AWS EC2 Spot Instances for Mission Critical Services

Jerry Danysz, Víctor del Rosal, Horacio González–Vélez
Cloud Competency Centre, National College of Ireland
E: jerry.danysz@dancom.eu, {Victor.DelRosal,horacio}@ncirl.ie

KEYWORDS

AWS; spot instance; utility computing; elastic provisioning; SLA; random forest regression; cloud computing

ABSTRACT

For over a decade now, Amazon Web Services (AWS) has offered its spare capacity at a discounted price in the form of EC2 spot instances. This discount comes at the price of variable pricing and sudden instance termination. In this paper, we present a machine-learning solution to one of the challenges when using AWS Spot Instances, namely the termination of the instance on short notice. Our system, Spot Instance Management System (SimS), can effectively manage spot instances and keep up the availability at the desired level using 100-tree Random Forest Regression model. By using a risk assessment mechanism and proactive actions, SimS assures a three-nines SLA using AWS spot instances with lower running costs on workloads for a major European financial institution.

INTRODUCTION

AWS EC2 spot instances have offered a significant discount–up to 90% according to AWS–compared to their dedicated and on-demand/reserved models for over a decade. One of the base assumptions behind spot instances is that price is dynamic and can change anytime based on available capacity and current demand for the type of instance. Consequently, the attractive pricing comes with the trade-offs related to the availability of the spare capacity on a given region at a given period in time.

The use of spot instances requires customers to set a maximum instance price (per hour) they wish to pay and also to understand the risk, e.g. if the price of a given instance changes above the maximum price, the instance could be suddenly terminated and all data lost if not saved elsewhere. Until 2015, the instance termination was executed without prior notification to the customer. Since then, a different type of termination behaviours is offered, ranging from default termination through stopping the instance to hibernation of the instance. Nonetheless, the usage of spot instances, even if cost-effective, is normally circumscribed to applications that are either non-critical or designed for interruptions.

This paper presents the Spot Instance Management System (SimS) whose main goal is to counteract the sudden/unexpected termination of cloud services running on top of EC2 spot instances. SimS employs a parallelised Random Forest Regression [5], [6] model for continuous response with 100 trees to predict price fluctuations. In this case, rather the regression model allows systems to swiftly detect when price would change and calculate the risk of the instance termination, and then consider the appropriate actions to maintain a 99.9% (a.k.a three nines) Service Level Agreement (SLA).

The actions could be live migration to another availability zone or redeployment of spot instance with a higher maximum bid. It is worth to notice that all actions are done proactively, while the risky instance is still running, therefore minimising the potential downtime of any service using EC2 spot instances.

RELATED WORK

Spot Price Prediction (SPP) has been a topic of research since the introduction of AWS spot instances in 2009. Yehuda et al. [1] predict the spot price by deconstruction and reverse engineering of a hypothetical spot instance algorithm that, despite common assumptions on spot prices, does not necessarily lead to supply- and demand-driven fluctuations but alternatively where prices are randomly generated from dynamic hidden reserve pricing. To confirm their hypothesis, they analyse the spot market and divided it into three pricing epochs with each epoch change at the significant change of SPP pricing models. They acknowledge that there is a market element in the price, but prices are still driven from the hidden reserved price.

In contrast, Singh and Dutta [13] seem not to fully concur with Yehuda et al.'s conclusion given that their model for dynamic price prediction is accounting for global market trends and local seasonality. The dynamic price prediction model is presenting two types of predictions:

Communications of the ECMS, Volume 34, Issue 1,
Proceedings, ©ECMS Mike Steglich, Christian Mueller,
Gaby Neumann, Mathias Walther (Editors)
ISBN: 978-3-937436-68-5/978-3-937436-69-2(CD) ISSN 2522-2414

short-term (hourly based) and long-term (a week ahead) based on analysis of 9 months of historical data for the top ten most used spot instances. They present the prediction result with an average 9.4% prediction error in short-term prediction and 20% in long-term (five days and more) price prediction.

Accurate price prediction for cloud instances typically relies on assertive workload quantification, which is related to the application type, e.g. HPC [11], [12] or to extrinsic factors such as seasonality or social-demographic factors [14].

