International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-8 Issue-6, August, 2019

Learning Towards Failure Prediction of High Performance Computing Clusters by Employing LSTM

Kamaljit Kaur, Kuljit Kaur

***Abstract*: This Failure prediction of high-performance computing clusters (HPCC) is a crucial issue and a hot problem for many years. Previous works have failed to provide a robust method for real-time failure prediction of HPCC. The available techniques are old, unrealistic and provide low accuracy. This paper presents an efficient technique which provides robust failure prediction with good accuracy and state of the art models. We have employed the concept of long short-term memory (LSTM) with reinforcement learning to correct the prediction accuracy in real-time and provide a solution to the industry with reliable results.**

***Index Terms*: LSTM, Rainbow, HPCCs, failure prediction**

I. INTRODUCTION

With the rapid growth of available hardware resources, large datasets and complicated networks, high-performance computing clusters (HPCC) are increasing exponentially and they are expected to have millions of nodes in the near future [43]. With this scale, complexity, and dependability, researchers are facing key challenges for the proactive avoidance of node failures, timely mitigation and fast recovery with the lowest downtime in HPCC. There are several exemplary failures of HPCC in the previous decades, which triggered a huge loss for the parent organization in terms of time, money and human efforts [2],[4],[7],[8],[9],[10],[19]. Despite the fact that countless techniques are present for designing ~~of~~ extremely reliable components, the increasing cluster size and intricacy has overtaken the further enhancement in the reliability of components. To address the above key challenges, one possible solution is to actively avoid HPCC failures by improving fault tolerance thresholds at the application level. Representative works in this domain include failure aware components accommodation [44], runtime failure support [45], scheduling [22] and check-pointing [46],[47]. However, these fault fault-tolerant techniques are unstable due to a lack of efficient and proactive failure prediction support in HPCC. For example, fault-tolerant techniques with a reactive measure like check-pointing should have a robust prediction of failure to reduce cost by incorporating intelligent checkpoints instead of blindly calling the checkpoint functions periodically [46],[47]. Similarly, for proactive measures it is necessary to have timely alerts about the failure so a mitigation process can be triggered – thus, timely failure prediction has a critical impact on the development of reliable clusters with lowest possible downtimes. Previous literature on failure prediction can be categorized into two main categories; model-based techniques in which a probabilistic model is derived from the detailed analysis of a system [23],[25],[26]. The other technique is data-driven in which historical data is exploited from already occurred events, main triggering factors of failures are considered and other correlated issues are derived, finally, a comprehensive data-driven model is prepared which can predict proactive failures of nodes. Due to the complexity and size of HPCC, the approaches based on probabilistic-model are too complicated and provide too many false alarms, which reduces its efficiency factor, on the other hand, data-driven methods are more reliable in terms of robust failure prediction [30],[32],[34],[39],[42]. To predict the occurrence of a failure, in-depth and comprehensive understanding of correlated factors is required. Past studies on IBM BlueGene/L [23] and LANL [5],[29], exposed significant trends for the root cause of failures in HPCC. There exists strong temporal co-relation (time-of-day and day-of-week patterns) alongside with failure types (spatial information) that may reveal the important properties of such events to predict the trend in HPCC failures. There are few works exploiting temporal and spatial relations of these trends for failure estimation and practical management in real-time [48],[26]. According to our observation, most of the spatiotemporal based failure prediction techniques are severely unrealistic, which employ heuristics to grasp the temporal and spatial information about failure event by using profiling. This technique is failed to provide real-time accuracy because these models lack to measure the spatiotemporal features with respect to different time-scales. An HPCC has a hierarchy of nodes and failures can happen in several places. Therefore, a robust failure prediction model should be able to forecast the component which can be the root cause for failure. In this work, we exploited temporal and spatial information of event data using state of the art technique LSTM. bMoreover, already presented studies focused on static and one-time training of model which means that the model will utilize one-time training data and provide the prediction results based on that training. However, the event data can be old, deterred or have some missing values.

Revised Manuscript Received on October 30, 2019.
* Correspondence Author
Kamaljit Kaur*, Department of Computer Engineering & Technology, Guru Nanak Dev University, Amritsar, Punjab, India.
Kuljit Kaur, Department of Computer Science, Guru Nanak Dev University, Amritsar, Punjab, India.

