An Ensemble Multivariate Model for Resource Performance Prediction in the Cloud

Jean Steve HIRWA*, and Jian CAO

Shanghai Jiao Tong University
Department of Computer Engineering
800 Dongchuan Road, Minhang District, Shanghai 200240,
P. R. China
E-mail: hirwasteve@hotmail.com
cao-jian@sjtu.edu.cn

Abstract. In cloud environment, multiple resources performance prediction is the task of predicting different resources by considering the differences from multiple task inferences based on the historical values to make effective and certainty judgmental decisions for the future values. One resource performance prediction can conclude the performance of another, which implies dependency (i.e., multi-resources) or independency (i.e., one resource), but that cannot be directly confirmed accurately. We use time series algorithms to investigate possible approaches, which can greatly assist us to analyze and predict the future values based on previously observed values. The goal of this paper is to review the theory of the common several models of multivariate time series, and to emphasize the practical steps to take in order to fit those models to real data and evaluate the outcome. Moreover, ensemble-learning algorithms are applied to the best-fit models to improve performance. Finally, we will discuss the results.

Keywords: Time series · Statistics · Ensemble learning

1 Introduction

In cloud computing [2], since applications complete for resources with unknown workloads from other users, resources contention causes host load and availability to vary over time [3], and makes the load prediction problem even more harder. The host can be viewed as a collection of resources i.e., CPU, memory usage and I/O. If one of the components in cloud computing will not work or at least will execute below par, cloud computing will never work. Certain support measures, for example [6], have to be implemented to prevent any form of downtime.

* The authors thank the reviewers for their valuable comments and effort to improve this paper. This work is partially supported by China National Science Foundation (Granted Number 61073021, 61272438), Research Funds of Science and Technology Commission of Shanghai Municipality (Granted Number 11511500102, 12511502704).

The behavior of cloud computing is highly dynamic [3], wherein the only way the process would be possible is through proper interaction of the application and hardware. Historical data can provide an adequate amount of information for modeling and predicting components in cloud computing behaviors [14]. In [15], [16], [1] the accuracy of prediction is subject to the choice of model chosen, which in turn may be limited by characteristics of the time series observations and the availability of labeled training data.

The available literature on Forecasting consider time series predictions of the status of distributed systems resources both CPU and available memory [1], [2], [4], [9], and their predictions are based on historical information provided by monitoring systems. In [7], and [13], they have evaluated and compared univariate and multivariate normality, and performance by applying graphical and statistical procedures. Obviously, it makes sense to put all available resources into consideration, which would provide more information rather than investigating each resource independently. Ensemble learning methods would be applied to the best fit models, in order to improve performance [11].

2 Related work

Previously, research into resource prediction has focused on determing appropiate predictive models for a single or multiple resources [1], [8] for host behavior. Therefore, this work is mainly focused on multivariate model approach.

For example in [5], they proposed a new means of characterizing correlated workload patterns across Virtual Machines (VMs) resulted from the dependencies of applications running on them. Their applied multiple time series approach, and the workload was analyzed at the group level rather than at the individual VM level.

In [1], they proposed a multi-resource prediction model (MModel) that uses both the autocorrelation (a single resource) and the cross correlation (between two resources such as CPU and Memory usage). And their adaptation approaches were able to adapt to changing characteristics of the resources, especially for highly dynamic resources and long time predictions.

In [9], the approach focuses on the usage prediction for a specific node between CPU and Memory. They have found out that using both resources data can improve the forecasting performance. Moreover, a number of other techniques have been taken into account, from many and different factors of a cloud environment [3], [6], and [14].

In this investigation, regardless unstable behaviors may exist among resources [9] and interdependence, all resources available are taken into consideration as they can provide more information than any other autonomous resources. The idea of ensemble learning approaches is not new [11] [24], [27], but we intended to improve the outcome from the best fit models.

3 Prediction theory and techinques

In this work, we choose few models, which work better with univariate and multivariate time series similitaniously. In general, random variables may be uncorrelated but highly dependent [19]. But if a random vector has a multivariate normal distribution then any two or more of its components that are uncorrelated are independent. This implies that any two or more of its components that are pairwise independent are independent. But it is not true that two random variables that are separately, normally distributed and uncorrelated are independent.

Formally, dependence refers to any situation in which random variables do not satisfy a mathematical condition of probabilistic independence. Correlation can be considered as any departure of two or more random variables from independence, but technically it refers to any of several more specialized types of relationship between mean values.