Consequently, AWS spot price prediction has been previously modelled using a moving simulation model to create an artificial neural network-based algorithm for price prediction [16]. It employs historical data available for only medium size instances in a period of 7 months to train the MLP model, resulting in 4% prediction error in short-term prediction (hourly prediction) on average for medium size spot instances. This leads to the conclusion that neural network models are well suited for the prediction of price changes of spot instances. Zhao et al. [17] follow a different prediction approach by using a time-based series forecasting method, ARIMA Model (Auto-Regressive Integrated Moving Average) that is lighter compared to machine learning techniques like neural networking. Other approaches [13] have added a seasonal component to their ARIMA model effectively changing it to the SARIMA model. SARIMA has been used to analyse five months of historical data and create a prediction that is close to the average price for a period of 48 hours.

Random forest regression using historic AWS traces has been recently reported in the literature [8]. Similar approaches have previously used Support Vector Poly Kernel Regression (SMOReg), Gaussian Process and Linear Regression [3], and multiple discrete-time modelling [7]. All these models have been trained for the month with 12 months of historical data for the three most used types of instances. Their predictions have been typically generated for short (next hour), medium (half day) and Long (next day) periods. Per the conclusion, the neural network-based algorithms are performing better than others for medium (half day predictions) where SMOReg is better suited for predictions with highly variable months, and random forest regressions seem to deal better with workload variability.

From the above analysis, we can see that there are different heuristic approaches to prediction of spot instance pricing, ranging from reverse engineering and understanding what principles are behind the price level, through the classical statistical approach to the most modern use of artificial intelligence and machine learning techniques. For this research, we want to prove that ML Models, specifically Random Forest Regression, in combination with business-driven automation can achieve a 99.9% SLA whilst accurately predicting price and potential price variation. Our approach has been successfully evaluated to support mission-critical cloud workloads from a major European financial institution.

*

AWS Spot bidding strategies overview

Previous research has attempted to find the best strategy to find a golden bid to assure spot instance availability. Andrzejak et al. [2], question how bidding may be conducted with strict target dates or SLAs, focusing their research on bidding strategies with that goal. Li et al. [10], classify common bidding approaches into three types:

- White box approach where bidding strategies are taking into account interactions between different market participants and effectively bidding can influence a spot price.
- Grey box approach has more individual biding strategies wherein, contrary to the white-box approach, market interactions are not taken into account, but strategies are focusing mostly on workload, cost and availability of the resources.
- Black box approach, consisting of the most common strategies which derives bidding from historical spot pricing data and do not focus so much on workload, cost and availability, nor on interactions between market participants.

The five most classic strategies in the black box approach have been discussed by Li et al. [10] and by Voorsluys and Buyya [15]:

1. The minimum price, where the bid is based on historical minimum spot price
2. Mean, the bid price is set as the mean of all values of the historical spot price
3. High, the bid price on the maximum price observed in historical data
4. Current, the bid price is set as the value of current spot price
5. On-demand is the bid price equal to the on-demand price of the instance.

The above five strategies have also been incorporated to the solution for reliable provisioning of spot instances by Voorsluys and Buyya [15], combining all five strategies with fault tolerance techniques like migration to assure the most reliable solution for the limitations of spot instances. A survey on spot pricing by Kumar et al. [9] presents four bidding strategies: bidding on near to reserve price; bidding on above the average price calculated from the historical data; bidding close to the on-demand price; and, bidding over the on-demand price. Each of the aforementioned

	LOW	MEDIUM	HIGH
Availability			✓
Integrity			✓
Confidentiality			✓

TABLE I: Business Application Critically Matrix

techniques has its own benefits particularly for the final consumer, but also its costs in terms of the actual deployment and the business interest of the cloud provider.

In contrast to other attempts to use machine learning for AWS spot price prediction [4], the authors researched the possibility of maintaining the agreed service level of 99.9% that is common for a business-critical application while savings costs by running those applications on EC2 Instances. The aim was to test how EC2 spot instances can be used to reliably host mission-critical applications. Underpinned by a parallel Random Forest Regression model, we have employed a real application scenario borrowed from a major European Financial Institution to validate our findings.

METHODOLOGY

- Availability: impact if information availability is affected.
- Integrity: impact if information integrity is affected.
- Confidentiality: information confidentiality level.

