Towards Predictive Monitoring of Research Infrastructures

Jedrzej Rybicki
Juelich Supercomputing Center (JSC)
Juelich, Germany
Email: j.rybicki@fz-juelich.de

Abstract—**Nowadays, both modern computing infrastructures, as well as their scientific workloads exhibit far reaching complexity and diversity. Therefore, it is increasingly hard to comprehend and manage them in an efficient manner. In particular, it can lead to under- or overuse of resources. A prerequisite for efficient resource allocation is the ability to predict their usage. In this paper, we use real world workloads recorded in a modern research infrastructure to conduct load prediction. We use well established statistical model (ARIMA) and achieve good results in predictions. The big data challenge is here given not by the sheer size but rather by the speed of data collection and processing. The quicker the prediction is made, the more time is available for actions. Such predictions can be included in monitoring systems to give human operators better insights into the status of their infrastructures and lead to better load distribution.**

Keywords–Fast Data; Load prediction; Monitoring; Research infrastructures; Predictive model.

I. INTRODUCTION

This short paper marks the beginning of our research towards automatic monitoring and managing of distributed computing resources. Our main motivation stems from the increasing complexity of the distributed research infrastructures we have to manage. They comprise of High-Performance Computing (HPC), High-Throughput Computing (HTC), and Cloud resources. The resources are used by researchers from different disciplines cooperating across the borders to accomplish ambitious scientific goals. The usage patterns emerging from this far-reaching diversity are challenging for the underlying infrastructure. Yet, a mismatch in mapping of users demand on hardware resources may potentially result in high cost, low performance, and users' disappointment.

One of the prerequisites for an efficient resource allocation is the load prediction. Obviously, in case of increasing load more hardware resources need to be allocated, conversely decreasing load can trigger reallocation of idle resources. High and very high levels of load can also be an indicator that particular tasks were mapped on the wrong class of resources, e.g., Cloud nodes instead of HTC, and an adjusting action should take place. Given the fact that such an adjustment takes some time, it is of crucial meaning to predict the future demand directions to give the resource providers sufficient time for the required changes. Lastly, a sudden change of load can be an indication of a hardware failure or malicious action, potentially requiring human intervention. In this paper, we will examine if it is possible to predict future infrastructure workload based on the historical measurements.

Collecting and analyzing workload data is a challenging task. The single measurements are not large in size, but they are only meaningful if analyzed quickly. An application of decentralized approach for data evaluation would have clearly some interesting properties but our goal is to use existing monitoring infrastructure and extend it with a prediction capability. In this approach, data are collected locally and then transferred to the central unit for visualization, analysis, and storage.

Load analysis and prediction pave the way for an algorithm driven infrastructure, in which no operator intervention is required, but rather the infrastructure itself makes optimal usage of the available resources. Given the aforementioned complexity of the current research infrastructures, such an automatic, algorithm-driven support is not a fancy vision but rather an urgently required solution. It becomes increasingly hard for human operators to grasp the tendencies and problems in the managed infrastructures, and to make and implement reallocation decisions.

In this paper, we mainly focus on using the model for predicting future values of load in the test infrastructure. But a good model itself provides an insight in the underlying process which generated the experimental data. In our case, the model can potentially help us in understanding how the resources are used. This knowledge, in turn, can influence the choice of resources deployed and overall improve the quality of the service offered to the users.

The rest of the paper is structured as follows. In Section II, we shortly summarize some of the previous work on broadly defined predictions. Section III comprises our initial results. We provide information on our setup, analyze stationarity of the process we are going to predict on, and describe both the model used and its predictive performance. We conclude the paper with an outlook on future work in Section IV.

II. RELATED WORK

There is some research done in the broad field of predictions that might be relevant and inspirational for our work. Amjady analyzed electric load to issue short term prediction of future load demand [1]. This is crucial for the economic and secure operation of power systems. There are some differences between electric load and hardware utilization, especially as the first one exhibits a strong seasonal component. Nevertheless, the existing body of work in this field provides meaningful insights into possibility and methodology of predictions, as well as their economic relevance.

Statistical Process Monitoring (SPM) using either statistics or machine learning methods can help in modeling and diagnosing of industrial process operations and production results [2]. The overarching goal here is to conduct preventive service before faults occur. The setting is similar to ours but the systems analyzed differ in many ways. The work, however, underpins the needs of algorithm help in managing complex infrastructures and again give valuable hints on methodology.

