APPLICATION OF TIME SERIES TO PERFORMANCE ASSURANCE OF BIG DATA ENVIRONMENT

Y. BALASANOV , PhD,
University of Chicago

B. ZIBITSKER, PhD
President and CEO BEZNext, Emeritus professor of BSUIR

T. BAKANAS
University of Chicago Graham School of Continuing Liberal and Professional Studies

E. HAMMOND
University of Chicago Graham School of Continuing Liberal and Professional Studies

M. ISLAS-MARTINEZ
University of Chicago Graham School of Continuing Liberal and Professional Studies

CEO BEZNext, Chicago, USA
E-mail: bzibitsker@beznext.com

Abstract. The selection of the Big Data algorithms, YARN rules and infrastructure can affect accuracy, performance and scalability of Big Data Applications.

We will present a methodology and algorithms for proactive performance management. Every hour collected measurement data are aggregated into workloads representing each lines of business. Each workload has three profiles, including 1) performance (response time and throughput), 2) resource utilization and 3) data usage profiles. Profiles represent Workloads' Time series. This information is used as input for exploratory analysis techniques specific to time series data. The data are transformed into stationary Time Series and an analysis to select the best time series model (ARMA, VARMA) is conducted. Historical data are used to identify past exceedances which are utilized as predictors or outcome variables to build a classification model.

We will review short term prediction, seasonal peaks identification, diagnostic and root cause analysis Performance Assurance algorithms enabling proactive performance management of Big Data Application.

Key Words: Missing data, time series data, anomaly detection, performance prediction, big data, performance assurance.

*Introduction.*Big Data applications implementation success depends on design, implementation and management decisions.

Selection of Machine Learning (ML) algorithms and libraries, software timing parameters and rules assigning priorities and concurrency limitations. for software subsystems like YARN, Kafka, Spark, Storm and selection of the architecture and hardware configuration also can affect performance, scalability and cost.

Difficulty of managing complex multi-tier, distributed, virtualized, parallel processing environment creates an uncertainty and high risk of performance surprises for Big Data applications.

Failure to meeting Business Service Level Goals (SLG) can affect ability of business to make real time decisions. Therefore, our goal is to implement performance prediction technology justifying proactive measures necessary to meeting SLGs in growing and constantly changing environment.

Predictive models are based on performance measurement data continuously generated by each of Big Data subsystems. It includes response time and throughput, resource utilization, (CPU, memory, storage and network) by Systems, Users and Applications and Data usage, including read/write ratio, level of parallelism during accessing data. We aggregate measurement data into hourly profiles by business workloads / line of business.

Each workload use different set of applications. The number of users and volume of data are growing. The number of new applications is increasing as well. It increases the contention for resources and affects performance.

We aggregated performance measurement data into 44 business workloads, characterized by performance, resource utilization and data usage as a time series. All workloads compete for the same resources and affect each other performance, so they should be tested for the degree of cointegration to determine the type of time series analysis that can be conducted. Different applications are active during different time of the day. Hardware and software configuration periodically changes as a result of tuning or hardware outages. Many workloads have a random arrival rate and service time [Buzen].

Gaps in measurement data and randomness of the performance of workloads characterization affect the selection and use of the modeling algorithms.

Queueing network models for example, assume the Poisson distribution of the independent variables. Calibration of the multi-tier distributed, virtualized parallel processing environment supporting mix workloads have many challenges [z and S and L]. ML algorithms which we implemented use a lot of resources and do not scale well.

In this paper we will illustrate how time series algorithms can be used for performance prediction, determining seasonality and performing the diagnostics and root cause analysis to justify Performance Assurance decisions and implement dynamic performance management. We will review problems of data preparation, fitting and applying time series and categorical models. We will present a methodology of selecting the segment from an incomplete time series that maximizes the number of original data points while minimizing the amount of data that need be imputed, review a point forecasting model for each performance indicator for each workload to detect and predict extreme positive anomalies in key performance variables

Data Collection, Aggregation and Preparation. BEZNext agents continuously collect measurement data from Linux, YARN, Kafka, Storm, Spark, Cassandra and other Big Data subsystems [BZ]. Measurement data are aggregated by workloads in hourly performance, resource utilization and data usage profiles to represent the aggregated activity of users and applications supporting each line of business. [BZ]

Data set used during this study contains 44 workloads with 15 variables each. List of Variables includes: Date, Hour, Workload Name, Number of Parallel Jobs, Number of Parallel Requests, Number of Network Messages, Total Number of Execution, Total Arrival Rate, Average Response Time, Total CPU Time, Total # IO, Total Delay Time, Total CPU Utilization, Concurrency Limit, CPU Utilization Limit.

Any changes in number of users, changes in volume of data, implementations of new applications or modifications of existing applications and corresponding changes in performance and usage of resources are reflected in the hourly workload profiles. The Average Response Time and Total Number of Executions per hour are the Key Performance Indicators (KPIs), which are used in the time series and categorical models. All other variables are used as predictors for the KPI.

The methodology presented in this paper focus on data imputation, time series forecasting, seasonal peaks and anomalies determinations, performance problems and their root causes prediction and development proactive performance management recommendations.

Imputation. Time series data have a significant number of time intervals without observations.

In a 2010 paper in *the American Journal of Political Science*, two researchers, Honacker and King, apply a particularly apt analogy to the problem. If a data set is a cheese, most real data sets are Swiss, filled with holes. With some data sets, one could simply drop the missing data pieces. When dealing with time series data, however, the missing chunks cannot be dropped as both auto-regressive and moving average predictive techniques depend upon consecutive values.

Since for time series data melting the cheese down and reforming with no holes doesn't work, the other option is to fill the holes. Filling in long chunks of data, however, runs the risk of either artificially reinforcing or removing any seasonality in the time series. As Honacker and King put it, "if you fill the holes in the cheese with peanut butter, you should not pretend to have more cheese!" (Honacker and King, 2010).

Honacker and King suggest a method of data filling that is based upon building an imputation model and combining the imputed values with expected values from historical analysis of similar data sets (Honacker and King, 2010). Their work, while robust, does not cleanly apply to the problem of

Big Data systems performance forecasting, where some of the workloads might be not active for some time and some of the nodes might having problems and not available during period of time. Imputation over such a long period loses its validity.

