Forecasting Models for Self-Adaptive Cloud Applications: A Comparative Study

Vladimir Podolskiy, Anshul Jindal, Michael Gerndt
Department of Computer Architecture and Parallel Systems
Technical University of Munich
Boltzmannstr. 3, Garching (near Munich), Germany 85748
Emails: v.podolskiy@tum.de, anshul.jindal@tum.de, gerndt@in.tum.de

Yury Oleynik
Instana GmbH
55 E 3rd Ave, San Mateo, CA 94401
Email: yury.oleynik@instana.com

***Abstract*—With the introduction of autoscaling, clouds have strengthened their position as self-adaptive systems. Nevertheless, the reactive nature of the existing autoscaling solutions provided by major Infrastructure-as-a-Service (IaaS) cloud services providers (CSP) heavily limits the ability of cloud applications for self-adaptation. The major reason of such limitations is the necessity for the manual configuration of the autoscaling rules. With the evolution of monitoring systems, it became possible to employ the data-driven approaches to derive the parameters of scaling rules in order to enable the autoscaling in advance, i.e. the predictive autoscaling. The change in the amount of requests to microservices could be considered as a reason to adapt the virtual infrastructure underlying the cloud application. By forecasting the amount of requests to cloud application, it is possible to estimate the upcoming demand to replicate the microservices in advance. Hence, anticipation of the demand on the cloud application helps to evolve its self-adaptive properties.**

In the scope of the paper, the authors have tested various extrapolation models on the real anonymized requests time series data for 261 microservices provided by the industry partner Instana. The tested models are: various seasonal ARIMA models with GARCH modifications and outliers detection, exponential smoothing models, singular spectrum analysis (SSA), support vector regression (SVR), and simple linear regression. In order to evaluate the accuracy of these models, an interval score was used. The time required to fit and use each model was also evaluated. Comparative results of this research and the classification of forecasting models based on the interval accuracy score and model fitting time are provided in the paper. The study provides an approach to evaluate the quality of forecasting models to be used for self-adapting cloud applications and virtual infrastructure.

***Keywords*-requests forecasting; self-adaptive cloud; microservice; seasonal ARIMA; GARCH; SSA; SVR**

I. Introduction

The most important property of a cloud application is its scalability. Scalability allows to adjust the cloud application to the changing amount of requests by adding or removing virtual machine (VM) with microservices. Recently, the scaling was also introduced to the serverless architectures allowing to change the number of replicas of the function application; for example, Function-as-a-Service (FaaS) offering from Microsoft Azure monitors the rate of events in order to scale the number of functions horizontally[1]. In general, such adjustments can be done either in response to an observed change in the monitored parameters or by extrapolating the amount of requests and determining the necessary number of microservices replicas that could serve it [1]. The latter path is getting more attention both within the research community and within the companies. It is known under the name of *predictive autoscaling*.

Predictive autoscaling involves two major stages - forecasting of cloud application parameters that might be used to adjust the application, and adjustment of the application according to the forecast and Quality of Service (QoS) requirements. The quality of predictive autoscaling heavily relies both on the accuracy of the forecast and on the speed of its computation. The number of requests is widely used as a metric to be extrapolated as this parameter does not depend neither on the application characteristics nor on the underlying infrastructure, i.e. it cannot be directly influenced, e.g. by adding new virtual machines. Preliminary research shows that requests arrival rate patterns tend to be less noisy than hardware-relevant metrics, e.g. CPU or memory utilization. Hence, we focus on the evaluation of forecasting models used for the prediction of the number of requests.

A variety of forecasting techniques was already evaluated in a number of publications. These studies, however, usually demonstrate at least one of the following weak points: 1) the lack of comprehensiveness - usually, study focuses only on two or three forecasting models [2], [3], [4], [5]; 2) insufficient forecasting models selection and tuning or comparison of untuned model with the tuned [6]; 3) strong focus on the accuracy evaluation omitting the evaluation of time required to fit the model [2], [3]; 4) use of accuracy metrics intended for point forecasts which are vaguely relevant for practice [7], [2], [3], [8]. In the paper, we have tried to overcome these challenges to establish a solid foundation for the efficient self-adaptation of the virtual cloud infrastructure for the demand.

In the next section, we provide an introduction to the considered forecasting models. The third section of the

[1] docs.microsoft.com/en-us/azure/azure-functions/functions-scale

paper highlights the interval score used to evaluate the presented forecasting models based on the so-called prediction intervals. The fourth section describes the experimental setting and the implementation of the analysis framework. Additional information on the interactive forecasting models evaluation service is also provided in the same section. The fifth section presents the results of the comparative forecasting models study pointing at the necessary careful consideration of the forecasting model depending on the underlying time series. The sixth section discusses the results of the comparative study and outlines the directions to integrate the forecasting models into the predictive autoscaling solutions based on the complex evaluation of accuracy and performance. Related works studying different workload forecasting models and techniques are covered in the seventh section. The last section contains conclusions and future research directions.

II. Forecasting Models

A. ARIMA and GARCH

The AutoRegressive Integrated Moving Average (ARIMA) process models a time series, i.e. a sequence of values of some variable ordered in time. ARIMA can account for seasonality (SARIMA) and external regressors (ARIMAX). The generalized SARIMA model for the time series z_t is represented by the equation [9]:

$$\Phi_P(B^S)\phi_p(B)\nabla_S^D\nabla^d z_t = \Theta_Q(B^S)\theta_q(B)a_t \qquad (1)$$

The ARIMA process establishes the connection between the time series z_t and its previous values using the backward-shift operator B with modification ∇, e.g. $Bz_t = z_{t-1}$ and $\nabla = 1 - B$, and coefficient polynomials $\Phi_P(B^S)$ and $\phi_p(B)$ (autoregressive component). The random noise a_t is also considered with corresponding coefficients $\Theta_Q(B^S)$ and $\theta_q(B)$ (moving average component).

