Forecasting Price of Amazon Spot Instances Using Machine Learning

Manas Malik, Jain University, Bengaluru, India
Nirbhay Bagmar, Jain University, Bengaluru, India

ABSTRACT
An auction-based cloud model is followed in the spot pricing mechanism, where the spot instances charge changes with time. The user is bound to pay for the time that is initially initiated. If the user terminates before the sessional hourly completion, then the customer will be billed on the entire hourly session. In case Amazon terminates the instance then the customer would not be billed for the partial hour. When the current spot price reduces to bid price without any notification the cloud provider terminates the spot instance, it is a big disadvantage to the time of the availability factor, which is highly important. Therefore, it is crucial for the bidder to forecast before engaging the bids for spot prices. This paper represents a technique to analyze and predict the spot prices for instances using machine learning. It also discusses implementation, explored factors in detail, and outcomes on numerous instances of Amazon Elastic Compute Cloud (EC2). This technique reduces efforts and errors for forecasting prices.

KEYWORDS
AWS, EC2, Machine Learning, Random Forest

1. INTRODUCTION
Computation of cloud offers shared resources reachable over the Internet connectivity. Software and Hardware are both comprised of cloud computing resources. The different models of cloud services are- Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Network as a Service (NaaS). The cloud platform has various deployment models, they are-public cloud, private cloud, community cloud and hybrid cloud. Resources are enabled by the cloud on the basis of pay as you go model. The major provider for the computation of cloud-related resources is Amazon.

Amazon offers three instances: Spot Instances, On-demand Instances and Reserved Instances, around 16 regions across the globe.

Spot Instances has dynamic pricing, which makes it exclusive and distinguishable. Dynamic variation of the process of spot instances is built on request and delivery of cloud-related services in the data centers. Using an online auction platform, clients request a bid to attain spot instances. The auction platform offers an analysis to determine the market clearance price or spot price if the consumer bids above the aforesaid price, spot instance is obtained. Cloud providers deliver up-to-date

DOI: 10.4018/IJAIML.20210701.oa5
This article, published as an Open Access article on April 23, 2021 in the gold Open Access journal, International Journal of Artificial Intelligence and Machine Learning (converted to gold Open Access on January 1, 2021), is distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/) which permits unrestricted use, distribution, and production in any medium, provided the author of the original work and original publication source are properly credited.
and recent spot price data to aid clients in the bidding process. It provides customers with access to the web-based API for bidding spot instances. There are certain parameters to spot instance bid request, they are as follows:

1. Number of Instances
2. Availability Zone
3. Instance Type
4. Bidding Amount based on price/instance/per hour
5. Graphics Processing Unit (GPU) mode machines

2. TECHNIQUE FOR PRICING SPOT INSTANCE

Due to the fluctuations in prices and sudden termination possibility of instances, Amazon EC2 spot instance pricing becomes very complicated. Termination may be triggered by either the customer or the cloud organization.

The phases in the spot price billing system followed are:

1. User bids for one spot instance (in a specified availability zone) for a particular machine type.
2. Only when the bid price is greater than the current price, a spot instance is obtained.
3. A spot price is set as the cost of getting an instance on setup.
4. Pricing algorithm for spot instances works on an hourly basis in spite of the continuous variations, both user and cloud provider can interrupt computation.
5. The user is supposed to pay the entire hour for the estimate, even if the user interrupts the instance without the completion of the hour.
6. User doesn’t have to pay if the interruption is performed by the cloud provider. The user’s partial utilization will not be charged.
7. After an hour has been finished, the user has to pay for the computation and at the start of the new hour, a new price is set.
8. Bid prices remain the same once the instances are created.

Two important conclusions can be drawn from the above steps. Firstly, in case the cloud provider performs the termination of instance, the user gets free of cost of computational services. Furthermore, Cloud service termination of each instance results in a later start of a new instance. New instance boot time isn’t included in the simulation’s successful execution time.

2.1 Methods for Price Forecasting

The time-series approach and simulation approach is mainly two types categorized as the price forecasting mechanism. The approach to time series is based on past market prices. To compute an enormous amount of data, simulation approaches can be quite expensive. Machine Learning is amongst the most widely used techniques for forecasting time series-based prices of the spot case. The researchers have developed hybrid models to resolve the flaws in the individual models. Over the past few years study has shown there are various mechanisms developed by the researchers for predicting the prices. Machine learning is a great improvement over other techniques used for forecasting. Without actually being programmed explicitly, software applications tend to become more accurate, it is through the categories of algorithms which basically is machine learning. Algorithms built to receive input data and predicting output based on statistical analysis also update outputs as new data becomes available, this basically is machine learning.
2.2 Literature Review

One of the most important key areas of research in today’s world is the spot price prediction. Profit for the users is directly related to the accurate results of spot prediction. To save the computation time for a specific task and for significant cost reduction of the resource usage, accurate price prediction is highly important. In recent years, there has been quite a development in progressing predicting models.