We have proposed a robust method, which employed an LSTM based model with feedback and periodic training. The newer training will accommodate the latest events data and provide an up-to-date dynamic prediction. To summarize this, we provide the following contributions in the work:

- We have proposed a robust prediction method based on an advanced approach as LSTM [49];
- We have incorporated feedback and fine-tuned the model on the basis of reinforcement learning;
- We incorporated Rainbow [50] based reinforcement learning technique and compared the results with the current state of the art;

The rest of the paper is organized as follows. Section II provides exemplary failures and related works on HPCC failure. The detailed architecture of the proposed methodology is elaborated in Section III. Section IV presents the dataset description, experimental setup, and evaluation in terms of results. Finally, Section V deduces the conclusion with some discussions and future works.

II. EXEMPLARY FAILURES AND RELATED WORK ON HPCC FAILURE

Despite the fact, that the distributed systems are expected to be reliable and invulnerable to failure, engineers have to face un-maskable consequences incurred by the failure of hardware, human error, network, software or undetermined. This section will provide a chronological and comprehensive study of previous works on the topic of cluster failures its prediction and mitigation process. Moreover, exemplary failures that had affected the huge scale of services will be discussed in relation to cluster failures. This study is one of the major factors which leads us to provide a robust failure prediction system using state of the art techniques.

A. Exemplary Failures of HPCCs in recent decades

The famous Open Science cluster Grid3 [1] had to suffer from around 30.0% job failure because of the site issues like HDDs failure [2]. Similarly, a European ICT organization experienced catastrophic failure of Grid'5000 [3], on and off between 2005 to 2007. This organization was providing services to a massive army of scientists and had to reconcile serious penalties due to Grid'5000 failures [4]. The possible causes of failure were estimated to be a software stack's immaturity (software failure) [5] and mishandling of the scheduling system (OS failure) [6].

Overloading (software failure) of high-performance clusters can also lead to disastrous effects, even on peer-to-peer clusters (not to mention the fact they are scalable by design). For instance, the BitTorrent file-distribution had to suffer from the lowest performance in 2009 and 2010 due to memory overloads [7]. A multiplayer game, "World of Warcraft" have to suffer from downtime every week for 4 hours due to memory overloading issues, which are still costing them a huge amount of revenue [8]. Moreover, the IT infrastructure of the USA and Italy struck in 2003 due to a power cut and file system failure [9].

Similarly, in 2004 several tech giants (Google, Yahoo, and eBay) had to undergo downtime due to Akamai [11] failure (CPU snags) as they were counting on the content distribution of Akamai [10]. This costed Akamai, countless clients. On the same point, 20% of PlanetLab [12] resources remained inaccessible to scholars, several times throughout 2004 [13]. Facebook, Instagram, WhatsApp, and Twitter went down frequently in 2008, 2010 and 2019 due to memory failures in clusters [14],[15],[16]. Hotmail server went down for several hours in 2010, Goodreads had to face the same in the middle of 2012 [17]. Software handling, the network configuration of Amazon services led to downtime in April 2011 [19]. Due to software failure and memory overloading, BATS and Nasdaq failed in the middle of the Facebook public offering launch [18].

B. Previous work on Failure prediction of HPCC

Bianca, et al. [20] performed a statistical analysis of Los Alamos National Laboratory (LANL) [21] failures recorded data. On the basis of statistical analysis, they tried to explain the root causes of failures and mean time of failures. Moreover, they analyzed how much it took to repair a failed node. By following Weibull distribution, they stated that there are different failure rates for different systems starting from 20 to 100 in a year. Similarly, they stated the repair time between an hour to a day by exploiting lognormal distribution.

Weiwei Zheng, et al. [22] proposed an efficient approach that predicts the failures of clusters and computing elements. They designed a structure for the groups having a probability of shared risks and described a few correlated properties of those groups. They provided associations of these properties and updates them using rule mining techniques. These associations were discussed and exploited for upcoming failures. They have performed exhaustive testing on the LANL HPC dataset. After experiments and testing, they have claimed 89% Precision, 80% Recall, and 84% F-measure in terms of prediction for a cluster having 16 nodes.