The short-hand for the generalized model is $ARIMA(p,d,q) \times (P,D,Q)_S$. Seasonal and non-seasonal components in the Equation 1 are represented by polynomials of B of orders p, q, P, Q. Orders d and D point to the differentiation of time series in order to get the stationary time series required by ARIMA process. In case of non-integer value of d, we deal with the fractionally integrated (ARFIMA) process [10].

The parameters selection for ARIMA models includes finding the values of p, q, d, and their seasonal counterparts. This process is vaguely formalized and mostly encompasses residual values evaluation, i.e. a sequence of steps is conducted to fit ARIMA models with different parameters values and the difference between the fitted model and the actual values is compared with the white noise.

The Generalized AutoRegressive Conditional Heteroskedasticity (GARCH) model describes the variance of the error term in the source time series model, therefore this model is useful for time series with changing variance. The general formula for $GARCH(p,q)$ is [11]:

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i a_{t-i}^2 + \sum_{i=1}^{p} \beta_i \sigma_{t-i}^2 \qquad (2)$$

GARCH model is useful for time series that exhibit volatility, e.g. when there occurs an outage followed by the retry wave. GARCH models can significantly extend ARIMA models forecasting abilities in case of volatile time series. In the paper, GARCH was used precisely in this manner - it was combined with ARIMA models to fit residuals thereof.

Additional details on the described and following models are not provided due to space limitation.

B. Exponential Smoothing

The main idea of the exponential smoothing is to approximate the time series using the exponential window function. Exponential functions are used to assign decreasing weights to past observations thus favoring the recent observations when computing the predictions.

Exponential smoothing includes various models that differ in type of their components: error, trend, and seasonality. A trend could either be absent or could be of one of the following types: additive, additive damped, multiplicative, and multiplicative damped. Seasonal component also varies and is usually represented as absent, additive, or multiplicative. R. Hyndman et al. propose to use Akaike Information Criterion (AIC) to select a specific exponential smoothing model [12].

Holt's linear trend method of this class gives the prediction for h steps in the future by linear equation [13]:

$$\hat{y}_{t+h|t} = l_t + hb_t \qquad (3)$$

The exponential smoothing is hidden in l_t and b_t that determine the level and the trend. Holt-Winters extends Holt's method for seasonality in time series.

C. Singular Spectrum Analysis

Singular Spectrum Analysis (SSA) is a non-parametric method of time series decomposition that obtains spectral information on time series. SSA algorithm includes four steps [14]: embedding of original time series into the vector space of specific dimension; singular value decomposition resulting in a set of elementary matrices of rank 1; eigentriple grouping to group elementary matrices; diagonal averaging to receive the original time series representation as a sum of reconstructed subseries.

SSA applies the apparatus of linear algebra to time series in order to receive its decomposition. Consequently acquiring the covariance matrix for lagged time series, computing its eigenvalues, eigenvectors, and principal components, following with the summation of reconstructed

components, we receive a reconstructed time series. The results of SSA method are appropriate for forecasting using the linear homogeneous recurrence relation as was proposed by Golyandina et al [15].

D. Support Vector Regression

Support vector machines (SVM) were originally created with the purpose to classify data samples. SVMs are supervised learning models. Support vector regression (SVR) is an extension proposed by H. Drucker et al. [16].

SVR functions by conducting the nonlinear mapping of the input into a feature space. A linear model is constructed in the feature space using the supervised learning. This linear model is given by the equation:

$$f(x,\omega) = \sum_{j=1}^{m} \omega_j g_j(x) + b \tag{4}$$

To learn the model, SVR uses optimization techniques adjusted by several meta-parameters influencing the quality of the regression. Wrong selection of these parameters may result in underfitting or in overfitting.

E. Simple Linear Regression

In seasonal time series analysis, simple linear regression might be used only as a baseline for comparison with advanced forecasting techniques. Fitting of such a model is conducted by minimizing the squared distance between data points and the model. Graphically, the simplest linear regression model is represented as a straight line drawn through the data points.

III. Interval Score for Forecasts Evaluation

Scoring is used to compute a numerical value for a forecasting model based on out-of-sample prediction results. This allows to compare each specific model with other models in respect to accuracy. As the invention of a crystal ball is still far away, in practice the interval forecast is preferred over the point forecast. Interval forecasts are based on the so-called Prediction Interval (PI). PI is an interval which covers the future observation with a certain probability. In practice, PIs with the probability of 95% are usually used. An example of 80% and 95% PIs is provided in the Fig. 1.

In this paper, a *negatively oriented interval score* [17] was used to evaluate the forecasts:

$$S_\alpha^{int}(l,u;x) = (u-l) + \frac{2}{\alpha}(l-x)\mathbf{1}\{x<l\} + \frac{2}{\alpha}(x-u)\mathbf{1}\{x>u\}$$

This score includes the penalty $(u-l)$ for the wide PI; α is the probability of I type error, $\mathbf{1}\{y\}$ is an identity function used to transfer the result of logical expression into continuous space[2]. The lower the score is, the more favorable the evaluated forecasting model is.

