Time-Series Analysis for Price Prediction of Opportunistic Cloud Computing Resources

Sarah Alkharif , Kyungyong Lee, and Hyeokman Kim

Abstract Cloud computing resources are offered in various forms, and surplus of computing resources are provided at cheaper price. A leading cloud computing vendor, Amazon Web Services, provides such opportunistic resources as EC2 spot instance whose price changes dynamically based on the resource demand from users. We analyze the spot instance price logs and apply various predictive analysis algorithms to better predict future spot instance price. By applying various train dataset modeling heuristics, we uncover that the SARIMA algorithm achieves the best prediction accuracy in spot price prediction; it shows 17% more accuracy than other algorithms that are widely used for spot instance applications. By applying contributions in this paper, we expect that spot instance users can decrease monetary cost while improving system stability.

1 Introduction

Cloud computing provides various types of compute resources to serve diverse application scenarios. The cloud computing frees the burden of system administration overheads without incurring prohibitive initial hardware purchase cost. From the service provider's perspective, fully utilizing the already established hardware resources and services is crucial to maximize monetary gain. As the users' resource demand can vary from time to time, some cloud computing providers offer services at cheaper price than the regular price to maximize hardware/service utilization. For instance, Amazon Web Services (AWS), a leading cloud computing vendor, pro-

Sarah Alkharif
Kookmin University, Seoul. South Korea, e-mail: saraalkharif@kookmin.ac.kr

Kyungyong Lee
Kookmin University, Seoul. South Korea, e-mail: leeky@kookmin.ac.kr

Hyeokman Kim
Kookmin University, Seoul. South Korea, e-mail: hmkim@kookmin.ac.kr

vides its surplus of EC2 computing resources at a cheaper price in the form of *spot instance*. A user who wants to use a spot instance bids for a price that one is willing to pay, and if the bid price is higher than the spot price that is decided by the service provider, one can get the resource allocated and pays for the spot price in the hourly basis. Other than AWS, Google Cloud Engine provides such opportunistic resources in the form of *preemptive instances*, and Microsoft Azure provides its excess compute capacity as *low-priority VM*.

Though users can utilize the opportunistic resources at a cheaper price, sudden service termination can happen at anytime as the demand for the computing resource changes. To mitigate the chance of sudden service interruption, few works were conducted to better predict and model the price change of EC2 spot instance in literature. Ben-Yehuda et al. [1] and Zhao et al. [10] tried to predict the future spot instance price using various predictive analysis algorithms, but they all concluded that the spot price is rather random and hard to make meaningful prediction for future price changes. Since then, most of studies that are related to spot instance focus on the handling sudden service interruption [9, 2, 6, 5] or spot instance bid strategy [7, 11].

In this paper, we apply various time-series analysis algorithms to predict price change pattern of AWS EC2 spot instances. By carefully designing the period of train datasets, we could uncover that applying seasonal-arima (SARIMA) can improve the accuracy of price change prediction error by 17% on average comparing to the naive method that references the most recent price to predict future price [5]. In addition to the contribution of improved price prediction accuracy, we could also discover that the configuration values to get the best prediction accuracy differs significantly across different availability zones and instance types. Based on the extensive experiments and promising results, we bring up an opportunity of improving spot instance prediction accuracy that can result in significant cost gain for cloud computing users with increased system stability.

2 Time-Series Analysis for AWS EC2 Spot Instance Price

In this work, we use various time-series analysis algorithms to predict future spot instance price. First, we apply a simple *mean* method that uses the average of previous price to make prediction. In the *mean* method, assuming the price of instance in time $t-1$ is x_{t-1}, to predict the price of an instance at time t, we use previous instance price until lag that is notated as n (Equation 1).

$$\hat{y_t} = \frac{x_{t-1} + x_{t-2} + ... + x_{t-n}}{n} \tag{1}$$

Naive method is another simple model that references only the most recent spot instance price to make prediction for future time windows 2. For naive and mean methods, the expected value can be used to make prediction for longer period of time later than t.

