

The Analysis of Time Series Forecasting on Resource Provision of Cloud-based Game Servers

Esma Mouine, Yan Liu, Jincheng Sun
Concordia University
Montreal, Qc, Canada
e_mouine@encs.concordia.ca
yan.liu@concordia.ca
s_jinche@encs.concordia.ca

Mathieu Nayrolles, Mahzad Kalantari
Ubisoft
Montreal, Qc, Canada
mathieu.nayrolles@ubisoft.com
mahzad.kalantari@ubisoft.com

Abstract—The server workloads of large-scale online video games are elastic and on-demand. The workload can range from tens to thousands of server instances in short periods. In fact, a cloud-based video game ecosystem can reach a workload of millions of players every week. Given such a large scale, even a small portion of over-provisioning leads to a significant amount of resource idling and a high cost of waste. It is essential to define an effective forecasting model on the game session workloads given a time span. The effectiveness shall be measured by metrics representing Service Level Objectives (SLOs). In this work, we analyze time series forecasting models using ARIMA, Prophet, and LSTM to predict the number of virtual machines in need for cloud resource monitoring data. In addition, we define service-level metrics for measuring effectiveness based on factors of over/under provision and ratio of resource waste. We analyze models with 16 fleets with an average of 2754 game servers over a five-month-long period of time in the production environment. We observe that our LSTM model is the most accurate in forecasting the demand of virtual machines in terms of RMSE and MAE. Further analysis using metrics of SLOs, we observe that the LSTM model leads to more cases of under-provisioning than ARIMA and Prophet do. The LSTM model forecasts the demand of virtual machines with less over-provision ratio than ARIMA and Prophet do for 14 out of 16 fleets. Using the LSTM model, we further evaluate the forecasting effect across different time spans of a single fleet and across multiple fleets within the same time span.

Index Terms—Forecasting Time Series, Resource Provision, Machine Learning, Deep Learning, Service Level Objective

I. INTRODUCTION

Online high-budget video games (known as AAA) are adopting a cloud-based video game ecosystem. Such systems experience workloads up to millions of players in a typical week that demand the game servers of different scales. They can range from tens to thousands of games in short periods.

Game servers from cloud providers have two layers of scheduling. The typical life cycle of a game server begins with provisioning a number of virtual machines (VMs) in a fleet. A fleet is a pool of VMs with the same configuration such as capacity, region and operating system. A VM is initialized by starting a certain number of identical processes as game servers. A game session is a game in progress with players. Once a game session is created, an available game server process is assigned to handle the game session. Each VM is

shared by multiple game servers. When the session ends, the game server is back to an idle state and waits for the next session until the VM is reclaimed.

One of the service-level objectives (SLOs) is to meet the demand of virtual machines. The number of virtual machines should be in minimal excess to the real demand to reduce the wasting costs of idling virtual machines on cloud. The cloud platform’s solution is auto-scaling. Hanieh Alipour et al., [1] present a survey that explores the definitions of related concepts of auto-scaling and the taxonomy of auto-scaling techniques. The auto-scaling mechanism ensures the provision of VMs in one of the three kinds, (1) rule-based solutions that specify configurations such as time or a threshold to trigger scaling; (2) reactive solutions that adjust the virtualized resources in response to a changing demand resulting in the changing virtualized resources utilization at the application service level; and (3) proactive solutions that forecast the near future working load at the application service level and adjust the virtualized resources in advance.

Time series modeling has been applied to implement auto-scaling mechanisms for applications that exhibit a type of temporal pattern. In the case of cloud provisioning, previous works have used time series to predict the number of resources necessary in the future according to different factors [2]–[5]. Regression-based methods have been applied in cloud resource modeling. Bankole and Ajila in [6] propose a prediction model for VMs needed based on CPU’s traces, memory, and network throughput, and SLO metrics including response time. The author compare several machine learning techniques, including Support Vector Machine (SVM), Neural Network (NN), and Linear Regression (LR). K-nearest Neighbors (KNN) algorithm-based regression is used in [7] to predict over-utilized and under-utilized Performance Metrics. Calheiros [8] deploy the Autoregressive Integrated Moving Average (ARIMA) model on web requests and network throughputs to predict future workloads. Markov chain model is proposed in [9] to autonomously predict and decide the allocation of CPU, Memory and bandwidth on host VMs and thus reduce the cost in cloud services. Hurst exponent, which is widely used in network data flow prediction, is utilized in [10] to measure the predictability of a VM. Filtering and signal processing

methods are also used to predict future workloads in VMs. Gong [11] used Fast Fourier Transform (FFT) to detect VMs with a similar pattern in frequency windows. Wu [12] used the Kalman filter to minimize the error of measurement and make the forecasting model robust to noises and fluctuations. More recently, recurrent neural network models with data correlation analysis have been developed and used to analyze multi-variables of cloud metrics [13], [14]. The above work mainly focuses on the workload and resource usage forecasting with less attention on the waste estimation. In the game session context, the forecasting of virtual machine demands should be under the constraints of minimizing both the over-provision and under-provision cases.

In our work, we define metrics directly related to service level objectives, including (1) the average number of virtual machine wasted during a period of time; (2) under provision ratio; and (3) over provision ratio. These metrics calculated from actual demands and predicted workload are used to evaluate fleets of game servers. We analyze three kinds of predictive models that can predict the number of virtual machines with a minimal excess of real-demands, Long Short Term Memory (LSTM) networks, a traditional forecasting model of ARIMA and an additive model Prophet [15] suitable for strong seasonal effects. The observation helps to discover a baseline model for forecasting the future amount of virtual machines while being resilient to unexpected events in an effort to reduce operational costs using cloud resources. Our forecasting problem becomes forecasting the workload every 10 minutes ahead from data of 30 minutes in the past. Therefore, we derive the following context-specific questions in the model analysis:

- **RQ 1:** What are the metrics of service level objectives to measure and observe the learning performance of a forecasting model in addition to RMSE and MAE?
- **RQ 2:** Can a forecasting model achieve at the same time both the accuracy of forecasting and service level objectives of minimal over-provision and under-provision cases?
- **RQ 3:** How transferable a model trained on data from one fleet can be generalized to other fleets and different time spans?

