Classification in Sparse, High Dimensional Environments Applied to Distributed Systems Failure Prediction

José M. Navarro , Hugo A. Parada G., and Juan C. Dueñas

Abstract. Network failures are still one of the main causes of distributed systems' lack of reliability. To overcome this problem we present an improvement over a failure prediction system, based on Elastic Net Logistic Regression and the application of rare events prediction techniques, able to work with sparse, high dimensional datasets. Specifically, we prove its stability, fine tune its hyperparameter and improve its industrial utility by showing that, with a slight change in dataset creation, it can also predict the location of a failure, a key asset when trying to take a proactive approach to failure management.

Keywords: Online failure prediction · Machine learning · System management · Automatic feature selection · Logistic regression · Multivariate prediction

1 Introduction

Big Data has revolutionized the way industry works. From data-driven decisions to breakthrough discoveries through the analysis of massive data, exploiting all the available data we generate is improving and broadening our capabilities in ways we hadn't thought possible. This discipline relies heavily on distributed systems, as the power of big data platforms, such as Hadoop, resides on the fact that they scale horizontally, working over a grid of commodity hardware-based nodes that are coordinated to work together. In terms of failures, a distributed system has an inherent complexity due to the sheer amount of devices it is composed of, including all the network layer to connect the different hosts. In fact, the network layer is one of the key factors in ensuring fault transparency. In its book, Tanenbaum [21] remarks how "reliable networks simply do not exist" and that is the exact point we tackle in this paper. To do so, we employ a discipline called Proactive Fault Management. And, more specifically, one part

of it called Online Failure Prediction, which deals with identifying situations that will possibly cause a failure. We improve a method we already proposed in [10], validating and expanding it, to achieve a reliable failure prediction model for a large infrastructure IT network which could be integrated in a Reliability Solution for companies. Our key contribution is showing how this model can predict not only the possible occurrence of a failure, which has already been extensively treated [4][7][8], but where it will happen. This feature is crucial in ensuring fault transparency in large scale systems, as knowing that an event will happen will often not be enough to pinpoint it and correct it before it affects service quality. At the point of writing this paper, the authors are only aware of few efforts in this direction. What sets apart our paper from them is the dataset we use and its associated method. Our data present two problems: first, data sparsity, which happens when (assuming binary data) most of the dataset matrix is composed of zeroes (event absences), instead of ones (event presences). This situation is a tricky one for most usual machine learning classification methods, such as logistic regression, where they tend to overfit or not estimate correctly the target variable [15]. The second problem in our environment is the high complexity of our data: we start with more than 1300 input variables. This situation can also affect the performance of classification methods so we are forced to select which variables are actually affecting each event in each node. Genetic Algorithms (a technique used by other works in this area) suffer an exponential increase when the number of elements to mutate is large, so we consider our approach is more valid for this kind of environment.

Apart from the main contribution we exposed before, in this paper we show several additional contributions related to the problem at hand: we test the randomness introduced by our method's preprocessing phase, fine-tune the method's hyperparameters and analyze the effect of adding resource consumption information. In the following sections we analyze the current works in Online Failure Prediction, briefly comment the methodology we used, expose the scenario we worked on, detail the experiments we carried on and the results they yielded, draw some conclusions and expose future lines of work that could span from this research.

2 Related Work

In terms of the industrial problem we are solving, the current state of the art [22] shows that network hardware failures are still a key aspect of distributed systems reliability, which justifies the practical aspects of our contribution. In fact, [23][24][25] independently perform an analysis of failure logs from distributed systems and conclude that network errors are often present on them, along with closely correlated failures. So we now turn to which techniques have been applied to solve similar problems as the one we deal with. Dealing with critical systems, whose correct performance must be ensured, we can not rely on reactive approaches to maintain their reliability. We, thus, turn to Online Failure Prediction, which takes a proactive approach to system reliability [1]. This discipline covers from data cleaning and preprocessing to the actual creation of

