Task Failure Prediction in Cloud Data Centers Using Deep Learning

Jiechao Gao, Haoyu Wang and Haiying Shen
Computer Science Department
University of Virginia
Charlottesville, VA, USA
{jg5ycn, hw8c, hs6ms}@virginia.edu

***Abstract*—A large-scale cloud data center needs to provide high service reliability and availability with low failure occurrence probability. However, current large-scale cloud data centers still face high failure rates due to many reasons such as hardware and software failures, which often result in task and job failures. Such failures can severely reduce the reliability of cloud services and also occupy huge amount of resources to recover the service from failures. Therefore, it is important to predict task or job failures before occurrence with high accuracy to avoid unexpected wastage. Many machine learning and deep learning based methods have been proposed for the task or job failure prediction by analyzing past system message logs and identifying the relationship between the data and the failures. In order to further improve the failure prediction accuracy of the previous machine learning and deep learning based methods, in this paper, we propose a failure prediction algorithm based on multi-layer Bidirectional Long Short Term Memory (Bi-LSTM) to identify task and job failures in the cloud. The goal of Bi-LSTM prediction algorithm is to predict whether the tasks and jobs are failed or completed. The trace-driven experiments show that our algorithm outperforms other state-of-art prediction methods with 93% accuracy and 87% for task failure and job failures respectively.**

I. Introduction

Nowadays, cloud computing service has been wildly used because it provides high reliability, resource saving and also on-demand services. The cloud data centers include processors, memory units, disk drives, networking devices, and various types of sensors that support many applications (i.e., jobs) from users. The users can send requests such as store data and run applications to the cloud. Each cloud data center is composed with physical machines (PMs) and each PM can support a set of virtual machines (VMs). The tasks that are sent from users are processed in each VM. Such a large scale cloud data center can host hundreds of thousands of servers which often run tons of applications and receive work requests every second from users all over the world. A cloud data center with such heterogeneity and intensive workloads may sometimes be vulnerable to different types of failures (e.g., hardware, software, disk failures). Take software failures as an example [1], Ya-hoo Inc. and Microsoft's search engine, Bing, crashed for 20 mins in January 2015, which cost about $9000 per minute to reboot the system. Previous research [2]–[6] found that hardware failure, especially disk failure, is a major contributing factor to the outages of cloud services. These many different types of failures will lead to the application running failures. Thus, accurate prediction for the occurrence of application failures beforehand can improve the efficiency of recovering the failure and application running.

A job is comprised of one or more tasks, each of which is accompanied by a set of resource requirements. A job fails when one of its tasks fails. The previous works [3], [7]–[13] use statistical and machine learning approaches such as Hidden Semi-markov Model (HSMM) and Support Vector Machine (SVM) to predict the task and job failures in cloud data centers. They use CPU usage and memory usage, unmapped page cache, mean disk I/O time and disk usage as inputs and the task failure or job failure as the output. However, HSMM and SVM assume that all their inputs are stationary and independent of each other which are not true in the cloud data centers. Thus, they cannot handle the sequence data or high dimensional data, in which data in time points or different features may be dependent to each other. In the cloud data centers, the input features and noisy data are diverse in nature and have dependencies on the past events. Thus HSMM and SVM can't handle the failure prediction in cloud data centers.

Meanwhile, some other researchers applied deep learning approaches such as Recurrent Neural Network (RNN) and LSTM for the failure prediction [14]–[19]. They use CPU usage, memory usage, unmapped page cache, mean disk I/O time and disk usage as input and the task or job failure as output. However, the traditional RNN has a serious drawback for the data that has long-term dependencies [20], which means the past events can influence future events and a temporary memory of events that happened a while ago may be essential for producing a useful output action. Since the error signals are back-propagated through time, they may exhibit decay [21]. To overcome this issue, deep learning approaches such as LSTM can be used to better handle the failure prediction problem which can capture the long term dependencies. Some of the recent research papers [15]–[17] have already shown that LSTM performs better than HSMM, SVM and RNN in terms of accuracy in task or job failure prediction in cloud data centers.

