won't allow starting any process again [22].

2) Dynamic spot price changing

The leasing prices are dynamically changing based on supply and demand. If there is a high demand for a certain instance type, the spot price for that instance type will be increased. Figure 1 shows the spot leasing rate fluctuation compared with the stability of the on-demand rate. Both the spot and on-demand leasing rates were belonging to R4.14xlarge instance type. The figure shows how the spot leasing rate was changing in the period between 1st of April and 31st of May 2018. R4.16xlarge was in zone A inside region CA-Central-1 region. The data were taken from a dataset that was published in 2018 [23]. The figure reflected how the spot instance rate started at 3$/hour, then got increased gradually until it reached to the on-demand leasing rate around 4.6$/hour on the 8th of April. On 12 of April, the spot leasing rate started to reduce gradually until it reached 1$/hour on the 26th of April. This changing happed due to the continuous changing on the demand for R4.16xlarge instance type. Such kind of rate changings affects the spot users, who were using the spot scheme to minimize their costs.

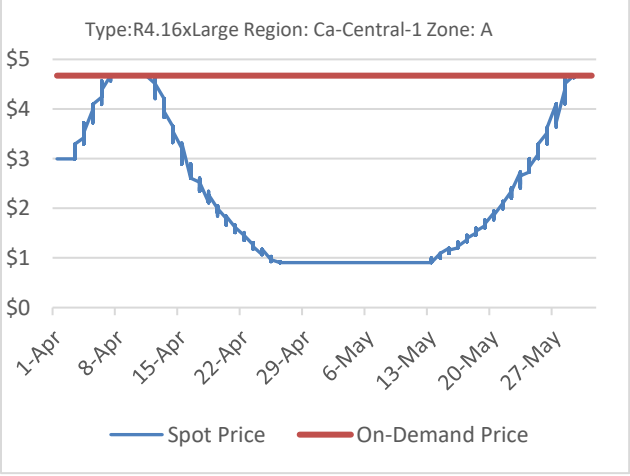

Figure 1. Spot leasing rate fluctuations

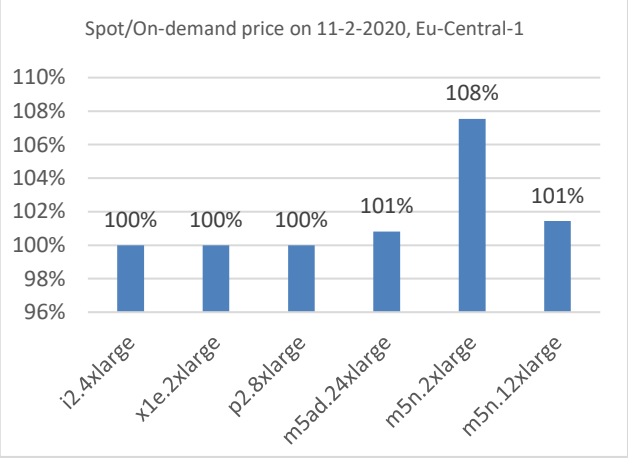

Figure 2. Spot rate increasing over on-demand rates

3) Increasing over the on-demand rate

The spot leasing rate can continue climbing up to become higher than the reserved and on-demand rates. Price rate changing can work reversibly against the users, who were using the spot scheme to reduce the leasing costs. Figure 2 shows the spot price over the on-demand price for some instance types on the 11th of February 2020. The data was collected from availability zone A from Eu-central region. The instance types were using Suse Linux operating system. The figure shows some cases were the spot prices became equivalent to the on-demand prices, same as what happened with i2.4xlarge, x1e.2xlarge and p2.8xlarge instance types. The figure also shows three more cases where the spot prices became higher than the on-demand prices, just like what happened with m5ad.24xlarge, m5n.12xlarge and m5n.2xlarge.

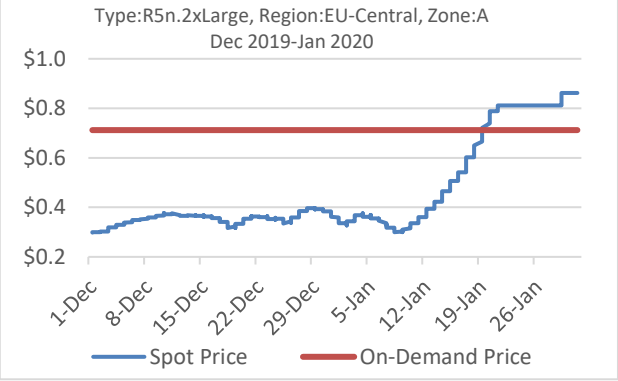

Figure 3. Spot leasing rate became over on-demand leasing rate

Furthermore, the spot leasing rate could get increased to become more than the on-demand leasing rate. Figure 3 shows the spot and on-demand leasing rates for R5n.2xlarge on the period between 1st of December 2019 and the 31st of January 2020. the data was taken from dataset no. 2, from zone A from Eu-central-1 region. The figure presents a case where the spot leasing rate became higher than the on-demand rate. As shown in the figure, the spot leasing rate for R5n.2xlarge started with 30 cents/hour on the 1st of December 2019 and stayed fluctuating until the 9th of January. On the 8th of January, the spot rate started climbing up until it became 70 cents/hour, which was equivalent to the on-demand rate. The spot rate counited climbing until it reached 85 cents/ hour. At such case, using the spot leasing scheme can be considered as a bad choice, that because it lacks for the required availability and has a higher price compared with the on-demand that have both features.