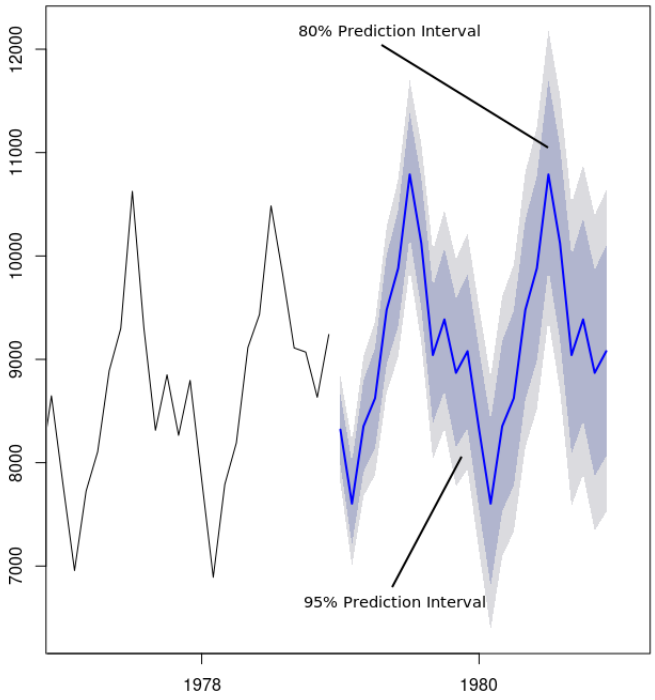

Figure 1. Example of prediction intervals acquired with for monthly totals of accidental deaths in the USA.

IV. Experimental Setting & Implementation

A. Data & Hardware

The experimental data set provided by company Instana was cleared prior to experiments and therefore consists of 261 non-zero 4 week-long time series (673 values in each) with interpolated missing values. These time series were collected for distinct microservices under the real load. Each value represents the amount of requests per hour. The discretion is 1 hour.

The following hardware was used to conduct the tests: Intel core i7, 2.7 GHz, 16 GiB RAM. The operating system used is Archlinux (4.12.12-1-Arch).

B. Testing & Analysis Framework

The testing and analysis framework was implemented as a set of scripts in R. These scripts use the following CRAN libraries to establish the forecasting functionality: forecast, rugarch, tsoutliers, arfima, Rssa, e1071. The wrapping script ForecastingService.R accepts path to the file with the data set, starting time for the series and type of processing as

[2] $\mathbf{1}\{y\}$ can be substituted:
- in case of $\mathbf{1}\{y<l\}$ by $max(\text{sign}(l-y),0)$;
- in case of $\mathbf{1}\{y>u\}$ by $max(\text{sign}(y-u),0)$.

parameters and starts model fitting and forecasting. In order to decrease the overall testing time, multiple cores of the processor are used to fit the models.

The following models were fitted to each time series: exponential smoothing, seasonal ARIMA, seasonal outliers-adjusted ARIMA, seasonal ARFIMA, linear regression, SSA, and SVR. Each ARIMA-based model was also extended with GARCH, which gives a total of 10 model classes which are further simply addressed as models[3]. The parameters for all models are selected automatically except for the pre-selected parameters for SVR model; particularly, for ARIMA models an implementation of the traditional Box-Jenkins approach [9] is employed. The implementation of the corresponding algorithms might be found using the link to the public repository in section IX.

The time required for parameters selection was not included in the duration evaluation as accounting for this time would significantly penalize the models using machine-learning techniques and requiring parameters to be set up prior to learning (e.g. SVR). The selection of such parameters is a procedure that is conducted very rarely, therefore in practice it would not influence the performance of model fitting in the majority of cases.

For testing and scoring the last 168 values (1 full week of data points) of each time series were used.

As an output of the testing framework, a table with 261 rows is produced. Each row corresponds to each time series, and contains the score and the duration for each forecasting model.

C. Interactive Web Service for Forecasting Models Evaluation

An interactive web application with microservice architecture was developed using Node.js and R to make the forecasting user-friendly and easily available. The architecture of the application is shown in Fig. 2.

User Interface (UI) provides a way to easily configure different parameters including starting timestamp, i.e. the timestamp of the first value in the time series, and the number of prediction steps, i.e. the number of predicted data points. UI allows the user to either upload and perform the forecasting with own data or test the service with the provided sample data. This component is also responsible for showcasing the forecasting graphs as well as the interval score and time taken to complete the forecast for all the models. A set of R scripts is the backbone of the web application - data cleaning and forecasting logic resides

[3]Generally speaking, each model class contains multiple models which differ in parameters used, e.g. the seasonal ARIMA model class includes seasonal ARIMA(1,0,0,12,1,0,0), seasonal ARIMA(2,0,1,12,1,0,0) and many more. For each time series only a single model of each class was selected based on the automated model parameters selection. Therefore, instead of using "model class", the word "model" is used in the text. Further evaluation of the model class is based on the evaluation of the model selected for the given class for each time series individually.

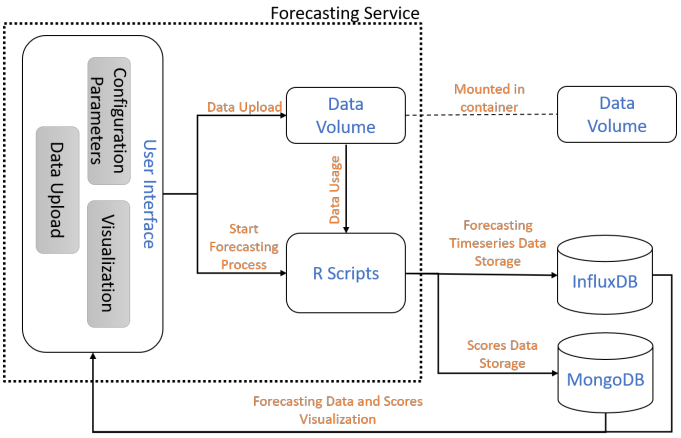

Figure 2. Architecture of the Forecasting Service.

there. A data volume is mounted for storing the uploaded data. The combination of the UI, R scripts and mounted data volume makes the forecasting service.

A typical workflow for the forecasting models evaluation web service includes:

1) storing the uploaded data into the mounted volume;
2) running R scripts for forecasting using the user-configured parameters and the data path;
3) storing the forecasted results inside InfluxDB for each model on the fly;
4) storing the interval score and the computation time for each model inside MongoDB;
5) visualizing the data stored in both DBs by web-UI.