3.1 Prediction theory

In many situations, it is desirable or necessary to transform a time series data set before using the sophisticated methods [21], [22]. Since we have nonstationary data, an approriate preliminary transformation of the data to get stationarity might succed in stabilizing the variance and then we might use one of the familiar time series models.

Even if some models can be applied directly to nonstationary time series without requiring a preliminary transformation of the data [17], in this case study, univariate functions can be applied point-wise to multivariate data to modify their marginal distributions. It is also possible to modify some attributes of a multivariate distribution using an appropriately constructed transformation [21]. Details for techniques to transform and analyze stationality are introduced in [22].

In the prediction of time series, based on the correlations over time and among the variables, we can estimate the future behavior of time series by using various information extracted from current and past observations [22]. In this study we are looking for an approach which would provide us more accurate information in term of correlation from the previous and current observations comparatively.

3.2 Prediction techniques

In our research process, we proposed two commonly used algorithms for our study and investigation:

Stable Vector Autoregressive Model The vector autoregression (VAR) model is one of the most successful, flexible, and easy to use models for the analysis of multivariate time series. [18], [20]. Forecasts from VAR models are quite flexible because they can be made conditional on the potential future paths of specified variables in the model, see [20], [18]. In its basic form, a VAR consists of a set of d endogenous variables $Y_t = (Y_{1t}, \ldots, Y_{kt}, \ldots, Y_{dt})$ for $k = 1, \ldots, d$. A VAR model of order p (Stable Vector Autoregressive Model VAR(p)) is a special case represented by:

$$Y_t = v + \phi_1 Y_{t-1} + \cdots + \phi_p Y_{t-p} + W_t, \tag{1}$$

with A_i are $(d \times d)$ coefficient matrices for $i = 1, \ldots, p$, and where W_t is a *Gaussian* white noise and d-dimensional process with $E(W_t) = 0$. The VAR(p) process is stationary, which means, Y_t generates stationary time series with time invariant mean, variance and covariance structure given sufficient starting values.

$$det(I_d - A_{1^z} - \cdots - A_{p^{z^p}}) \neq 0 \quad for \quad |z| \leq 1. \tag{2}$$

If the solution of the above equation has a root for $z = 1$, then either some or all variables in the VAR(p) process are integrated of order 1, i.e., $I(1)$. It might be the case, that cointegration between the variables does exist [28].

The stability of emprical VAR(p) process can be analyzed by considering the companion and calculating the eigenvalues of the coefficient matrix, and can be represented by:

$$\xi_t = A\xi_{t-1} + v_t, \tag{3}$$

Once a VAR(p) model has been estimated [18], the next step is to go for further analysis. Causal inference and forecasting are based upon Wold moving average decomposition for stable VAR(p) processes, which is given below as:

$$Y_t = \phi_0 W_t + \phi_1 W_{t-1} + \phi_2 W_{t-2} + \ldots, \tag{4}$$

with $\phi_0 = I_d$ and ϕ_s can be computed recursively according to;

$$\phi_s = \sum_{j=1} s\phi_{s-j} A_j \quad for \quad s = 1, 2, \ldots, \tag{5}$$

where $A_j = 0$ for $j > p$.

Therefore, forecasts for horizons $h \geq 1$ of an emprical VAR(p) process can be generated recursively according to;

$$Y_{T+h|T} = A_1 Y_{T+h-1|T} + \cdots + A_p Y_{T+h-p|T}, \tag{6}$$

where $Y_{T+j|T} = Y_{T+j}$ for $j \leq 0$.

Dynamic Linear Models Dynamic Linear Models (DLM) are represented as a special case of general state space models, being linear and Gaussian e.g., [23], [25]. State-space models can be used for modeling univariate or multivariate time series, also in presence of non-stationarity, structural changes, irregular patterns.

Estimation and forecasting can be obtained recursively by the well know Kalman filter [25], and the first important class of state space models is given by Dynamic Linear Models (DLM), which are represented by the following two equations:

$$\begin{aligned} Y_t &= F_t\theta_t + v_t, \quad v_t \sim N_m(0, V_t) \\ \theta_t &= G_t\theta_{t-1} + w_t, \quad w_t \sim N_p(0, W_t), \end{aligned} \tag{7}$$

where G_t and F_t are matrices and (v_t) and (w_t) are two independent white noise sequences, with mean zero and covariance matrices V_t and W_t respectively. Y_t equation is named *observation equation* and θ_t equation is named *state equation.* Moreover, it is assumed that θ_0 has a Gaussian distribution.

$$\theta_0 \sim M_p(m_0, C_0), \tag{8}$$

for some non-random vector m_0 and matrix C_0, and it is independent on (v_t) and (w_t). In contrast to (7), the general state space model can be provided in the form:

$$\begin{aligned} Y_t &= F_t(\theta_t, v_t) \\ \theta_t &= G_t(\theta_{t-1}, w_t), \end{aligned} \tag{9}$$

with arbitrary functions F_t and G_t, it is therefore more flexible. Linear state space models specify f_t and g_t as linear functions, and Gaussian linear models add the assumptions of Gaussian distributions. Model details summary; filtering, smoothing and forecasting [23], [25].