However, the LSTM based prediction methods still have a few shortcomings. First, the methods [15]–[17], [19] only consider CPU usage, memory usage, cache memory usage, mean

978-1-7281-0858-2/19/$31.00 ©2019 IEEE

disk I/O time, and disk usage as input features. More input features may further increase the prediction accuracy. Second, the LSTM based prediction model used single-layer LSTM construction which can not handle multiple input features well compared to multi-layer construction [21]. Third, in the cloud data center, input features like CPU usage and memory usage are highly related over time. For a given time for prediction, the LSTM based prediction model always sets higher weights on the data closer to the time, and lower weights on the data further away from the time, with the assumption that the data further away from the time always has lower impact to the prediction. However, such settings cannot accurately reflect the impact degree as the further data may still have higher impact on the failure (e.g., failures in long term jobs). The performance can be better if the weights of data items are determined based on the real data trace. Thus, a new prediction model is needed to build to implement failure prediction in the cloud data center in order to achieve better accuracy in prediction.

To overcome the shortcomings, in this paper, we build a failure prediction model based on multi-layer Bidirectional LSTM and name it as Bi-LSTM. First, Bi-LSTM has more input features than previous methods, which include task priority, task resubmissions and scheduling delay. Second, Bi-LSTM has multi-layer structure which can better handle multiple input features for higher accuracy. Multi-layer construction can narrow the number of parameters in the calculating functions but still with the same number of neurons which can reduce the calculation time [21]. Third, Bi-LSTM can determine the weights of data items based on their real impact to the failure rather than simply setting higher weights to data item closer to the given time for prediction than the data items further away from the time. We perform the trace-driven failure prediction study by using Google cluster trace and we compare the performance of Bi-LSTM with other state-of-art prediction methods.

Our contributions in this paper are as follows:

(1) We present a deep learning based prediction model named Bi-LSTM for task and job failures prediction. We first input the data into forward state and backward state in order to adjust both the weight of both closer and further input features. We find that the further input features is essential to achieving high prediction accuracy.
(2) We compare Bi-LSTM with other representative models including statistical models, machine learning models and deep learning models in terms of accuracy, F1 score, precision, and recall. The results show that our algorithm detects task failures and job failures with an accuracy of 93% and 87% respectively.

The rest of the paper is organized as follows. Section II presents the related work. Section III presents the trace study. Section IV presents our proposed prediction method and experiment process. Section V presents the performance evaluation of our methods. Section VI concludes the paper with remarks on our future work.

II. Related Work

We classify the previous related work into two parts: failure analysis in cloud data centers and failure prediction methods.

Failure analysis in cloud data centers. Ford *et al*. [22] studied the impact of correlated failures on availability of distributed storage systems for Google clusters. They found that although disk failures can result in permanent data loss, the major reason for most unavailability in the Google cloud storage system is transitory node failures. Birke *et al*. [23] conducted a failure analysis on physical machines (PMs) and virtual machines (VMs) hosted on commercial data centers in IBM, using one-year-long data collected over 10K servers. Their analysis highlights the differences and similarities of PM and VM failure patterns. They found that VMs have lower failure rates than PMs, and show a surprising trend that, in contrast to PMs, increasing the computation intensity by VM unit does not increase failure rate.

Failure prediction methods. We further classify the failure prediction methods to three categories: statistical approaches, machine learning approaches and deep learning approaches. For these approaches, they use input features such as CPU usage, memory usage, mean disk I/O time, and disk usage.

Statistical approaches: Amin *et al*. [24] proposed a forecasting approach based on Autoregressive-Integrated-Moving-Average (ARIMA) and Generalized Autoregressive Conditional Heteroscedastic (GARCH) models to predict response time and time between failures in the web services. Zhao *et al*. [8] used HMM and Hidden Semi-markov Model (HSMM) to predict disk failures in cloud storage systems. They considered various features such as memory usage, disk usage and disk I/O time measured at consecutive time intervals for a disk drive as time series and modeled such time series to classify failed disks and good disks. However, statistical approaches such as HSMM assume that all their inputs are stationary and independent of each other which are not true in the cloud data centers. Thus, they cannot handle the sequence data or high dimensional data, in which data in time points or different features may be dependent to each other.