$$\hat{y_t} = x_{t-1} \tag{2}$$

Seasonal naive method is similar to naive method that uses the most recent observation, but it utilizes seasonal information in making prediction. Assuming the seasonal period is s, the predicted value at time t is the observed value at time $t-s$ [3].

$$\hat{y_t} = x_{t-s} \tag{3}$$

Auto Regressive Integrated Moving Average (ARIMA) is a popular statistical model and widely applied for time series datasets. The algorithm is a combination of auto-regressive (Equation 4) and moving-average model (Equation 5), with three parameters (p,d,q). p is the number of auto-regressive terms that indicates dependencies on past values, d is the degree of differencing to make input dataset stationary, and q is the number of lagged forecast errors in the prediction equation that depends only on random error terms.

$$\hat{y_t} = w_0 + \beta_1 y_{t-1} + \beta_2 y_{t-2} + ...\beta_n y_{t-n} + \varepsilon_t \tag{4}$$

$$\hat{y_t} = w_0 + \varepsilon_t + \delta_1 \varepsilon_{t-1} + \delta_2 \varepsilon_{t-2} + ... + \delta_n \varepsilon_{t-n} \tag{5}$$

If we set $d = n$, the equation becomes

$$\hat{y_t} = w_0 + \beta_1 y_{t-1} + \beta_2 y_{t-2} + ...\beta_n y_{t-n} + \delta_1 \varepsilon_{t-1} + \delta_2 \varepsilon_{t-2} + ... + \delta_n \varepsilon_{t-n} \tag{6}$$

The term β_i is the weight that is applied to prior values in the time series, δ_i is auto-correlation coefficients at lags, and ε_i is the residual error term.

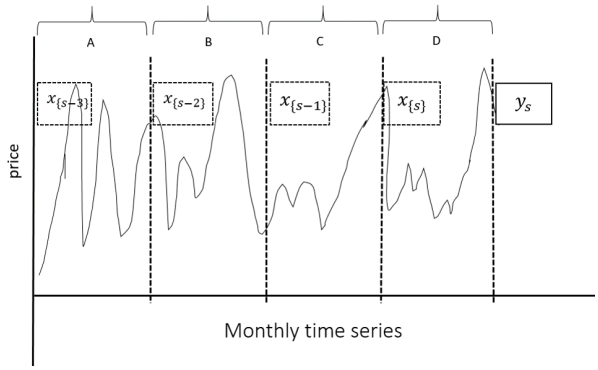

Fig. 1 SARIMA example

SARIMA stands for Seasonal ARIMA that allows ARIMA model to take into account seasonal characteristics that some seasonal patterns repeat along many periods in data sets. For example, in Figure 1, taking historical time series data X for duration of one year, we will predict y_s with a seasonal period, $s = 4$, then the model splits the length of time series by the number of seasonal period. We get 4 periods,

and each period contains 3 months of data. In the figure, assuming (A) as 4th lag, (B) 3rd lag, (C) 2nd lag, and (D) as the first lag, the seasonal lag time for X would be $(x_0, x_{s-1}, x_{s-2}, x_{s-3})$ and the number of periods will be chosen based on number of lags given by AR (Equation 4) and MA (Equation 5). If we set AR and MR coefficient as 2 then the AR would be β_s and β_{s-1} and the MA δ_s and δ_{s-1}.

Prophet model is developed by Facebook's Core Data Science team [8]. Prophet model forecasts time series dataset based on an additive and non-linear model fit with seasonality and holidays. The prophet algorithm predicts y_t by computing growth, seasonality, and holidays:

$$\hat{y_t} = g_t + s_t + h_t + \varepsilon_t \tag{7}$$

The term g_t is the growth function to compute how the series has grown and the expected values for continuing growth, and s_t seasonality change based on series behaviors, h_t the effects of holidays, ε_t the error term.