4 Ensemble learning approach

Ensemble learning is a machine learning paradigm where multiple learners are trained to solve the same problem [26], [27], and it performs better than single learning model and discovers regularities in dynamic. Generally, it is primarily used to improve the (classification, prediction, function approximation, etc.) performance of a model, or reduce the likelihood of an unfortunate selection of a poor one.

The bagging approach is being taken into consideration for this study. It is a device intended for reducing the prediction error of learning algorithms. And following, a brief process of a bagging algorithm:

- Bagging method:
  - Create many data sets by bootstrapping or cross validation.
  - Create one decision tree for each data set.
  - Combine decision trees by averaging or voting final decisions.
  - Primarily reduces model variance rather than bias.
- Results:
  - On average, better than any individual tree.

Therefore, given a data set $S = (x_1, y_1), \ldots, (x_N, y_N)$ of size N, where $x_n \in X$, $y_n \in Y = \{0, 1\}$, M base models h_m, bagging constructs M classifiers with bootstrap replicas Sm of S, where Sm is obtained by drawing examples from original the data set S with replacement, usually having the same number of examples as S. The diversity among the classifiers is introduced by independently constructing different subsets of the original data set [10], [12]. After constructing ensembles, the prediction of the class of a new example is given by majority voting.

Algorithm 1 Ensemble learning algorithm

```
Input: $S, M$
1: for $m = 1, 2, \ldots, M$ do
2: $S_m$ = Sample with replacement $(S, N)$
3: Train a base learner $h_m \rightarrow Y$ using $S_m$
4: end for
Output: $H(x) = argmax_{y \in Y} \sum_{m=1}^{M} I(h_m(x) = y)$
```

Moreover, it has been proved by [24] that the perfomance of bagging has goodness and badness. It does not simply reduce variance in its averaging process. But instead, it takes multiple random samples (with replacement) from training data set, and uses each of those samples to construct a separate model and separate predictions for test set, which in the end, those predictions are then averaged to create a more accurate and final predictive value. As result, it may underperform its ensemble members. In such situation, reweighting on training set is applicable to some of the learning algorithms.

Algorithm 2 Improved Ensemble learning algorithm

```
1: Initialize base models $h_m$ for all $m \in 1, 2, \ldots, M$
2: for all training examples do
3:   for   $m = 1, 2, \ldots, M$ do
4:   Set $\boldsymbol{w}$ = poisson (1) // a random variable $\boldsymbol{w}$ has poison distribution
5:   Update $h_m$ with the current examples with weight $\boldsymbol{w}$
6: Anytime output:
7: return:$H(x) = argmax_{y \in Y} \sum_{m=1}^{M} I(h_m(x) = y)$
```

Finally, we evaluate the outcome and investigate if an Ensemble Learning *Alg.* 1 underperformed its previous ensemble members. Furthermore, in order to reduce prediction errors, an Improved Ensemble Learning *Alg.* 2 should be applied afterward. The results and discussion are given in *Section*6.

Table 1: An overview of ensemble learning algorithm

$$\begin{tikzcd}
Y \arrow[r] & model_1 \arrow[rd] & & \\
Y \arrow[r] & \dots \arrow[r] & \sum \arrow[r] & Y_{ens}(t) \\
Y \arrow[r] & model_m \arrow[ru] & &
\end{tikzcd}$$

5 Evaluation Techniques

The datasets are localized from different datasets collected from different time sets. To conduct the numerical analysis, the dataset are selected from 2 different clusters and 3 variables are chosen from each cluster with the same time set and host: CPU, Memory usage and I/O. Furthermore, 1000 observations were randomly taken for each study separately. We analyze and investigate our data dependently and independently to reach our goal.

The models used for our study have many parameters, and they may be difficult to interpret due to complex interactions and feedback between variables in the model. As a result, we tried to find common various types of structural analysis, but putting more emphasize into correlation relationships Moreover, our samples are transformed stationary and normally distributed, i.e., $Y \sim N(0, \sigma^2)$; each variable Y is independent and identically distributed with mean zero, standard deviation σ^2, normal variate, in order to maintain the same condition for a good investigation.