The contribution of this paper is three-fold:

- 1) We define metrics to measure over-provision and under-provision effects of a forecasting model in addition to general accuracy based evaluation;
- 2) We develop a baseline model using LSTM to analyse the forecasting accuracy across fleets and temporal spans;
- 3) We analyze the models using data from 16 fleets with an average of 2754 game servers over 5 months time spans in the production environment. Our LSTM model is the most accurate in forecasting demand of virtual machines in terms of RMSE and MAE. We observe the LSTM model leads to more cases of under-provisioning but fewer cases of over-provision than ARIMA and Prophet do.

II. RELATED WORK

Recent studies have been conducted regarding dynamic cloud resource provisioning. Various approaches and mechanisms for different purposes have been proposed in both academia and industry. They have already been deployed in saving cost in large-scale cloud services. Cloud services require sophisticated resource auto-scaling methods to operate under varying workloads. Auto-scaling approaches aim to acquire and release resources dynamically while maintaining an acceptable quality of service. Rule-based autoscaling are the most popular approaches offered by different platforms such as Amazon EC2 or Microsoft Azure. Conditions and rules in those approaches can be defined based on one or more performance metrics such as system level, CPU load, or service level metrics of the average response time or requests in each unit time. The advantage of the rule-based methods is the simplicity which makes them easy to use in cloud providers. However, the rules are predefined. When the service level objective is sensitive to avoid under-provision or to minimize over-provision, such a rule-based method is not adaptable to the varying demands on resources.

Dutreilh et al. [16] investigates horizontal auto-scaling using threshold-based and reinforcement learning techniques. In [17], the authors describe a lightweight approach that operates fine-grained scaling at the virtual machine level scaling in order to improve resource utilization while reducing cloud provider costs. Hasan et al. [18] extend the typical two threshold bound values and add two levels of threshold parameters in making scaling decisions. Chieu et al. [19] propose a simple strategy for dynamic scalability of web services based on the number of active sessions. The strategy scales the number of virtual machines if all instances have active sessions exceeding particular thresholds. Al-Sharif et al. [20] have a rule-based approach where they use Autonomic Computing components that include CPU, memory and bandwidth to improve the system's reliability, availability, and utilization level by scaling resources in response to changes in the cloud system states.

Other methods aim to use time series analysis to collect and study the past patterns of historical data to generate future prediction for the series. Forecasting models such as ARIMA focus on the direct prediction of future values. For example, Morais et al. [3] use time series based approaches to achieve auto-scaling services by proactive and reactive methods. Huang et al. [4] propose a prediction model (for CPU and memory utilization) based on double exponential smoothing to improve the forecasting accuracy for resource provision. Mi et al. [21] use Brown's quadratic exponential smoothing to predict the future application workloads alongside a genetic algorithm to find a near-optimal reconfiguration of virtual machines. Roy et al. [5] present a look-ahead resource allocation algorithm to minimize the resource provisioning costs while guaranteeing the quality of service in the context of auto-scaling elastic clouds.

Regarding machine learning approaches, Bankole and Ajila [6] propose a machine learning model to predict virtual ma-

chine resources based on CPU’s traces, memory, and network throughputs, and service-level agreement metrics including response time. Islam et al. [22] predicted resource demand based on neural network and linear regression. Toukir, et al. [23] predict workload tendency of a cloud platform based on time-delay neural network and regression methods. Google Autopilot [24] combines time series analysis and reinforce learning algorithms to scale the number of containers and associated CPU/RAM.

The above research works mainly focus on satisfying the demands using rules or accurate prediction. When the service level objectives have constraints on the occurrence of over-provision or under-provision, existing work shall be further extended to represent these constraints. In this paper, we discuss our systematic method of integrating metrics, model development process, and model evaluation to analyse the effectiveness of a forecasting model. An effective forecasting model is essential to automate the autoscaling operations.

III. THE RESOURCE FORECASTING METHOD

Game servers are stateful, long-lived and have two layers of scheduling. Game servers have run-time at the layer of Platform-as-a-Service (PaaS) on cloud. The run-time implements the life cycle of game servers and exposes a simple control on the underlying scaling model. We design the resource scaling workflow by adding in forecasting on-the-fly within a short time interval to fine-tune the rule-based scaling automation.

A. The Resource Scaling Workflow

The cycle of life of game servers is as follows: a virtual machine is requested from the cloud; the virtual machine starts a number of identical processes sharing the state of the virtual machine; and the virtual machine is reclaimed according to a de-provision policy. During this lifecycle, the initialization of a stateful server requires loading the initial state from the data source. Hence, the process of booting and the initialization of the game servers takes a certain amount of time to complete before being ready for handling players’ commands. Another level of complexity for long-lived stateful game servers is sharing their states across servers. Therefore scaling mechanisms for servers running stateless functions are not suitable to stateful game servers. To mitigate the delay of initialization, the current solution consists of initializing a certain number of virtual machines running idle processes. When a new game server process is required, one of the idle processes is selected. Figure 1 depicts the resource planning center sends scaling orders to a cloud manager in the Infrastructure-as-a-Service layer that distributes the players over the game fleets. Each game fleet is composed of several virtual machines hosting the game servers.

The current planning center uses rule-based service of AWS *GameLift*. *GameLift* tracks a fleet’s system level metrics and determines when to add or remove virtual machine instances based on a set of alerts and actions defined. For instance, if there are fewer than 10 available game server processes for 5

minutes, then add 10 cloud instances. Either whole number of instances or percentages of the current instances can be used to define the values in a condition or in an action. There are two methods of auto-scaling to choose:

- **Target-based scaling** – *GameLift* scales the fleet up or down as a certain percentage of current idling server processes. Given a fleet with thousands of game servers, a small difference of the percentage (such as one percent) may have large changes of the server amount (such as hundreds of game servers).
- **Rule-based scaling** – More fine-grained control of scaling actions is given. Each policy specifies an alert condition that triggers a scaling action. Servers within a fleet have unbalanced workloads. The control is performed at the fleet level as a cluster, but not fine-grained for each virtual machine.