The authors published a paper using the gradient descent algorithm to estimate the weighted data coefficients from the preceding month. The linear equations are solved using a gradient descent. This algorithm is the reformulation of a linear equation into a quadratic minimization problem. The difference between the predicted value or result and the actual spot price is our calculated error. The article uses a kind of artificial neural network, a multilayer perceptron. Results obtained show, ANN is a good approach to predict the spot prices and can be useful to users while bidding for the spot instances. The authors also suggested an algorithm in the article for using available computing clusters to optimize the cost and time of running simulations. It is possible to achieve the time required or referred to as the efficient time at a specified minimum reference price point. Fault tolerance is quite needed, as the compute resources are not guaranteed to spot instances and can be stopped at any point in time. Computation results are often lost because of the abrupt termination. If no fault-tolerant techniques are applied the reliability decreases.

In checkpointing techniques have been discussed by the authors, it also includes work migration techniques. This can be used to minimize the monetary costs and maximize reliability. The demand for reserved and on-demand instances increases spot instances prices, as per the current implementation. As a result, there is a sudden revocation of spot instances. So as to increase in the total cost of execution, the setting of high bids is a possible solution.

The study explores increasing the predictability of pricing, reducing the risks. For Amazon Web Services (AWS) EC2 instances study includes pricing over specified time-interval using the machine learning and LSTM techniques (long short-term memory). It is a neural network technique, comparing some of the results with ARIMA (auto regressive integrated moving average) modelling. As when compared to the ARIMA model, the results show an average reduction in mean absolute percent error (MAPE), this happens by at least 95%.

In the use of Elastic spot instances, there is a trade-off between the cost and the time of computation estimate. For minimizing the resource provisioning cost and unpredictability, a set of bidding mechanisms are proposed. Running instance to be preserved by the user with an increase in bid price for the completion of a task within a time limit. Users have access to set the limit to increase in the price, for the crossed limit termination of instance can be planned. In terms of job completion time and monetary costs, various adaptive checkpointing schemes have been explored. Authors have used a linear programming-based model to get a bidding scheme which is optimally randomized. To find a bidding mechanism that is optimal, the problem is being formulated as a Constrained Markov Decision Process (CMDP). There are certain traits of cloud pricing models.

The three main parameters on which user will evaluate a cloud provider are:

1. Approximate Cost Pricing
2. Service Quality
3. Utilization Period

Without consideration of quantity or size, constant price with unit rate or rate of unit price with the determined quantity of purchase, a fixed price approach is the pricing method without considering the defined quantity of purchase. A quality of service document holds what service provider provides the customers. A utilization period is the duration of time a user has full access to utilize the resources of the provider depending upon the SLA.
2.3 Methodology

We tackle the obstacles; we have the steps to set up the spot instances. This supports the workflow for deep learning training and minimizes the loss of progress in training if spot disruption occurs.

Our goal is to introduce a configuration that has the following features (Figure 1):

- Decouple items from computing, processing, and coding, and holding the compute instance stateless. This is to allow a quick rehabilitation and preparation after a case has been dismissed and replaced.
- Use a dedicated volume for data sets, progress training (checkpoints), and logs. The level will remain insistent and should not be influenced by instance termination.
- For the software tool, use a version control system (e.g. Git). It ensures traceability and prevents code changes from being lost when the instance is terminated.
- Minimize computer changes in the training paper. This guarantees the training document is developed independently, and backup and snapshot activities are carried out outside of the training manual.
- Automatize the creation of replacement instances after termination, add EBS volume data set and checkpoints at launch, transfer volumes through Availability Zones, restore instance status, restart training, and terminate instance after completion of training.

3. PROPOSED METHOD

The proposed method uses Machine Learning for the spot price prediction. The input to the training model is the dataset having AWS parametric labels, output is the analysis of spot prices across different regions and accuracy of spot prediction prices.