Roberts et al. [3] used profiles of power consumption to detect malware infections on the host machines. The deviation of recorded power consumption during predefined task were detected with kernel-based Support Vector Machines (SVM). The authors recorded four voltage and four corresponding current channels and achieved perfect detection of malware infections. Their problem statement differs substantially from ours, yet clearly shows the potential of analyzing CPU loads generated by applications.

Tao Li et al. [4] conducted a relevant study in a field similar to ours. They proposed an algorithm combining linear regression and the improved Knuth-Morris-Pratt match to predict the next moment load in the examined Cloud environment. Furthermore, they conducted initial studies on automatic Cloud resource reallocation based on this prediction. This work serves as a good comparison for our results as we use a slightly different setting and different predicting algorithm. Particularly, the authors used Cloud simulator for data generation whereas we use traces from real infrastructure. There exists also a more generic Patent by Wolters [5] on predictive monitoring of IT structures. It provides less technical details, but proves the commercial relevance of the work. There also exists a body of work which uses prediction for efficient scheduling of the computation jobs [6] [7]. The techniques used range from neuronal networks to statistical models with no clear favorite.

Our cross-disciplinary survey was intended to give us an overview on methodology used in different kinds of predictions. We have seen that techniques ranging from simple statistical models up to multi-layer neural networks are used. An open question is what method to use when predicting future values of a given feature. Artificial Neural Networks (ANNs), tend to perform well in some situations and fail in others. The question "Neural networks: Forecasting breakthrough or passing fad?" posed first by Chattfield [8] remains open. Thus, in this first attempt on the problem, we decided to stick to classical, and well-understood methods and explore the potential of the idea. In the future, work we might turn towards more sophisticated methods. However, there is some evidence that simple modeling methods tend to perform better than their more sophisticated counterparts [9]. We were also partly motivated by the need to better understanding of the process that generates the experimental data. Such an understanding is better achieved with simple models.

III. EVALUATION

In our evaluation, we follow the well-established Box-Jenkins methodology [10]. In this approach, the analysis comprises of three phases: model identification, parameter estimation, and model checking.

A. Data and tools

For our experiments, we used data collected within our production infrastructure. It consists of a number of hosts offering different kinds of services for academic users. The services range from single-sign-on systems, through simple storage offerings, up to Grid-based computing end points. We use nagios [11] to perform periodic checks of the hosts and services running. The tests comprise of host reachability, disk usage, and relative CPU load, among others. Measurements were done over two days with 5 minutes frequency. For this initial evaluation, only load values were used.

TABLE I. STATIONARITY TESTS.

	Train	Test
Mean	0.0158	0.0183
Variance	0.0007	0.0008

The analysis was done with Jupyter Notebooks [12] and we used popular Python libraries like pandas [13], statsmodels [14], and matplotlib [15]. In this project, we follow the best practices for structuring data science projects [16].

B. Model creation

To model and predict load changes, we use a very popular Autoregressive Integrated Moving Average (ARIMA) model. In the identification phase, we check the applicability of the model. The first question that needs to be answered when analyzing time series, is the stationarity of the process. We used two metrics for that. Firstly, the series was split in a 60-40 ratio into parts which we later used for training and testing of the model. Then, both mean and variance for the two intervals were calculated. Only slight differences in values obtained for both intervals (see Table I), suggest the stationarity of the series. A more sophisticated way of establishing if a series is stationary is the Augmented Dickey–Fuller test [17]. The value calculated in the test was -3.611 laying beyond the 1% significance threshold (-3.441 in our case). Given the results of both tests, we can assume the series is close to stationary and proceed in model creation that should be able to predict future values. The small changes in the variance will be accounted for by differencing in the ARIMA process.

The ARIMA model is characterized by three parameters. The number of lags for autoregression is denoted as p. Parameter d defines the number of times the observations are differenced. Finally, the q parameter describes the size of the moving average window of the model. Since p basically defines the relationship between subsequent observations, a good hint for estimating its value is to use autocorrelation plot. On Figure 1, we can see that there is strong autocorrelation within the series, the first time the curve crosses 0 for lag 3, thus we take $p = 3$ in our model. Because the process seems to be close to stationary, we used a small value for $d = 2$. Finally, the remaining q was taken as 1, meaning that we perform, more or less exponential smoothing of the values, what makes sense for CPU loads. Furthermore, it is common to set at least one parameter value to one, as otherwise a risk of overfitting is high.