Thomas J. Hacker, et al. [23] developed an approach to understanding the reliability concept of IBM Blue Gene systems [24]. This was four steps comprehensive approach: the first step is comprised of reducing the dependent events from logs, which are less correlated; the second step was to define the statistical analysis and state the methodologies that can predict the mean time to failure (MTTF) efficiently. In the third step, they modeled the spatial and temporal information of log data. Finally, they used a combination of the Markov model, Weibull distribution, exponential distribution, and gamma distributions to predict the health status of a node. They have claimed comparable results for their prediction technique. Daniel Ford, et al. [25] applied the clustering heuristic to classify the failure types of distributed systems. They identified some issues which are not directly related to failure but can be stepping stones towards it, for example, the placement of systems, reboot times and frequency, sudden power failure and battery transfer. Based on their findings, they provided feedback to the data engineers working at Google about how to use proactive approaches to reduce the failure rates in Google's high-performance computing systems. Hertong Song, et al. [26] introduced a homogeneous framework for availability modeling of high-performance systems during runtime. They applied Markov Models for the implementation of the proposed system. Their system can collect event logs, filter the failure correlated logs and provide a detailed analysis of a real-time cluster.

This framework provides comprehensive resource availability and guarantees accurate results. They have demonstrated the results by testing this model on a real-time Lawrence Livermore system [27].

Praveen Yalagandula, et al. [28] analyzed traces of PlanetLab [12], DNS [29] and a cluster with 100 nodes to find the failure causes in these systems. Despite previous findings, their work focused on significant properties of initial discoveries. They have presented the following findings:

- The availability does not always mean better MTTF or MTTR;
- By utilizing a good predictive model, MTTF and MTTR may be accurately predicted;
- Engineers should not rely on the building design to have predictive alerts;
- While designing the high computing systems, correlated factors should be kept in mind.

Song Fu, [61] designed an infrastructure of a distributed virtual machine which is reconfigurable in terms of node selections for proactive failure prediction and management. They suggested node selection methods for this reconfigurable machine to make it failure-aware by taking performance and reliability metrics into account. Their methods provide effective and reliable node selection methodologies and deduced best-fitting nodes. They conducted tests on the NAS Parallel Benchmark [31] and stated that the job accomplishment rate can be enhanced by the factor of 17.6% with the new task completion rate of 91.7%.

Chokchai, et al [32] provided a solution of high availability for open source cluster by employing a robust technique of component redundancy. They proposed an efficient failure repair model which can predict the availability factor for the open source cluster. For modeling the behavior and high availability they employed Stochastic reward nets [33] and provided a robust technique. After exhaustive testing, they provided System Availability of 0.99% with Mean cluster downtime 34.9 minutes per year for cluster having four nodes. Jiexing Gu, et al. [42] presented a two-way dynamic prediction technique based on meta-learning. They introduced a continuous increase of training data while the system is in the running state and the technique to transform the rules of failure by tracing the prediction in terms of accuracy. They stated 70% accuracy in failure prediction of the runtime system with 10% false alarm rate.

Liang, et al. [34] collected comprehensive logs of IBM BlueGene/L [38] and showed that log data can be converted into classification data by using minimal resources and computation efforts. They further classified that data into sub-classes of failure prediction by employing different classifiers like Support Vector Machines [35], RIPPER [36], KNN [37] and their own customized K-NN. Their results suggested that customized KNN outperformed other classifiers in terms of failure prediction with an F measure of 70.0% for a 12-hour and 50.0% for a 6-hour prediction timeframe.

Li, et al. [39] proposed an adaptive method FT-Pro which provides fault management methodology by selecting the appropriate action in case of failure. The system can select from data migration, checkpoint or take no action based on the failure estimation by calculating the cost of a decision. They tested this system on event logs from NCSA [40] and stated that if a prediction technique even with minimum failure prediction accuracy is applied, their system can outperform all the traditional fault recovery techniques with the 30% margin rate and it can reduce the execution time of the application significantly. Similarly, Fulp, et al. [41] proposed an SVM based approach that can predict the failure of the cluster based on its previous event logs. They applied sliding-window based message passing technique which can robustly predict an upcoming failure. They tested their system on a Linux cluster and stated 73% accuracy for advance failure prediction.

Song Fu, [30] developed the spherical-covariance based model PREdictor, having adjustable parameters for timescale to learn the spatiotemporal features of the failure prediction data. They provided correlations of important features with the failure events and predicted the future failures. For the evaluation of performance of PREdictor they employed various types of offline and online prediction tests and achieved 76.0% precision in offline estimation of failure and 70.0% accuracy in online prediction.