The Root Mean Square Error (RMSE) is moslty applied to measure the difference between values predicted by a model and the values actually observed from the environment that is being modelled, and those differences are *residuals*. The RMSE of a model prediction with respect to the estimated variable Y_model is defined as the square root of the mean squared error:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(Y_{obs,i} - Y_{model,i})^2}{n}} = \sqrt{\frac{\sum_{i=1}^{n}(residuals)^2}{n}} \tag{10}$$

A granger causality analysis has been also carried out in order to assess whether there is any potential predictability power of one indicator for the other. A granger causality requires that the series has to be covariance stationary [18], [28], so an Augmented Dickey-Fuller test has been calculated. For all of the series the null hypothesis H_0 of non stationarity can be rejected at a 5% confidence level. A granger causality is normally tested in the context of linear regression models [29].

Finally, an Ensemble learning algorithm has been introduced, see *Table 1*, it involves combining multiple predictions derived by different techniques in order to create a stronger overall prediction. In [27] an ensemble with two techniques that are very similar in nature will perform more poorly than a more diverse model set. In this situation, we select the best performed model from each group and combine it with best from a different group.

Table 2: Ensemble learning outcome for multivariate data (Initially = Initial state, Ensemb. = Ensemble learning approach, and Impr. Ens. = Improved Ensemble learning approach)

Time	**Ensemble: CPU and Memory**			**Ensemble: CPU and I/O**		
	Initially	**Ensemb.**	**Impr. Ens.**	**Initially**	**Ensemb.**	**Impr. Ens.**
*Cl.*1						
60 sec	1.0850	1.0456	0.9171	1.0968	1.0719	0.9640
10 min	1.0410	1.0211	0.9694	1.0467	1.0223	0.9290
60 min	1.0403	1.0251	0.9492	1.1083	1.0776	0.9742
*Cl.*2						
60 sec	1.1431	1.1295	0.9855	1.0697	1.0661	0.9980
10 min	1.0758	1.0412	0.9346	1.0692	1.0579	0.9872
60 min	1.0516	1.0167	0.9281	1.0698	1.0592	0.9563

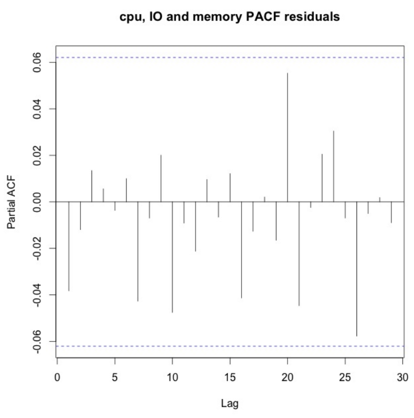
6 Discussion

We have selected our dataset from 2 different clusters and each cluster we only study one host. We have chosen 3 variables from each cluster with the same time set and host: cpu, memory usage and I/O.

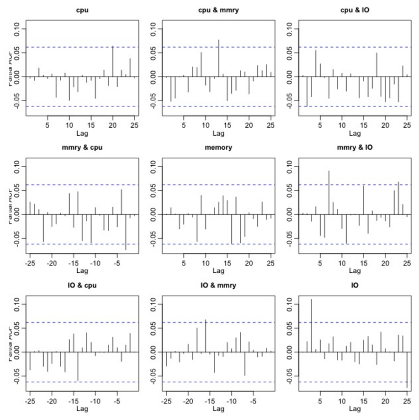
The first step is to analyze the accuracy; by looking onto $RMSE$, it shows that mostly, a multivariate model always has to provide promising results over its univariate counterparts. Surprisingly, few cases of univariate series performed better, as shown in *Tables 5-12*, (all numbers in bold). For VAR(p); once we increase the value of p-lags, more we have a better outcome from multivariate model and it distiguishes itself from its equivalent univariate, as provided in *Tables 5-8.* This might be different from a DLM Model, where the results stay constant *Tables 9-12.* Once we set an initial p-lag value "1" to both selected multivariate algorithms (VAR(p) and DLM); VAR(p) performed poorly than its coequal univariate, but while increasing the p-lag value, a multivariate model outperformed its univariate model.

Compare both used models, we came to the conclusion that it depends on the condition of the initial values of the parameters. At the time p-lag was set to "1", DLM model *Fig. 1-(a)* performed well than VAR(p) model *Fig. 1-(b)*, but once VAR(p) increased its p-lag values *Fig. 2-(a) and 2-(b)*, it performed much better. Therefore, we cannot conclude directly that this model is better than another without looking into its initial state. Generally speaking, VAR(p) would be a promissing model once a p-lag is much higher in our situation *Fig. 2-(b)*.