4. PROPOSED FRAMEWORK

A framework has been proposed to mitigate the associated risk of spot leasing. The framework was designed to utilized spot leasing scheme to reduce the total leasing cost, and at the same time to minimize the spot leasing associated risks. The framework was planned to be installed on a single instance trying to provide multiple features including cloud monitoring, dynamic selection and self-migrating.

The framework monitors instance resources (CPU and RAM) utilization rates to determine the optimum configuration needed to run the hosted application. Simultaneously, the framework keeps monitoring the cloud markets within the hopping range and migrates (hop) to the cheapest available instance that has the required application resources. In case any spot leasing risks have been faced (such as increasing the leasing rate of the currently leased instance or receiving a reclaiming notification, or even a cheaper instance has been found) then the framework will migrate to the cheaper instance. The hopping ability allows the framework to migrate between different instance types across different zones and regions. Furthermore, the framework monitors the on-demand leasing rates as (as the last resort) to be used if the spot leasing became more expensive or not available due to the high demand.

The hopping operation is basically to migrate the running application from the currently used instance to the new instance. To achieve such migrating operation, a life container migration technology can be used. Moving a container is considered faster than charring the whole virtual machine which contains an OS that might take some time due to its large size. The hopping operation is used to mitigate the risks associated with spot leasing scheme. moreover, the hopping operation is used to reduce the total leasing cost. The following section describes more about the hopping operations types.

A. Hopping Types

There are different types of hopping operations that can be achieved by the framework. the hopping types including cross-schemes hopping, cross-zones hopping, cross-regions hopping, cross-instance-types and resource minimization hopping types. The following subsection describes each hopping type:

1) Cross-schemes hopping

Cross-scheme hopping allows the framework to hop across different leasing schemes, such as hopping from an instance leased were leased based on spot scheme to an identical instance but using different leasing scheme (cross-schemes hopping). This type of hopping is considered important to the framework especially when dealing with the spot leasing scheme, where such hopping type allows the frame to mitigate the spot leasing risks generated by CSP reclaiming and dynamic price increasing.

2) Cross-zones hopping

Cross-zones hopping is a hopping type that is used to achieve short distance hopping to change the zone but to stay within the same region. Such type of hopping can be used to minimize the leasing cost while preserving the quality of service for end-user experience. Since all the zones were located inside the same region and they are connected using high-speed links, then changing the zone will not affect the network bandwidth or the network latency. This will also be going to minimize the time needed to achieve the hopping operation compared with the long-distance hopping operation.

3) Cross-regions hopping

Cross-regions hopping is used to achieve long-distance hopping operations, where the framework can hop from one region to another. This hopping type can be used to obtain a more reduced cost compared to the short-distance (cross-zones). on the other hand, this hopping type might affect the end uses service quality. If the offered service by the VM hosted application was considered sensitive and requires high network bandwidth or low latency.

4) Instance-types hopping

This type of hopping can be used to allow the framework to hop (vertical scaling) to different instance type that might have the same (or more) resources as the current instance, even if the new instance was from a different family. As long as the new instance is having the ability to maintain the service up and running, then it can be considered as a hopping target. Such hopping type can be used when a certain type of instances is facing high demand, while the other types are not utilized. In that case, the old instance price will be increased due to the demand increasing, while the different instance types are going to become cheaper even when it has more resources compared with the old instance.

5) Resource minimization hopping

Resource minimization hopping allows the framework to hop (vertical scaling) to different instance types based on the current instance resources utilization percentages. The difference between the resource minimization hopping and the cross-instance-types is that in the resource minimization hopping the framework can hop to instances that have less resources compared with the current. the new instance is going to be selected based on the current resource utilization percentage. That type of hopping enables the framework to hop smaller instance to reduce the cost if the running application is not using the resources available in the current instance. In contrast, if the application needs to have more resources, then this type of hopping will allow the framework to find a larger instance that could fulfil the application needs to maintain the performance level. The benefit of this leasing type that it can dynamically hop to different instances to match the application needs and ant the same time it helps in reducing the leasing cost. the main drawback of this hopping type is that the framework might need to hop more frequently compared with other hopping types.

B. Framework hopping controllers

The proposed framework was designed to give server administrators the ability to control framework decisions to fits the hosted application. Several input controllers were used to doing that, including the hopping range, instance types, upper and lower resources utilization percentages. The following subsection describes each controller.

1) Hopping range

Hopping range is an input variable that is used to allows the server administrator to control the hopping distance that the framework can cover. This input variable can affect the framework searching scope. The administrator can decide the suitable markets to run he hosted application and the limit the framework hopping operations to be within that defined markets. The hopping range can include zones and regions. more hopping range can increase the chance of getting cheaper costs, while shorter hopping range can use to deliver a better users experience.

Adjusting the hopping range have many trade-offs. As an example, Extending the hopping range can increase the number of markets in the searching scope. Therefore, it increases the chance of getting a cheaper instance, but it might also affect the user experience (lower network bandwidth and higher latency). Such a case might happen when the cheapest instance found was located far from the end-users.

On the other hand, shortening the hopping range can be used to define fewer markets, and focuses on searching for the cheapest option in the nearby markets only. Therefore, preserving the better quality of services for user experience (higher internet speed and lower latency), but it would reduce the chances of having a cheaper cost compared with the longer hopping range.

2) Instance types

This input variable can be used by the administrator to select the types of instances which contain the required resources that suites the hosted application. The framework hopping operations will be limited to those instances types that have been selected. In some cases, the hosted application requires to have a certain amount of

resources to operate properly. Using the instance types input variable gives the administrator the ability to select what are the instance types that contain the required resources which can fulfil the application needs. The administrator can add many instances types including the types that have more resources than what is needed, knowing that the framework will search for the cheapest instance among the listed. If the server administrator defines only one instance type, then the framework will be only searching to that type, while adding many types will allow the framework to search for more options which will increase the chances of getting a cheaper instance.