Fig. 3 shows an example for the requests number forecasting graph using SARFIMA with GARCH model for 22 prediction steps (hours). The data for the graph was provided by the industry partner Instana. The depicted lines are:

- *SARFIMA+GARCH.av* - the actual value;
- *SARFIMA+GARCH.ub* - the upper bound of PI computed based on the point forecast;
- *SARFIMA+GARCH.lb* - the lower bound of PI computed based on the point forecast;
- *SARFIMA+GARCH.pf* - the point forecast.

The graph in Fig. 3 shows that all the actual values of the request time series are inside the prediction intervals computed based on point forecasts. Hence, we can assume that by using the prediction intervals a sufficient coverage of the possible future values for the number of requests variable may be reached. The smaller the area between the upper and the lower bound of the prediction intervals is, the more accurately an exact position of the predicted values could be identified. An attempt to make this area as narrow as possible could lead to some of the values being outside of the prediction intervals. The case of an actual value lying above the upper bound could be considered a significant issue as it implies that the number of requests

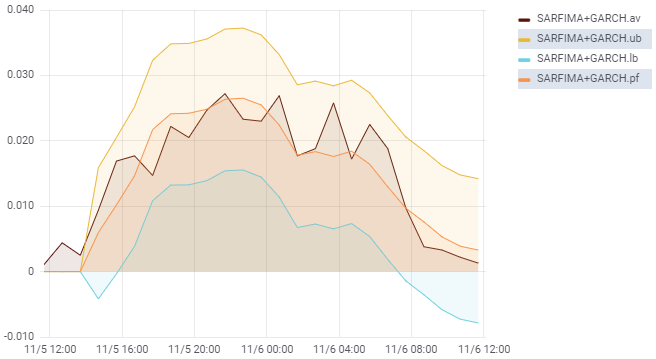

Figure 3. Prediction intervals for SARFIMA with GARCH model.

is underestimated and therefore the virtual infrastructure deployed for the anticipated number of requests will not be enough.

Despite actual values of the requests time series being able to fit accurately inside the area between lower and upper bounds of the predictions in Fig. 3, the prediction intervals exhibit some redundancy. This redundancy is attributed to the negative values of the lower bound of prediction intervals for some of the points. This fact points at the opportunity to make the prediction interval more accurate integrating the semantic information about the time series in the prediction. That way, prediction interval for such characteristics as e.g. the amount of requests, amount of memory used, should always stay above zero.

SLA-constraints on the prediction intervals width might also be introduced as a quality measure for the forecasts. For example, such constraints might be imposed on the discussed interactive forecasting service. That way, an additional filtering might be added that provides the forecasts only of appropriate quality as specified by SLA. As the prediction interval might extend with the time, the SLA-constraints on the width of the prediction intervals should increase with the number of prediction steps for which the forecast is provided.

V. Comparison of Forecasting Models

Due to the large amount of time series (261, one series per each service), we won't discuss distinct forecasts, but will rather focus on the overall comparison. The comparison methodology is based on two parameters - model fitting duration and interval score discussed previously. By identifying these two parameters for the combination of each time series with each forecasting model (out of 10 listed before), we receive a total of 2610 points in the two-dimensional space. A part of these points is shown in Fig. 4.

Evaluating the forecasting models *based purely on acquired interval scores*, we may note that SARFIMA-based models jointly outperform other models in almost 51% of all best cases. The overall rating is provided in Table I. Linear regression didn't make it into rating as its interval score was worst in all the test cases.

Table I
Forecasting Models Rated Only by Interval Score

Rank	Model	Cases with Top Score	% of Cases
1	SARFIMA with GARCH	79	30.27
2	SARFIMA	53	20.31
3	SSA	31	11.88
4	SVR	31	11.88
5	SARIMA with outliers	25	9.58
6	SARIMA with GARCH	14	5.36
7	SARIMA with outliers with GARCH	12	4.60
8	SARIMA	11	4.21
9	Exponential Smoothing	5	1.92
	Total	**261**	**100**

Although the SARFIMA with GARCH modification tops other models, it is the best model only for slightly more than 30% of test cases as shown in the fourth column in Table I. When considering the requests forecasting for such highly dynamic and business-critical entity as cloud, it is important that the forecasts are produced in time so that the cloud infrastructure adaptation could still happen in time, before the request arrive. The consideration of the second evaluation parameter, namely model fitting duration, becomes necessary for practical applications.

Fig. 4 presents the zoomed-in version of the interval score / duration space for all the test cases with the evaluation results near the origin (0, 0). As we can see, SARFIMA and its GARCH-modification which demonstrated the best interval score for more than half of the time series, do not appear in the zoomed-in graph. The reason is the highest model fitting time reaching 253.79 seconds on average for pure SARFIMA, whereas the graph is limited at around 1.8 s. of model fitting duration (Y-axis). Another coordinate (X-axis) corresponds to the interval score. As was previously discussed, the interval score is engineered in such a way that the lower this score is, the more accurately the forecasted area covers the future values of the time series. As an example of reading the graph, we can see that outliers-adjusted SARIMA demonstrates the score of 0.6408 which is the lowest score for the 55th time series. Most likely, finding this single point on the plot would be problematic, therefore for the sake of analysis the whole space of evaluation points was divided into 4 subspaces which correspond to evaluation classes of the forecasting models.