We applied algorithm that selects a time frame based on maximizing length while minimizing hours filled is selected as the best method of imputation for this data. In terms of the cheese analogy mentioned in the Background section, it identifies as large a portion of the Swiss as it can where all the holes all remain under a certain size. The idea is that small holes can be reliably filled with a simple algorithm without biasing the data. Using this methodology can produce unbiased data sets capable of supporting further analysis.

The algorithm developed has two parameters, neighborhood size and minimum size. The neighborhood size is the number of values before and after a missing value that will be used for imputation. The minimum size is the minimum length of the selected segment. There are cases where a Workload has multiple segments that meet these requirements. In those cases we select the most recent segment.

The proposed methodology is described below:

- Select a time series and identify all missing values within
  - For our analysis, each workload is examined over the length of one year.
- For each missing value extract a neighborhood of adjacent values.
  - Our analysis used a total neighborhood size of ten, which includes the five values preceding the missing point and five after.
- Each missing value is evaluated.
  - If there is at least one true value on both sides of the point within the neighborhood, then missing value is imputed with average of all the existing values within the neighborhood.
  - If there are no existing values on both sides of the point, then the point remains missing.
  - If a value cannot be imputed, then the segment is ended and a new segment starts at the next non-missing value.
- The best subset is selected.
  - The short-term analysis uses the most recent segment.
  - The anomaly prediction analysis uses the largest segment.

This subset selection methodology allows for the consistent treatment and cleansing of time series data across many distinct time series with differing degrees of missing data.

In the graphs below we see how the imputation algorithm selects both the ‘largest’ and the ‘best’ segment for the Workload DBC. For DBC the ‘Largest’ segment occurred from 2015-05-03 through 2015-09-18. The imputation algorithm selected a different, more recent, time frame for the ‘Best’ segment. Both perform as they are supposed to, a larger set of data is selected to feed the anomaly prediction from ‘Largest’ and the most recent set of data is selected to feed the time series model from ‘Best.’

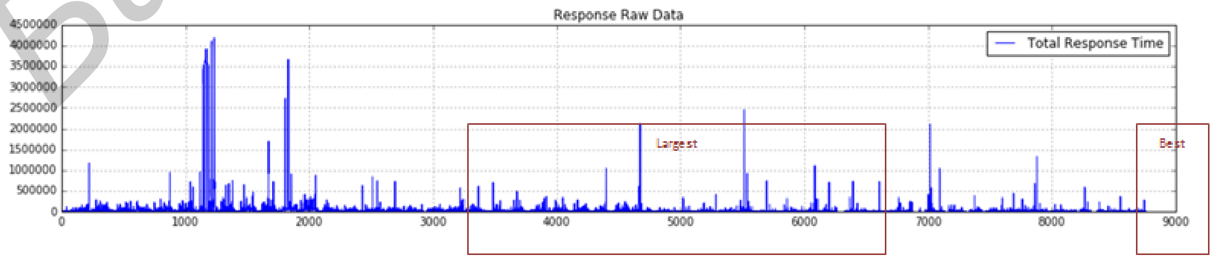


Fig. 1. DBC Workload Total Response Time Full Year

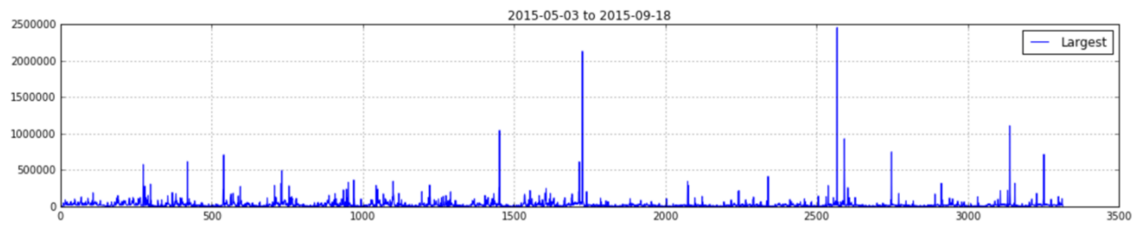


Fig. 2. DBC Workload Total Response Time Largest Segment

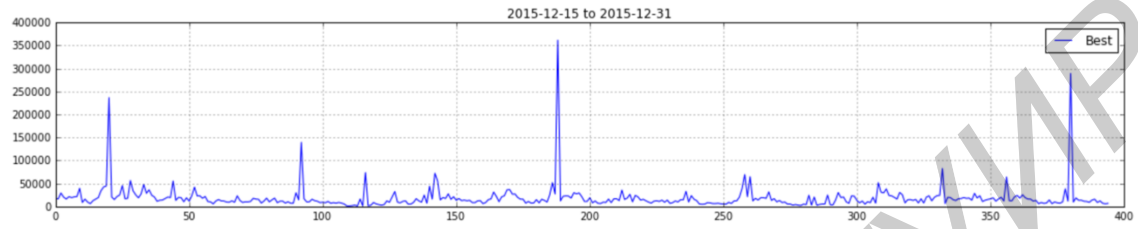


Fig. 3. DBC Workload Total Response Time Best Segment

The Workload Best and Longest segments have return a variety of lengths across the 44 Workloads. In many cases a segment from a Workload is selected as both the Best and the Longest. As can be seen from the histograms below the 'Best' segments trend shorter than the 'Longest' segments. This is due to the requirement that the 'Best' segment be the more recent available.

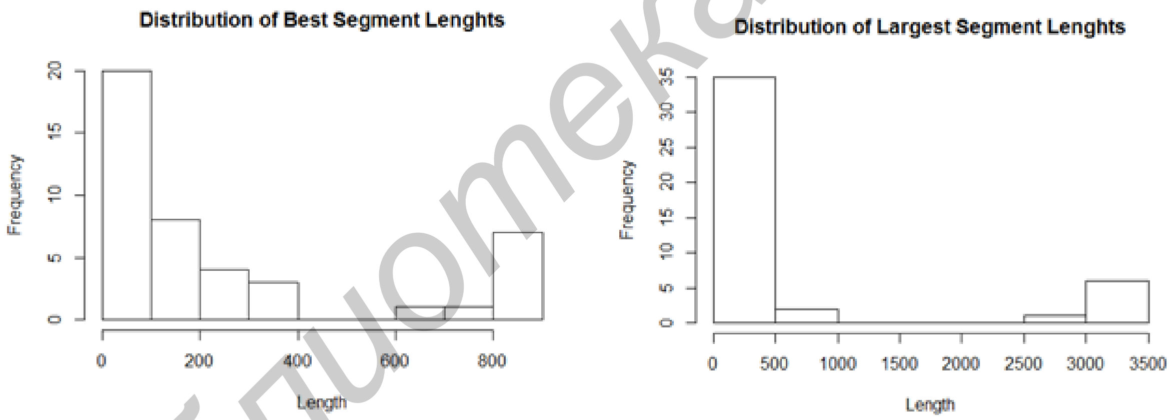


Fig. 4. DBC Workload distribution of best and largest segment lengths

The imputation methodology is reliable and flexible. From 44 Workloads with varying data quality it selected time periods capable of supporting two types of analysis. We used methods based on linear Time Series models which do not require very long samples.