Too small granularity of value may not be responsive to the increasing game server process demands that leads to under-provision and players’ unsatisfactory experience. On the other hand, too large granularity incurs extra costs of idling virtual machines. We extend the rule-based planning center with the forecasting of future virtual machine demands every 10 minutes ahead. This prediction value is then used by the planning center to configure the scaling policy. In our work, we aim to achieve two goals of forecasting : (1) the forecasting accuracy and (2) minimize over-provision. Any forecasting with residual to the real demand results in either over-provision or under-provision. The current business goal is to reduce the idling server cost that means to minimize over-provision cases.

B. The Dataset

The historical data for forecasting is retrieved from AWS *CloudWatch* logs that contains run-time metrics of fleets, including the number of active fleets during their life cycle every minute; the duration of activity for each fleet; the number of active virtual machines of each fleet, the total number of active game servers and free game servers. A fleet’s lifetime is up-to two months. Fleets may not have overlapping on the timeline. The dataset used in this paper consists of 16 fleets in total in two time spans with 10 days overlapping. Eight fleets that are active from 2019-12-12 to 2020-02-16 and another eight fleets that are active from 2020-02-06 to 2020-03-02. Figure 2 plots the statistics.

First, we test the stationarity of the time series before evaluating the forecasting models. We perform Augmented Dickey-Fuller test (ADF) [25], a kind of unit root test that determines how strongly a time series is defined by a trend. The null hypothesis of the test is that the time series can be represented by a unit root, that it is not stationary. The alternate hypothesis is that the time series is stationary. We interpret these results using the p-value from the test. A p-value below the threshold 0.5 suggests we reject the null hypothesis and the time series is stationary, otherwise, a p-value above the threshold suggests we accept the null hypothesis with a unit root that indicates non-stationary. Figure 3 plots the With the number of active game servers of *Fleet-1*. The ADF test

Fig. 1. The Resource Provision Workflow for Stateful Game Servers

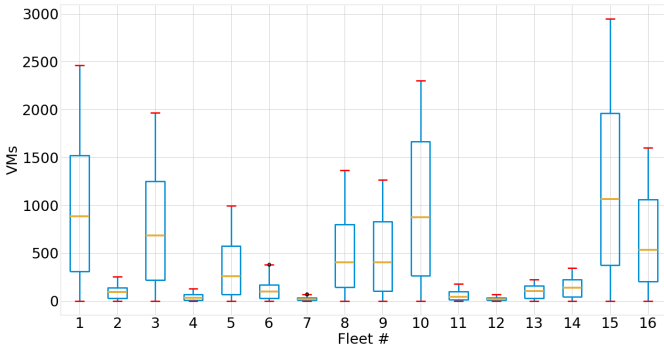
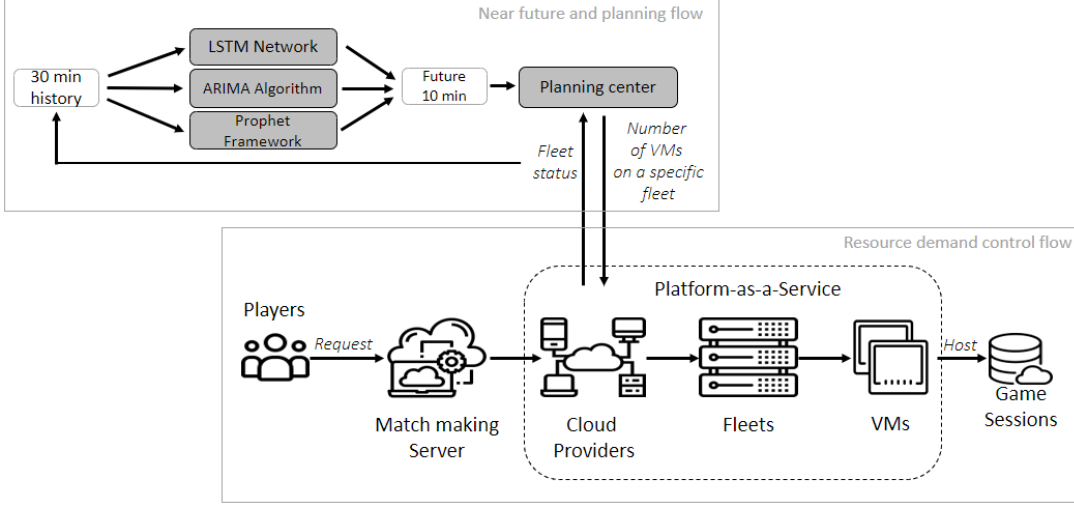


Fig. 2. The aggregation of active numbers of virtual machines on 16 fleets

implies that the series of *Fleet-1* is not stationary. Therefore we consider forecasting models of ARIMA (autoregressive model), Prophet (an additive model for non-linear trends), and the LSTM recurrent neural network model.

C. Metrics for Service Level Objectives

We define metrics for service level objectives to consider the cases of under-provision and over-provision for a forecasting result. The over and under-provisioning cases are denoted O and U , respectively. We define the service level objectives of two kinds. One kind measures the ratio of over-provision and under-provision. The other kind measures the average number of virtual machines wasted. The virtual machine demand produced from a rule-based system is used as benchmark to evaluate any forecasting models.