We had a look at Granger causality whether there is any potential predictability of one indicator for the other. It is hard to judge, but in general more chances will be given to CPU. *Tables 3 and 4*, show that the granger causality value have changed in disorder, which does not make any sense at all. And we conclude that, it is always beneficial if we may apply Granger Causality for each initial Models' conditions independently to find out a variable has a potential predictability, without making a general and fixed conclusion.

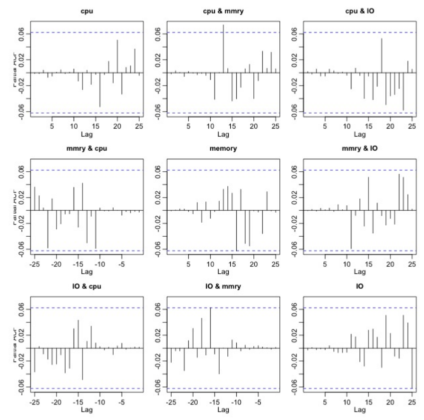


(a) DLM model overall Residuals (CPU, Memory, IO)

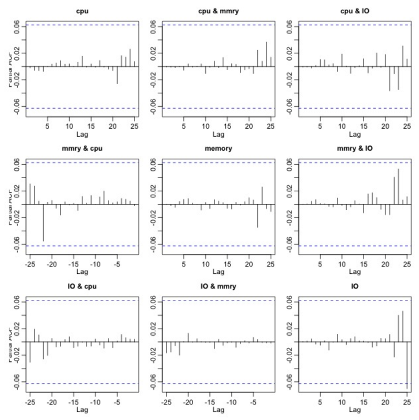


(b) VAR(p) model Residuals: *p*-lag "1" (CPU, Memory, and IO)

Fig. 1: DML and VAR(p) models for multivariate data



(a) VAR(p) model Residuals: *p*-lag "10" (CPU, Memory, and IO)



(b) VAR(p) model Residuals: *p*-lag "20" (CPU, Memory, and IO)

Fig. 2: VAR(p) models for multivariate data

Finally, as seen above multivariate models are promising approaches. If we want to make effective use of information extracted from them, it may always be beneficial to combine the best performed models from each group to obain the best overall and highest performance accuracy *Table 2*.

7 Conclusion

In this work we provide a more comprehensive look at the issue of investigating performance by using more appropriate statistical tests of comparative predictive

Table 3: Cluster-1: Granger Causality

Time	**Granger: CPU and Memory**		**Granger: CPU and I/O**	
	cpu-memory	**memory-cpu**	**cpu - I/O**	**I/O - cpu**
$p = 1$				
60 sec	0.0825	0.3053	0.2835	1.0385
10 min	0.5562	1.0261	0.1984	1.4754
60 min	0.8310	1.0622	0.5492	1.0950
$p = 10$				
60 sec	1.6182	2.1973	0.0765	0.0514
10 min	1.4778	0.6243	0.9124	0.5372
60 min	1.0580	0.6848	1.3795	0.8603
$p = 20$				
60 sec	0.3008	0.0601	0.2541	2.6377
10 min	0.9812	0.6081	1.5446	1.4462
60 min	1.1616	0.7966	1.3042	1.2351

Table 4: Cluster-2: Granger Causality

Time	**Granger: CPU and Memory**		**Granger: CPU and I/O**	
	cpu-memory	**memory-cpu**	**cpu - I/O**	**I/O - cpu**
$p = 1$				
60 sec	0.2696	0.7267	0.2333	0.4009
10 min	1.0079	2.3926	0.4070	1.1025
60 min	0.8015	1.3143	0.7693	0.9387
$p = 10$				
60 sec	0.7965	0.0346	0.2333	1.4252
10 min	0.6610	0.8483	0.5458	1.0681
60 min	0.8015	1.0465	0.7770	0.8215
$p = 20$				
60 sec	0.0653	2.5657	0.2036	1.8821
10 min	0.9035	1.5681	0.6837	1.1332
60 min	0.8310	1.0750	0.6173	1.0845

ability. Moreover, we compare univariate versus multivariate models to provide evidence based real data experiments. We conclude that, in general multivariate outperform univariate counterparts. However, we cannot just conclude the best multivariate model because as seen from the results, it all depends on the initial condition of each model.

This was not enough, we have applied a granger causality to find out if there is any potential predictability of one indicator for the other, and we conclude that CPU has many chances than other resources, but it cannot always be a definitive conculsion according to the unstable results have been evaluated. Therefore, as multivariate contains more information, it might be reasonable to take advantage of ensemble learning approach and apply it to the best fit models to improve the performance from different groups.