Time Series Forecasting. Big Data systems produce multiple time series system status variables that are likely to be related. Aboagye-Sarfo, Mai, Snfilippo, Preen, Stewart and Fatovich (2016) present VARMA forecast model as a technique designed for modeling multiple time series simultaneously. Aboagye-Sarfo et all use VARMA for predicting emergency department demand in Western Australia. They compare results that use benchmark univariate autoregressive moving average (ARMA) and Winters' techniques. Their research demonstrates that a VARMA model provides a more accurate forecast than the normal or standard univariate ARMA and Winters' methods.

We followed a similar methodology. After assessing stationarity for each variable in each workload a VARMA and ARMA model is fit to each of the KPI variables. We expect the VARMA model to be the strongest fit, but by running both time series techniques we are able to customize the results for each variable workload combination..

Example of Time Series Forecasting for DBC workload

Overall, ARMA models had the best forecasting performance with all three models out-performing the naïve *Yt* – 1 forecasting model. Average Response Time had an MASE value over one which indicates that using *t* – 1 for a prediction would have been more accurate than the VARMA model.

Table 1. DBC 1-Hour Forecast MASE

Variable	**ARMA**	**3-VARMA**
Average Response Time	0.9439	1.240
Total IO	0.8184	0.835
Total CPU	0.9268	0.896

Extreme outliers are a major obstacle in our forecasts, with the VARMA models proving to be particularly sensitive. The figures below show that the VARMA forecasts an extreme outlier around hour 85, where none occurs in the original data.

Stationarity. The first step in building a time series forecasting model is testing for stationarity in the data. If the time series is not stationary, then it must be transformed into a stationary state otherwise the model will not be stable enough to generate accurate forecasts. Testing for stationarity can be done quantitatively using statistical tests such as the Augmented Dicky-Fuller (ADF) and the Kwiatkowski–Phillips–Schmidt–Shin (KPSS) tests. These tests evaluate a time series for the presence of a unit root or trend-stationarity. Stationarity testing can also be done be qualitatively by examining autocorrelation plots for exponential decay.

Numerous techniques (de-trending, differencing, seasonal adjustment, log transformation, smoothing) can be used to transform non-stationary data into stationary data. This analysis uses the differencing and seasonal adjustment techniques to achieve stationarity in the data.

Overall, the stationarity tests indicate that our differencing selection methodology produced accurate results. Before each model was constructed ADF and KPSS tests verified that the data used to train the series models was stationary. No variables were found to be non-stationary after differencing.

Differencing. A stationarity transformation algorithm was developed for this analysis. The algorithm is a brute force approach that differences each variable with a set of predefined differencing functions. Then, metrics are calculated from the differenced data sets and ranked. The differencing functions were selected based on our knowledge of the problem domain and segment lengths identified in the imputation stage. The differencing techniques evaluated for each variable are:

– No difference.
– First-order difference.
– Second-order difference.
– Daily difference.
– Weekly difference.
– Monthly difference.
– First-order + daily difference.
– First-order + weekly difference.
– First-order + monthly difference.

Each differencing function is applied to each variable of each Workload and a set of ranking parameters are calculated. These parameters are used to select the optimal differencing function. The ranking parameters are

– Mean absolute scaled error of ARMA in-sample predictions after integration (lower is better).

– Root mean squared error of ARMA in-sample predictions after integration (lower is better).

– Count of autocorrelation function points with the confidence interval (higher is better).

– Count of partial autocorrelation function points with the confidence interval (higher is better).

– Absolute average of autocorrelation function points (lower is better).

– Absolute average of partial autocorrelation function points (lower is better).

– The differencing function is automatically rejected if one of the Augmented Dicky Fuller, Kwiatkowski–Phillips–Schmidt–Shin and Phillips–Perron tests indicate nonstationary.

This brute force process allows for many independent time series with different types of stationarity and seasonality to be accounted for in an automated fashion. Each differencing function also has a corresponding integration function that is used to transform the forecasts generated by the ARMA/VARMA models. Refer to Appendix 2 for example Python code of the differencing functions used in this analysis.

Five unique differencing functions were selected. However, the most commonly selected approach was to use no differencing. The most common differencing function was first-order differencing. Thirteen variables had seasonality detected by our algorithm.

Table 2. Differencing Functions Selected

Function	Count
no_difference	52
first_order_difference	41
weekly_difference	7
daily_difference	5
second_order_difference	3
first_order_weekly_seasonal_difference	1

For instance, the differencing algorithm determined that the three variables in the DBC Workload were already stationary and selected the “No Difference” differencing function. In contrast, the Total CPU variable of the SALES Workload was non-stationary. We will illustrate the differencing selection algorithm by using the SALES Workload.

The autocorrelation function of the SALES Workload indicates non-stationarity and possible seasonality.

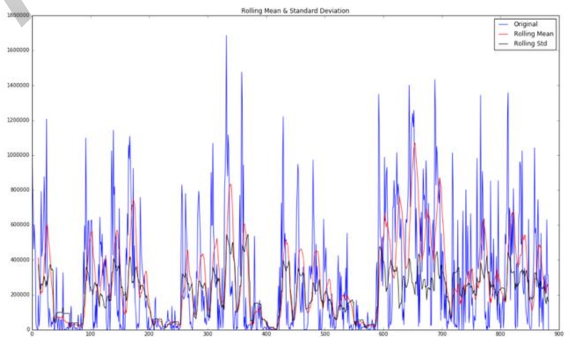


Fig. 5. SALES Workload Total CPU raw data

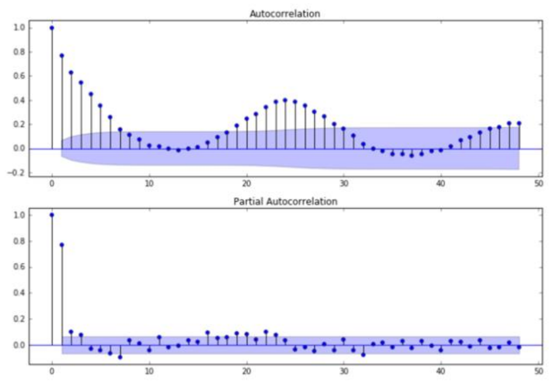


Fig. 6. SALES Workload Total CPU ACF and PACF plots

The periodogram is extremely noisy, with no clear indication of seasonality.