At a given time t , the number of wasted virtual machine is defined as $pro_t - dem_t$, the difference between pro_t the number of virtual machines that the resource planning system/framework provides at time t and dem_t the number of virtual machines actually needed to host all the game sessions at time t . Over a time span with N timestamps, the

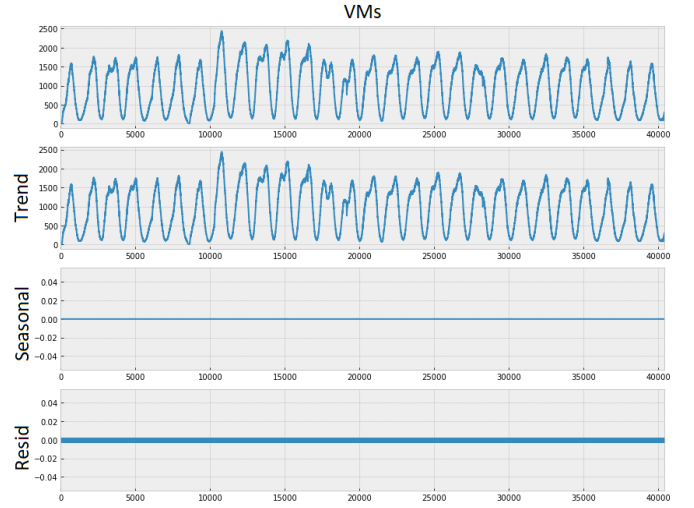


Fig. 3. Seasonal Decomposition of the game sessions of one of the fleets.

TABLE I
TERMS FOR DEFINING METRICS OF SLOS

Notation	Description
O	The set of over-provision cases
U	The set of under-provision cases
T	The time span of the time series
N	The number of data samples
pro_t	The number of virtual machines provisioned at time t
dem_t	The number of VMs demanded at time t
Wr_t	The ratio of wasted virtual machine at time t
Mr_t	The mis-forecasting ratio Mr_t at time t
Mr_o	The average over-forecasting ratio across all the time steps
Mr_u	The average under-forecasting ratio across all the time steps

average demand given all the over-provision cases is defined as $\frac{\sum_{t=1}^N dem_t}{N}$, $t \in O$. We then define the ratio of wasted virtual machine at time t as follows:

$$Wr_t = (pro_t - dem_t) \div \frac{\sum_{t=1}^N dem_t}{N}, \quad t \in O \quad (1)$$

We compute the percentile of $Wr_t, t \in O$ (1) over the whole time span of the dataset and take the 50_{th}, 90_{th}, 95_{th} and the 99_{th} percentile of Wr as the evaluation metrics.

The number of provisioned virtual machines is also the output from forecasting models. To evaluate the ratio of either under-forecasting or over-forecasting amount compared to forecasting result, we define the metrics of mis-forecasting ratio Mr_t at time t as follows:

$$Mr_t = \frac{pro_t - dem_t}{pro_t} \quad (2)$$

when $Mr_t > 0$, the ratio represents over-forecasting at time t and when $Mr_t < 0$, the ratio indicates under forecasting at time t . We define the average mis-forecasting ratio across all the time steps as follows:

$$Mr_o = \frac{\sum_{t \in O} Mr_t}{|O|}, (Mr_t > 0) \quad (3)$$

$$Mr_u = \frac{\sum_{t \in U} Mr_t}{|U|}, (Mr_t < 0) \quad (4)$$

We define the over-provision ratio (OPR) and under-provision ratio (UPR) to categorize the two cases. We have the *OPR* as the ratio of all the over-provision cases to N ; and *UPR* as the ratio of all the under-provision cases to N .

$$OPR = \frac{\sum_{t=1}^N op_t}{N} \times 100\%, \quad (5)$$

$$UPR = 1 - \frac{\sum_{t=1}^N op_t}{N} \times 100\%, \quad (6)$$

This flag is defined as a Boolean variable at time step t as:

$$op_t = \begin{cases} 0 & pro_t < dem_t \\ 1 & pro_t \geq dem_t \end{cases}, \quad (7)$$

IV. THE TIME SERIES FORECASTING MODELS

A time series consists of three components: the trend, the seasonality and the noise. The trend can be defined as the upward and downward movements of the data over a period of time. Seasonality is the repeating short-term cycle in the series. The noises are the spikes and decreases at random intervals. Our dataset is extracted from the logs of cloud computing. The intervals are not regular and the time spans of each fleet are not the same, thus forming irregular time series. Time series also add the complexity of a sequence dependence among the input variables. We consider the forecasting models of the autoregressive model of ARIMA, additive model for dataset with trends of Prophet, and non-linear neural network model of LSTM.

A. Time Series Modeling in ARIMA

Our ARIMA model performs univariate time series forecasting by using the previous values of active virtual machines to predict the future values. ARIMA is an acronym that stands for Auto-Regressive Integrated Moving Average. It is a class of models that captures a suite of different standard temporal structures in time series data. ARIMA's parameters are:

- p , the lag order;
- d , the degree of differencing;
- q , the order of moving average.

The fact is that our dataset contains observations taken at different times, which necessitates value analysis of the number of lag observations and specification of the size of the moving windows. In this paper, we setup the ARIMA model using the `auto.arima` function. The `auto.arima` returns the best ARIMA model according to either AIC (Akaike Information Criteria), AICc (Corrected AIC), or BIC (Bayesian Information Criteria) values. The function conducts a search over possible models within the order constraints provided. Using our data, `auto.arima` returned the parameters, $p = 1$, $d = 1$ and $q = 2$. We fit the ARIMA model with the parameter order (1,1,2).

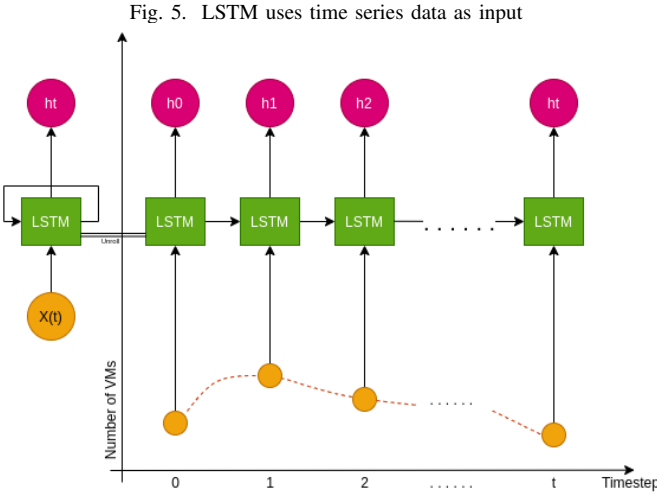
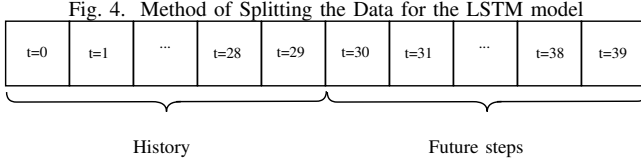
B. Time Series Modeling in Prophet