3) Upper and lower resource utilization limits

The upper and lower resource utilization limits allow the sever administrate to define the threshold that can be used to trigger the framework to hop (scale vertically). if the upper threshold limit has been reached for a predefined period (defined by the administrator), then the framework will be triggered to hops to a larger instance with more recourses. In contrast, reaching the lower limits for and staying for some time will trigger the framework to hop to a smaller instance with less resources. those input limits are associated with resource-minimization hopping type, which can be used to minimize the cost by hopping to the optimum instance that got the required resources.

Figure 4 shows how the framework chooses between the different leasing schemes. Before taking any decision, the framework is going to look for the cheapest available instance, whether it was a spot or an on-demand instance. In some cases, like when the of spot instance becomes not available, the framework might issue a hop request to switch from spot to on-demand and vice versa. Another case of using an on-demand instance is that when the spot price became more than the on-demand price, then the on-demand will be considered as the cheapest option to reduce the total cost. While using the on-demand instance, the selector will keep looking for any alternative instance for leasing. If any suitable instance has been found, then the selector is going to migrate from the on-demand to the spot instance. This is done dynamically according to price changing, resources utilization and the availability of the instances.

C. Migration/hopping triggers

The instance selector monitors many triggers. Triggers including the platform reclaim notification. This notification is sent if the cloud platform is going to reclaim or revocation the instance. After receiving such notification, the instance is given a limited time slot to act before being interrupted. In this case, the instance must be migrated immediately before the allowed interruption time ended.

Another trigger that needs to be considered is the resources utilization of the current instance. If the running service on the instance is utilizing a high percentage of the currently available resources, then the service must be migrated to another instance that has more resources compared with the currently hosted instance. Doing this will maintain a certain level of service performance and will prevent the instance from reaching its maximum capacity. Vice versa, if the resources utilization percentage is low, then a new instance with lower specifications can be used to run the service while reducing the leasing costs. Hopping dynamically based on the utilization percentage is providing better performance while maintaining the price as low as possible.

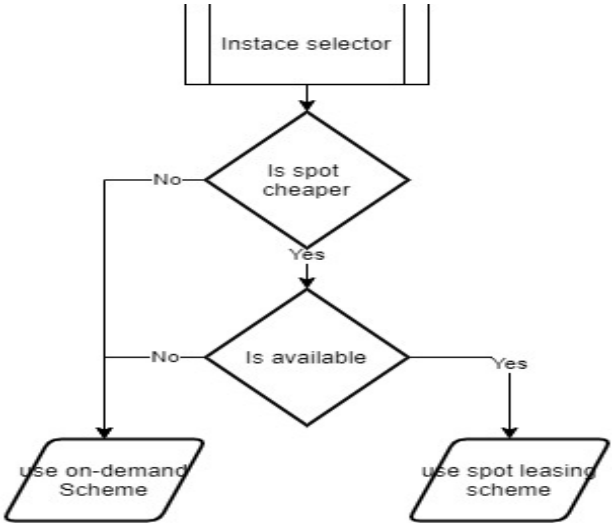

Figure 4. Hopping between Spot and on-demand leasing schemes

One more trigger that is considered important is the current instance price (Spot, on-Demand). Since the Spot price is dynamically changing due to the supply and demand, there is a chance that it would become high, more than the on-Demand price. If the currently leased instance price got increased, then the framework needs to find an alternative instance that is offering the needed recourses at a cheaper price.

Table 1 shows the triggers that can start the migration process with the actions that need to be taken by the proposed framework. As an example, when a reclaim notification generated by the cloud platform got received by the proposed framework, then the framework is going to do an immediate migration to the cheapest suitable instance from the updated price list. Such action is needed to be done because of the short period that is given by the cloud platform. If that period is finished and the instance is still running, the cloud platform is going to reclaim the instance by force. On the other hand, if the instance is starving for more resources, then the selector process is going to issue a hop request to migrate the instance to another instance that contains more recourses that suits the needs. The last trigger is the leasing cost. Depending on the cost changing, the selector process is going to decide ether to stay or to migrate. If migrate is option is chosen, then the destination target is needed as well.

TABLE I. HOPPING OPERATION TRIGGERS FOR THE PROPOSED FRAMEWORK

Trigger		Action	Details
Cloud Platform Reclaim notification		Immediate Migration to the cheapest available instance (spot, on-demand). And start the time counter	Reclaim will be done within a fixed time slot determined by the cloud provider. If spot instance is available then it will be selected, otherwise, an on-demand will be selected.
Resource utilization percentage	CPU Or RAM utilization>= upper threshold limit for the defined period [24]	Find an instance that has resources more than the current instance, then start the migration.	If the current resource percentage has reached the upper limit and stayed for a period, then the application must be migrated to another instance with higher resources. -allows providing better performance
	CPU and RAM utilization<= lower threshold limit for the defined period [24]	Find an instance that has resources lower than the current instance, then start the migration.	If the current resource percentage has reached a lower threshold limit and stayed for a period, then the service can be migrated to another cheaper instance with lower resources to reduce the cost.
Instance leasing price	If a cheaper instance with the required resources has been found	if a cheaper option has found. Options obtained from both including spot and on-demand leasing schemes.	Find the cheapest instance that has the needed resources. Search in both leasing schemes. Migrate to the cheapest instance to minimize the total leasing cost

D. Proposed Framework structure

Figure 5 presents the proposed framework architecture. The framework monitors three variables simultaneously which might trigger the hopping process. Firstly, the framework monitors the currently leased instance resources. if the resources utilization levels became high and stayed for some time, then the framework will migrate the application container to another instance type with more resources to sustain the application performance. Verse versa, if the resource utilization levels were low, then the framework is going to migrate the container to another instance that has lower resources to minimize the total cost. Secondly, the framework monitors the cloud market and prepare a list of all available instances that might be applicable as hopping targets. if a cheaper price got detected, then the framework will consider the cheapest instance for the next hop. Doing this allows the framework to maintain the cheapest price and to save time whenever an immediate hopping is required. Thirdly, the framework monitors the notification sent by the cloud platform to the currently running instance. If the instance has received a release/reclaim notification, then it will use the pre-prepared list to choose a new instance as a migration target.