The space is divided into 4 quadrants as is shown in Fig. 4. These quadrants are determined by two median values: one for interval scores acquired in tests (4.5958) - vertical line, and one for model fitting duration (0.8966 seconds) - horizontal line. Each quadrant corresponds to a single class of models in the scope of test case:

- Class I - appropriate interval score and appropriate

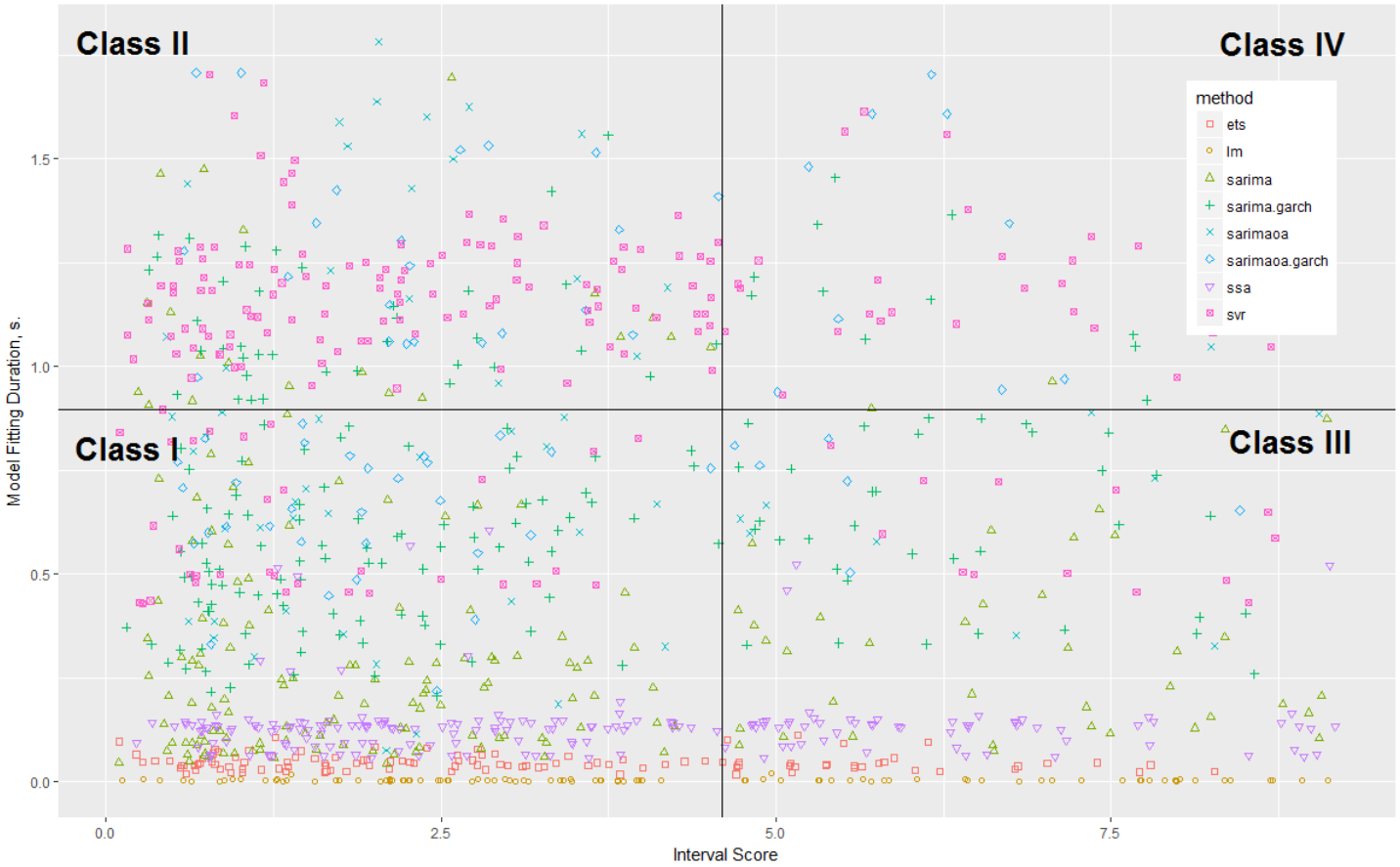

Figure 4. Forecasting Models in Score/Duration Space (zoomed-in)

model fitting duration (both interval score and the fitting time are less than corresponding median values);

- Class II - appropriate interval score but not optimal model fitting duration (only the interval score is less than the corresponding median value);
- Class III - appropriate model fitting duration but not optimal interval score (only the model fitting duration is less than the corresponding median value);
- Class IV - worst score and model fitting duration (both interval score and the fitting time are greater than corresponding median values).

The relative attribution of the forecasting model to the specified classes is shown in Figure 5. This plot shows how many cases of the forecasting model application to the time series from Instana data set were attributed to one of the four aforementioned classes.

The selection of the medians as the base for the forecasting models classification allowed to establish the primary groups of models on average exhibiting the similar quality characteristics. The selection of these borderlines was conducted arbitrarily as no domain-specific information was provided along with the data. The classification borderlines might be selected based on the application and domain specifics, e.g. in the case of strict requirements on the forecasts accuracy the median value for the score might be substituted by lower percentile, e.g. 25-percentile.

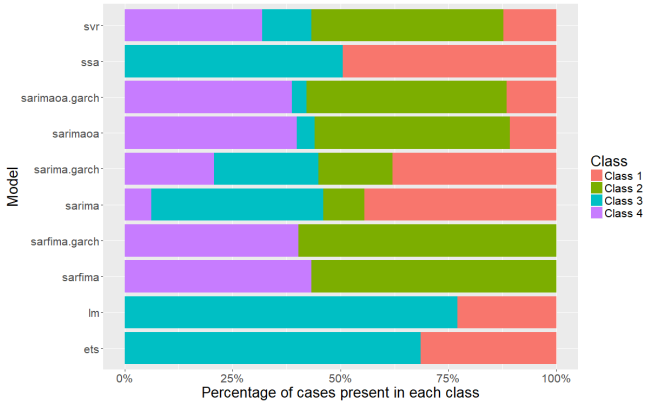

Figure 5. Relative Presence of Forecasting Models in each Class

It is necessary to mention that not all the evaluation points are provided in the graph. An attempt to visualize all the

points on the same graph leads renders the visualized data unreadable.