After fitting the model to data, we estimated its quality. The plot of model residuals distribution (Figure 2) shows that mean value of residuals is close to zero (0.381×10^{-3}) and the distribution is normal, confirming the good model fit.

C. Prediction

After successfully modeling the CPU load changes with ARIMA model, we tested how good it can predict the future values of load. This metric is crucial for the envisioned automatic steering of resource allocation. If the model is unable to predict correct values it would not be possible to account for them.

The predictions were made in a rolling manner. Firstly, we split the available data in two parts: train and test. The

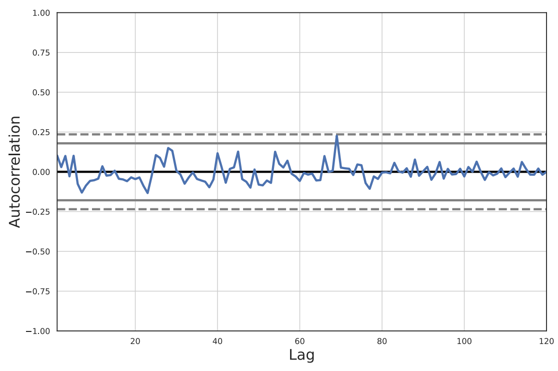

Figure 1. Autocorrelation plot of the measured CPU load.

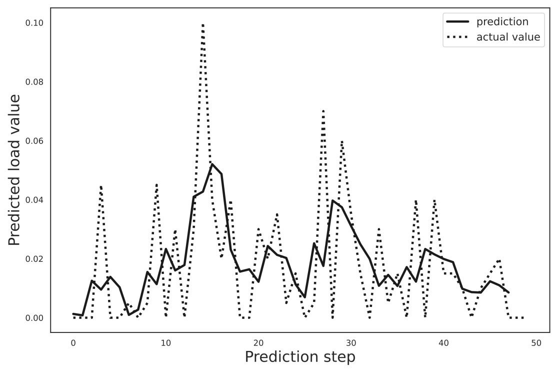

Figure 3. Predictions of CPU load made with the model.

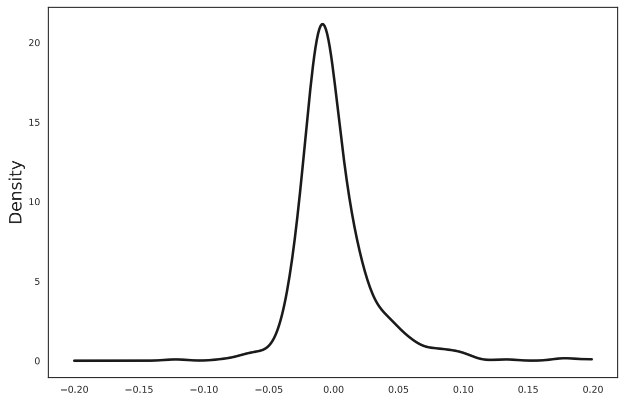

Figure 2. Distribution of model residuals.

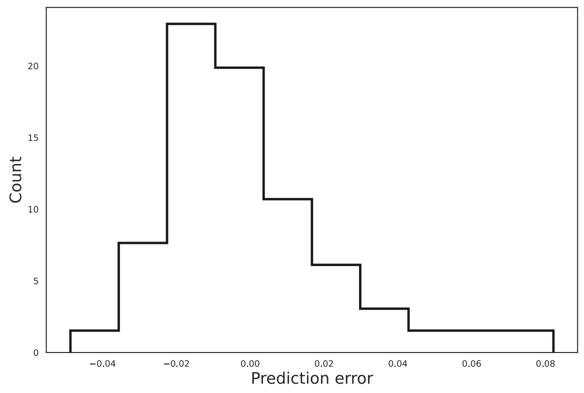

Figure 4. Distribution of prediction errors.

training set comprised of 350 measurements and was used to fit ARIMA model, with previously selected parameters $p = 3$, $d = 2$, and $q = 1$. Subsequently, a series of 40 predictions (corresponding to about 3.5 hour interval) were made. After each step, the predicted value was compared to the actual value from the test set. Finally, the model was retrained with the train set extended by the actual value. Figure 3 shows the quality of the predictions made by our model. The dashed line shows values from the test set, solid line represents predictions. One can see that the model is a little bit conservative and sluggish but overall predicts the values pretty well. The mean squared error of all predictions was 0.698×10^{-3}. Prediction error can be calculated as a difference between the actual and predicted value. The distribution of such an error is depicted on Figure 4. Most of the errors reside in the area close to 0.