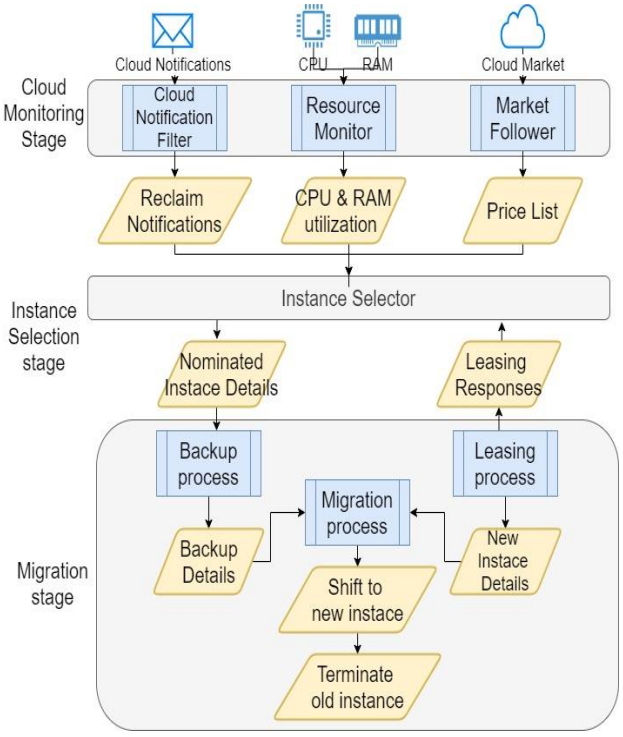

Figure 5. Proposed Framework Block Diagram

To find the target instance, the framework starts looking for candidates that can be leased with the minimum cost fees, while having the resources required to keep the service running. It starts by looking for a spot instance since it might be found cheaper. If the framework could not succeed to acquire any spot instance, then the framework will go to the second option which is leasing an on-demand instance. Concurrently with the searching operation, the framework starts backing-up the data from the running instance and migrate to the newly selected target.

The framework contains three stages including pre-selecting stage, instance selection stage and the post-selection stage. The following subsections describe the process in more details.

1) Preselection stage

The spot instances are exposed to be reclaimed by could platform at any moment. If the cloud provider requests to take over a certain spot instance, then a notification will be sent to that instance to back up the data and stop any running processes. The instance will be

given a time slot to complete its running tasks. Each CSP has a different reclaim period. In the case of Amazon AWS, the reclaim notification period is 120 second [25] [26]. After finishing the time slot, the cloud provider will interrupt the instance and will take over its resources. The cloud service provider allows the user to choose the suitable interruption behaviour from the three available options which are hibernating, stop and terminate. The pre-selection stage contains three processes, including the cloud notification filter, the resources monitor and the market follower. The following subsection describes each process in more details.

a) Cloud Notification Filter

The cloud notification filter process is responsible for continuously monitoring the notification sent by the CSP. The process looks for any release/reclaim request among the received notifications. If any reclaim notification has been received, Then the filter is going to generate a trigger to start the instance selection stage processes.

b) Resource Monitor

The second process is the Resource Monitor, which is used to monitor the current instance resources utilization percentages. Including the CPU and RAM utilization percentages compared to their maximum capacity. Monitoring the utilization percentages can be useful in triggering the instance to perform vertical hopping, which means to hop to another instance type that is either bigger or smaller than the current. Bigger instance type refers to an instance that has more resources, while smaller instance type refers to an instance that has fewer resources.

c) Market follower

The third Process in the pre-selection stage is the market follower. The function of this process it to monitor the price in several cloud markets that were located within the framework hopping range. The framework hopping range can be configured to control the searching scope.

The process is going the check the prices of several instance types that are found in its hopping range. Then, it is going to generate a list of candidate instances. at the same time, it is going to generate a trigger to the instance-selector stage informing to notify about a cheaper price has been detected. The Candidates list contains both spot and on-demand instances. That list is going to be used if any migration is required. The cloud monitor process keeps updating the candidates list if there is any cost change happened. Updating the list will reduce the time needed while looking for an alternative instance.

2) Instance selector stage

Instance selector process is responsible for choosing an instance that is going to be used as a migration target. Choosing the best instance is based on several factors, such as the cheapest price, leasing scheme, required resources and the location. The selector takes the outcome of the pre-selector stage as an input. at the begging, the selector waits for any trigger generated by pre-selection processes. If any trigger has been received, then the selector is going to start a countdown timer. Moreover, it is going to run a sequence of checking conditions that will end with nominating several instances as hopping targets. Those targets mainly contain spot instances and might contain on-demand instances as a backup. After that, the selector is going to send the best-nominated instance for the post-selection stage to place a leasing request and trying to acquire that instance.