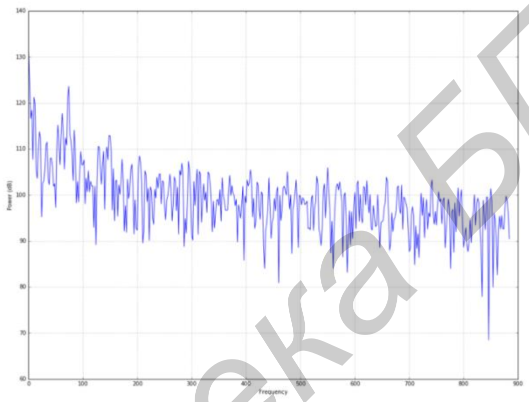


Fig. 7. SALES Workload Total Periodogram plot

Again, there is a disagreement between the periodogram and the autocorrelation plots. Autocorrelations indicates that there is daily seasonality, while the periodogram does not. The differencing selection algorithm selected “First-Order Difference” as the differencing function.

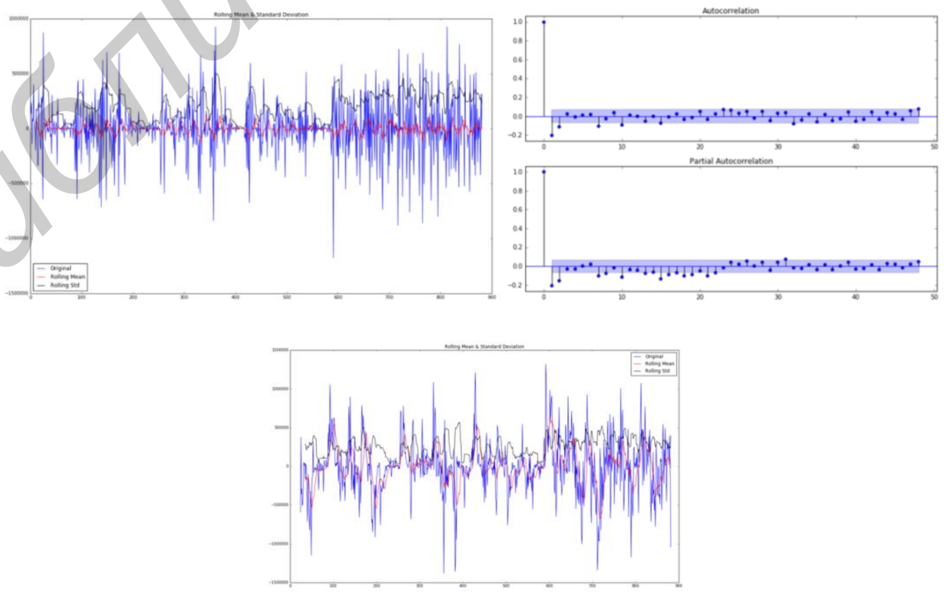


Fig.8. SALES Workload Total CPU First-Order Difference

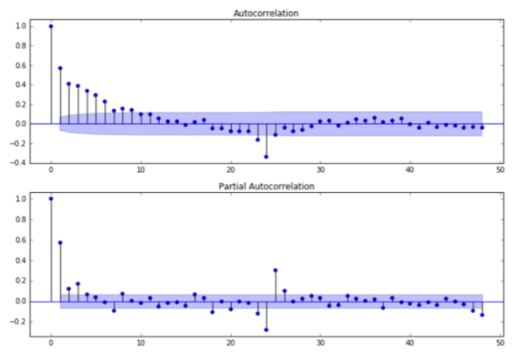


Fig. 9. SALES Workload Total CPU Daily Difference

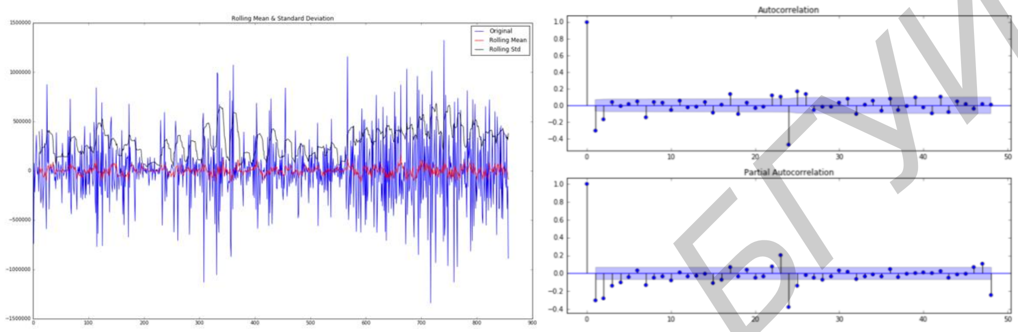


Fig. 10. SALES Workload Total CPU First-Order + Daily Difference

Three autocorrelation plots above show that the differencing selection algorithm correctly identified the best differencing technique. The "First-Order" differencing function has the best autocorrelation plot of the three.

Cointegration. We hypothesize that the Workload data from BEZNext contains variables that are dependent on each other. This lends itself to VARMA time series analysis. A requirement of a VARMA model, however, is that data be cointegrated.