Classes I and II are the most relevant for the practical use (best interval scores). Rankings inside these classes are provided in Table II and Table III correspondingly. An interesting observation follows for Class I which has no obvious leader - SSA and SARIMA seem to exhibit high interval accuracy and computation-efficiency. If we recall the results in Table I, we will see that seasonal ARFIMA model is more accurate, however it is widely represented only in Class II (Table III) which implies its slow fitting procedure thus making its use impractical. High ranks of outliers-adjusted models in Table III are caused by the presence of time series with requests bursts in the data set - due to the inclusion of the terms responsible for capturing the outliers, these models obtain increased accuracy.

Table II
CLASS I: APPROPRIATE INTERVAL SCORE AND APPROPRIATE MODEL FITTING DURATION

Rank	Method	Number of Cases in the Class	% of all the cases
1	SSA	129	49
2	SARIMA	116	44
3	SARIMA with GARCH	99	38
4	Exponential Smoothing	82	31
5	Linear regression	60	23
6	SVR	32	12
7	SARIMA with outliers with GARCH	30	11
8	SARIMA with outliers	28	11

The results provided in the Table II show that the linear algebra (SSA) and simple statistical models (SARIMA) could be used to achieve both accurate and fast forecasting. Despite the low ranks of the forecasting models taking the outliers into account (i.e. aiming to forecast the structural outliers alongside the general prediction), for certain industrial cases the ability of the model to take the outliers into account might be crucial. Hence, the features of the use-cases should be carefully considered when selecting the appropriate forecasting model, though the interval accuracy score and the model fitting time alone can provide an appropriate evaluation for making the decision about the use of the specific model in practice.

Additionally, the results of the conducted research (see both tables) imply that none of the models is equally accurate and fast to compute for every request pattern obtained for the microservices. This point is especially important in the light of the diversity of the cloud management activities - these activities may differ both in the level of the importance for the cloud user and in the time required to accomplish the specific activity. Therefore, the requirements on the speed with which the corresponding management activity gets its input in the form of the forecast may also differ. The variety of load patterns also contributes to the requirements imposed on the forecasting models. That way, it becomes possible to adapt the selection of the forecasting models for each specific time series based on its usage requirements and on the type of the load pattern. A following example could be provided - the smart scaling of a logging service with a slowly changing load pattern might be not so demanding on the speed with which it gets the forecasted number of requests. In this case, the forecasting model with a higher fitting time and the higher accuracy could be used. Hence, the scaling for such a service might be conducted more accurately, thus fulfilling the demand and at the same time minimizing the cost. Naturally, other cloud management activities might demonstrate the near real-time requirements (e.g. an activity should be conducted in the scope of milliseconds or seconds) or dynamically changing load pattern, which definitely demands the use of forecasting models with the lower fitting time. Though, such models might be less accurate.

The smart cloud operations on distinct microservices require an individual evaluation of forecasting models. The minimization of the evaluation activities might be reached with the generalization of the requests time series to some common patterns. This approach, however, requires a significant amount of data, i.e. at least several months worth of the data.

Table III
CLASS II: APPROPRIATE INTERVAL SCORE BUT NOT OPTIMAL MODEL FITTING DURATION

Rank	Method	Number of Cases in the Class	% of all the cases
1	SARFIMA with GARCH	156	60
2	SARFIMA	148	57
3	SARIMA with outliers with GARCH	121	46
4	SARIMA with outliers	118	45
5	SVR	116	44
6	SARIMA with GARCH	45	17
7	SARIMA	25	10

VI. DISCUSSION

Evaluation of the forecasting models both in the dimension of the accuracy and in the dimension of the time allows to better select the forecasting models appropriate for particular use-cases. By including the model fitting duration in the evaluation, one may determine whether a particular model would be sufficient for an operation which needs to be conducted fast, i.e. requiring that the inputs (forecasting results) are also provided in the timely manner. The accuracy of the forecasting models evaluated by means of the interval score is the basic quality metric for the prediction model - it allows to understand whether a particular model would provide the sufficient foundation for the accurate decision-making, e.g. for determining the number of VMs to be

allocated. Looking at both metrics simultaneously, one might easily identify the models appropriate for the use-cases with complex requirements.

The metrics set could be extended to deepen the analysis of the forecasting models applicability. By taking into account the time necessary for identifying the parameters values for the forecasting models requiring machine learning, one might easily draw a line between the models that could only be useful in the cases where the data do not exhibit frequent changes in patterns and models that do not require much time to be adapted and thus that could be useful in such cases. As we have already demonstrated in the previous section with the SARFIMA-based models, the high quality of the forecasting model in one of the dimensions might come at a price in another dimension. The same applies for the introduction of other possible parameters of evaluation. For example, in the case of model parameters identification time the models with the higher time could exhibit the higher accuracy, though this might be inapplicable for specific cases with frequently changing data patterns. Hence, there is no universal parameters set that could be used for the evaluation of the forecasting models - such evaluation parameters should be selected for each particular use-case in advance.

As was previously stated, the predictive autoscaling is a particular use-case where the forecasting of the requests amount enables the self-adaptation of the cloud virtual infrastructure and cloud applications. Such basic evaluation parameters as the accuracy and the model fitting time could be considered sufficient in selecting the forecasting model for the predictive autoscaling as the cloud application type and underlying business logic are generally unknown. Preliminary collection of the data containing timestamps with the number of the requests to the microservices enables the preliminary forecasting model selection based purely on the batch data. In the case of business-to-customer (B2C) online services, we could state the presence of the reoccurring demand patterns which depend on different conditions (e.g. day of the week, time of the day, national holiday, event, cloud provider outage). Such a use-case would be ideal for the batch-based evaluation of the forecasting models for predictive autoscaling. The stability of patterns implies that the re-selection of the forecasting model is either not necessary or should be conducted rarely thus not seriously influencing the overall performance of the cloud application.