Prophet [15] was created by Facebook to capture trends and seasonality in time-series data. Prophet is able to capture daily, weekly and yearly seasonality along with holiday effects, by implementing additive regression models. The mathematical model of Prophet is $y(t) = g(t) + s(t) + h(t) + e(t)$. $g(t)$ represents the linear curve for modeling non-periodic changes in time series. $s(t)$ represents periodic changes in weekly, monthly or yearly). $h(t)$ represents the effects of holidays on irregular schedules. $e(t)$ is the error term accounts for any unusual changes not accommodated by the model. Prophet works best with time series that have strong seasonal effects and several seasons of historical data. By default, the Prophet model automatically computes the parameters according to the data given. A `Prophet()` object is defined to fit on the dataset by calling the `fit()` function and passing the input data. The fit function takes the time series data in a specific format. The first column of data contains the dates and times. The second column contains the sample observations or values. The forecasting is made by calling the `predict()` function by passing the dates and times for all the intervals that we aim to predict.

C. Long Short-Term Memory

We define a LSTM model of two layers. The first layer has a 100-dimensional hidden state followed by 200 hidden units for the second layer. These two layers are followed by a fully-connected layer to output the prediction of active virtual machines in the next 10 minutes. This 10-minute interval is set for the delay of virtual machine to start and receive new players. The model's input data has the shape [samples, timesteps]. We transform our dataset into $X = x^1, x^2, \dots, x^n$, where x^t is a vector of active virtual machines of the 30 previous time steps as shown in Figure 5.

To train the LSTM model, the data is split in a particular way to prepare the historical and future data. As shown in Figure 4, we divide the data into multiple input/output segments of which the current time step and the previous 29 time steps are used as inputs. Future 10 time steps are used as output of the LSTM model. We normalize the data in the range [0.01, 0.99] using a Min-Max scaler.



V. THE MODEL ANALYSIS

In this sections we perform different experiments to answer three research questions.

- RQ 1: What are the forecasting effects based on metrics of both the general accuracy measures and service level objectives?
- RQ 2: Is a model trained with data from one fleet transferable to forecast to other fleets that are active during the same time period ?
- RQ 3: Is a model trained with data from one fleet transferable to other fleets that are active during a different time period?

We analyze the models using two sets of metrics: error metrics for measuring the analysis accuracy and metrics defined in section III-C for measuring effects on service level objectives. Mean Square Error (RMSE) is calculated by taking the average of the square of the difference between the original

and predicted values of the data. Mean Absolute Error (MAE) is the average of all absolute errors. The MAE of a model refers to the mean of the absolute values of each prediction error on all values of the test data-set.

$$MSE = \frac{\sum_{t=1}^n (p_t - x_t)^2}{n} \quad (8)$$

$$MAE = \frac{1}{n} \sum_{t=1}^n |x_t - p_t| \quad (9)$$

A. Forecasting Effects (RQ.1)

In this experiment, we aim to observe the model's forecasting effects in both model accuracy and how they perform on over-provision and under-provision cases. We use the current rule-based active virtual machines as the baseline to compare the effects of forecasting models. Table II lists the RMSE and MAE for rule-based baseline and three forecasting model across 16 fleets. Each fleet is trained for one forecasting model, and such a model is applied to predict the virtual machine demand for the next time minutes. The experiment results show that the LSTM model produces the least errors for 15 fleets out of 16 fleets.

We further observe the effects of over-provision and under-provision. Table III lists each model's two metrics OPR and UPR across 16 fleets. The results show that the LSTM model outperforms other forecasting models with least over-provision ratio. Compared to the rule-based model, the ratio is reduced up to 90%. For the ratio of under-provision, the LSTM model is not the optimal model. The Prophet model has the least under-provision ratio for 8 fleets out of 16 fleets, and the ARIMA model obtains the least under-provision ratio for 5 out of 16 fleets. The LSTM model have higher under-provision ratios than the other two models. In terms of the amount of virtual machine under provisioned in ratio to the amount predicted by the model, Table tab:comp-missprovision shows the LSTM model has comparable ratios with other two forecasting models. For the case of over-provision, the LSTM has better results than other two models. This indicates the metrics UPR only measures the number of cases that are under provisioned. When consider with the metrics of Mru that measures the differences of quantity in mis-forecasting, we can observe that LSTM has the best forecasting errors with lowest over-provision occurrence and quantity over time. The quantity of virtual machine under-provisioned is comparable to other two models.

We further observe the quantity of over-provisioned virtual machine over time. This effect is measured by the ratio of wasted virtual machine Wr_t at each time step. Table V lists the 50th, 90th, 95th and 99th percentile of each forecasting model. Across 16 fleets, the LSTM model demonstrates the least wasting of the virtual machines over time steps. This indicates the LSTM has stable forecasting over time for the 30 minutes historical data in learning and 10 minutes data forecasting.