Machine learning approaches: Guan *et al*. [11] applied supervised learning based on Bayesian classifiers and decision tree classifiers to forecast future system failure occurrences in the cloud. Pitakrat *et al*. [12] proposed a hierarchical online failure prediction approach called Hora. Hora employed a combination of a failure propagation model and software system failure prediction techniques based on Bayesian networks. Zhang *et al*. [13] designed and implemented a new tool based on Random Forest (RF) called PreFix, for accurately predicting whether there will be a switch failure in the near future. However, machine learning approaches such as SVM have the same shortcomings just like statistical approaches, so they cannot handle the sequence data well in cloud data centers.

Deep learning approaches: Chen *et al*. [18] used RNN for predicting failures via various features and performance time series data in the Google cluster traces. Islam *et al*. [19] performed a failure characterization study of the Google

cluster workload trace and presented LSTM to predict the task failures. However, deep learning approaches such as RNN and LSTM also have several shortcomings. For RNN, it cannot handle the data with long-term dependency. For LSTM, it overcomes the drawback in RNN. But LSTM still has drawback that it sets higher weights on the data closer to the time, and lower weights on the data further away from the time, with the assumption that the data further away from the time always has lower impact to the prediction.

To overcome all the above shortcomings, we propose a deep learning based failure prediction model: Bi-LSTM, for task and job failures prediction which can adjust the weights of both closer and further input features to achieve better prediction performance.

III. Trace Analysis

The Google cluster trace [25] starts at 19:00 EDT on Sunday May 1, 2011, and it records the resource utilization of CPU and memory usage of each task on the Google cluster of about 12.5k machines for 29 days. The trace contains 672,075 jobs and more than 48 million tasks in the 29 days. This trace is a randomly-picked 1 second sample of CPU and memory usage from within the associated 5-minute usage-reporting period for each task. Each job is composed of one to tens of thousands of tasks which are shown with information such as CPU usage, memory usage, cache memory usage, mean disk I/O time, disk usage, task priority, the number of task resubmissions, task scheduling delay and etc. The priorities are classified into five categories: lowest, low, middle, high and highest priority by task scheduler and ranging from 0 to 11 which is set by cloud service provider [26].

Termination status means whether certain task is finished or not. Each job and task has several possible termination statuses. These are: (1) evicted, (2) killed, (3) failed (due to an exception or abnormal condition), and (4) finished (successfully terminated). We found that around 42% of the jobs and 40% of the tasks are not finished. In the unfinished jobs and tasks, 41% of the jobs are failed and 39% of the tasks are failed. This is because evicted and killed jobs and tasks are very rare termination status. Therefore, we focus mainly on finished and failed jobs and tasks in our work. Specifically, we consider two kinds of failures: job failure and task failure. The job failure means the job is descheduled due to task failures. Because in the trace, a job consists of at least one task, and each task is constrained by scheduling and resource usage limits. The task failure means the task is descheduled due to a task failure such as software bugs.

1) Data Preprocessing: In order to build a reliable prediction system with high accuracy performance, we should understand the trace data and extract the relevant features. The previous research on Google cluster trace [25], [26] indicate that in addition to the resource usage, the following features are also correlated with job and task failures. Therefore, our prediction model additionally use these features to leverage the prediction performance.

Task priority decides whether a task is scheduled on a machine. The work in [26] indicates that that the highest and the lowest priority jobs experience higher rate of failures than the middle priority batch jobs.