III. PROPOSED METHODOLOGY

A. Long Short-Term Memory Units (LSTMs)

To grasp the spatiotemporal information of HPCC's data for robust failure prediction, we employed the network proposed by [49] which is called Long Short-Term Memory (LSTM). This network is an optimized version of Recurrent neural network (RNN) [54] with various enhancements in terms of robustness and a solution to the vanishing gradient problem [59]. LSTM network remembers the errors from a long state of time (solving a long-term dependency problem). This error is back-propagated over time through hidden layers. It can optimize the loss over several time stamps even more than 1000 – thus, it is easier to maintain a constant error throughout the training, which provides a strong analysis of trends in time series data [53]. Similar to RNN, LSTM has a chain like an assembly of repeating modules/nodes/cells. However, instead of depending upon a single layer of the neural network (in the repeating nodes), LSTM has four layers of NN, interacting with each other in a specified way. The chain-like the assembly of LSTM can be visualized in Fig. 1.

As discussed above, the repeating cell of LSTM retains information in a different way as compared to the normal drift of the RNN cell. The LSTM cell chooses intelligently from the prescribing decisions of read, write, discard and store, with the help of gates much similar to the computer memory. However, the configuration of these gates\activation functions is analog, which makes it differentiable and appropriate for the backpropagation. Technically, these activation functions, activate the input signals, the same way as in convolutional neural network [54]. These functions discard or retain information by measuring its importance and strength with the help of weights. These weights are learned during the recurrent networks' training process. Fig. 2, demonstrates the data flow in the recurrent cell and the control of gates.

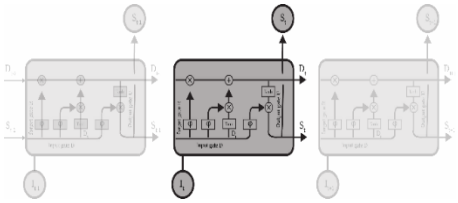

Fig. 1 Chain like the assembly of repetitive cells in LSTM

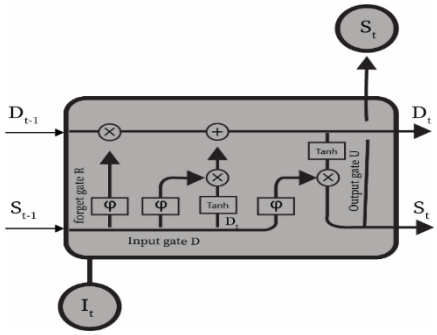

Fig. 2 Single repetitive cell of LSTM architecture

In the diagram above, the first sigmoid function represents the forget gate that provides the information, that should be retained from the previous cells' output. The second and third activation functions show the input gate and save vector as Tanh and sigmoid respectively. The final sigmoid illustrates the output gate which highlights, what information will be provided to the hidden state. As it can be clearly visualized from Fig. 2, an LSTM repetitive cell has the following four types of gates:

- Forget Gate represented by R;
- Input Gate represented by D;
- Input Modulation Gate or save vector represented by D_t;
- Output Gate represented by U;

B. Forget Gate

Forget gate takes the output from the previous cell S_{t-1} as an input signal and take decisions about what information is ineffective, which should be discarded in the output state S_t, this helps to retain only relevant information for impactful learning. This gate is comprised on a Sigmoid activation function [55] which squeezes the input signal between 0 and 1. The mathematical expression of forget gate can be described as (1)

$$R_t = \varphi\,(Weight_R[S_{t-1}, I_t] + b_R) \qquad (1)$$

Where W_f is weights, S_{t-1} is the input from the previous cell, I_t is current input and b_R is biases. The Fig. 3 illustrates the forget function.

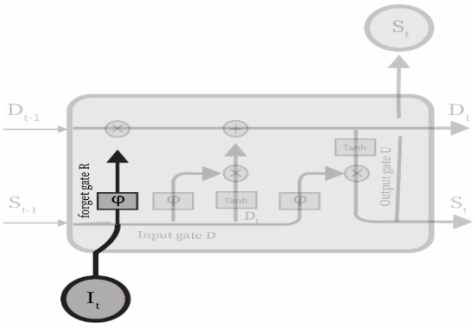

Fig. 3 Visualization of forget gate in LSTM

C. Input Gate and Save Vector

The block decides, what information to save in the cell's current state by using the combination of two gates: the input gate D_t and the save vector $\check{D}_t$. The input gate is comprised of a Sigmoid activation function as well, which decides the values that will be updated in the current state. The Save vector gate produces a vector of candidate values (what to output) based on the decision provided by the input gate. It chooses Tanh [56] as an activation function. The input gate and save vector can be represented as an equation (2) and (3) respectively.