Summary. To answer the first research question, we observe the experiment results by viewing metrics for both accuracy

TABLE II
FORECASTING ERRORS FOR RULE-BASED AND THREE MODELS ACROSS 16 FLEETS

Fleet NO.	$RMSE_{Rule-based}$	$RMSE_{LSTM}$	$RMSE_{ARIMA}$	$RMSE_{Prophet}$	$MAE_{Rule-based}$	MAE_{LSTM}	MAE_{ARIMA}	$MAE_{Prophet}$
0	322.58	93.62	1221.39	328.28	290.86	66.01	1221.39	284.28
1	101.1	7.57	82.66	30.04	90.64	7.57	82.66	25.16
2	90.66	124.12	1028.68	253.97	261.10	124.12	1028.68	212.89
3	56.08	5.10	44.35	24.01	51.66	5.10	44.35	20.66
4	201.52	42.75	688.31	131.31	173.94	42.75	688.31	112.91
5	105.92	11.22	109.22	23.28	94.23	11.22	109.22	16.59
6	38.58	3.82	15.77	7.12	37.74	3.82	15.77	5.43
7	223.23	46.34	603.58	555.78	206.49	46.00	603.58	494.45
8	228.88	30.11	436.10	149.97	197.77	30.11	436.10	111.51
9	340.87	58.01	757.06	314.64	289.68	58.01	757.06	248.20
10	74.50	6.29	43.61	12.65	64.41	6.29	43.61	10.15
11	40.64	4.16	17.134	8.72	38.95	4.16	17.13	6.87
12	105.49	6.95	60.83	22.90	94.35	6.95	60.83	19.13
13	114.06	11.73	90.97	52.94	103.29	11.73	90.97	43.71
14	361.17	59.86	670.44	508.44	316.93	59.86	670.44	457.67
15	242.68	39.72	384.08	277.55	230.41	39.72	384.08	251.39

TABLE III
MODEL FORECASTING EFFECTS ON OVER-PROVISION AND UNDER-PROVISION ACROSS 16 FLEETS BY OVER/UNDER PROVISION RATIOS

Fleet NO.	$OPR_{Rule-based}$	OPR_{LSTM}	OPR_{ARIMA}	$OPR_{Prophet}$	UPR_{LSTM}	UPR_{ARIMA}	$UPR_{Prophet}$
0	31%	5.7%	1.3%	40%	75%	99%	25.4%
1	54.8%	4.7%	40.3%	35.6%	77%	84%	13.9%
2	33.9%	27%	56.39%	40%	56%	0%	23%
3	61.7%	18%	63.3%	51.06%	44%	21.26%	19.9%
4	42%	5.2%	65.97%	48.1%	83%	0.05%	29.1%
5	51.7%	20%	54.3%	26%	37%	10.97%	31.4%
6	69.9%	25%	58.11%	33.9%	45%	23.72%	26.4%
7	36.6%	6.3%	2.8%	55.4%	43%	99.99%	5.1%
8	40%	9.8%	61.83%	14.26%	39%	17.60%	71.15%
9	31%	3.5%	61.57%	30.3%	72%	28.41%	15.7%
10	59.4%	6.9%	52.62%	18.61%	64%	73.94%	66%
11	65.5%	16.3%	52.32%	32.7%	45.3%	22.75%	20.19%
12	55.7%	5.1%	64.49%	30.8%	62.5%	56.62%	12.36%
13	48%	5.3%	56.6%	31.6%	58%	30.38%	8.05%
14	28%	10.4%	58.9%	37.3%	59.5%	58.52%	0%
15	34.9%	6.47%	46.72%	37.5%	71.4%	66%	0%

TABLE IV
THE RATIO OF OVER/UNDER PROVISION AMOUNT COMPARED TO THE FORECASTING AMOUNT

Fleet NO.	Mr_o				Mr_u		
	RB	L	A	P	L	A	P
1	0.32	0.06	0.01	0.40	-0.08	-18.38	-0.11
2	0.55	0.05	0.40	0.36	-0.95	-7.46	-0.06
3	0.34	0.27	0.56	0.40	-0.11	-0.01	-0.08
4	0.62	0.18	0.63	0.51	-0.10	-0.17	-0.11
5	0.42	0.05	0.66	0.48	-0.24	-0.01	-0.17
6	0.52	0.20	0.54	0.26	-0.06	-0.10	-0.11
7	0.70	0.25	0.58	0.34	-0.17	-0.28	-0.47
8	0.37	0.06	0.03	0.55	-0.23	-6.74	-0.10
9	0.40	0.10	0.62	0.14	-0.15	-0.08	-2.15
10	0.31	0.04	0.62	0.30	-1.65	-0.10	-1.48
11	0.59	0.07	0.53	0.19	-3.00	-4.73	-1.15
12	0.66	0.17	0.52	0.33	-0.31	-0.21	-0.20
13	0.56	0.05	0.64	0.31	-1.13	-0.82	-0.05
14	0.48	0.05	0.57	0.32	-0.20	-0.22	-1.04
15	0.29	0.10	0.59	0.37	-0.05	-0.96	0.00
16	0.35	0.06	0.47	0.38	-0.07	-1.73	-0.27

and service level objectives. Our metrics provide indicators to observe the occurrence and quantity of over/under-provision. The LSTM model produce stable forecasting performance across fleets and over time. Thus the LSTM model is further evaluated for transferability of the model trained on one fleet and one time span to other fleets and other time spans.

B. Forecasting on different fleets (RQ2.)

For the forecasting, we use previous values to predict the future ones. In the case where a new fleet is active, there is not previous observations that can be used to predict the future ones. So we experiment using data from other fleets that are active during the same time period to predict the future values of other fleets. With this experiments, we aim to evaluate the following research question:

Research question 2: *Can a model trained with data from one fleet generalize to other fleets that are active during the same time period ?*

From the rule based system, we have data from different fleets. We use the fleets that are active between 2019-12-12 and 2020-02-06. WE train with fleet 0 and predict the other fleets from 1 to 7 to predict their future values using our LSTM model.

Table VI shows the result of the experiment. Compared to the rules-based system, the LSTM managed to predict the future values with a lower waste. However with a higher value of under-provisioning rate.