Task resubmissions means that the tasks can be resubmitted multiple times after abnormal terminations during the life cycle of a job. We observe that the number of task resubmissions for the failed jobs is much higher than the task resubmissions for the finished jobs. The ratios of jobs with tasks that execute more than once for failed and finished jobs are 35.8% and 0.9% respectively and about 75% of the jobs have tasks that are resubmitted at most four times which is the maximum resubmission time for certain tasks.

Scheduling delay means the waiting time for certain task. We found that tasks which cannot be finished have a significant long scheduling delay compare to the finished tasks.

2) Input Feature Determination: In the training phase, the previous research [18] built the prediction model in cloud by using five classes of resource usage measures which are CPU usage, memory usage, cache memory usage, mean disk I/O time, and disk usage. In our training phase, we enlarge the set of input dimensions to improve the failure prediction accuracy. So we additionally add the following features to the inputs: task priority, the number of task resubmissions, and scheduling delay. We set all the measures of these features as a *feature vector* so we can put them together at any single time point as an input. For job failure prediction, we consider that a job is predicted to be failed when any of its tasks is predicted to fail.

IV. Main Design

Next, we will introduce how we use Bi-LSTM to predict task and job failures. There are two phases: training and testing. In the training phase, we input the time series data for each task one by one to the model. Each task is labeled by failed or finished. We adjust the parameters of the model according to the comparison between the label and predicted output. In the testing phase, for a given set of time series input data of one task, the model will calculate the failure possibility of one task from the end time of the input data to the task completion time.

A. Model Architecture

The architecture of Bi-LSTM model contains one input layer, two Bi-LSTM layers, one output layer and the Logistic Regression (LR) layer to classify whether the tasks and jobs are failed or finished. As mentioned earlier, our Bi-LSTM based method is more advantageous than the LSTM based prediction methods in the following aspects. First, Bi-LSTM has more input features than previous methods, which include task priority, task resubmissions and scheduling delay. Second, Bi-LSTM has multi-layer structure which can better handle multiple input features for higher accuracy. Third, Bi-LSTM can determine the weights of data items based on their real impact to the failure rather than simply setting higher weights to data item closer to the given time for prediction than the data items further away from the time, which helps achieve

higher prediction accuracy. Below we introduce the input layer and output layer.

1) Input Layer: Given a dataset, we divide it into training set (e.g., 90% tasks) and testing set (e.g., 10% tasks). The training set is used to learn the Bi-LSTM model, and the testing set is used to predict whether a task will be failed or finished in the next time point. Each input is a feature vector at a time point. There are 100 inputs from time point $t-100$ to time point $t-1$. The output is whether the task is failed or finished at time point t. For each task, we got its data for the last 101 time points for training and testing. If the number of time points for a certain task is less than 100, we set the feature vector input as 0.

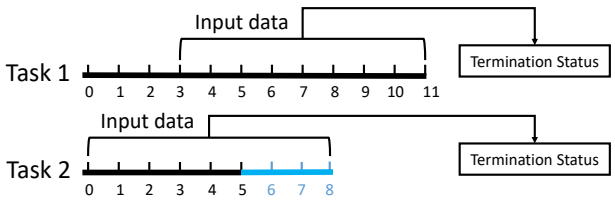


Fig. 1. An example of prediction procedure.

Figure 1 shows a simple example of training and testing procedure. Each axis means the time sequence of one task. Take the testing procedure for instance, the black brackets represent the input data of the testing and the number of the data points in the brackets means the number of input data w. In this example, in task 1, the input data is the value of time = 3 to the value of time = 11, so the total number of input data is w = 9. The black part that is not in the bracket represent the data points that are not used in the testing process. In task 2, the input data is the value of time = 0 to the value of time = 8, so the total number of input data is w = 9 as well. However, task 2 is finished or stopped at time = 5. To ensure the input data is in the same format, we expand the input data in task 2 from time = 5 to time = 8. So the total number of input data can be w = 9 as well, and we set the value of time = 5 to time = 8 as 0. The blue part in the bracket represents that the task has less than 9 time points, so it is expanded by value = 0. Task 1 represents the tasks that have more data points than input data, and task 2 represents the tasks that have less data points than input data. The squares represent the predicted termination status, which represents the task is failed or finished.