The migration process is going to require a new instance to be leased before starting the migration. There are several factors to be considered before selecting a new hopping target. The following sections describe the instance selection factors.

a) The leasing cost

Such as the price of the currently leased instance compared with other instances from the market. The instance that needs to be selected preferred to have a cheaper price than the current instance. In case that the cheapest available spot instance price has become higher than the cheapest on-demand, then leasing based on-demand scheme is going to be considered as the best option.

b) Hopping Range

A third factor that needs to be considered is the hopping range, which determines the availability zones and regions that will be scanned by the framework while searching for the new instance. Longer range means more markets and a higher chance to find a cheaper price.

c) Instance resource utilization

Another factor is the current instance resources utilization parentage. This factor is going to assist in determining suitable instances based on the application's requirements. As an example, assuming that the current instance is having 8 GB of RAM, and the current utilization percentage has reached the upper threshold limit (assume 80%), that is showing that the current instance might reach its maximum capabilities and gives an indicator that the application should be migrated to another instance with more resources of its stayed there for a while. Moreover, having the resources and the utilization percentage can be used to indicate the size of the RAM that must be available in the new instance. Based on that, the new instance must have more than 8 GB of RAM to be considered as a valid migration target.

3) Post-selection Stage

The post-selection stage is responsible for achieving the hopping operation. It includes three processes which are the leasing process, back up process and the migration process. The post-selection stage processes are going to be activated by receiving a trigger from the instance selector. Once a start trigger has been received, the leasing and the backup processes will be started immediately. The third process (migration process) is

going to wait for the other two processes to get finished their jobs before it can start. The following subsections explain more details about each process.

a) leasing process

The function of the leasing process is to communicate with the CSP to acquire the selected instance. The leasing process receives details related to the target instance that was nominated by the instance selector. The Selected instance details include the instance type (family and size), leasing scheme, region, and availability zone. The leasing process starts by creating a leasing request for the selected candidate and submit it to the CSP. Then, the leasing process is going to wait for the CSP feedback related to the submitted request. The CSP will respond to the leasing request Based on the availability of the selected instance. If the leasing request was rejected, then the leasing process will remove the rejected instance from the price list that was used by the instance selector, and it will trigger the selector again to nominate another instance. The processes will be repeated until receiving an approve response and process with spot leasing [27], or the countdown timer reached a critical point, which will force the selector to nominate an on-demand instance. Upon the completion of this process, a new instance should be leased and ready to be used. The details of using the new instance will be forwarded to the migration process including the IP address, access credentials.

b) Backup process

The backup process is responsible for backing up the important data and store it in an external location just like Elastic Block Store EBS or Amazon Simple storage service S3. This data will be migrated to the new instance, but it will be moved separately from the application container.

The backup process will start concurrently with the leasing process. While the leasing process is waiting for the cloud platform to reply, the backup process keeps backing up the data for the hopping operation. The preparation includes taking a snapshot for the application container, copying the data to cloud storage. Once the backup process finished from data copying operation, it will send the backup location to the migration process to be launched in the new instance.

c) Migration process

The migration process is responsible of achieving the hopping operation. The migration process will start after the end of both leasing and backing up processes. To start the hopping operation, the migration process requires some details. First, it needs to receive the new instance details, which has been recently leased by the leasing process. Details such as the IP address, the access credentials, accessing method, the region and availability zone. Second, it needs to receive the details of the data backup that has been created by the backup process. Details include the location of the backup, accessing credentials, decryption keys. Then, the migration process is going to check the status of the new instance and install any needed tools such as the virtualization tools and containers manager. If the new instance is up and ready to host the application container, then the migrator process is going to live-migrate the application container to the new instance. Upon the completion of the live migration, the migration process needs to make sure that the application is running correctly before proceeding with terminating the old instance.

5. PROPOSED FRAMEWORK EVALUATION

Evaluating the proposed framework has been done using a dataset that contains real data. the following subsection describes the used dataset in more details.

A. Used dataset

The dataset that has been used in evaluating the proposed framework was collected directly from Amazon. The data was collected in the period between the 10th of November 2019 and 8th of February 2020. The dataset contains spot price history related to 16 out of 22 regions. Some regions like ap-east-1, ap-northeast-3 and me-south-1 were designed for local access only or require an authorized AWS account to gain access. That is why those regions were not included in the dataset. The account that was used to access the AWS to collect the data was a personal account. Creating a personal AWS account requires using a valid credit card, even if that account was created to access the free tier. API requests were generated using the AWS CLI tool [28]. Each request is related to a certain instance type. The requests got reported to cover all instance types in all zones and Regions for a specific period. The API response from AWS is containing the Spot instance prices for the period that was defined in the request. On the other hand, only concurrent spot instances prices were published on AWS website, which is got refreshed every 5 minutes [29]. Collecting the spot price using CLI tool allows collecting spot prices history that was published 90 days ago, which is not available on the AWS website. The On-demand prices were also needed to conduct the evaluation. Since the on-demand prices are not changing dynamically, it can be collected either by using the AWS website or the CLI. Table 2 shows the dataset properties and their associated ranges.

TABLE II. DATASET PROPERTIES AND RANGES

Attribute	Ranges
Date	From 10-11-2019 to 8-2-2020
No. of Regions	16
Availability Zones	50
Instance types	219
Operating system	4 (windows, Red hat, Suse and Linux/Unix)
Dataset size in MB	415

B. Data Pre-processing

pre-processing calculations were needed to be done on the datasets to use their contents in the evaluation process. AWS publishes spot prices based on multiple events, such as having a price changing event or a regular periodic price logging. Both datasets containing the data as it was published by AWS without any changes. The period between every two consecutive rows is not fixed and may vary depending on the spot dynamic changings. Moreover. The published spot leasing rates presenting leasing cost per hour. Those issues have created a need to do pre-processing calculations for the data before using it in the simulation. The pre-processing was done by calculating the number of hours between every two consecutive rows, and then multiplying it by the leasing rate to find out the leasing cost for that period. The pre-processing has been done using a script that was integrated with the target simulator (refer to the appendix section for more details).