The assumption is a scenario where the IT Service Provider is responsible for providing business-critical batch processing and ERP application for a European Financial Institution with agreed Availability of the service on the level of 99.9%. The IT Service Provider and the European Financial Institution have agreed upon a project to run the batch processing application using only AWS Spot Instances for the cost-effectiveness. The criticality matrix shown in Table I determines the application criticality level of Availability, Integrity, and Confidentiality.

As seen above, the application is highly critical to the core business of the European Financial Institution. If the availability of the system and, therefore, the application is compromised, the institution's ability to operate properly is degraded, potentially leading to significant profit loss and reputational damage. Compromise of information integrity may lead to substantial profit losses as well as posing a risk to confidentiality.

Since the system holds sizeable amounts of confidential information subject to General Data Protection Regulation (GDPR), in case of a confidentiality breach, the Institution could face legal actions based on GDPR provisions as well as widespread loss of customer trust, negatively impacting the ability to conduct business.

Based on the criticality matrix, the following service level requirements are presented:

- Application Criticality: High
- AWS Region: eu-central-1
- Cumulative Downtime including maintenance 3.65 days per year
- Mean Time to Respond: 2 minutes
- Mean Time to Resolve: 15 minutes

Based on the above, in the Service Level Agreement (SLA), the service level has been agreed at three nines (99.9%). The service Level has been measured via the monitoring system *site24x7* [1], which calculates system availability using HTTP response codes and general host response.

We have deployed a Python data analysis module based on random forest regression model using 100 trees, parallelised using 4 instances at a time. To save computing time and costs associated with it, we have decided to limit data analysis to only the `EU-CENTRAL-1` region and three selected types of instances `c5.xlarge, t3.micro and t3.medium`.

The Spot Instance Management System (SiMS) has been developed composed of four main modules:

- *The Data Collection Module*: responsible for downloading and aggregating historical data for EC2 Spot prices.
- *The Data Pump Module*: responsible for moving collected data from S3 Landing zone Bucket to S3 Staging Zone bucket after data collection, where later this data is used by Data Analysis Module for training the machine learning algorithm.
- *The Data Analysis Module*: responsible for analysing historical data gathered by the data collection module and then is responsible for applying the machine learning model based on random forest regression. Execution is started via Amazon Lambda function that is responsible for starting the AWS Fargate Task(Figure 1).
- *The Risk Assessment and Automation module*: responsible for risk analysis of the instance interruption in each of availability zones in the configured region and next for taking the appropriate actions concerning the level of the risk(Figure 2).

The prime idea behind the system is to have automated mechanism that with use of machine learning, can predict the price of the spot instance in the next hour and act accordingly by migrating the affected spot instance to the next availability zone. In case when all availability zones would