3 Evaluation

To evaluate the effectiveness of applying various algorithms to predict spot instance price, we fetch 11 months (from March. 2016 to Feb. 2017) of spot price log files from the AWS public API service. From the log files, we extracted the timestamp, spot price, availability zone, and instance type. The price of each instance is stored hourly-basis using *time-series* data format in R with the period of 24 hours. The on-demand instance price is different for different instance types in different regions, we normalize the spot instance price to that of on-demand instance. The normalized value indicates the cost gain that one can expect while using spot instances; smaller value indicates more gain.

At the time of writing, there are over 60s of AWS EC2 instance types that are served in over 30 availability zones. It becomes prohibitive to present the experiment results from all instance types, and we select representative instances in General, GPU, Compute-, Memory-, and Storage-optimized types that are m4.2xlarge, g2.2xlarge, c3.2xlarge, r3.2xlarge, and i2.2xlarge, respectively. The instances are not served in all availability zones, and we choose 18 zones that provide the aforementioned instance types for experiments.

We evaluate naive, seasonal naive, mean, seasonal ARIMA, linear regression, and Prophet algorithms using packages in R 3.2.4. Among them, linear regression and Prophet always perform worse than Arima, and we do not show the result from them. Different algorithms have distinct heuristics to choose the train dataset window. Naive and seasonal naive methods simply reference values from the previous observations. For mean method, we use previous 1, 3, 7, 15, 30, 60, 90, and 120 days of prices to get mean value for prediction. However, using only the most recent data (1 day) shows the best result, and we exclude other results. For seasonal Arima, we differentiate the training dataset period as 30, 60, 90, 120, and 150 days. In the

prediction step, we use a model built by *auto.arima* method of R. After building a model, we use the model for the next 1, 4, 8, 15, and 30 days. Overall, seasonal Arima has two configurations in the modeling data, *previous days used in modeling, the number of days for a model to be used*, and we notate the value using parenthesis. For all algorithms, we predict the normalized spot instance price for the next 24 hours and calculate root-mean-squared-error to evaluate each model.

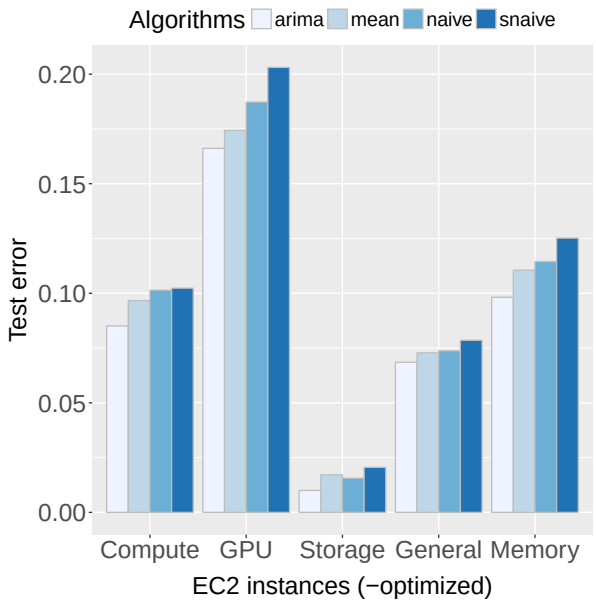

Fig. 2 Test error rates of different predictive algorithms. Regardless of the instance types, Arima shows the least error rate (lower is better).

Figure 2 shows the test error of different algorithms. For seasonal Arima and mean, we select the best performing configuration values. Each algorithm is executed in all 18 availability zones, and the mean test error is presented. Regardless of instance types, Arima algorithm shows the best prediction accuracy among other methods. Previous works on predicting spot instance price insisted that using predictive analysis algorithms did not help to improve the prediction accuracy. With that, most systems using spot instances usually rely only on the very recent price (naive method) only. However, with thorough experiments and train data configuration, we could uncover the effectiveness of using Arima model to predict spot instance price for the first time. In the figure, we can observe that different instance types show different test error rate, and we expect the different hardware specifications, such as GPU cards, can result in distinct supply and demand pattern.