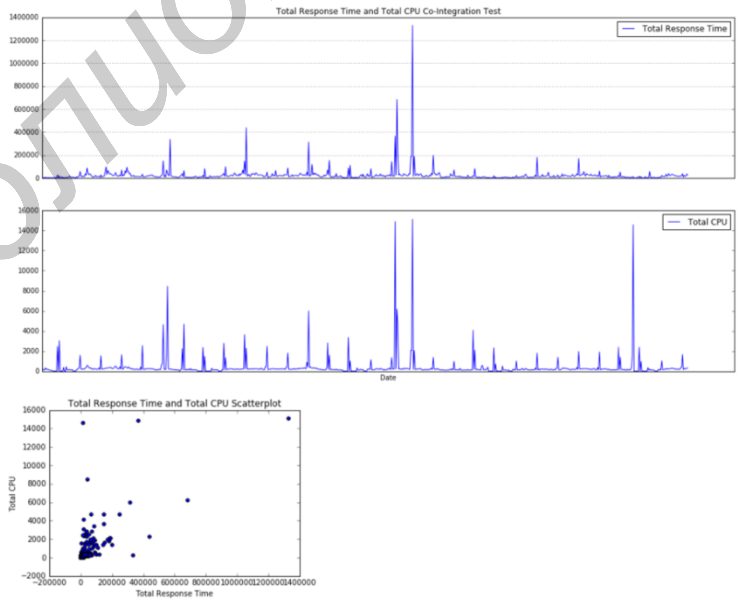


Fig. 11. DBC Workload response time vs CPU Cointegration results

Our approach uses the Augmented Dickey-Fuller and the Johansen tests to determine if any variables are cointegrated. These are standard tests that are applicable to a wide range of time series data.

We find that the Key Performance Indicators mostly are cointegrated with each other. In some Workloads, the Average Response Time Variable is not cointegrated and is excluded from the VARMA time series model.

The ADF and Johansen tests both show cointegration between the CPU and IO variables. We run these tests for each Key Performance Indicator before building the VARMA model for each Workload. Any analysis of Workload data from Big Data systems should undergo testing to determine if its descriptor variables display cointegration.

Model Construction. ARMA and 3-variable VARMA forecasting models are constructed for each variable, Both models follow a similar workflow:

1 If VARMA, test for cointegration.
2 If necessary, difference data.
3 Train models for all *p, q* combinations between 0, 0 and 5, 5.
4 Select best *p, q* values using AIC as the evaluate metric.
5 Generate forecasts.
6 If using differenced data, integrate previous value(s).

The univariate ARMA approach is the most straight-forward model and directly follows the above workflow. One model is constructed for each of the Key Performance Indicators in each Workload. This approach has the fastest training and cross validation times by a large margin.

VARMA forecasts are generated for variables without cointegration. A single VARMA model is generated for each Workload.This approach has slower training and evaluation times compared to ARMA.

This approach constructs three VARMA models per Workload, one for each Key Performance Indicator, with each model containing the fields cointegrated with the target Key Performance Indicator. This approach seems to the most through and our initial hypothesis was that this technique would be the most accurate method. However, this approach proved to be very cumbersome, with extremely long training and evaluation times. The large number of variable combinations often resulted in models with invalid states. Many models did not converge or had errors when fitting the model to the data.

Refer to Appendices 3 through 5 for Python code samples showing the model construction and cointegration testing.

Model Evaluation. Cross validation evaluates the forecasting accuracy of the ARMA and VARMA models. A rolling-origin forecast cross validation procedure follows these steps:

1 Create an 80-20 train test split within each Workload time series.
2 Train the model on the train data set.
3 Forecast $T + 1$ value.
4 Roll the origin forward in the train data set by adding the Salesl $T + 1$ value.
5 Retrain the model.
6 Forecast $T + 2$ value.
7 Continue until all values in the test dataset have been forecasted

We find the VARMA usually outperforms the ARMA model in the rolling forecast origin cross validation. However, we also want to see how these forecasts perform with multiple forecasts instead of just forecasting a single hour. We look at 3 to 6 hour forecast periods. The forecast accuracy decreases as it predicts further to the future.

We also noticed that in some cases accuracy increases proportionally to the forecasting intervals. In these cases, the models revert to predicting a constant value around 4 hours in the future. This lead to accurate forecasts in T + 4 or later if we had properly selected the differencing function and there were not a significant number of anomalies in the data.

Seasonality. Generally, qualitative approaches are used for identifying seasonality in time series data. Patterns indicating seasonality can often be spotted in the power spectral density or autocorrelation plots after careful review. However, these approaches can result in inconsistent interpretations and tend to be very time-consuming.

A periodogram plot of a time series' power spectral density (PSD) is one of the most common ways of identifying seasonality. When focused on a single time series, techniques can be applied to draw a signal out of noisy data (Fernandez et all, 2016). However, proper interpretation can be extremely challenging. The plot below shows the Total IO variable from the DBC Workload. At approximately daily intervals there are spikes in the time series which indicates that there may be some seasonality in this data.

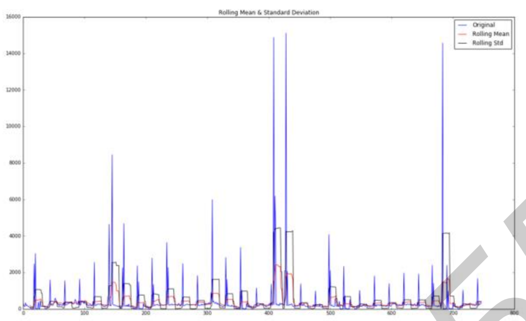


Fig. 12. DBC Workload Total CPU raw data

The periodogram below indicates that there might be a daily seasonal pattern as well. The left plot is the standard power spectral density while the plot on the right is the transformed PSD plot with hours on the X-axis instead of frequency. There is a large spike at hour 24 in the transformed plot indicating a daily seasonal pattern.

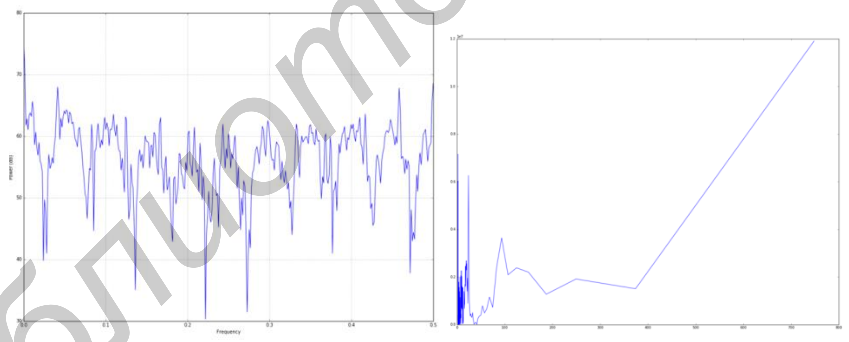


Fig. 13. DBC Workload Total CPU Periodogram

An autocorrelation plot can also be used to identify seasonality. The autocorrelation and partial autocorrelation plots are shown below and clearly indicate that there is no seasonal pattern. In fact, the data already appears to be stationary with very little autocorrelation.