C. Data Samples

To evaluate the framework, one sample has been taken from each dataset. each sample represents an instance family that includes multiple instance types. The instance families were taken from multiples availability zone located within two AWS regions (EU-Central-1 and US-East-1). From each family, a single spot instance type has been selected. The selected instance types were suffering from the spot leasing associated risks, which make them good testing subjects to the proposed framework.

The sample has been taken from AWS EU-Central-1 and US-East-1. From those two regions, M5 instance family has been selected. M5 is the fifth generation of AWS general-purpose instance family [30]. Similar to M4 instance from Sample 1, M5 was designed to be used for web servers, application servers, small and mid-size database servers, clustering, gaming and more. M5 family came in different options, such as supporting for a high-speed network (M5n) or faster disk (M5d) that can be chosen from many different sizes (between m5.large up to m5.24xlarge). Table xx shows the M5n instance family and types that are offered by AWS [31]. The table also shows the resource configuration for each instance type including the number of CPU cores and the size of RAM allocated to each type. M5n.24xlarge EC2 instance type has been selected as a sample. M5n.24xlarge came equipped with 96 CPU cores, 384 Gigabyte of RAM. It also has an internal solid-state hard disk drive and very fast network interface card that can reach to 100 Gbps. Those specifications make M5n.24xlarge as a highly preferred option for many applications.

The evaluation process has been done for two months period, started on December 1st, 2019 and ended on January 31st 2020 with a total of 1488 hours. Table 3 shows AWS M5n.24xlarge EC2 sample parameters. The table shows the regions that were the data has been taken from. Moreover, it shows the Availability zones belong to each region. Furthermore, the table shows the minimum, the maximum, the average, the standard deviation, and the variance that are related to spot leasing scheme. The table also shows the on-demand leasing price rate per hour (in bold). As shown in the table, M5n.24xlarge from zone A located in Eu-Central-1 has a high variance comparing with same instances from the other zones. Another issue to be considered was M5n.24xlarge from zone C had high spot price rate that reached the on-demand rate and exceeded it in some points.

Figure 6 shows the spot and on-demand leasing rates per hour for instance type M5n.24xlarge for the period between the 1st of December 2019 and 31st of Jan 2020. The mentioned instance type was in AWS Frankfort region (EU-central-1) in availability zone A. The figure shows that on the 1st of December the spot leasing rate/hour was around 1.8$. On the 17th of December, the spot rate started increasing, until it became equivalent to the on-demand by around 7$/hour, and then remains until the end of January. the shown instance was suffering from one of the spot leasing risks which is the price increase due to the demand changing.

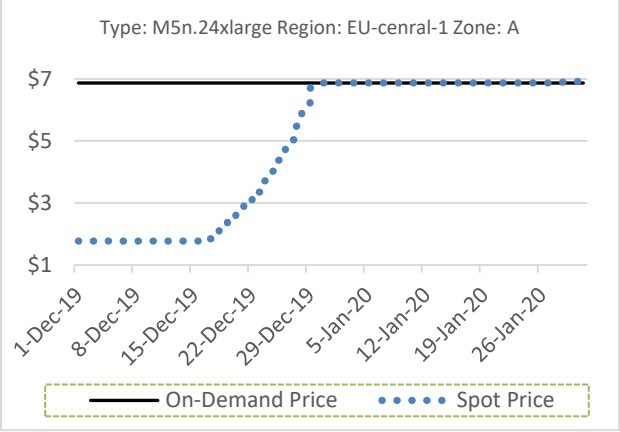

Figure 6. Spot and on-demand leasing rate for the sample

D. Case studies

Six case studies have been conducted to measure the effectiveness of using the proposed framework. The case studies were designed to measure the effects of increasing both the hopping range and the number of instance types. Each case study has different configurations than the others. Table 4 shows the hopping range and instance types configurations for all the cases. The hopping range defines the framework searching scope including the regions and availably zones, while the instance type column shows the types of instances that were used in each study. Cases 1, 2, 3 were teasing the hopping range without adding any extra instance type. case study 1 was used to test the framework ability to switch between spot and on-demand leasing schemes within a single zone. Case 2 tests the framework adaptability after adding extra zones to the hopping range. In case 3 the hoping range was extended again by adding an extra region to the hopping range. The added region contains 3 zones.

TABLE III. M5N.24XLARGE SAMPLE PARAMETERS

Instance Type	regions	AZ	On-Demand price ($/Hour)	Spot Price ($/Hour)				
				Min	Max	Average	standard deviation	Variance
M5n.24xLarge	EU-Central-1	**A**	6.87	**1.77**	**6.92**	**4.46**	**2.12**	**4.45**
		B		1.77	1.82	1.77	0.01	0.001
		C		6.87	6.92	6.87	0.01	0.001
	US-East-1	A	5.81	1.73	1.73	1.73	0.001	0.001
		B		1.73	1.74	1.73	0.001	0.001
		C		1.73	1.73	1.73	0.001	0.001

On the other hand, case studies no.4, 5 and 6 were used to test the framework responses after allowing it to hop between M5n,24xlarge and the newly added instance type (M5dn.24xlarge). Case 4 was used to measure the changes after adding the new instance but with limited hopping range to a single zone. Case 5 used the same instance types as case 4 but with increasing the hopping range to include several zones, while in case 6 the hopping range became including two regions.