From Figure 2, we observed that the Arima algorithm shows the best performance. As noted earlier, we use various combinations of modeling data period and

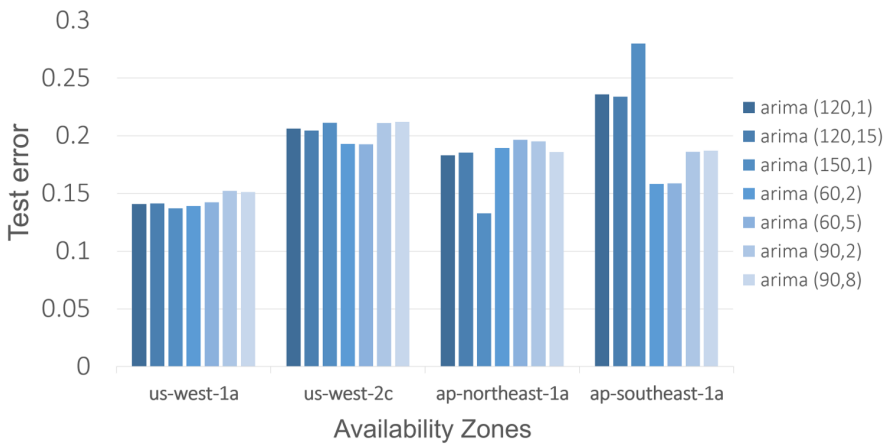

Fig. 3 The impact of train data configurations to the overall test error rate of GPU instance (lower is better).

model use days. To see the effect from the different parameters, we show the test errors of GPU instances for different train data configurations in Figure 3. A group of bars in the left two clusters show the test error value of us-west-1a and us-west-2c that show the least impact from the distinct parameters. The right two bars show the test error of ap-northeast-1a and ap-southeast-1a that show the most impact from the parameters. Other availability zones that are not shown in the chart show the pattern in-between. In the legend, the first value separated by comma means the number of days used in the SARIMA modeling step, and the second value after the comma indicates the number of days a model is used after it is built. In ap-southeast-1a zone, the worst configuration shows 50% more error rate than the best configuration. Furthermore, the worst configuration in ap-southeast-1a (150, 1) is the best configuration for ap-northeast-1a. We suspect that such diversity was not considered in the previous works that try to predict spot instance price, and they could not eventually find a model to make better prediction.

To check if there is optimal Arima train data configuration, we list the best performing train data parameters in Table 1. From the table, we can see that there is no globally optimal configurations. Contrary to general belief, building a model with train dataset with shorter time window sometimes performs better than longer train dataset. Furthermore, using a model longer period of time occasionally perform better. From the table, we conclude that predicting spot instance price needs careful consideration in building train dataset, and the configuration needs to be dynamically updated.

Availability Zones	General	Compute	Memory	Storage
ap-northeast-1a	(60,4)	(60,4)	(60,4)	(120,15)
ap-northeast-1c	(60,2)	(60,2)	(120,15)	(120,15)
ap-southeast-1a	(150,1)	(150,1)	(150,1)	(150,1)
ap-southeast-1b	(150,1)	(60,2)	(150,1)	(150,1)
ap-southeast-2a	(60,4)	(60,4)	(150,1)	(150,1)
ap-southeast-2b	(60,4)	(60,2)	(150,1)	(120,15)
eu-west-1a	(60,4)	(90,8)	(30,2)	(150,1)
eu-west-1b	(60,4)	(60,2)	(30,1)	(120,1)
eu-west-1c	(60,2)	(60,4)	(30,1)	(150,1)
us-east-1a	(50,1)	(150,1)	(30,1)	(60,2)
us-east-1c	(150,1)	(150,1)	(30,1)	(150,1)
us-east-1d	(60,2)	(150,1)	(30,2)	(150,1)
us-east-1e	(60,4)	(150,1)	(30,1)	(60,4)
us-west-1a	(60,2)	(150,1)	(30,1)	(120,15)
us-west-1b	(60,4)	(150,1)	(30,1)	(120,15)
us-west-2a	(60,2)	(150,1)	(150,1)	(150,1)
us-west-2b	(60,4)	(150,1)	(30,1)	(150,1)
us-west-2c	(60,2)	(60,2)	(60,2)	(150,1)