Figure 1. Workflow
The prediction model contains a Jupyter notebook that contains code and plots for:

- Data fetching using boto3
- Exploratory data analysis
- Data insights
- Data import
- Feature extraction and engineering
- Data preparation
- Predictive Model built using random forests

The initial stage for spot price forecasting is to get the market history provided by AWS. The first basic move is an account with AWS, allowing the use of the AWS Command Line Interface (CLI) for managed services. The approach suggested above is developed in language Python. Random Forest is the machine learning algorithm used to train and test the model for better accuracy in prediction.

A first glance to the data we retrieved Amazon Spot Price data which was collected by a third-party person. Data has 5 fields: <Timestamp, ProductDescription, InstanceType, SpotPrice, AvailabilityZone>

According to data fields:

- Timestamp (TS) is a time when the tuple was collected.
- Product Description (PD) is referring to a kind of operating system on an instance of Virtual Machine (VM) where the Operating System will be installed according to customer’s demands. It is consisting of 6 unique operating systems.
- Instance type (IT) is referring to the type of VM. Since IT can be picked with respect to the business’ goal of the customer, it was taken down into wide brands with 33 unique VM types by AWS.
- Spot Price (SP) is showing the current market price for each IT and Availability Zone (AZ).
- AZ is consisting of 22 unique zones in different countries across the world.

### 3.1 Requirements for Implementation

With the help of Python language, we can implement machine learning concepts. The latest version of Scikit-learn should be installed by the users. Users can make use of either pip or conda manager. Before implementing random forest and other machine learning algorithms, the following libraries should be installed:

1. **Pandas:** This library (Iosup et al., 2011) is used for testing the datasets. Pandas is a software library written for the Python programming language, for data manipulation and analysis.
2. **Matplotlib:** Matplotlib (Javadi et al., 2013) is updated to plot the input data dataset and output. For plotting graphs and charts, this is the particular library of Python.
3. **Keras:** To develop models of in-depth learning, Keras provide high-level building blocks. It relies on a well-optimized tensor manipulation. There are currently two backend implementations of Keras, they are:
   a. TensorFlow
   b. Theano
4. **NumPy:** It is the main program using Python for scientific computing.
5. **Random Forest:** A Random Forest is an ensemble method that performs regression and classification responsibilities with the use of more than one decision tree and Bootstrap Aggregation. Random Forest method including the bagging, comprises schooling every selection tree on a different statistics pattern in which sampling is done with replacement. The basic idea
behind this is to combine multiple decision trees rather than depend on individual decision trees to determine the final production.

Next, we move the data set’s features (X) and variable values based on int(y) to the method developed for the model of random forest regression. Using the sklearn library’s grid search cross-validation method to determine the ideal values from a specified range of values to be used for the hyperparameters of our model.

6. **Boto**: Boto is the IDE for Python on the Amazon Web Services (AWS). It lets Python developers build, configure, and control AWS resources like EC2 and S3. Boto includes an object-oriented, easy-to-use API and low-level access to the AWS tools.

Algorithm for Spot Price Prediction

Dataset: \( D = \{d_0, d_1, d_2, \ldots, d_l\} \)
Input 1 (Training Set): \( T = \{X_0, X_1, X_2, \ldots, X_i\} \)
Input 2 (Test Set): \( S = \{S_i, S_{i+1}, S_{i+2}, \ldots, S_l\} \)
Output: \( P = \{P_i, P_{i+1}, P_{i+2}, \ldots, P_l\} \)

Step 1: Build a random seed to ensure the reproducibility of the results.
Step 2: Using Scaler to normalize the dataset.
Step 3: Label the dataset, accordingly, perform Random Forest Regressor to the dataset.

**Approach:**
- Select data points from the training set at random K.
- Create the decision tree connected to those K data points.
- Choose the Ntree number of trees you wish to build and run the epochs.
- Have each of the Ntree trees forecast the value of \( Y \) for the data point for a new data point and give the new data point the sum over all the \( Y \) values expected.

Step 4: Calculate the accuracy and the error for the following algorithms, i.e. the prediction values (Figure 2).

Where \( A \) is the actual value, and where \( F \) is a value forecast. The difference between \( A \) and \( F \) is again broken down by the actual value \( A \). For each expected point in time, the absolute value in this equation is summed up and divided by the number of installed points \( n \). The multiplication by 100 makes it an error in measure.