TABLE IV. CASE STUDIES CONFIGURATIONS

	Hopping range		Instance types
	Zones	Regions	
Case 1	C	Eu-Central-1	M5n.24xlarge
Case 2	A, B, C	Eu-Central-1	M5n.24xlarge
Case 3	A, B, C	Eu-Centra-1, Us-East-1	M5n.24xlarge
Case 4	C	Eu-Central-1	M5n.24xlarge, M5dn.24xlarge
Case 5	A, B, C	Eu-Central-1	M5n.24xlarge, M5dn.24xlarge
Case 6	A, B, C	Eu-Centra-1, Us-East-1	M5n.24xlarge, M5dn.24xlarge

6. RESULTS

Table 5 shows the results of applying the proposed framework for each case study. the table shows the period for the data that were used in the simulation process, which includes December 2019 and January 2020.

Leasing M5n.24xlarge EC2 instance for that period using on-demand and spot leasing scheme costs 5621.76$ and 2504.41$ accordingly. The table also shows the results that were obtained by using the framework for each case. Moreover, the table shows the cost reduction obtained by the framework over the on-demand and the spot leasing schemes. All cost reduction percentages were obtained from a single hopping operation.

The framework has been simulated to test the effects of different type of hopping. Table 6 shows the result of the 6 case studies that were applied to the proposed framework. The table shows the effect of increasing the hopping range horizontally (going from left side to right side). While going vertically from (up to bottom) shows the effect of enabling hopping between different instance types. The table also shows that increasing the hopping range had given positive effects on reducing the total leasing cost. Comparing between the case studies results horizontally such as (1 and 2), (2 and 3) shows that with increasing the hopping range, more discount rates were obtained. Furthermore, comparing the case study results vertically from up to bottom such as case (1 and 4) shows that increasing the instance types -that the framework allowed to use- has a positive impact on reducing the total leasing cost. moreover, complaining several hopping types together improve the cost reduction even more. The best results were obtained by enabling the framework to use the longest hopping range with having multiple instance types as shown in case No. 6.

TABLE V. FRAMEWORK RESULTS FOR THE USED CASE STUDIES

Leasing period	December 2019 and January 2020					
Cost using on-Demand Scheme ($)	10219.58					
Cost using Spot scheme ($)	7275.2					
Framework results	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
Total leasing using the Framework ($)	7272.55	2637.9	2577.67	2638.5	2637.9	2577.66
Framework to on-demand cost reduction (%)	28.8%	74.19%	74.78%	74.18%	74.19%	74.77%
Framework to spot cost reduction (%)	0.04%	63.73%	64.57%	63.72%	63.73%	64.56%
Number of Hops	1	1	1	1	1	1

TABLE VI. THE EFFECT OF INCREASING THE HOPPING RANGE AND THE NUMBER OF INSTANCES TYPES

	Hopping range								
	Eu-central-1 (A) Single zone			Eu-central-1 (A, B, C) Multiple zones			Eu-central-1 (A, B, C) & Us-east-1 (A, B, C) Multiple region		
Instance types (hopping targets)	Case study	Reduction/ On-demand	Reduction/ spot	Case study	Reduction/ On-demand	Reduction/ spot	Case study	Reduction/ On-demand	Reduction/ spot
	1	28.8%	0.04%	2	74.19%	63.73%	3	74.78%	64.57%
	4	74.18%	63.72%	5	74.19%	63.73%	6	74.77%	64.56%

7. LIMITATIONS

There are a few limitations that were encountered. One of them was the difficulty in collecting the live spot instances price. Even accessing to instances price history was restricted to only those who are registered with Amazon. Such data is published on the internet, but when it comes to using an API to collect the data, it will require having a valid username and password. Another limitation was faced when trying to measure the impact of resource utilization on the selection algorithm by using simulation. This feature requires to have actual resources utilization percentages from a running server, such an algorithm can be tested using a real implementation for the proposed system using a running server.

8. CONCLUSION AND FUTURE WORKS

The cloud service providers are managing data centers that contain the physical infrastructure and equipment's. According to the supply and demand, there are instances left without being used. The service providers are offering those instances for leasing with low prices, with a chance to be reclaimed at any moment. Utilizing such instances can reduce the total leasing cost even more, but it will be increasing the risk of losing the instances as well. To utilize the extra instances while mitigating the risks, we proposed an approach that is using instance hopping. The proposed instance is monitoring the prices in different regions and migrate the instance if a less price has been found. Multiple case studies have been simulated to test the feasibility of the proposed approach using actual data obtained directly from AWS. We found that the proposed approach has successfully reduced the cost of cloud instance provisioning. furthermore, the proposed algorithm gives better results along with increasing the hopping range and the number of instance types.

This work can be improved even more if different cloud providers were included. Such a thing can be done by providing a mechanism to crossmatch different instance models based on hardware resources, which in turn is going to help to compare the difference in the prices between the providers. That might open the door to inter-cloud hopping when it came to cut the costs. On the other hand, the proposed algorithm needs to be implemented and tested on a real instance to evaluate its impact on the cost and count the average migrations. Moreover, if instance utilization percentage is known, it will allow hopping between different instance types, and that is going to reduce the cost even farther.

REFERENCES

[1] Zhang, Q., Zhu, Q., & Boutaba, R, "Dynamic resource allocation for spot markets in cloud computing environments," in *Fourth IEEE International Conference on Utility and Cloud Computing*, 2011.

[2] Sharma, U., Shenoy, P., Sahu, S., & Shaikh, A, "A cost-aware elasticity provisioning system for the cloud," in *31st International Conference on Distributed Computing Systems*, 2011, June.