The unstable demand patterns, however, require the frequent adjustment of the model, which could be achieved by incorporating the new measurements provided by the monitoring infrastructure into the model. If the amount of changes becomes high and the initial model cannot reflect the reality anymore, it is necessary to conduct the forecasting models evaluation procedure once more. Due to problems getting the access to production monitoring systems, the dynamic adaptation of the forecasting models and evaluation thereof was not widely studied in practice, though some theoretical models exist [18], [19]. In the following studies, we aim to address this challenge.

VII. Related Works

The forecasting of the number of requests to the cloud services is usually used to derive the number of virtual machines necessary to meet the anticipated demand thus serving as a cornerstone of the predictive autoscaling [20], [21]. The use of the conventional models (e.g. models implemented in R libraries) does not require extensive discussion, therefore only some really interesting or even outstanding examples of forecasting for the self-adaptation of cloud applications and virtual infrastructure are considered below.

One of the most interesting uses of different forecasting models was proposed by H. Fernandez et al. The authors approach the challenge of the requests forecasting model selection from the point of view of the patterns demonstrated by time series [22]. In the Predictor component of the discussed autoscaling system, linear regression is used to forecast the time series with underlying linear trends, ARMA is used for linear trends with small oscillations, exponential smoothing is coupled with daily and seasonal trends, whereas autoregression and vector autoregression capture the correlated trends. The pattern-centric approach to selection of the forecasting model is promising although such strict coupling of models with patterns might lead to the inaccuracy in the forecasted values compared to other models which were not considered for the particular pattern.

The majority of the related works, however, focuses on the resources parameters forecasting, e.g. CPU utilization, used memory. Such parameters are usually used as a base for the smart management of the cloud infrastructure directly by cloud services providers (e.g. migration of VMs).

M. Barati et al. present a hybrid model combining the feed-forward artificial neural network (FFANN) for the prediction of the filtered output of highly dynamic CPU utilization time series with the GARCH model for the prediction of the difference of the smoothed time series and the actual time series [23]. In order to get rid of the oscillating effects on the CPU utilization measurements, authors propose to average the measurements. Although such an approach helps to adapt the data for the forecasting techniques, it also can damage the accuracy of the technique as the important outliers could be lost rendering the workload scheduling at the CSP inaccurate.

J.-J. Jheng et al. have approached the cloud data centers workload prediction with another forecasting model at hand - the authors have adapted the Grey interval forecasting to the data centers workload forecasting followed by the VM migration [5]. The main objective of the forecasting-based VM migration is stated as the reduction of the power consumption at the data center. The work showed the applicability of the Grey interval forecasting to the prediction

of the workloads - this approach could also be adapted to the forecasting of the amount of the requests to the cloud application. The major drawback of the model is, however, its strong focus at the prediction of the trend which could also be approached with other techniques or models. The presentation of the model lacks the comparison with other commonly known analogues thus leaving the question of practical usability of the model.

J. Patel et al. have made a contribution into predicting the CPU load by grouping the CPU utilization time series and deriving the clusters based on the dynamic time warping technique [24]. This approach was applied to the Google trace data. The extraction of clusters was done based on 500 randomly selected tasks of day 18 for this 1 month-long dataset. The cluster workload pattern was evaluated based on the time series forming each of 17 clusters derived. Identification of the pattern close to one of the clusters allowed to estimate the expected workload for a particular new task thus enabling the flexible allocation of the resources. The presented approach could not be considered as a pure forecasting in the classical sense as it does not deal with the extrapolation of the time series but rather with the classification of the time series. Authors have extended this approach with the multilayered artificial neural networks used for forecasting of the CPU utilization, thus enabling the cluster-specific prediction for the resource usage [25]. The conducted studies have demonstrated the appropriateness of the combination of clustering techniques with multilayered artificial neural networks to the tasks of the load forecasting.

The discussed approaches focusing on the prediction of the resources utilization could be useful when the problem of efficient resource allocation for virtual machines is considered by the cloud services provider (CSP). Usually, the Quality of Service (QoS) is not considered by studies in this area due to the shifted focus and the lack of user data. This, and the oscillations in such parameters as CPU utilizations could be considered as a major limiting factors for works in this direction.

VIII. Conclusion and Future Work

In the scope of the paper, 10 forecasting models were evaluated on the requests time series originating from the 4 weeks of observation conducted on 261 distinct microservices. The evaluation was based on the combination of the forecasting accuracy interval score with the model fitting time. The use of the median values for both evaluation parameters enabled the grouping of the forecasting models in four classes. Statistical and linear algebra models show a good compromise of accuracy with the performance on most of the time series presented in the dataset.

The further analysis showed that the fitting time as the additional evaluation parameter contributes to the forecasting models selection when the requirements on the timeliness of the input parameters for the smart cloud management activities differ. Introduction of this parameter might lead to the selection of more accurate forecasting model with higher fitting time for the cloud management activity that is conducted once per hour. By including the application-specific characteristics into the forecasting model fitting procedure, we might also expect that the self-adaptation of the cloud applications to the demand will be timely and as accurate as possible thus ensuring the QoS requirements and minimizing the cost of the virtual infrastructure.

Further research includes the study of fine-grained parameters of forecasting models as well as their influence on the accuracy and the performance of forecasts. The set of forecasting models might further be extended by inclusion of LSTM artificial neural networks and of additional hybrid models. The evaluation of the forecasting models will also profit from inclusion of the quality estimates that compare the time necessary for model fitting and forecasting with the time needed by the cloud services to scale.

Automatic selection and tuning of forecasting models for cloud operations tasks is yet another research direction that is tightly coupled with the research advances on other directions. Last but not least, the significant amount of work will be devoted to the study of online adaptation of the forecasting models based on the dynamical data and of the impact of the models adjusted accordingly on the accuracy and performance with the real workloads.