[1] `https://www.site24x7.com`

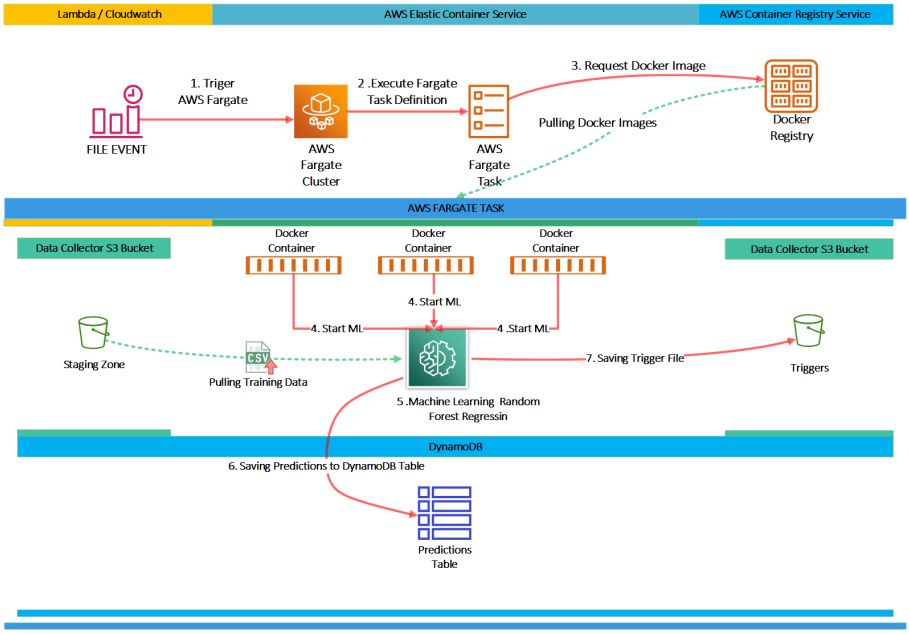

Fig. 1: The Data Analysis Module

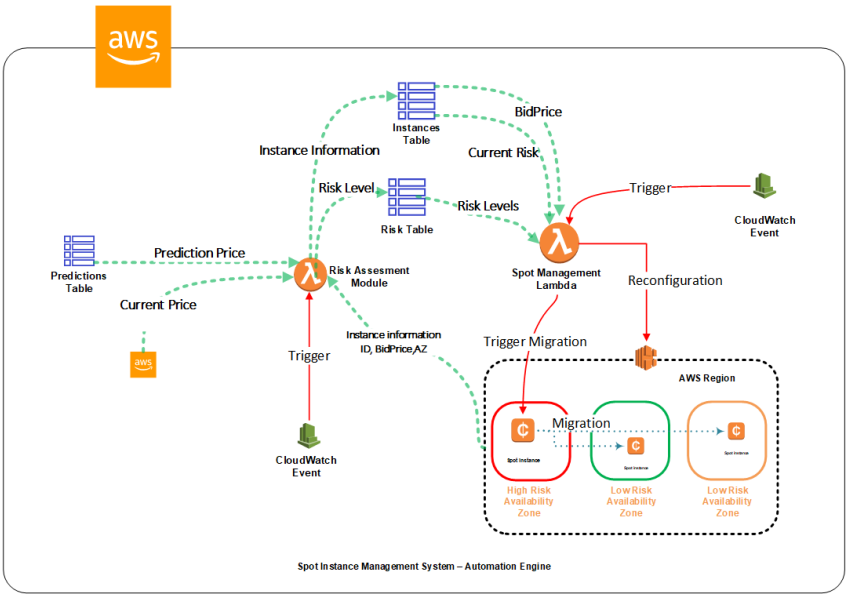

Fig. 2: The Risk Automation Module

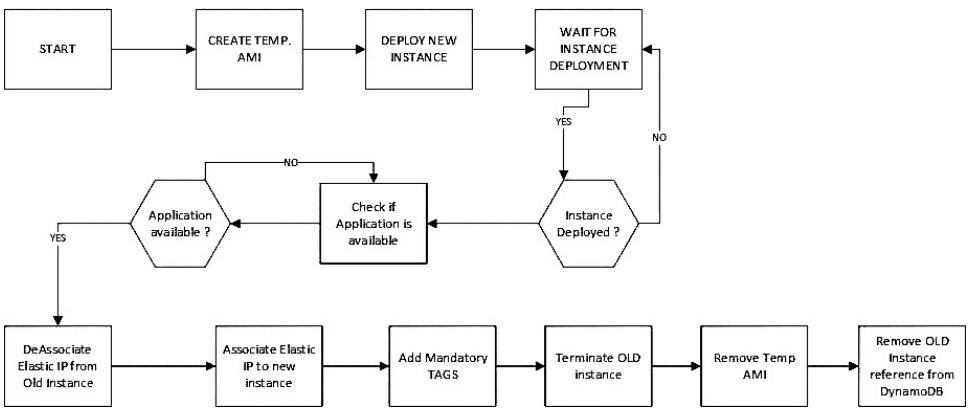

Fig. 3: Instance Migration Process

be labelled as High Risk, the system will redeploy the spot instance with a new bid price increased by 20%.

In the worst-case scenario, where there would be no spare capacity to spin the machine, the system would run the on-demand machine only when capacity is unavailable.

The risk engine is the part of the risk and automation module responsible for the calculation of the risk of instance interruption in each of the availability zones. The risk is calculated based on the maximum bid price, current price, predicted price and the threshold for each of the risks level Low Lr, Medium Mr, and High Hr.

The risk is calculated for each of the availability zones in a given region with the following formulae using Current price (Cp), Maximum bid price (Mb), and Prediction price(Pp) :

$$Lr = Cp < \left(\frac{55}{100} x\ Mb\right)\ AND\ \ (Pp <\ Mb)$$

$$Mr = Cp > \left(\frac{55}{100} x\ Mb\right) AND\ Cp < \left(\frac{8}{10} x\ Mb\right) AND\ \ (Pp <\ Mb)$$

$$Hr = Cp > \left(\frac{8}{10} x\ Mb\right)\ \ AND\ \ (Pp <\ Mb)$$

In case prediction price (Pp) is larger than the maximum bid price, risk is calculated as High.