the prediction model. Though most of the work has been carried out in the creation phase, there have also been efforts in the preprocessing phase, such as [16][17]. They filter and clean data to improve a prediction model's performance. There has also been a whole array of techniques applied to predicting failures in distributed system and computing clusters, where the works of Watanabe et al. [2][3], who show a method of pattern learning for the prediction of failures, Salfner et al. [6][11], who model a system using Semihidden Markov Models and add fuzzy logic to the OFP scenario, and, specially, Zheng et al. [5][7] are the main ones. The latter authors have worked for several years with the IBM Blue Gene supercomputer and have a long streak of papers related to the issue at hand. Apart from these main works, there have also been separate relevant works in the area, like the combination of time series analysis and fault trees [4], an anomaly detection approach to OFP [9], the creation of failure clusters measured by their correlation [8] and the research presented in [13] by Pitakrat et al., that proposes a full framework for OFP and tests several Machine Learning methods such as Naive Bayes or Support Vector Machines to test their performance. Compared to all these previous works, our proposal has a key feature that separates it from them and a minor one: the major one is that we include location awareness in our prediction, this is, our method not only predicts which event will happen but also indicates its node. As we stated in the introduction, the only work that also addresses this point is the one present in [12] by Zheng et al. The main advantage of our system over Zheng's work is its ability to work in a difficult environment (a high dimensional sparse one) and produce valid, sparse outputs. Additionally, the final model we train in this paper has two different data input sources: system events and system resource consumption. Most previous works only have a single kind of data source. As a side note, even though some works we have discussed do not model distributed systems' networks they are still relevant to our research, taken that, from an OFP point of view, they can be modelled using similar techniques.

The model creation algorithm we use, which we presented in [10], uses several approaches found on literature to work in the complex environment we defined before. Regarding the high dimensionality of the problem, we use a technique called Elastic Net, proposed by Zou et al. in [14], which allows the user to select the amount of two regularization types he prefers. This feature expands the capabilities of our model to use the optimal regularization amount for each event by optimizing it through a grid search. To work with sparse data we follow the advices given in [15], which suggests to trim the amount of zero (absences of the target event) instances included in the training dataset to optimize logistic regression performance. Summed up, the algorithm we use follows these steps:

1. Dataset randomized separation in training, validation and test datasets with zero-trimming to fulfill a user defined zero-to-one proportion.
2. For each event to model, train eight different logistic regression models with different regularization options (L1 and L2 proportions). Each of them is also internally cross validated to optimize model complexity and perform feature selection.

3. Test each model against the validation dataset to select the best performing one.
4. Test the best model against the test dataset to obtain its score.

Using this model we showed in [10] how we were able to successfully predict system failures in a distributed system. In this work we validate and expand it to improve its usefulness for real Big Data environments. This method also fulfills the requirements given in [20] by Trendafilov et al., as it produces a robust, sparse solution and allows the method to work over large datasets with a minimal amount of valuable data without producing a linear combination of variables as a results of the anaylisis, which allows for an easy interpretation of the output. Other interesting approaches to sparse data found in literature are the method proposed by Chickering and Heckerman in [18], which adapts standard machine learning methods to work with dense matrices instead of sparse ones and the work of Li et al. in [19], which is centered around calculating data distances from a conditional sample of the dataset. They are not directly appliable to our method, though, as we do not use distances and to modify the elastic net to work with dense matrices is out of our research scope.

3 Experiments and Results

3.1 Scenario

As in [10], the dataset we worked with was obtained from a big Spanish bank's IT network infrastructure. It was composed of two different structures, an intranet and an internet-connected section. They were a total of 36 devices, whose structure was divided in:

- Eighteen switches.
- Two DNS.
- Four routers.
- Six firewalls.
- Six load balancers.

We had system event logs from every device listed before, comprising a total of 22823 training instances. These events were categorized by their severity, the node they happened in, their timestamp and the event ID. Our key objective is to be able to forecast as many events' occurrence as possible in the distributed systems environment with the highest attainable detail. Some examples of these events were the three critical events present on our dataset: "'99% CPU usage threshold has been surpassed"', "'A single device is down or is not reachable by SNMP messages"' and "'A chassis is down or is not reachable by SNMP messages."'.