The above example shows how difficult it can be to perform these qualitative interpretations. The raw data seems to contain some form of seasonality. The periodogram while very noisy shows some element seasonality may be present. However, the autocorrelation plot is a stark contrast. It shows no indication of seasonality at all.

The other challenge in identifying seasonality is the sheer number of Workloads in our data set. Evaluating only the Key Performance Indicators in each of the 44 Workloads requires 132 separate analyses and a full evaluation of all variables requires 528 analyses.

Consistent interpretation in a timely manner requires an automated approach. Differencing can be used to both transform a time series into a stationary form and to capture any seasonal patterns if they exist. The key is to select a differencing function that addresses both conditions.

Identification of the workload's seasonality enables organization of the proactive planning of changing workloads' management rules, including priorities, concurrency and resource allocation.

Diagnostic and Root Cause Analysis. Many businesses are making business decisions in real time. It creates a pressure to develop real time Big Data applications capable to extract necessary attributes from the stream of data supported by Kafka or Flume and apply ML algorithms processing data by Spark or Storm in memory. It is critical to be able to detect anomalies and predict performance problems and their root causes in order to proactively and dynamically make necessary changes to continuously meet SLGs for each workload. Alexander Lavin and Subutai Ahmad ("Evaluating Real-time Anomaly Detection Algorithms – the Numenta Anomaly Benchmark", 2015) propose a benchmark anomaly detection methodology called Numenta Algorithm Benchmark (NAB). The technique applies a variety of anomaly detection and prediction algorithms and selects the combination it deems has the highest score. The score is determined by rewarding early detection while penalizing false results.

When deciding how to approach the anomaly detection and performance problems prediction, a real time streaming approach such has NAB was considered. However, because Workloads' variables are hourly aggregations the approximate real time and simpler techniques may be applied to achieve the same effect. Since models can be re-fit in the intervening hour between data point arrivals, past extreme positive anomaly detection and future problem prediction can be treated separately.

The anomaly detection approach followed in this paper focuses on detecting both global as well as local anomalies through applying the Generalized Extreme Studentized Deviate test. This technique can be used to identify one or more outliers in a data set. It holds the advantage that the number of outliers does not need be preset, but instead is determined within the algorithm.

In addition to NAB, a real time streaming model that combines Markov models and Bayesian classification methods can be used to predict anomalies. Experiments show that this approach efficiently predicts and diagnoses extreme positive anomalies with high accuracy (X. Gu and H. Wang, 2016). However, these models are highly complex, and due to the slightly less than real time nature of our data, such techniques need not be applied. This paper focuses on applying traditional classification models such as logistic and extremely randomized trees to the identified anomalies while using lagged variable data as the predictors.

For Big Data Applications, extreme positive anomalies on system status variables represent when resources are being stretched to capacity. The goal for this stage is to design a framework that enables automated prediction of these anomalies within Workload KPI variables and thus provide a warning to Big Data Application administrators of when changes to the configuration may be necessary.

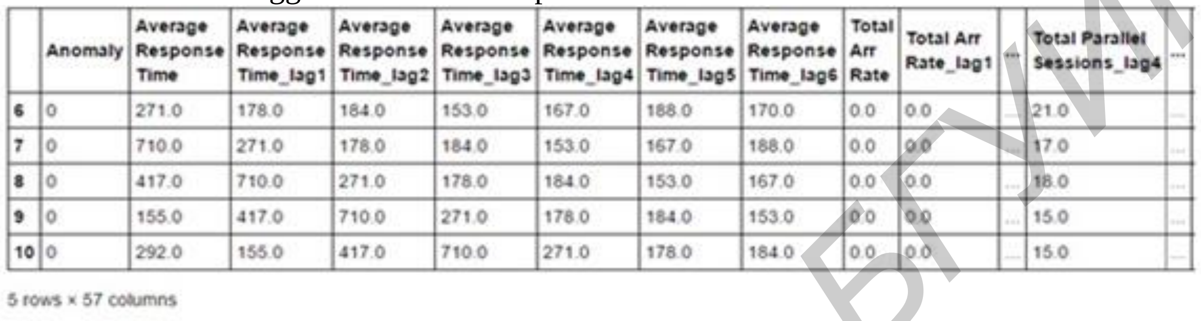
Anomaly Detection. To train a model to predict anomalies, historical anomalies must first be identified. Two techniques from previous methodology sections are applied to generate a clean and continuous data set for anomaly detection. The imputation and best segment methodologies provide the data set for each Workload's anomaly detection. However, where the time series analysis used the most recent segment above a certain size, it was deemed more important for the anomaly detection to find the largest segment available to provide more anomalies against which to train.

This Generalized ESD (gESD) algorithm is then applied to the data set generated for each Workload and KPI variable. The gESD works best on data that approximates normal. The code package used automatically normalizes the Workload segment fed to it to meet this requirement. The generalized ESD algorithm assumes there can be up to n anomalies. The algorithm iterates by removing the point with highest G value (the point farthest away from the mean of the sample) calculated from the similarly iterated sample mean and standard deviation. The critical value changes with the number of points that are removed from the sample. The number of anomalies is the condition with the most outliers with a G above the critical value.

*Anomaly Prediction.*Now that the anomalies of each Key Performance Indicator are identified, the next step is to design an anomaly prediction model that will alert a company in time to change its system configurations. To do this, we run four probabilistic machine learning models for each workload. The dependent variables are a time series of binaries for each workload KPI variable: 1 if anomaly detected in that hour, 0 if not. The independent variables are time lagged values of all 15 variables. It is this lagging that necessitated the anomaly detection be done on a continuous set of data.

A lagged data frame is created for each Workload KPI variable (6 hours of lags for the DBC Workload, as the example on Table 2 outlines). This effectively increases the number of independent variables from 15 to 90. It also removes the data from its time series format and directly pairs each set of dependent and independent variables.