If current risk is Medium and there is at least one availability zone with low-risk assessment than migration operation is starting to that zone. If there is no low zone, there is no action taken. If current risk is high and low or medium zones are available, then the migration process will move the instance to the lowest risk zone. If the risk is high in all availability zones, the machine is redeployed by the automation engine to the same availability zone but with a 20% higher maximum bid price. The migration process is shown in Figure 3.

EVALUATION

For simulation purposes, a Python script has been developed to execute the following steps:

- Change the predicted price for the next full hour to one of the values selected randomly on every execution (on-demand price, predicted price, maximum bid price, current price, mean of all above).
- Execute Risk Recalculation.

After risk is recalculated, we would relay on the automation module to evaluate and execute necessary actions against the SimS Managed instance. For the instance that is not managed (the

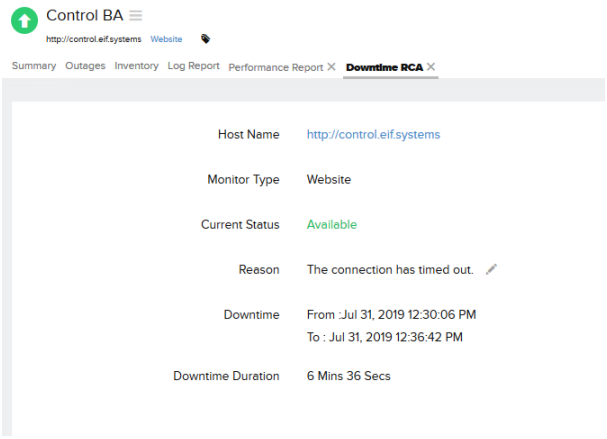

Fig. 4: Root Cause Analysis Report from Monitoring System

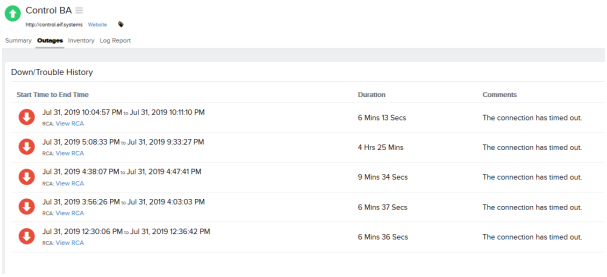

Fig. 5: Outage reports of control system

control instance), the script will execute the following steps: evaluate if the maximum bid price is lower than the predicted price and If the predicted price is higher, terminate the instance after 2 minutes. The authors have also included a simulation of an engineer acting on a given incident created by the monitoring system. A manual intervention required to bring the system back online has been calculated to take approximately four minutes and forty five seconds.

The data collection bucket contains landing zone, staging store and triggers, where trigger files at the end of the execution of data collection and data pump are stored.

To gather availability data, it was necessary to find a monitoring system that would be independent of the solution and would provide SLA-type reports and SLA configuration. The authors opted for the Site24x7 SaaS offering and configured it for website monitoring, checking connectivity to HTTP ports of defined targets, response times, DNS response time and general availability by ping, in one minute intervals.

During the migration execution, the following outputs are generated:

- New Temporary Amazon Image (AMI) created based on the current instance
- New Instance deployed to target availability zone based on the created image