We will now describe our main contributions divided in the different experiments we carried out and their results. The first two ones are related to validating and fine tuning the model we use, and the last two experiments expand and improve the forecasting capabilities of our method, adding a new prediction dimension and testing new information sources.

3.2 Preprocessing Method Stability

We started by testing the preprocessing method of our algorithm in terms of its stability. As it randomizes the dataset before splitting it, it may introduce random noise in the output models' performance. To check the validity of this assertion, we ran the algorithm for every model twenty times, as we considered this number of iterations large enough to show any random behaviour that could harm the models' performance, and analyzed each created model.

The first metric we extracted was the amount of created models for each iteration. The minimum amount was 53 and the maximum one was 56, with a mean $\pm$ standard deviation of 54.15 ± 0.9333 models. So, in terms of amount of created models, the algorithm is stable. This, indeed, does not seem like a large rate, unless it affects some critical events, which could lead to not modelling a key objective. This is not the case, though, as the only one that has a slightly high standard deviation is associated with a manual change in the network, as experts confirmed us. Lastly, we checked each model's performance in terms of average F-score and its standard deviation. Fig. 1 shows the obtained results. Only six events suffer from a deviation of more than 0.1 and none of them is a critical one. On the other hand, most models have a very narrow deviation bar and keep a stable score over the whole process. We, thus, consider the stability of the preprocessing phase sufficiently justified for the environment we are working in. We strongly emphasize the fact that this approach has only been proven to be suitable for this specific dataset. The large deviation found in the six anomalous events would lead us to deduce there are separate information clusters in those data, where modelling one cluster is not enough to forecast other ones. We would have to test this assertion to confirm it. Had more models presented this behaviour, we would have had to change our preprocessing phase to a more complex one, such as k-fold cross validation. Taking into account the number of models to be created, short computation time for each model's creation is a requisite; so we prefer simpler methods whenever they perform good enough. We consider, thus, that our preprocessing phase stability has been proven.

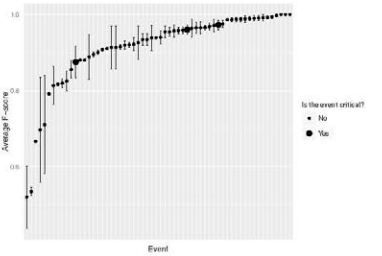

Fig. 1. Average F-score for 20 iterations of the algorithm for each event. Critical events and most non-critical ones have a stable performance.

3.3 Zero to One Proportion Study

Another step in the preprocessing phase is the pruning of excessive zeroes in the dataset, according to the advice given by King et al. in [15], where they suggest to start from a 1 to 1 proportion and start adding zeroes from there. In order to find the optimal amount of zeroes to add to our datasets, we performed a grid search over the zero proportion value, running our elastic net model for values ranging from 1 to 15. To compare its performance with a standard elastic net logistic regression approach, we also ran the algorithm without varying the zero to one proportion in the dataset. Considering that the stability of the preprocessing method was proven in the previous section, we only ran each model once. We will now compare each result in three dimensions: number of created models, computation time and average peformance for every event. We set as the main criteria for proportion selection the highest possible performance, unless the disparity in one of the other two metrics were unreasonably high.

For the first two metrics, created models with an f-score higher than zero and computation time, the distribution can be found on Table 1. In terms of created models, apart from an outlier at 1 to 1 proportion, a growing tendency is clear in the data. Assuming the second value is spurious, the optimal zero to one value would be 13, though the other ones are close to it. Regarding computation time, the total execution time we can draw two main points: the first one is that pruning zeroes has a drastic effect on computation time, reducing it by a factor of, at least, 22. The second conclusion is that, when pruning zeroes, computation time is linearly related with the zero to one proportion. As every value is, at least, 22 times less than the standard total execution time, we conclude that every zero to one proportion is equally valid for our experiment and, at the same time, justify the use of zero pruning as a time saving tool.