Table 3. Workload lagged data frame sample



	Anomaly	Average Response Time	Average Response Time_lag1	Average Response Time_lag2	Average Response Time_lag3	Average Response Time_lag4	Average Response Time_lag5	Average Response Time_lag6	Total Arr Rate	Total Arr Rate_lag1	...	Total Parallel Sessions_lag4	...
6	0	271.0	178.0	184.0	153.0	167.0	188.0	170.0	0.0	0.0	...	21.0	...
7	0	710.0	271.0	178.0	184.0	153.0	167.0	188.0	0.0	0.0	...	17.0	...
8	0	417.0	710.0	271.0	178.0	184.0	153.0	167.0	0.0	0.0	...	18.0	...
9	0	155.0	417.0	710.0	271.0	178.0	184.0	153.0	0.0	0.0	...	15.0	...
10	0	292.0	155.0	417.0	710.0	271.0	178.0	184.0	0.0	0.0	...	15.0	...

5 rows × 57 columns

A logistic regression model using all fields is implemented. Then, recursive feature elimination is conducted to trim the variables to only the most important. A new model is trained on the selected variables. It is tested on a 70/30 train/test data split.

The same sequence is conducted with an Extremely Randomized Trees classification model. The ERT model was selected over the standard random forest due to the faster training time and better classification accuracy (Geurts et all, 2006). The ERT is run using all 90 lag variables. Recursive feature selection is conducted on the results. Then the ERT is refit with just the selected variables and tested on a 70/30 train test split.

The logistic regression model does not always handle the skewed nature, few anomalies spread over many data points, of the datasets correctly. This results in high occurrences of false-negatives. The extremely randomized trees model handles the skewed dataset slightly better, reducing false-positives and false-negatives in many cases.

All the models are run for each Workload and the error rates are collected. The model with the lowest error rate is selected as best.

Anomaly Detection and Prediction Example. The plot below demonstrates an example of the gESD algorithm used with the DBC Workload data. The red points represent the anomalies that are detected. The anomalies are displayed against the undifferenced data demonstrating that we selected the largest values as anomalies. This is good as it is the largest values that affect the performance of a Big Data system.

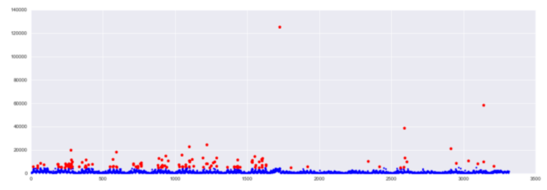


Fig. 14. DBC Workload – Average Response Time

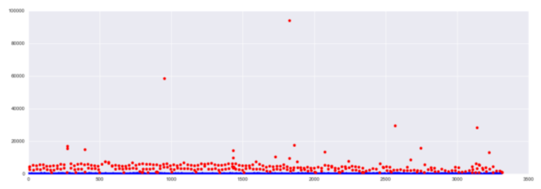


Fig. 15. DBC Workload –Total CPU Time

Confusion matrices from the both the naïve (non-feature selected) and feature selected logistic regressions run against DBC Workload data are shown in Figure 28. Generally running the feature selection and removing some of the lagged predictor variables improved the quality of the logistic models. Figure 29 shows which 10 independent lag-variables were the most frequently selected across all Workloads and KPI response variables by the recursively feature selected logistic models. It is fluctuations in those variables that are most closely tied to whether the logistic regressions predict an anomaly or not. These 10 lag-variables would be a good place to begin a root cause analysis of the source of an extreme positive anomaly.

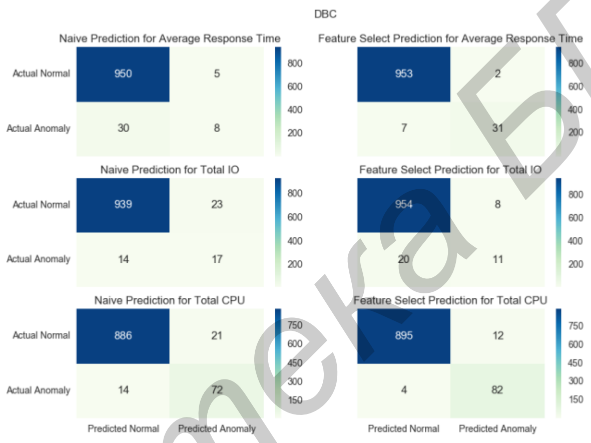


Fig. 16. Naïve and Feature Selected Logistic Regression Confusion Matrix

Top Features selected in Logistic Regression include Total Time Delay, Total Parallel Sessions, Average Response Time,

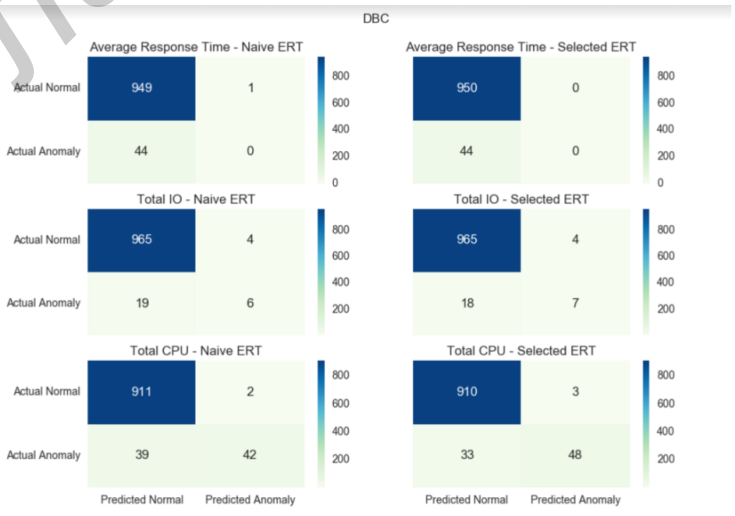


Fig. 17. Naïve and Feature Selected Extremely Randomized Trees Confusion Matrices

Top Features Selected in Extremely Randomized Trees include Total I/O, Total Priority Index, Total Intercon Messages

Forecast Accuracy:

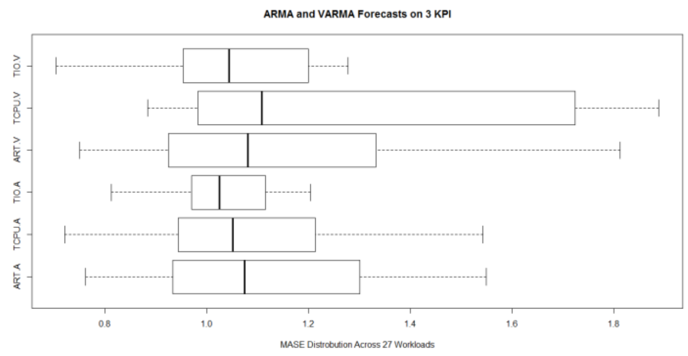


Fig. 18. Overall Forecasting MASE Values

The accuracy results were mixed with a roughly 60/40 split between models that performed worse than a naïve forecasting and models that performed better. The ARMA models tended to outperform the VARMA with 67% of the ARMA models scoring a lower MASE value than the VARMA models.