References

1. J. Liang, K. Nahrstedt, and Y. Zhou, "Adaptive Multi-Resource Prediction in Distributed Resource Sharing Environment", IEEE International Symposium on Clus-

ter Computing and the Grid, Page(s) 293 - 300, 2004.

2. M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A View of Cloud Computing", Communications of the ACM, Vol. 53, No. 4, Apr. 2010.
3. R. Yang, R. D. Van Der Mei, D. Roubos, F. J. Seinstra, and H. E. Bal, "Resource optimization in distributed real-time multimedia applications", Multimedia Tools and Applications,Vol 59, Issue 3, Aug. 2003.
4. R. D. Hu, J. F. Jiang, G. M. Liu, and L. X. Wang, "Efficient Resources Provisioning Based on Load Forecasting in Cloud", the 10th International Conference on Services Computing, Jul. 2013.
5. A. Khan, X. F. Yan, S. Tao, and N. Anerousis, "Workload Characterization and Prediction in the Cloud: A Multiple Time Series Approach", Network Operations and Management Symposium (NOMS), Page(s) 1287-1294, Apr. 2012.
6. A. F. Antonescu, T. Braun, "Improving Management of Distributed Services Using Correlations and Predictions in SLA-Driven Cloud Computing Systems", Conference Proceeding, 14th IEEE/IFIP Network Operations and Management Symposium (NOMS), Krakow, Poland, May 2014.
7. T. Burdenski, "Evaluating Univerariate, Bivariate, and Multivariate Normality Using Graphical and Statistical Procedures", ERIC, Page 61, Apr. 2000.
8. P. A. Dinda, "Design, Implementation, and Performance of an Extensible Toolkit for Resource Prediction in Distributed Systems", IEEE Transactions on Parallel and Distributed Systems, Vol. 17, No. 2, Feb. 2006.
9. J. Tan, P. Dube, X. Q. Meng, and L. Zhang, "Exploiting Resource Usage Patterns for Better Utilization Prediction", 31st International Conference on Distributed Computing Systems Workshops, Page(s) 20-24, Jun. 2011.
10. D. Z. Xiao, S. Cao, F. Wong, "Optimization of bagging classifiers based on SBCB algorithm", Machine Learning and Cybernetics (ICMLC), Vol. 1, Page(s) 262-267, Jul. 2010.
11. Dr. A. Chitra, and S. Uma, "An Ensemble Model of Multiple Classifiers for Time Series Prediction", International Journal of Computer Theory and Engineering, Vol. 2, No. 3, Jun. 2010
12. Q. He, F. Z. Zhuang, X. R. Zhao, Z. Z. Shi, "Enhanced Algorithm Performance for Classification Based on Hyper Surface using Bagging and Adaboost", Machine Learning and Cybernetics, Vol. 6, Page(s) 3624-3629, Aug. 2007.
13. R. C. Williges, and B. H. Willinges, "Univariate and Multivariate Evaluation of Computer-Based Data Entry", Proceedings of the Human Factors and Ergonomics Society Annual Meeting, Vol. 25, Issue 1, Page(s) 741-745, Oct. 1981.
14. F. Guim, A. Goyeneche, J. Corbalan, J. Labarta, and G. Terstyansky, "Grid computing performance prediction based in historical information", Proceedings of the 7th IEEE/ACM international conference on grid computing, 2006.
15. A. McGovern, D. H. Rosendahl, R.A. Brown, and K. K. Droegemeier, "Identifying Predictive Multi-dimensional Time Series Motifs: An Application to severe weather prediction", Data Mining and Knowledge Discovery, Vol. 22 Issue 1- 2, Page(s) 232 - 258, Jan. 2011.
16. J. G. De Gooijer, R. J. Hyndman, "25 Years of Time Series Forecasting", International Journal of Forecasting, Vol. 22, issue 44:1, Page(s) 443-473, Jan. 2006.
17. D. Kugiumtzis, and E. Bora-Senta, "Simulation of Multivariate Non-Gaussian Autoregressive Time Series with Given Autocovariance and Marginals", Similation and modelling Practice and Theory, Elsevier, Mar. 2014.
18. H. Lutkepohl, "New Introduction to Multiple Time Series Analysis", Springer, Page(s) 13-26, 31-39, 41, 90-100, 102-106, New York, 2006.