Table 1. Secondary metrics for each zero to one proportion

Proportion amount	**No**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Created models	36	44	37	37	37	38	40	41	40	42	42	41	42	43	41	42
Computation time (s)	8275	57	97	113	132	153	176	194	214	230	255	275	298	324	340	362

But the time saving that we have just studied would be completely irrelevant if performance decreased when zeroes are pruned. To study it, we plot in Fig. 2 the average F-score of models created with each proportion value. In this figure the model with a proportion of 3 to 1 of zeroes to ones is the one with best overall average performance, combined with a really low standard deviation. This settles the discussion of which proportion to use. As the amount of created models is not too low and the performance is the best one, 3 is the appropriate proportion of zeroes for our dataset.

We want to comment on the possibility of adding the optimization of this hyperparameter to the actual model training. We declined this option for two main reasons: computation time and dataset limitations. Adding another grid search over the already convoluted process of multiple cross validations would have imposed a burden on training time and would have forced us to split our

dataset into even more groups. As it is not a specially big one, we preferred to obtain a suboptimal, though usually good value for the zero proportion and take that as a fixed value for every experiment.

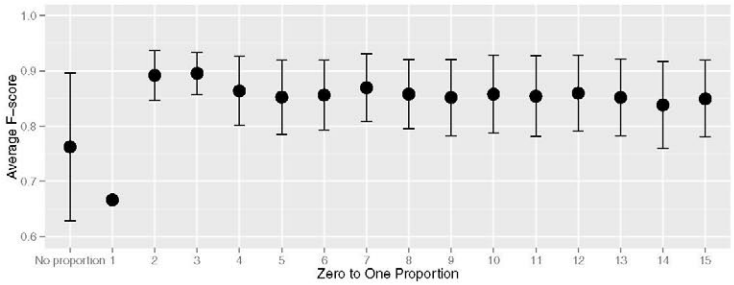

Fig. 2. Average F-score for each zero to one proportion

3.4 Locality Awareness

Now that we know that our algorithm is fine tuned and validated, we set on to improve its industrial interest by adding another prediction dimension to it. This section is divided in three different experiments, all of them related to locality and being able to predict where a failure will happen. We will compare them through the number of correctly created models, the distribution of the obtained performances and, numerically, by the average F-score and its standard deviation. We will first state the purpose of each experiment and then compare them all together. The first experiment is the most simple one and starts from the hypothesis that the occurrence of an event in a specific node is only affected by previous events on that node. To test this hypothesis, we divided our dataset in as many different datasets as nodes were in the system, and trained the elastic net model for each event on each node separately under the assumption previously stated.

The second and third experiments are, in essence, just a subtle change in the dataset preparation, but they completely alter the working environment and can enhance the practical utility of the models. This change we speak of deals with changing the input and output variables. Previously, our input features where "Event X has happened in the system" and the variable to predict was "Event Y will happen in the system". Now, when preprocessing our data, we deal with new input variables: we divide the n previous features in $\beta \leq \alpha n$ variables, where α indicates the number of nodes in the system and W the chosen observation window, where each feature now indicates "Event A has happened in Node ψ in the last W minutes". Inversely, our output now changes to "Event B will happen in node ψ' in the next W' minutes". With this subtle change, without altering the actual algorithm in any way, our problem changes (the number of variables is greatly increased), forcing our model to work in a much harder environment, but we also increase its industrial utility. The difference between these two experiments is that in the second one, both the input and the output

are divided by locality, whereas in the third one, only the output is divided by node, this is, it takes as a input just what has happened in the system and tries to predict where and which event will happen.

We can see in Table 2 the amount of models that yielded a higher than zero performance (measured in F-score), their average F-score and their standard deviation for the three experiments. The first observation we can draw from the first table is the drastic decrease in the percentage of created models over the total possible amount. We see two possible reasons for this behaviour, comparing it with the high amount percentage of created models obtained in previous experiments: the first one would be to assume that more data is needed as the number of features is increased. But we must also take into account the conditions of previous and current experiments: modelling events on a system, a steady pattern in a node can make an event predictable, even if its appearances are random in other nodes. Predicting failures in a specific node only yields models for nodes that actually exhibit a certain pattern. Thus, this severe decrease in created model percentage is not, necessarily, an ominous sign, as it may just indicate that not every event in every node exhibits any kind of pattern. Apart from this remark, we can also state that events in a node are influenced by events in other nodes. This is, each node is not an isolated system. This supports the correlated error conclusion that is exposed in [24] and [25]. If we had to choose which model better predicts our environment, we would tend to choose the option that yields better models as a whole, which is the second one, the Complex Input Complex Output Node Aware Elastic Net.