2) Bi-LSTM Layer: In the structure of Bi-LSTM layer, the hidden states are divided into two parts: forward state and backward state. The basic idea is to present each sequence of inputs forwards and backwards to two separate hidden states to capture information, respectively. Then the two hidden states are concatenated to form the final outputs to the logistic regression functions. The forward state consists of the memory cells connecting adjacent neurons form cycles that are self-connections of a neuron to itself across time. The input of memory cells includes the data x_i $(i = 1, 2, ...m)$ from the previous layer as well as the data n_{i-1} $(i = 1, 2, ...m)$ from themselves of the previous position. Different from the forward status, in backward state, n_{i-1} $(i = 1, 2, ...m)$ are from the next position. Therefore, the output of the Bi-LSTM layers can adjust the weights of data items from closer data point to the given time for prediction than the data items further away from the time . The structure of memory cell in Bi-LSTM is very suitable for processing data with sequential and time dependencies on multiple scales since the connections pass state information across time steps allowing previously processed data to affect subsequent data.

3) Output Layer: In the output layer, from an input sequence $\{x_1, x_2 \ ... \ x_m\}$, the memory cells in the Bi-LSTM layers will produce a representation sequence $\{h_1, h_2 \ ... \ h_m\}$. We define the mean of these outputs as the mean pooling, which is expressed as follows:

$$\widehat{h} = \frac{1}{m}\sum_{i=1}^{m} h_i \tag{1}$$

where h_i is the i^{th} element in the representation sequence.

4) Logistic Regression Layer: In the logistic regression layer, the result from output layer is fed to the logistic regression functions whose target is the class label (i.e., failed or finished) associated with the input sequence. It means that the termination statuses generated in the Bi-LSTM model are produced based on the input features. We set up a threshold for the probability value of the failure to determine the termination statuses. Based on $\widehat{h}$, the logistic regression functions calculate the probability value of failure. If it is smaller than the threshold, it classifies as the termination statuses into failed, and otherwise, it classifies the termination statuses into finished.

V. Performance Evaluation

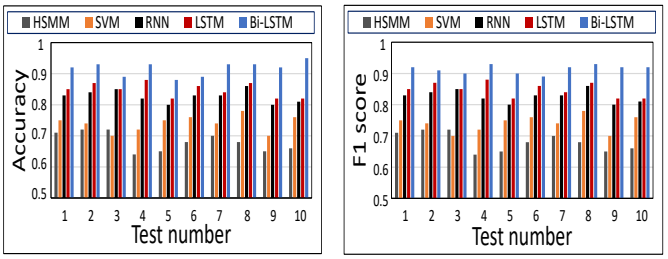
A. Experiment Setting

The experiments are deployed in our local server with a GTX 1080 GPU. The trace is originally stored in separated files of approximate size 200GB, and the data features are represented by key-value pairs. We load these data into a MySQL database for ease of analysis. The prediction method is applied based on Tensorflow in Python. According to our experiment, to achieve better performance, we set 17 memory cells in each layer. We chose 555555 tasks in total in our experiments, in which 50,000 tasks are for training and 55555 tasks are for testing.

The batch size is set as 100 which means we use 100 tasks for one training step. We have 5000 training steps for each epoch, and the number of training epochs was set to 1000, which means we used the 500000-task dataset 1000 times for training. For each training and testing, we used the data of the first 100 time points of a task since many tasks are longer than 100 time points [27], [28]. The threshold we select for Figure 2(a), 2(b), 3(a) and 3(b) is 0.8.