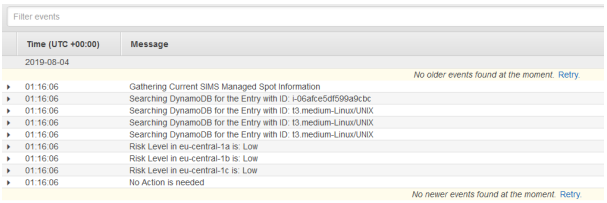

Fig. 6: Cloud Watch Log representing No Actions

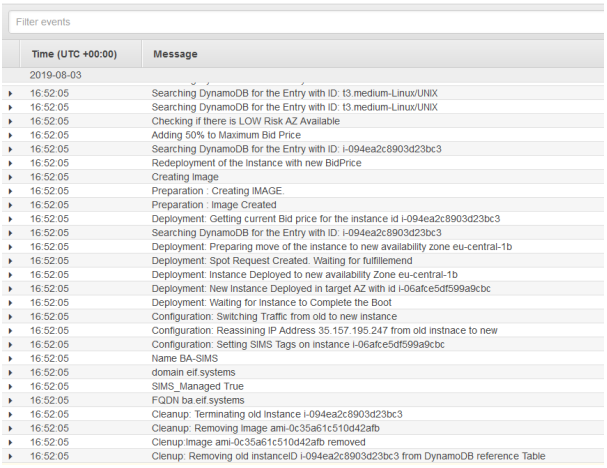

Fig. 7: Redeployment of Instance with new Bid Price

- Old Instance Data removal from DynamoDB Instances Table.

As part of the interruption of Control System, we have the following:

- Scenario: Classic Termination of EC2 Spot Instance
- Actors: Simulator
- Desired Outcome: System Terminated and restored

The simulation script to terminate EC2 instances operates with 2 minute delays, simulating the Amazon Notification grace period. It also later simulates the system engineer's input by restoring the service.

The control system during the experiment has been terminated a number of times causing reported outage (Figure 5) and unavailability of the application.

SimS Automated Actions low Risk

- Scenario: risk Level low low low
- Actors: SimS System
- Desired Outcome: no migration initiated

Expected behaviour, if all availability zones are low risk, then no action is taken (Figure6). As presented above, the automation module evaluated the risk and did not perform any actions.

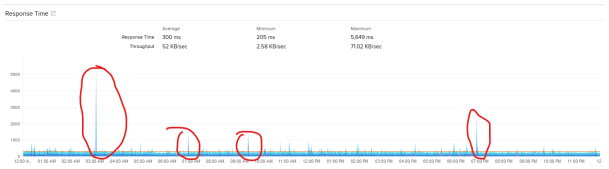

Fig. 8: Response time monitoring of the System

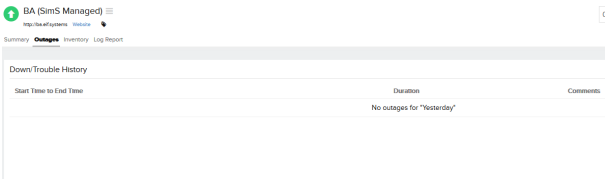

Fig. 9: No outage reported by the system. Monitoring system did not report system outage, the only indication of migration is a short spike in response time.

SimS Automated Actions High Risk

- Scenario: Risk Level high high high
- Actors: SimS System
- Desired Outcome: instance redeployed with higher bid price, no system downtime reported by monitoring system

During the execution, the Automation Module will evaluate risk in all availability zone and based on the result will move the instance to another availability zone or redeploy to current with the higher bid price.

In this scenario, The Sims System detected that all availability zones in the region are high risk. In this situation, the system is designed to redeploy the instance with a higher bid price to mitigate the high risk of interruption and therefore to maintain desired availability level by setting up (migrating new instance) and reassigning the elastic IP from Risky instance to newly deployed one, therefore allowing traffic to reach the system without any issue.

5-Day SLA Monitoring

- Scenario: 5-Day SLA Monitoring
- Desired Outcome: SLA Level of 99.9%

A simulation script has been scheduled and running in four hour intervals affecting the classic EC2 Spot instance as well as risk analysis data for the SimS System. Due to random price selection, the exact time and date of the next interruption were not known.

Results show that usage of a proactive system managing spot instances can prove to be valuable if our main goals are low costs and availability of the system using the spot instances.

We can see that if automation is designed to act while a risky instance is still running, automated switch-over is almost seamless but at the price