Table 2. Performance metrics for each node-awareness experiment

	Single Nodes	**Complex Input Complex Output**	**Simple Input Complex Output**
Models Amount	1140	1336	1336
Correctly Created Models Amount	420	601	564
Correctly Created Models Proportion	36.84%	44.99%	42.21%
Average F-score	0.826	0.931	0.9254
F-score Standard Deviation	0.201	0.102	0.108

To further study how each different option affects the models' performance, in Fig. 3 we show the distribution of the F-scores obtained for each option, which allows us to study in finer detail each option's effect. This figure reinforces the conclusions we previously drew. Events in a certain node are affected by events in different nodes and both experiments of locality are satisfactory, though, at the expense of computation time, performance can be slightly improved (and the amount of information extracted from the model) by using a complex input, this is, using every event in every node as a single input feature.

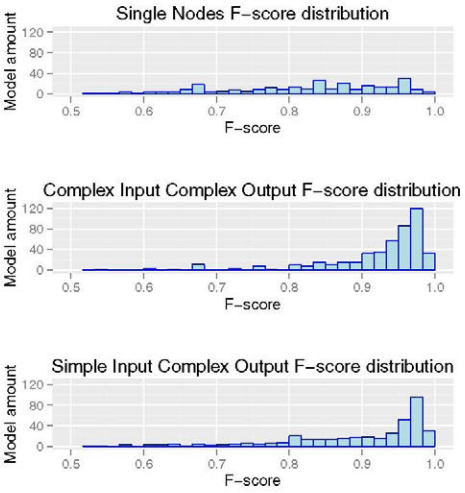

Fig. 3. F-score distribution for each node-awareness experiment.

3.5 Resource Consumption Addition

Once we have proved that the elastic net logistic regression model is able to forecast failures and their location in a much harder environment than previous experiments, we then tried to improve the node-aware model by adding resource consumption information. The information we had available was CPU, memory, hard drive and network interfaces usage, expressed in time series with a five-minute sampling frequency. For every node to forecast failures in, we add its resources consumption as new input features. We ran several experiments with different ways of creating these new variables, taking the Simple Input Complex Output Node-Aware Elastic Net model we trained in the previous section. We used this model to reduce the number of input variables, which would allow new input variables to have a greater influence on the model. The first experiments we ran added one single resource to the model, but it did not affect the model's performance in any remarkable way, so we will not include their detailed results in this work. The two experiments that provided good results were the following ones, each of them centered around different ways to transform a time series to ensure its stationarity: first, using a technique called E-Divisive with Medians proposed by James et al. in [26] to detect changes in the mean of a time series, we divided each resource information in two different time series: one with its mean value for every moment and one with the variation over the mean for each specific moment. The second experiment we ran took a different approach: instead of separating each time series, the input we fed to the algorithm was the difference of the time series, this is, for any specific moment T, the value of the series in T substracted the value of the series in $T-1$. This difference in input variables shows two underlying assumptions about our dataset: in the first experiment we consider that changes in the mean as well as variations over

Table 3. Performance metrics for each resource consumption experiment

	Separated Resource Usage	**Differenced Resource Usage**
Models Amount	1336	1336
Correctly Created Models Amount	554	546
Correctly Created Models Proportion	41.47%	40.87%
Average F-score	0.946	0.929
F-score Standard Deviation	0.089	0.104

the mean are significant to the forecasting of events, whereas in the second one we only consider that the value difference after a certain moment is what holds useful information. We now present the results of these two experiments. Looking at Table 3 allows us to study in detail these models' performance. The main conclusion to draw would be that events in this system do not seem to be affected by resource consumption or that there are no examples of resource-affected events in our dataset. Actually, average F-score for the first experiment is higher than previous ones, but at the expense of less created models. We consider this variation part of the normal randomness of the model, though.