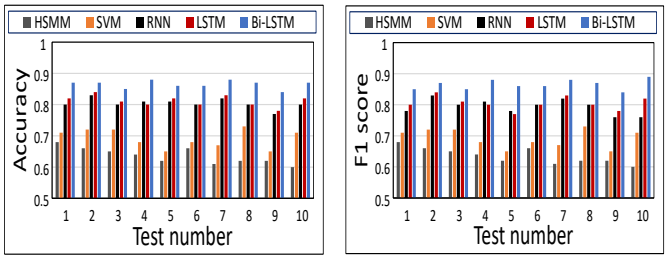
We compared our Bi-LSTM model with the prediction models used in previous research, which are HSMM [8], SVM [10], RNN [18] and LSTM [19].

1) Hidden Semi-Markov Models (HSMM) [8]: HSMM is an extended model of HMM. Different from HMM, HSMM considers the state-resident probability distribution as explicit. It adds time components to the structure of the defined HMM



(a) Accuracy (b) F1 score

Fig. 2. Task failure prediction.



(a) Accuracy (b) F1 score

Fig. 3. Job failure prediction.

and overcomes the limitations of HMM which is the prediction only depends on nearby state.

2) Support Vector Machine (SVM) [10]: SVM is also widely used in such failure prediction according to previous research. It has better performance when dealing with high-dimensional problem [29]. Also, it basically does not involve probability measures and laws of large numbers, so it is different from existing statistical methods. In essence, it avoids the traditional process from induction to deduction.

3) Recurrent Neural Network (RNN) [18]: RNN is recently used in this topic. RNN can not only deal with stationary input and output patterns but also with pattern sequences of arbitrary length [30] which means RNN can achieve better performance when handles the time series data.

4) Long Short Term Memory (LSTM) [19]: LSTM is also recently used in this topic. It is a deep learning model which is developed from Recurrent Neural Network (RNN). Since the error-signals could exhibit exponential decay as they are back-propagated through time, which leads to long-term signal being effectively lost as they are overwhelmed by undecayed short term signals. LSTM can overcome the drawback for data with long-term dependencies also capable of modeling the temporal connections between hidden states.

We keep the same input features as they mentioned which are CPU usage, memory usage, unmapped page cache, mean disk I/O time and disk usage.

B. Metrics

To illustrate the performance of the our method, we use three metrics to determine the better results.

1) Accuracy and F1 score: We determine the performance of different models by using accuracy and F1 score. The prediction accuracy is calculated by:

$$A_n = 1 - \frac{|P_n - R_n|}{R_n} \tag{2}$$

where A_n is the prediction accuracy of n^{th} prediction, P_n is the predicted value of n^{th} prediction and R_n is the real value in n^{th} prediction. The Y axis value is prediction accuracy (A_n) for each method. The F1 score is calculated by:

$$\begin{aligned} PPV &= TP/(TP+FP) \\ TPR &= TP/(TP+FN) \end{aligned} \tag{3}$$

$$\begin{aligned} F1 &= \frac{2 \cdot PPV \cdot TPR}{PPV + TPR} \\ &= \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \end{aligned} \tag{4}$$

where TP is true positive value, FP is false positive value, FN is false negative value. PPV is the positive predictive value, which is also known as $Precision$, TPR is the true positive rate, which is also known as $Recall$. The F1 score takes a balance that both precision and recall rates can reach the highest point at the same time.

C. Experimental Results

Now we evaluate the performance of our method and compare with previously proposed prediction methods [8], [10], [18], [19].

1) Task Level Failure Prediction: We first evaluate the task level failure prediction. At the task level, we classify the termination statuses of task submissions based on the attributes and performance data. In all the target classes, the status finish is considered as one class, and the status failed is considered as other class.

Figure 2(a) shows the accuracy of each prediction method in each test among tasks. The result follows HSMM<SVM<RNN<LSTM<Bi-LSTM. For HSMM, as we discussed before, HSMM only depends on each state and its corresponding observation object. It cannot handle the sequence data in the whole trace or the high dimensional data well. SVM has better accuracy performance than HSMM because SVM can handle the data which has high dimension. However, the performance of SVM is worse than RNN and LSTM. The reason is that in data center, dataset such as Google cluster trace has a huge among of data. SVM only has better performance when the dataset is not so big [31]. For LSTM, it overcomes the drawback from RNN, which is for data with long-term dependencies. However, as we discuss before, LSTM cannot adjust the weight of further data point for a given time in the time series dataset. For Bi-LSTM, it has the forward and backward states which can more accurately determine the weights of data items that are closer and further to the given time in the prediction. We can observe that LSTM can achieve 85% of accuracy and Bi-LSTM can achieve 93% of accuracy which is higher than LSTM.