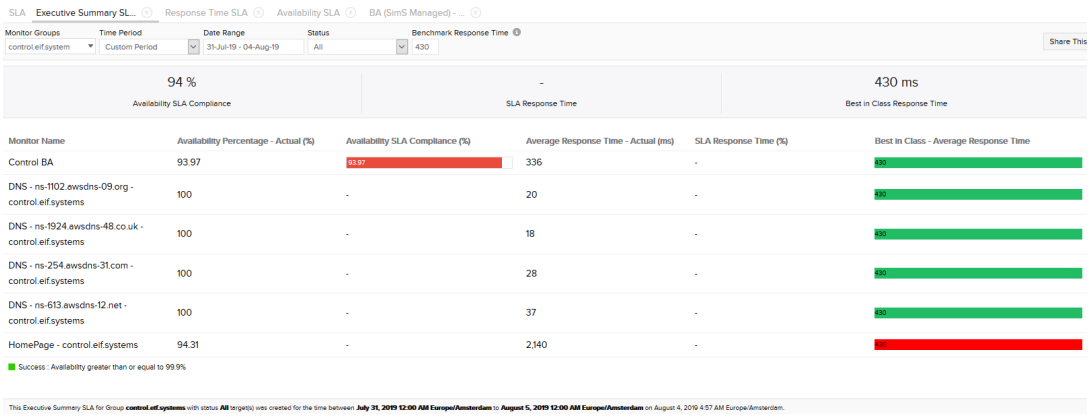

Fig. 10: 5-Day SLA Report for Control System

Fig. 11: 5-Day availability report for Control System

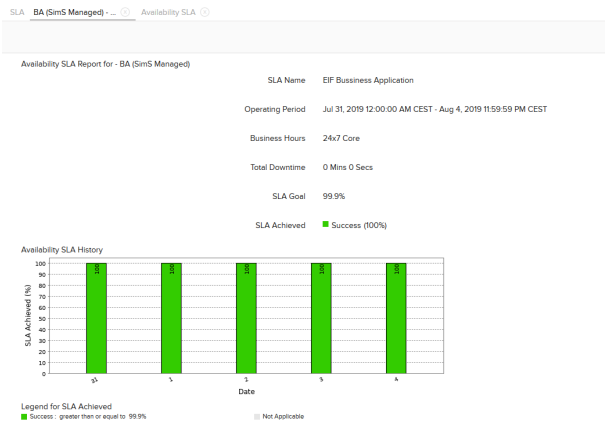

Fig. 13: 5-Day availability report for SimS Managed System

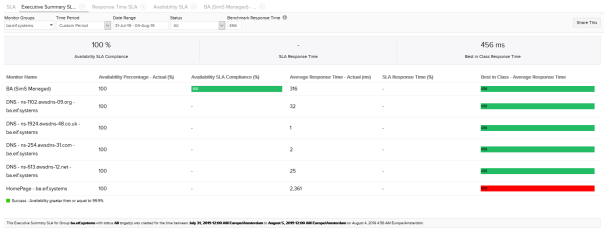

Fig. 12: 5-Day SLA Report for SiMS Managed System

of a drop in performance during the migration of the instance.

For applications that are very response time-sensitive this could be still the issue, as well as for the application where data is written continuously as during the migration the checkpoint (image) of the machine is created and any data written in the time between image creation and switch over of the traffic to the newly deployed machine would be lost.

The question in here would be the trade-off, in case of classic spot instance the customer can face unexpected termination and if they do not have any solution that would periodically save the data from that instance while running they could face complete data loss in comparing to few seconds of loss in case of SimS Managed System, therefore automated system as one presented in this research can significantly expand possibilities of the application of spot instances as well as can help with reduction of operational costs.

The Spot Instance Management system for its risk analysis is using Random Forest regression-based machine learning model, while this model during our research proved sufficient, the model itself was not a part of in-depth testing and therefore could not be as accurate as it could be hoped. Necessary comparison testing showed us that price predicted are the same as current prices or difference in price is minimal.

Our prediction machine learning model requires a lot of computing power, to calculate price predictions for 150 types of EC2 spot instances for every availability zone in EU region would require over 26 hours running in 4CPU docker containers provided by AWS. Even with those lim-

itations, it has been proved that smart, proactive system that can move around spot instances based on the predicted price and therefore risk calculated from it, can be effective in achieving not only 99.9% availability but even 100%

The authors' research question poses how can we assure SLA Level of 99.9% for service running on EC2 Spot instance, as mentioned above and shown in the evidence, the SimS System is able to effectively manage spot instances and keep up the availability on the desired level and achieve required three nines SLA thanks to its Risk assessment mechanism and proactive actions before any terminate can happen.

With automatic bid rise in case of high risk in all availability zone at the current stage, this could lead to higher than desired bid price and therefore not always attain cost-effectiveness.