*Overall Accuracy for each Key Performance Indicator:*The Extremely Randomized Trees generally do better than the logistic regression at predicting the anomalies though it is harder to tell what the relevant variables are. The accuracy rate tables demonstrate that the Naïve (non-feature selected) ERT model generated the most accurate results at the highest rate.

Conclusion. Performance Assurance algorithms reduce uncertainty and risk of performance surprises during design, implementation and performance management of Big Data applications. Algorithm of the short term prediction based on time series is applied to identification of current anomalies and future performance problems. Diagnostic and root cause analysis reduces the scope of analysis and enables organization of the proactive performance management. Determination of the seasonality of performance characteristics improves accuracy of recommendations. Data collection, workload aggregation is performed every hour. Models are retrained every day and short term prediction algorithms are applied every hour to predict anomalies and develop proactive recommendations.

In general, the forecasting models were unable to outperform the naïve forecast. The VARMA models struggled with low forecasting accuracy and long training times. Anomalies created extreme outliers in many Workloads which were nearly impossible to forecast and caused the VARMA models to generate inaccurate outlier forecasts. Some Workloads seemed to contain additive outliers that impacted the results.

Cointegration and outliers are present in some of the Workloads which negatively impact the VARMA models. Switching to a Vector Error Correction Model might be an appropriate way to address cointegration. The anomaly detection methodology outlined in this paper or the approach described by Chen & Liu (1993) for identifying additive outliers in time series data could be used to detect significant outliers. Smoothing, or removing, the outliers may yield before training may also improve the VARMA model forecasting.

The multiple models trained for anomaly prediction provide fairly accurate results. The main issue is that there are too many false negatives, instances where an anomaly occurs and it was not predicted. The number of false negatives can be addressed by lowering the probability threshold for an anomaly but would come with a corresponding increase in false positives. The most commonly selected lagged predictor variables would be a good place to begin an analysis into what causes an extreme positive anomaly.

Two additional time series techniques may also improve the validity of the short-term predictions. First, there are methods to deal with the additive outliers that may be affecting the ARMA and VARMA fits and forecasts. Second, the cointegration results indicate that a Vector Error Correction Model may be more appropriate than a VARMA for some of the Workloads and KPIs.

Future Work. This paper focuses on implementation of the time series algorithm for short term prediction, diagnostic and root cause analysis. We are planning to apply similar approach to Performance Engineering focusing on new Big Data applications and development a Prescriptor generating recommendations related to selection of the ML algorithms and ML libraries and dynamic performance management of Big Data environment.

First, the use of a smoothing algorithm may eliminate the extreme value anomalies. The smoothing can be applied to the predictor variable time series before it is used in the VARMA model. This would prevent sudden spikes in the predictor time series from passing into the response variable prediction.

Additionally, it would be interesting to see how well a neural network could learn and predict the Workload performance. The month, day and hour previously encompassed by the time series nature of the data could be transformed into dummy variables. Pairing the dummies with the values of the Workload interaction variables would create a dataset which could be fed to a neural network. The neural network would be able to handle the high dimensionality of the data given a sufficiently large sample size.

Limitation of the common time series algorithm is a requirement not to have gaps, and Have equally spaced data. Nonlinear Time series and other algorithms providing high accurate results and do not having strict requirements to data can be evaluated. BUT from the practical point of view common time series algorithms provide acceptable accuracy and relatively fast solutions . If necessary training can be done more frequently

Finally, both the short-term forecasts and anomaly predictions can be included into a framework for monitoring and maintaining Big Data systems. Using the outputs from these models it is possible to proactively adjust resource allocation and system settings to prevent downtime or significant drops in performance.

References

[1]. Aboagye-Sarfo, P., Mai, Q., Sanflippo, Frank M., Preen, David B., Stewart, Louise M., Fatovich, Daniel M. (2015). A comparison of multivariate and univariate time series approaches to modelling and forecasting emergency department demand in Western Australia. Retrieved on October 16, 2016 from https://www.deepdyve.com/lp/elsevier/a-comparison-of-multivariate-and-univariate-time-series-approaches-to-bwna7dWZny?#

[2]. Fernández-Ros, M., Parra, J. A., Salvador, R. M., & Castellano, N. N. (2016). Optimization of the periodogram average for the estimation of the power spectral density (PSD) of weak signals in the ELF band. Measurement, 78, 207-218. doi:10.1016/j.measurement.2015.10.006

[3]. Gałecka-Burdziak, E. (2016). Aggregate matching in Spain. Time series analysis using cointegration techniques. CONTEMPORARY ECONOMICS, 10(1), 5-12. doi:10.5709/ce.1897-9254.194

[4]. Gu, X., & Wang, H. (2009). Online Anomaly Prediction for Robust Cluster Systems. IEEE International Conference on Data Engineering. Retrieved October 15, 2016, from http://dance.csc.ncsu.edu/papers/icde09-xhhx.pdf

[5]. Honaker, J., & King, G. (2010). What to Do about Missing Values in Time-Series Cross-Section Data. American Journal of Political Science, 54(2), 561-581. doi:10.1111/j.1540-5907.2010.00447.x

[6]. Lavin, A., & Ahmad, S. (2015). Evaluating Real-time Anomaly Detection Algorithms – the Numenta Anomaly Benchmark, in 14th International Conference on Machine Learning and Applications (IEEE ICMLA '15). Retrieved October 13, 2016, from https://arxiv.org/ftp/arxiv/papers/1510/1510.03336.pdf

[7]. Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., & Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. Journal of Big Data, 2(1). doi:10.1186/s40537-014-0007-7

[8]. Villalpando, L. E., April, A., & Abran, A. (2014). Performance analysis model for big data applications in cloud computing. Journal of Cloud Computing J Cloud Comp, 3(1). doi:10.1186/s13677-014-0019-z

[9]. Zibitsker B, Lupersolsky A, CMG 2016, "Performance Engineering for New Big Data Applications

[10]. Zibitsker B, CMG 2016, "Performance Assurance for Big Data Applications"