Figure 2(b) shows the F1 score of each prediction method in each test among tasks. The results follow the same trend and order as in Figure 2(a) due to the same reasons. The result shows that LSTM can achieve 84% of F1 score and Bi-LSTM can achieve 92% of F1 score which is higher than LSTM.

2) Job Level Failure Prediction: At the job level, we classify the termination statuses of task submissions based on

the attributes and performance data. In all the target classes, the status finish is considered as one class, and the status failed is considered as other class.

Figure 3(a) and 3(b) show the accuracy and F1 score of each prediction method in each test among jobs. The results follow the same trend and order as in Figure 2(a) and 2(b) because of the same reasons. We can observe that LSTM can achieve 81% of accuracy and 80% of F1 score. Bi-LSTM can achieve 87% of accuracy and 86% of F1 score which are higher than LSTM.

VI. Conclusion

In cloud data centers, high service reliability and availability are crucial to application QoS. In this paper, we proposed a failure prediction model multi-layer Bidirectional LSTM (called Bi-LSTM). Bi-LSTM can more accurately predict the termination statuses of tasks and jobs using Google cluster trace compared with previous methods. In our method, we first input the data into forward state and backward state in order to adjust the weight of both closer and further input features. We then find that the further input features is essential to achieving high prediction accuracy. Secondly, in the experiments, we compare Bi-LSTM with other comparison methods including statistical, machine learning and deep learning based methods and evaluate the performance with three metrics: accuracy and F1 score, receiver operating characteristic and time cost overhead. The results show that we achieved 93% accuracy in task failure prediction and 87% accuracy in job failure prediction. We also achieved 92% F1 score in task failure prediction and 86% F1 score in job failure prediction. Our prediction method Bi-LSTM also have low FPR which can also indicate the proactive failure management based on prediction results become more effective. We also observe that the time cost overhead for Bi-LSTM is almost the same compared with RNN and LSTM, which means Bi-LSTM can achieve higher prediction performance with no further time cost.

Acknowledgements

This research was supported in part by U.S. NSF grants NSF-1827674, CCF-1822965, OAC-1724845, CNS-1733596, Microsoft Research Faculty Fellowship 8300751, and AWS Machine Learning Research Awards.

References

[1] "https://techcrunch.com/2015/01/02/following-bing-coms-brief-outage-search-yahoo-com-goes-down-too/, [Accessed in APR 2019]."

[2] M. Sedaghat, E. Wadbro, J. Wilkes, S. De Luna, O. Seleznjev, and E. Elmroth, "Diehard: reliable scheduling to survive correlated failures in cloud data centers," in *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2016.

[3] T. Chalermarrewong, T. Achalakul, and S. See, "Failure prediction of data centers using time series and fault tree analysis," in *2012 IEEE 18th International Conference on Parallel and Distributed Systems*, 2012.

[4] S. Mitra, M. Ra, and S. Bagchi, "Partial-parallel-repair (ppr): a distributed technique for repairing erasure coded storage," in *Proceedings of the eleventh European conference on computer systems*, 2016.

[5] H. Wang, H. Shen, and Z. Li, "Approaches for resilience against cascading failures in cloud datacenters," in *Proc. of ICDCS*, 2018.

[6] H. Wang and H. Shen, "Proactive incast congestion control in a datacenter serving web applications," in *Proc. of INFOCOM*, 2018.

[7] R. Baldoni, L. Montanari, and M. Rizzuto, "On-line failure prediction in safety-critical systems," *Future Generation Computer Systems*, 2015.