CONCLUSIONS

The research question poses how a service level of 99.9% can be assured. In the paper, the authors show that, in principle, Spot instance Management system (SimS) can effectively manage spot instances to keep an availability level of 99.9%

For a system like SimS to be effective and accurate, there is a requirement for a reliable machine learning module to predict spot price in the next hour. In hourly five-day tests, we have simulated a number of interruptions of classic EC2 spot instances, and we combined this with the change of the risk level in availability zones to force the SimS system to act and migrate affected machine if are located in a high risk availability zone. Migration is using image creation (checkpointing).

Currently, due to the limitation in computing power and the aim to run the system as serverless as possible, the support for more than just a subset of instances is limited by the computational power of AWS Fargate docker containers.

Future researchers could take this solution a step further by selecting more sophisticated machine learning models that would take into account not only the historical data but also seasonality, allowing support for a more significant number of instances and availability zones.

In our research, we have presented the theoretical application of the SiMS system in the financial sector to run important CRM application. As an overall conclusion, the authors posit that with the minor adjustments mentioned in this chapter, the solution could be of commercial value in a variety of sectors where service availability, as well as low cost, play a pivotal role.

References

[1] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir. Deconstructing Amazon EC2 spot instance pricing. *ACM Transactions on Economics and Computation*, 1(3):16, 2013.

[2] A. Andrzejak, D. Kondo, and S. Yi. Decision model for cloud computing under SLA constraints. In *MASCOTS '10*, pages 257–266, Miami, Aug. 2010.

[3] S. Arévalos, F. López-Pires, and B. Baran. A comparative evaluation of algorithms for auction-based cloud pricing prediction. In *IC2E*, pages 99–108, Berlin, Apr. 2016. IEEE.

[4] M. Baughman, C. Haas, R. Wolski, I. Foster, and K. Chard. Predicting Amazon Spot Prices with LSTM Networks. In *ScienceCloud'18*, Tempe, June 2018. ACM.

[5] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[6] A. Cutler, D. R. Cutler, and J. R. Stevens. Random forests. In C. Zhang and Y. Ma, editors, *Ensemble Machine Learning*, pages 157–175. Springer, Boston, 2012.

[7] R. Keller, L. Häfner, T. Sachs, and G. Fridgen. Scheduling flexible demand in cloud computing spot markets: A real options approach. *Business & Information Systems Engineering*, 62:25–39, 2020.

[8] V. Khandelwal, A. K. Chaturvedi, and C. P. Gupta. Amazon EC2 spot price prediction using regression random forests. *IEEE Transactions on Cloud Computing*, 8(1):59–72, 2020.

[9] D. Kumar, G. Baranwal, Z. Raza, and D. P. Vidyarthi. A survey on spot pricing in cloud computing. *Journal of Network and Systems Management*, 26:809–856, 2018.

[10] Z. Li, M. Kihl, and A. Robertsson. On a feedback control-based mechanism of bidding for cloud spot service. In *CloudCom '15*, pages 290–297, Vancouver, Nov 2015. IEEE.

[11] A. Marathe et al. Exploiting redundancy and application scalability for cost-effective, time-constrained execution of HPC applications on Amazon EC2. *IEEE Transactions on Parallel & Distributed Systems*, 27(9):2574–2588, 2016.

[12] D. Petcu et al. Next generation HPC clouds: A view for large-scale scientific and data-intensive applications. In *Euro-Par 2014*, volume 8806 of *Lecture Notes in Computer Science*, pages 26–37, Porto, Aug. 2014. Springer.

[13] V. K. Singh and K. Dutta. Dynamic price prediction for amazon spot instances. In *2015 48th Hawaii International Conference on System Sciences (HICSS)*, pages 1513–1520. IEEE, 2015.

[14] P. Smith, H. González-Vélez, and S. Caton. Social auto-scaling. In *PDP 2018*, pages 186–195, Cambridge, Mar. 2018. IEEE Computer Society.

[15] W. Voorsluys and R. Buyya. Reliable provisioning of spot instances for compute-intensive applications. In *AINA 2012*, pages 542–549, Fukuoka, Mar. 2012. IEEE.

[16] R. M. Wallace et al. Applications of neural-based spot market prediction for cloud computing. In *IDAACS '13*, volume 2, pages 710–716, Berlin, Sept. 2013. IEEE.

[17] H. Zhao, M. Pan, X. Liu, X. Li, and Y. Fang. Exploring fine-grained resource rental planning in cloud computing. *IEEE Transactions on Cloud Computing*, 3(3):304–317, 2015.