Table 1 Best Arima model configuration for different instance types in distinct availability zones

4 Discussion and Future Work

With thorough analysis about spot instance price prediction algorithms, we uncover the improved prediction accuracy as well as challenges in making better prediction. Based on the observation, we are going to further improve the algorithm in the following way.

The Good: Spot Price Change Prediction Most of previous works that tried to predict spot instance price concluded that the price is random, and applying predictive analysis algorithms does not really help to improve prediction quality [1, 10]. In this work, we applied multiple time-series analysis algorithms by carefully designing the period of modeling data and parameters. With extensive evaluation, we could uncover that applying predictive analysis algorithms improves the price prediction accuracy over 17% on average comparing to a method that uses only the most recent price [5, 7].

The Challenge: No Globally Optimal Model Despite of increasing prediction accuracy by applying various techniques, we could not find the globally optimal algorithm and training data specification for different availability zones and distinct instance types. It makes challenging to apply the algorithms for real applications that can be deployed in any environments.

The Promising: Applying Hybrid Models Even with the diversity of prediction accuracy for different algorithms and train data configuration, it is observed that the train error and test error show high correlation. Pearson product-moment correlation coefficient of train and test error is 0.904 - note that the coefficient has a value from -1.0 to 1.0, and the value of 1.0 means a perfect positive linear correlation, -1.0 means a negative correlation, while 0.0 means no correlation. We currently work on

referencing the train error to better choose the algorithms and train data period. We are going to apply the heuristic to an application that utilizes GPU-based AWS EC2 spot instances to execute deep learning tasks in a cost efficient way [5].

The Benefit: Lower Cost while Using Spot Instances With the improvement in the prediction accuracy, we expect it will result in the cost gain by cherry-picking few availability zones and instance types with lower prices. We are working on a theoretical model that specifies correlation between the prediction accuracy and the real cost gain. We are also working to utilize the predicted outcome to anticipate instances that are likely to incur unexpected service interruption that is the crucial factor of making users reluctant to use spot instances. In the prediction step, we try to anticipate the spot instance price of the next 24 hours. In an ideal case, if the task migration cost among different availability zones and instance types are negligible, we can issue more frequent migrations by relying on prediction module that has lower prediction error rate as the prediction window becomes shorter. By applying task migration heuristics that are proposed in literature [6, 5], we expect to decrease the prediction time window to increase the accuracy.

5 Related Work

Since the introduction of spot instance from AWS EC2 service, many attempts were made to predict price change pattern in the near future. After thorough investigation of spot instance price logs, Ben-Yehuda et al. [1] concluded that applying predictive analysis algorithms in the prediction of spot instance price is meaningless as it changes randomly. Javadi et al. [4] tries to apply statistical model to the price logs by applying MLE method. Though the approach helps to understand the spot price distribution, it does not help to predict future price. Similar to our work, Zhao et al. [10] applied ARIMA model to make spot instance price prediction, but they could not uncover the findings as we do in this paper. The authors performed experiments only for one instance type in a single region. As shown in this paper, the price change pattern differs significant among different environments, and we expect the authors missed the characteristics.

As it was widely known that the spot instance price is hard to be predicted, most of recent work focused on increasing stability of applications that run on spot instances. DeepSpotCloud [5] and Flint [6] proposed fast task migration mechanisms when a service interruption event happens. As the spot price prediction is challenging, both approaches used the naive method [11, 7], and we believe that the finding in this paper can significantly improve the cost gain and system stability of the previous approaches.