19. N. Ebrahini, G. Hamedani, E. S. Soofi, and H. Volkmer, "A Class of Models for Uncorrelated Random Variables", Journal of Multivariate Analysis, Vol. 101, No. 8, Sept. 2010.
20. J. Hamilton, "Time Series Analysis", Princeton University Press, Princeton, 1994.
21. K. Rehfeld, N. Marwan, J. Heitzig, and J. Kurths, "Comparison of Correlation Analysis Techniques for Irregularly Sampled Time Series", Nonlinear Processes in Geophusics, Vol. 18, Page(s) 389-404, Jun. 2011.
22. G. Kitagawa, "Introduction to Time Series Modeling", Page(s) 8-29, CRC Press, Boca Raton, 2010.
23. M. West, and J. Harrison, "Bayesian Forecasting and Dynamic Models, Second Edition", Springer, New York, 1997.
24. Y. Grandvalet, "Bagging equalizes influence", Journal of Machine Learning. Vol. 55, Issue 3, Jun. 2004.
25. J. Durbin, and S. J. Koopman, "Time Series Analysis by State Space Methods", Oxford University Press, Vol. 24, Jun. 2001.
26. T. G. Dietterich, "Approximation Statistical Tests for Comparing Supervised Classification Learning Algorithms", Neural Computation, Vol. 10, No. 7, Pages 1895-1923, Mar. 2006.
27. T. G. Dietterich, "Ensemble Methods in Machine Learning", First International Workshop on Multiple Classifier Systems, Lecture Notes in Computer Science, Vol. 1857, Page(s) 1-15, Jun. 2000.
28. R. F. Engle, and C. W. J. Granger, "Cointegration and Error Correction: Representation, Estimation, and Testing", Economica, Vol. 55, No. 2, Page(s) 251 - 276, Mar. 1987.
29. B. Zhidong, W. K. Wong, and B. Zhang, "Multivariate linear and nonlinear causality tests", Mathematics and Computers in Simulation 81.1, Page(s) 5-17, Jun. 2010.

8 Appendix

Table 5: Cluster-1: VAR(p) model "CPU and Memory usage"

Time	**Multivariate**		**Univariate**			
	RMSE	**Cov**	**RMSE**		**Cov**	
	cpu-memory	**cpu-memory**	**cpu**	**memory**	**cpu**	**memory**
$p = 1$						
60 sec	**0.9743**	**0.0105**	**0.9631**	**0.9833**	**0.9304**	**0.9718**
10 min	0.9857	-0.3560	0.9756	0.9976	0.9527	0.9983
60 min	**0.9811**	**-0.0169**	**0.9657**	**0.9862**	**0.9441**	**0.9908**
$p = 10$						
60 sec	0.9652	0.0105	0.9631	0.9833	0.9304	0.9718
10 min	0.9767	-0.3560	0.9756	0.9976	0.9527	0.9983
60 min	0.9696	-0.0169	0.9657	0.9862	0.9441	0.9908
$p = 20$						
60 sec	0.9600	0.0105	0.9631	0.9833	0.9304	0.9718
10 min	0.9665	-0.3560	0.9756	0.9976	0.9527	0.9983
60 min	0.9610	-0.0169	0.9657	0.9862	0.9441	0.9908

Table 6: Cluster-2: VAR(p) model "CPU and Memory usage"

Time	Multivariate		Univariate			
	RMSE	**Cov**	**RMSE**		**Cov**	
	cpu-memory	**cpu-memory**	**cpu**	**memory**	**cpu**	**memory**
$p = 1$						
60 sec	**0.9789**	**0.0167**	**0.9645**	**0.9904**	**0.9426**	**0.9820**
10 min	**0.9924**	**-0.0012**	**0.9601**	**1.0211**	**0.8524**	**0.5819**
60 min	0.9660	0.0530	0.9866	0.9841	0.9743	0.9719
$p = 10$						
60 sec	0.9681	0.0167	0.9645	0.9904	0.9426	0.9820
10 min	0.9857	-0.0012	0.9601	1.0211	0.8524	0.5819
60 min	0.9728	0.0530	0.9866	0.9841	0.9743	0.9719
$p = 20$						
60 sec	0.9561	0.0167	0.9645	0.9904	0.9426	0.9820
10 min	0.9737	-0.0012	0.9601	1.0211	0.8524	0.5819
60 min	0.9660	0.0530	0.9866	0.9841	0.9743	0.9719

Table 7: Cluster-1: VAR(p) model "CPU and I/O"