[8] Y. Zhao, X. Liu, S. Gan, and W. Zheng, "Predicting disk failures with hmm-and hsmm-based approaches," in *Industrial Conference on Data Mining*, 2010.

[9] J. Murray, G. Hughes, and K. Kreutz-Delgado, "Machine learning methods for predicting failures in hard drives: A multiple-instance application," *Journal of Machine Learning Research*, 2005.

[10] I. Fronza, A. Sillitti, G. Succi, M. Terho, and J. Vlasenko, "Failure prediction based on log files using random indexing and support vector machines," *Journal of Systems and Software*, 2013.

[11] Q. Guan, Z. Zhang, and S. Fu, "Ensemble of bayesian predictors and decision trees for proactive failure management in cloud computing systems," *Journal of Communications*, 2012.

[12] T. Pitakrat, D. Okanović, A. van Hoorn, and L. Grunske, "Hora: Architecture-aware online failure prediction," *Journal of Systems and Software*, 2018.

[13] S. Zhang, Y. Liu, Z. Meng, W.and Luo, J. Bu, P. Yang, S.and Liang, D. Pei, J. Xu, and Y. Zhang, "Prefix: Switch failure prediction in datacenter networks," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2018.

[14] C. Xu, G. Wang, X. Liu, D. Guo, and T. Liu, "Health status assessment and failure prediction for hard drives with recurrent neural networks," *IEEE Transactions on Computers*, 2016.

[15] Y. Cheng, H. Zhu, J. Wu, and X. Shao, "Machine health monitoring using adaptive kernel spectral clustering and deep long short-term memory recurrent neural networks," *IEEE Transactions on Industrial Informatics*, 2019.

[16] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[17] A. Das, F. Mueller, C. Siegel, and A. Vishnu, "Desh: Deep learning for system health prediction of lead times to failure in hpc," in *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*, 2018.

[18] X. Chen, C. Lu, and K. Pattabiraman, "Failure prediction of jobs in compute clouds: A google cluster case study," in *2014 IEEE International Symposium on Software Reliability Engineering Workshops*, 2014.

[19] T. Islam and D. Manivannan, "Predicting application failure in cloud: A machine learning approach," in *2017 IEEE International Conference on Cognitive Computing (ICCC)*, 2017.

[20] P. Angeline, G. Saunders, and J. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE transactions on Neural Networks*, 1994.

[21] F. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural Computation*, 1999.

[22] D. Ford, F. Labelle, F. Popovici, M. Stokely, L. Truong, V., C. Grimes, and S. Quinlan, "Availability in globally distributed storage systems," *Procdings of USENIX Symposium on Operating Systems Design and Implementation*, 2010.

[23] R. Birke, I. Giurgiu, L. Chen, D. Wiesmann, and T. Engbersen, "Failure analysis of virtual and physical machines: patterns, causes and characteristics," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2014.

[24] A. Amin, A. Colman, and L. Grunske, "An approach to forecasting qos attributes of web services based on arima and garch models," in *2012 IEEE 19th International Conference on Web Services*, 2012.

[25] C. Reiss, A. Tumanov, G. Ganger, R. Katz, and M. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proceedings of the Third ACM Symposium on Cloud Computing*, 2012.

[26] X. Chen, C. Lu, and K. Pattabiraman, "Failure analysis of jobs in compute clouds: A google cluster case study," in *2014 IEEE 25th International Symposium on Software Reliability Engineering*, 2014.

[27] J. Kumar, R. Goomer, and A. Singh, "Long short term memory recurrent neural network (lstm-rnn) based workload forecasting model for cloud datacenters," *Procedia Computer Science*, 2018.

[28] H. Wang, H. Shen, and G. Liu, "Swarm-based incast congestion control in datacenters serving web applications," in *Proc. of SPAA*, 2017.

[29] T. Joachims, "Making large-scale svm learning practical," Technical report, Tech. Rep., 1998.

[30] Z. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv preprint arXiv:1506.00019*, 2015.

[31] J. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, 1999.