$$D_t = \varphi\,(Weight_D[S_{t-1}, I_t] + b_D) \qquad (2)$$

$$\check{D}_t = tanh\,(Weight_D[S_{t-1}, I_t] + b_D) \qquad (3)$$

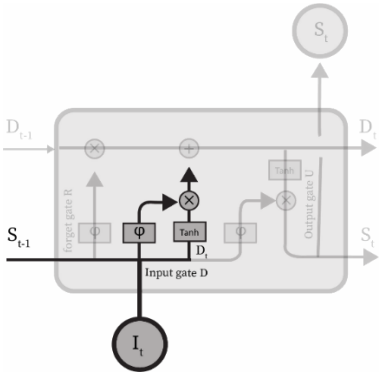

Fig. 4 Visualization of input gate and save vector in LSTM

Based on the concatenated candidates provided by the input gate and save vector, the old cell state is updated from D_{t-1} to D_t, which is the current cell's state. For the implementation of that, the output of R_t (forget gate) is multiplied with the old cell state D_{t-1} and then added with $I_t \times D_t$. The resultant state $\check{D}_t$ has new candidate values, decided by concatenation of input gate and save vector as described in (4)

$$\check{D}_t = R_t \times D_{t-1} + I_t \times D_t \qquad (4)$$

D. Input Gate and Save Vector

Finally, the output is decided based on the updated cell's state via output gate. This gate consists of two activation functions, Sigmoid and Tanh. First, the sigmoid function decides, what part of candidates' vector will be the productive output. Then, with the help of tanh, the output of sigmoid is pushed between -1 and 1. This can be mathematically expressed as (5) and (6)

$$U_t = \varphi\,(Weight_U[S_{t-1}, I_t] + b_U) \quad (5)$$

$$D_t = U_t \times Tanh\,(\check{D}_t) \qquad (6)$$

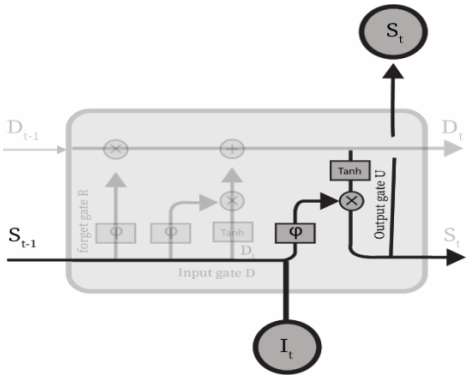

Fig. 5 Visualization of output gate in LSTM

E. Reinforcement learning for hyperparameters optimization

Due to the obvious reason that machines are highly unpredictable in work patterns, the output trends change all the time. Even if LSTM is employed to generate extremely accurate failure prediction models, the results can vary due to HPCC' behavior and unpredictability. This means the models should be constantly optimized in real-time working, so we don't have to suffer from such issues. To make this happen, the possible solutions can be:

- Addition, removal or change focus towards more correlated features;
- Improve the model's prediction through the incorporation of feedback, use appropriate values for hyper parameters;
- Use the loss values for current predictions vs. real-time value and add the mechanism of reward/punishment (Reinforcement Learning); due to

Because of the reason, we have a limited number of pages, rather than explaining the basis of RL, it is decided to jump into implementation details of the RL approaches. The only basic knowledge needed to understand here is: that RL algorithms work exactly like the human learn some gaming techniques through reward/punishment mechanism. If he/she gets the rewards of a particular score, he/she will continue to use that approach and learn it for further reuse. On the other hand, if he/she gets the punishment/lose the scores, it is obvious to lose that move and don't rely on it. From the implementation point of view, we will utilize the Q-learning approach where the value of an action (from state "A" to state "B") is calculated. For this, we will exploit the Rainbow technique which is a blend of 7 Q-learning based algorithms. For the critical purpose of setting accurate reward policy, we have employed following mathematical expression:

$$R = P_{accuracy} + 2 \times P_{loss} \quad (7)$$

Where R is a reward, $P_{accuracy}$ is the accuracy of prediction and P_{loss} is a loss calculated by RL techniques.