[3] Menache, I., Shamir, O., & Jain, N, "On-demand, spot, or both: Dynamic resource allocation for executing batch jobs in the cloud," in *11th International Conference on Autonomic Computing*, 2014.

[4] He, X., Shenoy, P., Sitaraman, R., & Irwin, D, "Cutting the cost of hosting online services using cloud spot markets," in *24th International Symposium on High-Performance Parallel and Distributed Computing*, 2015, June.

[5] Yi, S., Kondo, D., & Andrzejak, A, "Reducing costs of spot instances via checkpointing in the amazon elastic compute cloud," in *IEEE 3rd International Conference on Cloud Computing*, 2010, July.

[6] J. Zheng, T. E. Ng, K. Sripanidkulchai, and Z. Liu, "Pacer: Taking the guesswork out of live migrations in hybrid cloud computing," Rice University Technical Report, Jan 2013.

[7] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schioberg, "Live widearea migration of virtual machines including local persistent state," in *VEE*, 2007.

[8] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *INFOCOM*, USA, 2010.

[9] Lu, T., Stuart, M., Tang, K., & He, X, "Clique migration: Affinity grouping of virtual machines for inter-cloud live migration," in *9th IEEE International Conference on Networking, Architecture, and Storage*, 2014, August.

[10] Shastri, S., & Irwin, D, "HotSpot: automated server hopping in cloud spot markets," in *In Proceedings of the 2017 Symposium on Cloud Computing*, 2017, September.

[11] S. Shastri, "System Support for Managing Risk in Cloud Computing Platforms.," 2018. [Online]. Available: https://scholarworks.umass.edu/dissertations_2/1389/.

[12] Chi, Y., Cai, W., Hong, Z., Chan, H. C., & Leung, V. C., "A privacy and price-aware inter-cloud system," in *7th International Conference on Cloud Computing Technology and Science (CloudCom)*, 2015, November.

[13] Lee, K., & Son, M., "Deepspotcloud: leveraging cross-region gpu spot instances for deep learning.," in *In 2017 IEEE 10th International Conference on Cloud Computing*, 2017, June.

[14] Ali-Eldin, A., Westin, J., Wang, B., Sharma, P., & Shenoy, P., "Spotweb: Running latency-sensitive distributed web services on transient cloud servers," in *In Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing (pp. 1-12)*, 2019, June.

[15] Ekwe-Ekwe, N., & Barker, A., "Location, location, location: exploring Amazon EC2 spot instance pricing across geographical regions.," in *In 2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID) (pp. 370-373). IEEE.*, 2018, May.

[16] Sanad, A. J., & Hammad, M., "Reducing Cloud provisioning Cost Using Spot Instances hopping.," in *In 2019 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT),IEEE*, (2019, September).

[17] Sanad, A. J., & Hammad, M, "An Approach to Reduce Cloud Spot Instances Cost," *International Journal of Computing and Digital Systems (IJCDS)*, 2020-under process.

[18] Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., & Leaf, D, "NIST cloud computing reference architecture.," NIST special publication, 2011.

[19] "Gatner," [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2019-04-02-gartner-forecasts-worldwide-public-cloud-revenue-to-g.

[20] "AWS," [Online]. Available: https://aws.amazon.com/ec2/pricing/reserved-instances/pricing/.

[21] "Spot Instance Interruptions," [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/spot-interruptions.html#interruption-behavior.

[22] AWS, "Instance Lifecycle," [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-lifecycle.html.

[23] S. Shastri, "Amazon EC2 spot price traces," 28 Aug 2018. [Online]. Available: https://umass.app.box.com/s/jfbuno6sgyvmul72ise99tc09lqnja5d.

[24] "How AWS Auto Scaling Works," [Online]. Available: https://docs.aws.amazon.com/autoscaling/plans/userguide/how-it-works.html.

[25] AWS, "Amazon EC2 Spot Two-Minute Warning," [Online]. Available: https://aws.amazon.com/about-aws/whats-new/2018/01/amazon-ec2-spot-two-minute-warning-is-now-available-via-amazon-cloudwatch-events/.

[26] "EC2 Spot Instance Termination Notices," [Online]. Available: https://aws.amazon.com/blogs/aws/new-ec2-spot-instance-termination-notices/.

[27] "Spot Instance Requests," [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/spot-requests.html.

[28] "AWS Command Line Interface," [Online]. Available: https://aws.amazon.com/cli/.

[29] "Amazon EC2 Spot Instances Pricing," [Online]. Available: https://aws.amazon.com/ec2/spot/pricing/.

[30] "Amazon EC2 M5 Instances," Amazon AWS, [Online]. Available: https://aws.amazon.com/ec2/instance-types/m5/.

[31] "Amazon Ec2 Instance Types," Amazon Web Services, [Online]. Available: https://aws.amazon.com/ec2/instance-types/.

Ali Jasim is a Demonstrator in the Department of Computer Engineering at the University of Bahrain. He is an M.Sc. student in the college of IT at The University of Bahrain. He holds a BS.C degree in Computer Engineering from University of Bahrain, 2009. His research interest includes cloud computing, IoT and machine learning

Mustafa Hammad is an Associate Professor in the Department of Computer Science at the University of Bahrain and Mutah University. He received his Ph.D. in Computer Science from New Mexico State University, USA in 2010. He received his Masters degree in Computer Science from Al-Balqa Applied University, Jordan in 2005 and his B.Sc. in Computer Science from The Hashemite University, Jordan in 2002. His research interests include machine learning, software engineering with focus on software analysis and evolution.