IV. Conclusion and future work

In this paper, we modeled the utilization of an IT infrastructure and used this model to predict future changes in workload. The proposed model seems to be able to predict changes with acceptable accuracy. The results obtained in this initial study corroborate our assumptions and motivate us to further work on this subject.

Currently, we only used load values measured over time, but it will be interesting to see if other values available like disk or memory usage could help to improve the predictions. The data are available, but the current model cannot really deal with multidimensional data well. The straight-forward solution would be to use few ARIMA models to run in parallel and predict changes in each time series separately. Subsequently, the results would be combined to give a prediction of the holistic state of the resource.

Since our motivation is of a practical manner, we would like to integrate the model predictions into our infrastructure, monitoring solution. This will be the first step in assisting the human operators in grasping the tendencies in the managed infrastructure but also a field test for the proposed model.

References

[1] N. Amjady, "Short-term hourly load forecasting using time-series modeling with peak load estimation capability," IEEE Transactions on Power Systems, vol. 16, no. 4, Nov. 2001, pp. 798–805, ISSN: 0885-8950.

[2] S. J. Qin, "Survey on data-driven industrial process monitoring and diagnosis," Annual Reviews in Control, vol. 36, no. 2, 2012, pp. 220–234, ISSN: 1367-5788.

[3] R. A. Bridges, J. M. H. Jiménez, J. Nichols, K. Goseva-Popstojanova, and S. J. Prowell, "Towards malware detection via CPU power consumption: Data collection design and analytics," in Proceedings of 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), 2018, pp. 1680–1684, ISBN: 978-1-5386-4388-4.

[4] T. Li, J. Wang, W. Li, T. Xu, and Q. Qi, "Load prediction-based automatic scaling cloud computing," in Proceedings of International Conference on Networking and Network Applications (NaNA), Jul. 2016, pp. 330–335, ISBN: 978-1-4673-9803-9.

[5] T. J. Wolters, "Predictive monitoring and problem identification in an information technology (IT) infrastructure," Patent US7 107 339B1.

[6] H. Jiang, H. E, and M. Song, “Multi-prediction based scheduling for hybrid workloads in the cloud data center,” Cluster Computing, vol. 21, no. 3, Sep. 2018, pp. 1607–1622, SSN: 1573-7543".

[7] X. Tang, X. Liao, J. Zheng, and X. Yang, “Energy efficient job scheduling with workload prediction on cloud data center,” Cluster Computing, vol. 21, no. 3, 2018, pp. 1581–1593, ISSN: 1386-7857.

[8] C. Chatfield, “Neural networks: Forecasting breakthrough or passing fad?” International Journal of Forecasting, vol. 9, no. 1, 1993, pp. 1–3.

[9] K. C. Green and J. S. Armstrong, “Simple versus complex forecasting: The evidence,” Journal of Business Research, vol. 68, no. 8, 2015, pp. 1678–1685, special Issue on Simple Versus Complex Forecasting.

[10] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, Time Series Analysis: Forecasting and Control, ser. Holden-Day series in time series analysis and digital processing. Holden-Day, 1976, ISBN: 978-0-81621-104-3.

[11] W. Kocjan and P. Beltowski, Learning Nagios, 3rd ed. Packt Publishing, 2016, ISBN: 978-1-78588-595-2.

[12] Project Jupyter. [Online]. Available: https://jupyter.org/ [retrieved: Jan., 2019]

[13] pandas: Python data analysis library. [Online]. Available: http://pandas.pydata.org/ [retrieved: Jan., 2019]

[14] S. Seabold and J. Perktold, “Statsmodels: Econometric and statistical modeling with Python,” in Proceedings of 9th Python in Science Conference, 2010, pp. 57–61, ISBN: 978-1-4583-4619-3.

[15] J. D. Hunter, “Matplotlib: A 2D graphics environment,” Computing In Science & Engineering, vol. 9, no. 3, 2007, pp. 90–95.

[16] J. Rybicki, “Best practices in structuring data science projects,” in Proceedings of 39th International Conference on Information Systems Architecture and Technology (ISAT). Springer International Publishing, 2019, pp. 348–357, ISBN: 978-3-319-99993-7.

[17] D. A. Dickey and W. A. Fuller, “Distribution of the estimators for autoregressive time series with a unit root,” Journal of the American Statistical Association, vol. 74, no. 366a, 1979, pp. 427–431, ISSN: 0162-1459.