F. Rainbow technique

The Rainbow is an advanced Q-learning approach proposed by [50] for deep reinforcement learning. It efficiently combines 7 Q-learning algorithms to provide robust results. This approach has utilized the following techniques to achieve the best results:

- DQN, which represents Q value by exploiting neural network-based learning to minimize the loss function to represent the Q value. Similar to supervised (deep) learning, in DQN we train a neural network and try to minimize a loss function;
- Double Q Learning is exploited to handle the problem of overestimation effectively;
- Prioritized replay chooses a mechanism of distribution to give higher priority to the samples, which causes higher Q loss in the last iterations instead of replying to the randomized samples;
- Dueling networks architecture having two streams of networks. One for the value optimization and other for advantage calculation. To merge both streams, we employed special aggregator [58];
- Multi-step learning computes the Q-values using N-step returns instead of learning it from just a single next step;
- Distributional RL can approximate the distribution of Q-values instead of using an averaging mechanism;
- Noisy Nets are employed in the network, which helps it to learn the difference between noise and targeted values;

Technically, the LSTM's prediction output is fed to the Rainbow network to estimate reward for advantage learning. We employed Bellman equation [60] to estimate the temporal difference error at time T as follows:

$$Temporal_{err}(T) = Value_{state}(T) + \frac{reward(T) + \gamma\, Value_{state}(T+1) - Value_{state}(T)}{\beta} - [A - dv\,(state(T), action(T))] \quad (8)$$

Where β is a constant for scaling the difference between optimal and targeted values. While, $Value_{state}$ is the value of the current state the at timestamp T.

IV. DATASET

The LANL is a publicly available dataset collected by Los Alamos National Laboratory (LANL) [5]. This repository has a dataset of HPCC for the past 9 years, which covers 8 HPCC, 22 cluster nodes, 4,750 nodes and 24,101 high computing processors. This dataset has records of each failure that causes a node downtime, anytime during the 9-year period. LANL covers all types of failures in HPCC including five major categories of failure of hardware, network, human error, software, undetermined. For every type of failure, the record has: the time it started, repairing time, the node affected, and the root cause of failure. The details about dataset have been provided in Table 1 below:

TABLE I
LANL DATASET DESCRIPTION

*Node Type	*Procs	*Failures	*Nodes	*Systems
128-256 proc. NUMA	9000	8464	78	4
2/4-way SMPs	15101	12607	4672	18

As described in Table 1, the cluster nodes are of two types: Nonuniform-Memory-Access (NUMA) or 2/4-way Symmetric-Multiprocessing (SMP), according to the memory sharing process. On the basis of memory and processing component (procs) models, the dataset is divided into 8 HPCC (named A-H).

V. RESULTSAND EVALUATION

A. Experimental setup

We have employed 3 layers of stacked `LSTM` with 25 input units and 500 hidden units, and one `Dense` layer with 1 output - the failure prediction. For the initializer, we employed Xavier initializer and we will use L1 loss. We used `Adam` optimizer with variable learning rate. We trained the LSTM model for 500 epochs with early stopping mechanism. Data preprocessing is exploited to fill the missing values with zeros. The redundant values from the dataset using PCA dimensionality technique. This experiment was conducted with the LANL dataset from the time span of May 1997 and April 2002. Then, we evaluated the online prediction performance from May 2002 to April 2005.

B. Results

The results provided by our proposed mythology are explained below:

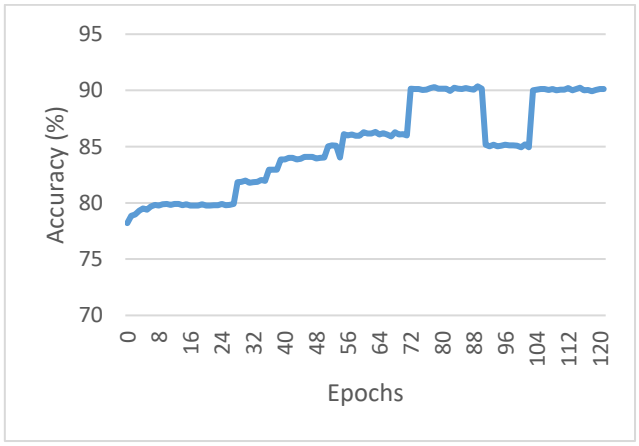

Fig. 6 Accuracy of hardware failure

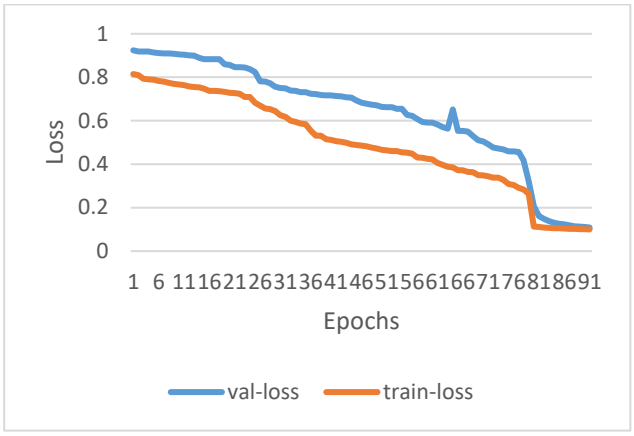

Fig. 7 Loss of Hardware Failures

Fig. 6 presents the visualization of the accuracy of validation for the Hardware failure prediction. As it can be clearly seen that accuracy started from 73.0% and went till 90.01% with quite a few fluctuations. As we have employed early stopping mechanism, we stopped our model at 121st epoch as we have achieved the best weights at this point. Similarly, Fig. 7 shows the loss of our model during a training session, the orange line represents training loss while the blue line represents validation loss. From the beginning, the loss continued to decrease steadily, till 73rd epoch and then with a sudden fluctuation it is decreased by the value of 0.1. After 80th epoch training and validation loss overlapped with a marginal difference.

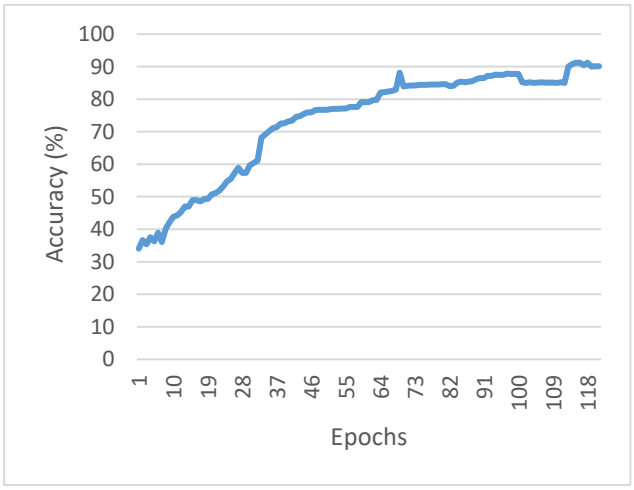

Fig. 8 Accuracy of human error failure

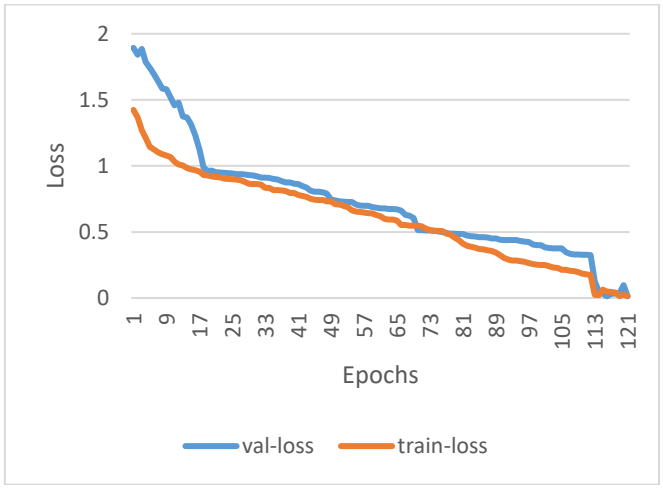

Fig. 9 Loss of human error failure

Fig. 8 provides the illustration of validation accuracy for the Human error-based failure prediction. The accuracy started with the low starting point at 36.2%, but it increased up to 83.60% around 110th epoch. Then it remained consistent with marginal variations for the rest of the epochs. Likewise, the loss graphs can be seen in Fig. 9, where blue trendline represents a validation loss while the orange trendline shows training loss. There was a huge error between two losses in the beginning but it became steady and overlapped around 21th epoch. The final value of loss was 0.03 at 121th epoch.