6 Conclusion

In this work, we try to predict future price for diverse instance types of AWS EC2 spot instances in 18 availability zones using various predictive analysis algorithms (naive, seasonal naive, mean, SARIMA, linear regression, and Prophet) to see if they can help to better predict the future spot instance price. Different from what is generally known, we uncover that SARIMA model performs better than simple methods. To the authors' best knowledge, this is the first work that uncovers applying predictive analysis helps to better predict the future spot instance price. To get better result, we need to tune the train dataset by differentiating the modeling period and model use time, and the tuning steps needs to optimized to further improve the accuracy.

Acknowledgements This work is supported by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIP) (No. NRF-2015R1A5A7037615 and NRF-2016R1C1B2015135), the ICT R&D program of IITP (2017-0-00396), the AWS Cloud Credits for Research program.

References

1. Agmon Ben-Yehuda, O., Ben-Yehuda, M., Schuster, A., Tsafrir, D.: Deconstructing amazon ec2 spot instance pricing. ACM Trans. Econ. Comput. **1**(3), 16:1–16:20 (2013). DOI 10.1145/2509413.2509416. URL http://doi.acm.org/10.1145/2509413.2509416
2. Gong, Y., He, B., Zhou, A.C.: Monetary cost optimizations for mpi-based hpc applications on amazon clouds: Checkpoints and replicated execution. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '15, pp. 32:1–32:12. ACM, New York, NY, USA (2015). DOI 10.1145/2807591.2807612. URL http://doi.acm.org/10.1145/2807591.2807612
3. Hyndman, R.J., Athanasopoulos, G.: Forecasting: principles and practice (2012). URL https://www.otexts.org/book/fpp
4. Javadi, B., Kondo, D., Vincent, J.M., Anderson, D.: Discovering statistical models of availability in large distributed systems: An empirical study of seti@home. Parallel and Distributed Systems, IEEE Transactions on **22**(11), 1896 –1903 (2011). DOI 10.1109/TPDS.2011.50
5. Lee, K., Son, M.: Deepspotcloud: Leveraging cross-region gpu spot instances for deep learning. In: 2017 IEEE 10th International Conference on Cloud Computing (CLOUD) (2017)
6. Sharma, P., Guo, T., He, X., Irwin, D., Shenoy, P.: Flint: Batch-interactive data-intensive processing on transient servers. In: Proceedings of the Eleventh European Conference on Computer Systems, EuroSys '16, pp. 6:1–6:15. ACM, New York, NY, USA (2016). DOI 10.1145/2901318.2901319
7. Sharma, P., Irwin, D., Shenoy, P.: How not to bid the cloud. In: 8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 16). USENIX Association, Denver, CO (2016)
8. Taylor, S.J., Benjamin, L.: Forecasting at scale. In: Facebook Technical Report (2017)
9. Yan, Y., Gao, Y., Chen, Y., Guo, Z., Chen, B., Moscibroda, T.: Tr-spark: Transient computing for big data analytics. In: Proceedings of the Seventh ACM Symposium on Cloud Computing, SoCC '16, pp. 484–496. ACM, New York, NY, USA (2016). DOI 10.1145/2987550.2987576. URL http://doi.acm.org/10.1145/2987550.2987576
10. Zhao, H., Pan, M., Liu, X., Li, X., Fang, Y.: Optimal resource rental planning for elastic applications in cloud market. In: Proceedings of the 2012 IEEE 26th Interna-

tional Parallel and Distributed Processing Symposium, IPDPS '12, pp. 808–819. IEEE Computer Society, Washington, DC, USA (2012). DOI 10.1109/IPDPS.2012.77. URL http://dx.doi.org/10.1109/IPDPS.2012.77

11. Zheng, L., Joe-Wong, C., Tan, C.W., Chiang, M., Wang, X.: How to bid the cloud. In: Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, SIGCOMM '15, pp. 71–84. ACM, New York, NY, USA (2015). DOI 10.1145/2785956.2787473. URL http://doi.acm.org/10.1145/2785956.2787473