IX. Availability

The source code for the testing and analysis framework is available under the following link:

`https://github.com/Remit/LPE`

The dataset used for the analysis is available per request.

Acknowledgments

We express our gratitude to our industry partner Instana for generously providing the data set with requests time series. We would also like to thank the anonymous reviewers that have generously provided their comments to improve the paper. We have tried our best to address all the comments provided.

References

[1] J. A. L. Tania Lorido-Botran, Jose Miguel-Alonso, "Auto-scaling Techniques for Elastic Applications in Cloud Environments," University of the Basque Country, Department of Computer Architecture and Technology, Tech. Rep., 09 2012.

[2] V. G. Tran, V. Debusschere, and S. Bacha, "Hourly server workload forecasting up to 168 hours ahead using seasonal arima model," in *2012 IEEE International Conference on Industrial Technology*, March 2012, pp. 1127–1131.

[3] R. N. 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, no. 4, pp. 449–458, Oct 2015.

[4] G. Tzagkarakis, M. Papadopouli, and P. Tsakalides, "Trend forecasting based on singular spectrum analysis of traffic workload in a large-scale wireless lan," *Performance Evaluation*, vol. 66, no. 3, pp. 173 – 190, 2009, modeling and Analysis of Wireless Networks: Selected Papers from MSWiM 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166531608001053

[5] J.-J. Jheng, F.-H. Tseng, H.-C. Chao, and L.-D. Chou, "A novel vm workload prediction using grey forecasting model in cloud data center," in *The International Conference on Information Networking 2014 (ICOIN2014)*, Feb 2014, pp. 40–45.

[6] Y.-C. Chang, R.-S. Chang, and F.-W. Chuang, "A predictive method for workload forecasting in the cloud environment," in *Advanced Technologies, Embedded and Multimedia for Human-centric Computing*, Y.-M. Huang, H.-C. Chao, D.-J. Deng, and J. J. J. H. Park, Eds. Dordrecht: Springer Netherlands, 2014, pp. 577–585.

[7] N. R. Herbst, N. Huber, S. Kounev, and E. Amrehn, "Self-adaptive workload classification and forecasting for proactive resource provisioning," *Concurrency and Computation: Practice and Experience*, vol. 26, no. 12, pp. 2053–2078, 2014. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.3224

[8] Y. Han, J. Chan, and C. Leckie, "Analysing virtual machine usage in cloud computing," in *2013 IEEE Ninth World Congress on Services*, June 2013, pp. 370–377.

[9] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis: Forecasting and Control*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2008.

[10] J.-R. Granger, C. W. J., "An introduction to long-memory time series models and fractional differencing," *Journal of Time Series Analysis*, vol. 1, pp. 15–30, 1980.

[11] R. Engle, "Garch 101: The use of arch/garch models in applied econometrics." *Journal of Economic Perspectives*, vol. 15, no. 4, pp. 157–168, 2001.

[12] R. Hyndman, A. Koehler, R. Snyder, and S. Grose, "A state space framework for automatic forecasting using exponential smoothing methods," *International J. Forecasting*, vol. 18, no. 3, pp. 439–454, 2002.

[13] C. C. Holt, "Forecasting trends and seasonal by exponentially weighted averages," *International J. Forecasting*, vol. 20, no. 1, pp. 5–10, 2004.

[14] N. Golyandina and A. Zhigljavsky, *Singular Spectrum Analysis for Time Series*. Heidelberg, Germany: Springer, 2013.

[15] N. Golyandina, V. Nekrutkin, and A. Zhigljavsky, *Analysis of Time Series Structure: SSA and related techniques*. Washington, D.C.: Chapman & Hall, CRC, 2001.

[16] H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola, and V. N. Vapnik, "Support vector regression machines," *Advances in Neural Information Processing Systems*, vol. 9, p. 155161, 1996.

[17] T. Gneiting and A. E. Raftery, "Strictly proper scoring rules, prediction, and estimation," *Journal of the American Statistical Association*, vol. 102, no. 477, pp. 359–378, 2007. [Online]. Available: https://doi.org/10.1198/016214506000001437

[18] O. Anava, E. Hazan, S. Mannor, and O. Shamir, "Online Learning for Time Series Prediction," *ArXiv e-prints*, Feb. 2013.

[19] C. Liu, S. C. H. Hoi, P. Zhao, and J. Sun, "Online arima algorithms for time series prediction," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI'16. AAAI Press, 2016, pp. 1867–1873. [Online]. Available: http://dl.acm.org/citation.cfm?id=3016100.3016160

[20] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "A cost-aware elasticity provisioning system for the cloud," in *2011 31st International Conference on Distributed Computing Systems*, June 2011, pp. 559–570.

[21] Q. H. Qing Li, Qinghai Yang and K. S. Kwak, "Profit-Maximizing Virtual Machine Provisioning Based on Workload Prediction in Computing Cloud," *KSII Transactions on Internet and Information Systems*, vol. 9, no. 12, pp. 4950–4966, 2015.

[22] H. Fernandez, G. Pierre, and T. Kielmann, "Autoscaling web applications in heterogeneous cloud infrastructures," in *2014 IEEE International Conference on Cloud Engineering*, March 2014, pp. 195–204.

[23] M. Barati and S. Sharifian, "A new hybrid model for request rate prediction in mobile cloud computing," in *2015 23rd Iranian Conference on Electrical Engineering*, May 2015, pp. 775–780.

[24] J. Patel, V. Jindal, I. L. Yen, F. Bastani, J. Xu, and P. Garraghan, "Workload estimation for improving resource management decisions in the cloud," in *2015 IEEE Twelfth International Symposium on Autonomous Decentralized Systems*, March 2015, pp. 25–32.

[25] Y. Yu, V. Jindal, I. L. Yen, and F. Bastani, "Integrating clustering and learning for improved workload prediction in the cloud," in *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, June 2016, pp. 876–879.