Time	Multivariate		Univariate			
	RMSE	**Cov**	**RMSE**		**Cov**	
	cpu-I/O	**cpu-I/O**	**cpu**	**I/O**	**cpu**	**I/O**
$p = 1$						
60 sec	**1.0078**	**0.0032**	**0.9818**	**1.0177**	**0.9649**	**1.0687**
10 min	**1.0147**	**-0.0198**	**0.9955**	**1.0321**	**0.9920**	**1.0680**
60 min	**0.9810**	**-0.0094**	**0.9845**	**0.9770**	**0.9702**	**0.9554**
$p = 10$						
60 sec	0.9951	0.0032	0.9818	1.0177	0.9649	1.0687
10 min	1.0030	-0.0198	0.9955	1.0321	0.9920	1.0680
60 min	0.9690	-0.0094	0.9845	0.9770	0.9702	0.9554
$p = 20$						
60 sec	0.9829	0.0032	0.9818	1.0177	0.9649	1.0687
10 min	0.9897	-0.0198	0.9955	1.0321	0.9920	1.0680
60 min	0.9592	-0.0094	0.9845	0.9770	0.9702	0.9554

Table 8: Cluster-2: VAR(p) model "CPU and I/O"

Time	Multivariate		Univariate			
	RMSE	**Cov**	**RMSE**		**Cov**	
	cpu-I/O	**cpu-I/O**	**cpu**	**I/O**	**cpu**	**I/O**
$p = 1$						
60 sec	**1.0040**	**-0.0619**	**1.0421**	**0.9597**	**1.0871**	**0.9347**
10 min	**1.0095**	**-0.0365**	**1.0252**	**0.9850**	**1.0521**	**0.9896**
60 min	0.9851	-0.0125	1.0271	0.9456	1.0560	0.8957
$p = 10$						
60 sec	0.9959	-0.0619	1.0421	0.9597	1.0871	0.9347
10 min	1.0040	-0.0365	1.0252	0.9850	1.0521	0.9896
60 min	0.9811	-0.0125	1.0271	0.9456	1.0560	0.8957
$p = 20$						
60 sec	0.9837	-0.0619	1.0421	0.9597	1.0871	0.9347
10 min	0.9878	-0.0365	1.0252	0.9850	1.0521	0.9896
60 min	0.9734	-0.0125	1.0271	0.9456	1.0560	0.8957

Table 9: Cluster-1: DML model "CPU and Memory usage"

Time	**Multivariate**		**Univariate**			
	RMSE	**Cov**	**RMSE**		**Cov**	
	cpu-memory	**cpu-memory**	**cpu**	**memory**	**cpu**	**memory**
$p = 1$						
60 sec	**1.0432**	**0.0026**	**1.0429**	**0.7815**	**1.0871**	**0.9623**
10 min	0.9931	0.0161	0.9911	0.9965	0.9855	0.9924
60 min	**1.0024**	**0.0273**	**1.0026**	**0.9307**	**1.0046**	**0.8657**

Table 10: Cluster-2: DML model "CPU and Memory usage"

Time	**Multivariate**		**Univariate**			
	RMSE	**Cov**	**RMSE**		**Cov**	
	cpu-memory	**cpu-memory**	**cpu**	**memory**	**cpu**	**memory**
$p = 1$						
60 sec	0.9780	0.0209	0.9769	1.0066	0.9560	1.0143
10 min	0.9383	0.0357	0.9390	0.9820	0.8809	0.9634
60 min	**1.0126**	**-0.0015**	**1.0120**	**0.9966**	**1.0243**	**0.9925**

Table 11: Cluster-1: DML model "CPU and I/O"

Time	**Multivariate**		**Univariate**			
	RMSE	**Cov**	**RMSE**		**Cov**	
	cpu-I/O	**cpu-I/O**	**cpu**	**I/O**	**cpu**	**I/O**
$p = 1$						
60 sec	**1.0412**	**-0.0619**	**1.0429**	**0.9673**	**1.0871**	**0.9347**
10 min	1.0020	-0.0051	1.0018	1.0113	1.0030	1.0220
60 min	0.9770	0.0322	0.9768	1.0290	0.9545	1.0579

Table 12: Cluster-2: DML model "CPU and I/O"

Time	**Multivariate**		**Univariate**			
	RMSE	**Cov**	**RMSE**		**Cov**	
	cpu-I/O	**cpu-I/O**	**cpu**	**I/O**	**cpu**	**I/O**
$p = 1$						
60 sec	**1.0070**	**0.0299**	**1.0036**	**1.0085**	**1.0139**	**1.0165**
10 min	1.0172	-0.0171	1.0172	1.0266	1.0339	1.0554
60 min	0.9513	0.0076	0.9509	0.9409	1.9042	0.8852