Step 5: Analyze the data and plot the graphs for better understanding.
3.2 Input Dataset
Spot prices for 90 days are contained in a CSV file, this CSV file is the input dataset for the program code. Running the AWS CLI command on the console we obtain this CSV file which is converted from a json file. There are following steps involved in fetching the history of spot prices from AWS:

1. Provide all the necessary details to create an account on AWS, the details include details on credit card as well.
2. Go to IAM (Identity and Access Management Console), once the AWS account is activated.
3. Go to the “User” tab on the left side of the IAM Console to build a new account.
4. Users need some policies to be attached while creating a new user. Access issues are often faced when the user is not made the admin. Attach a policy named “Administrator Access” to the user.
5. Upon effective development of the account, a different access Id and a hidden key are given per AWS user. This is something the user has to remember for future references.

As the configuration settings are achieved, the installation of AWS CLI is the next step for fetching a history of spot prices. The prerequisite for running AWS CLI is an installation of Python. AWS tool is used to manage AWS services, it can be used to import, customize and monitor via command line. To automate scripts can be used. For transferring files to and from Amazon S3, AWS CLI provides simple commands. To fetch the data for the history of spot price into a json file format a command needs to run on the CLI, a sample command as below:

```
aws ec2 describe-spot-price-history --instance-types m1.large --start-time 2019-12-15T04:04:09 --end-time 2020-01-015T05:05:10
```

3.3 Instances Used for Testing
In 16 regions (6 in North America, 3 in European Union, 6 in Asia and 1 in South America) Amazon offers 64 types of spot instances. Representing the physical locations each region contains more than one availability zone.

Details of instances used for prediction is shown in Table 1.

Table 1. Instances used for Prediction

| Instance ID | Instance Type  | Availability Zone |
|-------------|----------------|-------------------|
| Ins1        | m5.2xlarge     | ca-central-1      |
| Ins2        | p2.8xlarge     | eu-west-1         |
| Ins3        | m5.xlarge      | eu-west-2         |
| Ins4        | m4.xlarge      | eu-west-1         |
| Ins5        | c4.4xlarge     | us-west-2         |
| Ins6        | c4.xlarge      | eu-west-1         |
| Ins7        | c5.4xlarge     | ca-central-1      |

4. DISCUSSION AND EXPERIMENTAL RESULTS
The code for the framework is implemented in Python. With 75% of training data and 25% of test data the algorithm performs over the model. The algorithm has been tested for 2 different zones of availability in different instances. Percentage of errors and precision are listed in Table 2.

Prediction graphs are shown in Figures 3-9. X-axis reflects the days, and spot rates are measured by the Y-axis.
5. CONCLUSION

It is possible to help customers by offering the correct price range according to their demands, time of day and machine type. This information will help customers to give the best possible price and decrease their cost of cloud services usage. In future work, at the end of this process, an emergence rule can be revealed by using rule induction.
A method for predicting spot prices is proposed using the spot prices provided by Amazon during the past 90 days. Although there have been many approaches used in recent years, Random Forest Regressor and few other machine learning techniques are used in the proposed method to assess predictability. The approach provides various implementations for instance analyzes regarding console, time and region.
All the prices predicted with respect to their time zones are authentic and dynamically calculated. The graphs above are the prediction graphs between days and spot rates. Spot rates are something that are fetched using the AWS CLI script integrated to run our machine learning algorithm. Interval is 10 days on the X-axis in each region, and depending upon the size of the instance the spot prices are plotted on the Y-axis. Curves on the graphs can be seen escalating initially for few regions and going an average constant, whereas for some it keeps increasing as the days increase. Three colored lines on the graphs are 3 different prediction results based on the algorithm used in the model training. Yellow or the orange line is taken to be the average case prediction for our model.

Results can be validated from calculations mapped from the AWS instance report found on the documentation page of AWS. We conclude the training by displaying all the data using graphs, making sure the results match with research conducted and reports online. More regions and sizes can be calculated using the same approach for regions across different continents and for multiple volumes. As AWS keeps coming with various schemes and spot prices would differ in the future trials. This particular techniques helps in complete understanding of the relationship between time and region.
REFERENCES

Angiuoli, White, & Mataalka, White, & Fricke. (2011). Resources and costs for microbial sequence analysis evaluated using virtual machines and cloud computing. *PloS One*, 6(10).

Ben-Yehuda, Ben-Yehuda, Schuster, & Tsafrir. (2013). Deconstructing amazon ec2 spot instance pricing. *ACM Trans. Econ. Comput.*, 1(3), 16:1–16:20.

Chaisiri, S., Kaewpuang, R., Lee, B. S., & Niyato, D. (2011). Cost minimization for provisioning virtual servers in amazon elastic compute cloud. *2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*.