4 Conclusions

In this research paper we started from the basis that network errors are still an important problem in assuring distributed systems reliability. To tackle this problem we presented a series of experiments over a machine learning model that stands as a suitable algorithm for performing Online Failure Prediction in a distributed system, in order to take a proactive approach to system failures. In the first two experiment sections we validate the algorithm's preprocessing method and find, using a grid search, the optimal value of the only fixed parameter it was using. Then we present how our model can predict, not only what event will happen, but where it will do so, greatly improving the industrial utility of this model in large-scale distributed systems. We also show how just knowing which events occurred in the system is enough to predict their location with almost the same performance as the more computing-intensive option. Lastly, we analyzed whether adding resource consumption information to the model improved its performance in any significant way. Results showed that it did not and, furthermore, taking into account the storage and computational costs of using resource information, we would discourage their usage for this environment. Again, we must state that this analysis and advices are completely dataset-dependent, e.g. if most events were caused by resource consumption sudden peaks, our advice would be the complete opposite.

Summed up, we have presented a viable model for distributed systems failure prediction that could be incorporated in an Online Failure Prediction system. Additionally, we have also shown that our model is able to work in situations

that are hard for standard algorithms but usual for network environments: a large number of devices and very infrequent failures.

5 Future Work

There are two possible lines of work that span from this research. One would be to, now that it has been tested and improved, compare our model's performance with some more complex state of the art algorithms, like Artificial Neural Networks or Semihidden Markov Models [6]. Such a change would be interesting because of the way certain models behave: modelling a system with a Semihidden Markov model, for example, would allow us to specify with more certainty when an event would happen, but we would also need an external, previous, feature selection method to filter the input. Indeed, that would be the case with most algorithms, unless we were to use one that includes feature selection in the optimization process, we would need to take two steps (feature selection and model creation) to replicate the work of our algorithm, which increases the dimensions to explore and test. The second line of work we could go for takes a more industrial approach to extending this model: after proving that our algorithm correctly models large-scale networks, we would like to apply it to higher levels of infrastructure: servers, virtual machines, application layers... to try and find correlations between errors and provide a complete solution to ensuring a distributed system's reliability.

Acknowledgments. The authors would like to express their gratitude to PRODUBAN who inspired and motivated this challenge as a real business case and provided all necessary assistance to carry out this work.

References

1. Salfner, F., Lenk, M., Malek, M.: A Survey of Online Failure Prediction Methods. ACM Computing Surveys (CSUR) 42(3), 10 (2010)
2. Watanabe, Y., Otsuka, H., Sonoda, M., Kikuchi, S., Matsumoto, Y.: Online Failure Prediction in Cloud Datacenters by Real-Time Message Pattern Learning. In: 2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom), pp. 504–511 (2012)
3. Sonoda, M., Watanabe, Y., Matsumoto, Y.: Prediction of Failure Occurrence Time Based on System Log Message Pattern Learning. In: 2012 IEEE Network Operations and Management Symposium (NOMS), pp. 578–581 (2012)
4. Chalermarrewong, T., Achalakul, T., See, S.C.W.: Failure Prediction of Data Centers Using Time Series and Fault Tree Analysis. In: 2012 IEEE 18th International Conference on Parallel and Distributed Systems (ICPADS), pp. 794–799 (2012)
5. Guan, Q., Zhang, Z., Fu, S.: A Failure Detection and Prediction Mechanism for Enhancing Dependability of Data Centers. International Journal of Computer Theory and Engineering 4(5) (2012)
6. Salfner, F., Malek, M.: Using Hidden Semi-Markov Models for Effective Online Failure Prediction. In: 26th IEEE International Symposium on Reliable Distributed Systems, SRDS 2007, pp. 161–174. IEEE (2007)