1) *Conclusion:* The result of the experiments show that it is possible to predict the future values of the fleet using a different fleet active during the same time period. The ability to generalize across fleets is critical in production. Multiple fleets are active during the same time period. A new active

TABLE V

PERCENTILE OF THE RATIO OF WASTED VIRTUAL MACHINE ALL THE TIME FOR RULE-BASED AND THREE FORECASTING MODELS ACROSS 16 FLEETS

Fleet NO. Model	Wr(50%)				Wr(90%)				Wr(95%)				Wr(99%)			
	RB	L	A	P	RB	L	A	P	RB	L	A	P	RB	L	A	P
0	0.27	0.01	0	0.33	0.58	0.03	0	0.55	0.69	0.04	0	0.60	0.79	0.07	0	0.70
1	0.89	0.03	0.05	0.27	1.51	0.08	0.11	0.53	1.55	0.10	0.12	0.59	1.63	0.15	0.14	0.70
2	0.32	0.01	1.44	0.31	0.62	0.03	2.16	0.49	0.72	0.04	2.18	0.52	0.85	0.09	2.20	0.59
3	0.97	0.04	1.32	0.49	1.73	0.12	1.77	0.88	1.84	0.16	1.81	0.94	3.21	0.24	1.84	1.04
4	0.44	0.02	2.29	0.32	0.79	0.05	2.89	0.54	0.82	0.06	2.91	0.63	0.90	0.11	2.92	0.76
5	1.04	0.03	1.13	0.13	1.24	0.08	1.79	0.26	1.28	0.11	1.81	0.29	1.38	0.19	1.84	0.46
6	1.92	0.09	1.07	0.25	2.29	0.28	1.50	0.62	2.39	0.36	1.56	0.76	2.75	0.55	1.61	1.05
7	0.45	0.01	0.02	0.98	0.57	0.04	0.02	1.69	0.63	0.05	0.02	1.81	0.75	0.08	0.02	1.91
8	0.55	0.02	1.43	0.11	0.69	0.05	1.95	0.37	0.73	0.07	1.97	1.21	1.75	0.28	1.99	1.89
9	0.28	0.03	1.18	0.28	0.50	0.08	1.59	0.56	0.59	0.10	1.61	0.63	0.67	0.14	1.63	0.75
10	0.86	0.09	0.11	0.11	1.48	0.23	0.19	0.34	1.59	0.27	0.21	0.46	2.84	0.37	0.23	0.59
11	1.56	0.10	0.86	0.29	1.89	0.29	1.43	0.55	2.59	0.35	1.47	0.61	3.37	0.52	1.56	1.09
12	1.26	0.05	0.62	0.19	1.47	0.13	0.79	0.38	1.52	0.15	0.81	0.45	1.63	0.20	0.85	0.58
13	0.93	0.05	0.93	0.35	1.09	0.13	1.35	0.55	1.12	0.15	1.37	0.64	1.50	0.23	1.40	0.82
14	0.25	0.03	0.53	0.41	0.49	0.05	0.67	0.68	0.56	0.05	0.68	0.79	0.65	0.07	0.77	1.00
15	0.42	0.01	0.27	0.42	0.51	0.04	0.38	0.69	0.55	0.05	0.39	0.79	0.71	0.08	0.51	0.95

TABLE VI

PREDICTION RESULTS FROM TRAINING AND TESTING FLEETS THAT ARE ACTIVE ON THE SAME TIME PERIOD. FLEET 0 IS USED TO TRAIN THE MODEL AND THE FLEETS FROM 1 TO 7 ARE USED TO TEST IT. FLEET 0 TO 7 ARE ACTIVE DURING THE PERIOD (2019-12-12 TO 2020-02-06). FOR THE RULE-BASED THE UNDER-PROVISIONING IS ALWAYS 0.

Fleet NO.	$OPR_{Rule-based}$	OPR_{LSTM}	UPR_{LSTM}
1	54.8%	12%	68.9%
2	33.9%	6.1%	74%
3	61.7%	16%	63%
4	42%	11%	66%
5	51.7%	10%	66%
6	69.9%	20%	55%
7	36.6%	5.9%	77%

fleets doesn't have previous values that can be used for future predictions. In the experiments where we train with fleet 0 to predict the future values of other fleets, the LSTM learned the behaviour of the fleets and predicted the new values of the other fleets with a waste lower than the rule-based system.

C. Forecasting on different time spans (RQ3)

The fleets are active during different time period. In this experiments we aim to evaluate the following research question:

Research question 3: *Can a model trained with data from one fleet generalize to other fleets that are active during a different time period ?*

We train our model with fleet 0 that is active during the period of time from 2019-12-12 to 2020-02-06 and the fleets that we test on are active during the period of time from 2020-02-06 to 2020-03-02.

1) *Experiment Results:* The results are shown in the table VII. According the table, the LSTM waste is lower than the waste of the rule-based system. However, the under-provisioning ratio is still high compared to the rule-based system that is always 0.

2) *Conclusion:* The results of the experiments show that it is possible to predict the future values of the fleet using a different fleet from a different time period. It shows that the LSTM is able to learn and transfer the behaviour of the fleets

TABLE VII

PREDICTION RESULTS FROM TRAINING AND TESTING FLEETS THAT ARE ACTIVE ON A DIFFERENT PERIOD. FLEET 0 IS USED TO TRAIN THE MODELS AND THE FLEETS 8 TO 15 ARE USED TO TEST IT. FLEET 0 IS ACTIVE DURING THE PERIOD (2019-12-12 TO 2020-02-06) AND THE FLEETS 8 TO 15 ARE ACTIVE DURING THE TIME PERIOD (2020-02-06 TO 2020-03-02)

Fleet NO.	$OPR_{Rule-based}$	OPR_{LSTM}	UPR_{LSTM}
8	40%	12%	66%
9	31%	6.5%	76%
10	59.4%	16%	61.4%
11	65.5%	19%	59%
12	55.7%	14%	71%
13	48%	9.4%	72%
14	28%	6.3%	74.8%
15	34.9%	6.4%	76%

between time periods. We evaluate the transferability cross period of the LSTM by training with a fleet from a period and testing with seven fleets from a different period. The LSTM model predicts the future values with a waste lower than the rule-based system.

VI. CONCLUSION AND FUTURE WORK

This paper presents three different time series models for cloud-based game sessions, LSTM, ARIMA and Prophet. To evaluate the performance and the accuracy of these models we use different evaluation metrics and perform experiments to answer three research questions. We compare the models prediction to a rule-based auto-scaling system. We effectiveness of the model prediction by comparing the waste and under-provisioning ratio of the prediction of future values of 16 fleets. The results also showed that the LSTM outperformed the ARIMA and Prophet that shown to be unstable in predicting the fleets. It also minimized the waste according to future values by returning prediction more accurate to the actual demand. However, there is still the problem of under-provisioning that is necessary to meet the SLO agreement that is still unresolved.