Dawoud, W., Takouna, I., & Meinel, C. (2012). Increasing spot instances reliability using dynamic scalability. *2012 IEEE Fifth International Conference on Cloud Computing*. doi:10.1109/CLOUD.2012.58

El-Khamra, Y., Kim, H., Jha, S., & Parashar, M. (2010). Exploring the performance fluctuations of hpc workloads on clouds. *2010 IEEE Second International Conference on Cloud Computing Technology and Science*. doi:10.1109/CloudCom.2010.84

Iosup, S., Ostermann, S., Yigitbasi, M. N., Prodan, R., Fahringer, T., & Epema, D. H. J. (2011, June). Performance analysis of cloud computing services for many-tasks scientific computing. *IEEE Transactions on Parallel and Distributed Systems*, 22(6), 931–945. doi:10.1109/TPDS.2011.66

Jangjaimon, I., & Tzeng, N. F. (2015, February). Effective cost reduction for elastic clouds under spot instance pricing through adaptive checkpointing. *IEEE Transactions on Computers*, 64(2), 396–409. doi:10.1109/TC.2013.225

Javadi, B., Thulasiram, R. K., & Buyya, R. (2013, June). Characterizing spot price dynamics in public cloud environments. *Future Generation Computer Systems*, 29(4), 988–999. doi:10.1016/j.future.2012.06.012

Kaminski, B., & Szufel, P. (2015). On optimization of simulation execution on amazon EC2 spot market. *Simulation Modelling Practice and Theory*, 58, 172–187. doi:10.1016/j.simpat.2015.05.008

Kushwaha, V., & Simmhan, Y. (2014). Cloudy with a spot of opportunity: Analysis of spot-priced vms for practical job scheduling. *2014 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*. doi:10.1109/CCEM.2014.7015488

Lancon, Kunwar, Stroud, McGee, & Slater. (2019). AWS EC2 Instance Spot Price Forecasting Using LSTM Networks. *SMU Data Science Review*, 2(2).

Mattess, M., Vecchiola, C., & Buyya, R. (2010). Managing peak loads by leasing cloud infrastructure services from a spot market. *2010 IEEE 12th International Conference on High Performance Computing and Communications (HPCC)*.

Mazrekaj, A., Shabani, I., & Sejdiu, B. (2016). Pricing schemes in cloud computing: An overview. *International Journal of Advanced Computer Science and Applications*, 7(2), 80–86. doi:10.14569/IJACSA.2016.070211

Mihaiescu & Teo. (2010). Dynamic resource pricing on federated clouds. 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing.

Mohan Murthy, M. K., Sanjay, H. A., & Ashwini, J. P. (2012). Pricing models and pricing schemes of iaas providers: A comparison study. In *Proceedings of the International Conference on Advances in Computing, Communications and Informatics, ICACCI ’12*. New York, NY: ACM. doi:10.1145/2345396.2345421

Ostermann, S., & Prodan, R. (2012). Impact of variable priced cloud resources on scientific workflow scheduling. In *Proceedings of the 18th International Conference on Parallel Processing, Euro-Par’12*, Berlin: Springer-Verlag. doi:10.1007/978-3-642-32820-6_35

Pandey & Upadhyay. (2016). Different price forecasting techniques and their application in deregulated electricity market: A comprehensive study. *2016 International Conference on Emerging Trends in Electrical Electronics Sustainable Energy Systems (ICETESES)*.

Singh, V. K., & Dutta, K. (2015). Dynamic price prediction for amazon spot instances. In *Proceedings of the 48th Hawaii International Conference on System Sciences, HICSS ’15*, Washington, DC: IEEE Computer Society. doi:10.1109/HICSS.2015.184
Tang, S. J., Yuan, J., & Li, X.-Y. (2012). Towards optimal bidding strategy for Amazon EC2 cloud spot instance. In Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing, CLOUD '12. Washington, DC: IEEE Computer Society. doi:10.1109/CLOUD.2012.134

Turchenko, V., Shults, V., Turchenko, I., & Richard, M. (2014). Spot price prediction for cloud computing using neural networks. International Journal of Computing, 12(4), 348–359.

Vecchiola, C., Pandey, S., & Buyya, R. (2009). High Performance cloud computing: A view of scientific applications. 2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks.

Yi, S., Andrzejak, A., & Kondo, D. (2012). Monetary cost aware checkpointing and migration on Amazon cloud spot instances. IEEE Transactions on Services Computing, 5(4), 512–524.

Yi, S., Kondo, D., & Andrzejak, A. (2017). Reducing costs of spot instances via checkpointing in the Amazon Elastic Compute Cloud. 2010 IEEE 3rd International Conference on Cloud Computing.