7. Guan, Q., Zhang, Z., Fu, S.: Proactive Failure Management by Integrated Unsupervised and Semi-Supervised Learning for Dependable Cloud Systems. In: 2011 Sixth International Conference on Availability, Reliability and Security (ARES), pp. 83–90 (2011)
8. Fu, S., Xu, C.-Z.: Exploring Event Correlation for Failure Prediction in Coalitions of Clusters. In: Proceedings of the 2007 ACM/IEEE Conference on Supercomputing, SC 2007, pp. 1–12. IEEE (2007)
9. Guan, Q., Fu, S.: Adaptive Anomaly Identification by Exploring Metric Subspace in Cloud Computing Infrastructures. In: 2013 IEEE 32nd International Symposium on Reliable Distributed Systems (SRDS). IEEE (2013)
10. Navarro, J.M., Hugo, A., Parada, G., Dueñas, J.C.: System Failure Prediction through Rare-Events Elastic-Net Logistic Regression. In: 2014 International Conference on Artificial Intelligence, Modelling and Simulation, AIMS (2014)
11. Troger, P., Becker, F., Salfner, F.: FuzzTrees-Failure Analysis with Uncertainties. In: 2013 IEEE 19th Pacific Rim International Symposium on Dependable Computing (PRDC). IEEE (2013)
12. Zheng, Z., Lan, Z., Gupta, R., Coghlan, S., Beckman, P.: A Practical Failure Prediction with Location and Lead Time for Blue Gene/p. In: 2010 International Conference on Dependable Systems and Networks Workshops (DSN-W), pp. 15–22. IEEE (2010)
13. Pitakrat, T., et al.: A Framework for System Event Classification and Prediction by Means of Machine Learning (2014)
14. Zou, H., Hastie, T.: Regularization and Variable Selection via the Elastic Net. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67(2), 301–320 (2005)
15. King, G., Zeng, L.: Logistic Regression in Rare Events Data. Political Analysis 9(2), 137–163 (2001)
16. Yu, L., Zheng, Z., Lan, Z., Jones, T., Brandt, J.M., Gentile, A.C.: Filtering Log Data: Finding the Needles in the Haystack. In: 2012 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 1–12 (2012)
17. Zheng, Z., Lan, Z., Park, B.H., Geist, A.: System Log Pre-Processing to Improve Failure Prediction. In: IEEE/IFIP International Conference on Dependable Systems Networks, DSN 2009, pp. 572–577 (2009)
18. Chickering, D.M., Heckerman, D.: Fast Learning from Sparse Data. In: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, pp. 109–115. Morgan Kaufmann Publishers Inc. (1999)
19. Li, P., Church, K.W., Hastie, T.J.: Conditional Random Sampling: A Sketch-Based Sampling Technique for Sparse Data. In: Advances in Neural Information Processing Systems, pp. 873–880 (2006)
20. Trendafilov, N., Kleinsteuber, M., Zou, H.: Sparse Matrices in Data Analysis. Computational Statistics 29(3–4), 403–405 (2014)
21. Tanenbaum, A.S., van Steen, M.: Distributed Systems: Principles and Paradigms. Pearson Prentice Hall, Upper Saddle River (2007)
22. Ahmed, W., Wu, Y.W.: A Survey on Reliability in Distributed Systems. Journal of Computer and System Sciences 79(8), 1243–1255 (2013)
23. Schroeder, B., Gibson, G.A.: A large-scale study of failures in high-performance computing systems. IEEE Transactions on Dependable and Secure Computing 7(4), 337–350 (2010)

24. Kondo, D., Andrzejak, A., Anderson, D.P.: On correlated availability in internet-distributed systems. In: Proceedings of the 2008 9th IEEE/ACM International Conference on Grid Computing. IEEE Computer Society (2008)
25. Gallet, M., Yigitbasi, N., Javadi, B., Kondo, D., Iosup, A., Epema, D.: A model for space-correlated failures in large-scale distributed systems. In: D'Ambra, P., Guarracino, M., Talia, D. (eds.) Euro-Par 2010, Part I. LNCS, vol. 6271, pp. 88–100. Springer, Heidelberg (2010)
26. James, N.A., Kejariwal, A., Matteson, D.S.: Leveraging Cloud Data to Mitigate User Experience from 'Breaking Bad', November 28 (2014), http://arxiv.org/pdf/1411.7955.pdf