VII. ACKNOWLEDGEMENT

This research is funded by Ubisoft La Forge and MITACS Accelerate program.

REFERENCES

- [1] H. Alipour, Y. Liu, and A. Hamou-Lhadj, "Analyzing auto-scaling issues in cloud environments," in *Proceedings of 24th Annual International Conference on Computer Science and Software Engineering*, ser. CASCON '14. USA: IBM Corp., 2014, p. 75–89.
- [2] V. R. Messias, J. C. Estrella, R. Ehlers, M. J. Santana, R. C. Santana, and S. Reiff-Marganic, "Combining time series prediction models using genetic algorithm to autoscaling web applications hosted in the cloud infrastructure," *Neural Computing and Applications*, vol. 27, no. 8, pp. 2383–2406, 2016.
- [3] F. J. A. Morais, F. V. Brasileiro, R. V. Lopes, R. A. Santos, W. Satterfield, and L. Rosa, "Autoflex: Service agnostic auto-scaling framework for iaas deployment models," in *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, 2013, pp. 42–49.
- [4] J. Huang, C. Li, and J. Yu, "Resource prediction based on double exponential smoothing in cloud computing," in *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*. IEEE, 2012, pp. 2056–2060.
- [5] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *2011 IEEE 4th International Conference on Cloud Computing*. IEEE, 2011, pp. 500–507.
- [6] A. A. Bankole and S. A. Ajila, "Predicting cloud resource provisioning using machine learning techniques," in *2013 26th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, 2013, pp. 1–4.
- [7] F. Farahnakian, P. Liljeberg, and J. Plosila, "Lircup: Linear regression based cpu usage prediction algorithm for live migration of virtual machines in data centers," 09 2013, pp. 358–364.
- [8] R. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using arima model and its impact on cloud applications' qos," *IEEE Transactions on Cloud Computing*, vol. 3, pp. 1–1, 08 2014.
- [9] C. Jiang, L. Duan, C. Liu, J. Wan, and L. Zhou, "Vraa: Virtualized resource auction and allocation based on incentive and penalty," *Cluster Computing*, vol. 16, 12 2013.
- [10] C.-T. Lu, C.-W. Chang, and J.-S. Li, "Vm scaling based on hurst exponent and markov transition with empirical cloud data," *Journal of Systems and Software*, vol. 99, 01 2014.
- [11] Z. Gong, X. Gu, and J. Wilkes, "Press: Predictive elastic resource scaling for cloud systems," 11 2010, pp. 9 – 16.
- [12] Y. Wu, Y. Yuan, G. Yang, and W. Zheng, "Load prediction using hybrid model for computational grid," in *2007 8th IEEE/ACM International Conference on Grid Computing*, 2007, pp. 235–242.
- [13] M. Balaji, A. K. Cherukuri, and G. Rao, "Non-linear analysis of bursty workloads using dual metrics for better cloud resource management," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, pp. 1–16, 12 2019.
- [14] N. Tran, T. Nguyen, B. Nguyen, and G. T. Nguyen, "A multivariate fuzzy time series resource forecast model for clouds using lstm and data correlation analysis," in *KES*, 2018.
- [15] S. Taylor and B. Letham, "Forecasting at scale," *The American Statistician*, vol. 72, 09 2017.
- [16] X. Dutreilh, A. Moreau, J. Malenfant, N. Rivierre, and I. Truck, "From data center resource allocation to control theory and back," in *2010 IEEE 3rd international conference on cloud computing*. IEEE, 2010, pp. 410–417.
- [17] R. Han, L. Guo, M. M. Ghanem, and Y. Guo, "Lightweight resource scaling for cloud applications," in *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*. IEEE, 2012, pp. 644–651.
- [18] M. Z. Hasan, E. Magana, A. Clemm, L. Tucker, and S. L. D. Gudreddi, "Integrated and autonomic cloud resource scaling," in *2012 IEEE network operations and management symposium*. IEEE, 2012, pp. 1327–1334.
- [19] T. C. Chieu, A. Mohindra, A. A. Karve, and A. Segal, "Dynamic scaling of web applications in a virtualized cloud computing environment," in *2009 IEEE International Conference on e-Business Engineering*. IEEE, 2009, pp. 281–286.
- [20] Z. A. Al-Sharif, Y. Jararweh, A. Al-Dahoud, and L. M. Alawneh, "Accrs: autonomic based cloud computing resource scaling," *Cluster Computing*, vol. 20, no. 3, pp. 2479–2488, 2017.
- [21] H. Mi, H. Wang, G. Yin, Y. Zhou, D. Shi, and L. Yuan, "Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers," in *2010 IEEE International Conference on Services Computing*. IEEE, 2010, pp. 514–521.
- [22] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 155–162, 2012.
- [23] M. T. Imam, S. F. Miskhat, R. M. Rahman, and M. A. Amin, "Neural network and regression based processor load prediction for efficient scaling of grid and cloud resources," in *14th International Conference on Computer and Information Technology (ICIT 2011)*. IEEE, 2011, pp. 333–338.
- [24] K. Rzacca, P. Findeisen, J. Świderski, P. Zych, P. Broniek, J. Kusmierek, P. K. Nowak, B. Strack, P. Witusowski, S. Hand, and J. Wilkes, "Autopilot: Workload autoscaling at google scale," in *Proceedings of the Fifteenth European Conference on Computer Systems*, 2020. [Online]. Available: <https://dl.acm.org/doi/10.1145/3342195.3387524>
- [25] S. J. Leybourne, "Testing for Unit Roots Using Forward and Reverse Dickey-Fuller Regressions," *Oxford Bulletin of Economics and Statistics*, vol. 57, no. 4, pp. 559–571, November 1995.