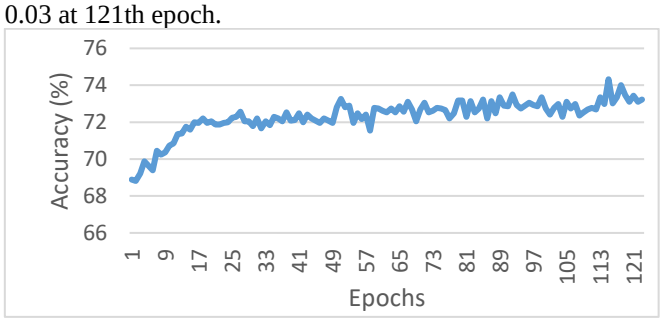

Fig. 10 Accuracy of network failure

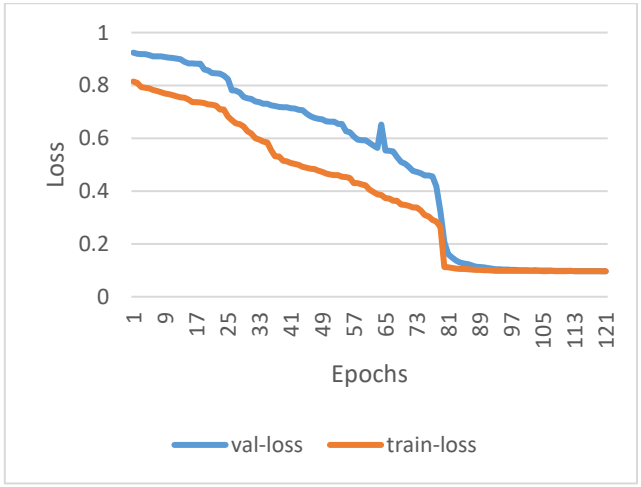

Fig. 11 Loss of network failure

The validation accuracy and loss graphs of network failure can be seen in Fig. 10 and Fig. 11 respectively. From Fig.10 it can be seen there are too many fluctuations in the validation accuracy of network failure with top accuracy of 74.35% at 116th epoch, while the losses have fewer fluctuations with a constant step of 0.03 at 82nd epoch. The reason for this strange behavior is unknown to us, we will try to investigate it in our future work.

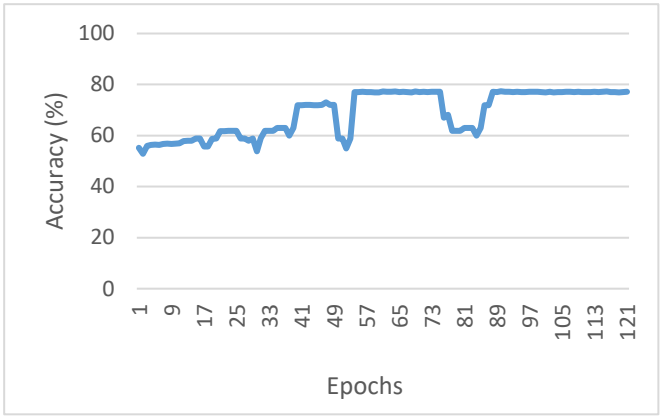

Fig. 12 Accuracy of software failure

Fig. 12 presents the validation accuracy of software failures. The trend started from a high point a 58.3%, but after some variations, it became stable the 47th epoch, next possible fluctuation can be seen at a 74th epoch which again stabilized at 97th epoch at the rate of 79.92% and continued for the rest of the epochs. One other thing worth noticing is a flat accuracy step for the most for the epochs can be visualized. This performance has been seen even after 121th s epochs till the 500 epochs.

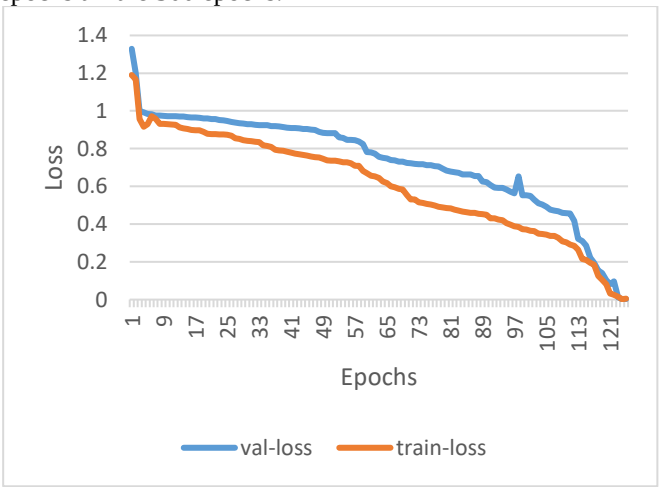

Fig. 13 Loss of software failure

The loss graph of software failure can be seen in Fig. 13 where blueline shows the validation loss and the orange line shows the training loss. Validation loss started from 1.3% and continued to decrease even after 121st epoch but for the better visualization purpose, we only included till 121st epoch. The final value of the loss was 0.0012% at 403rd epoch and remained constant after that.

Fig. 14 and Fig. 15 Shows the undetermined failures prediction's accuracy. It can be seen too many fluctuations in the validation accuracy with the final accuracy of 82.73 at the 120th epoch. While the losses are shown in Fig. 15 shows the huge loss error in initial epochs, but overlapped around 38th epoch and continued after that.

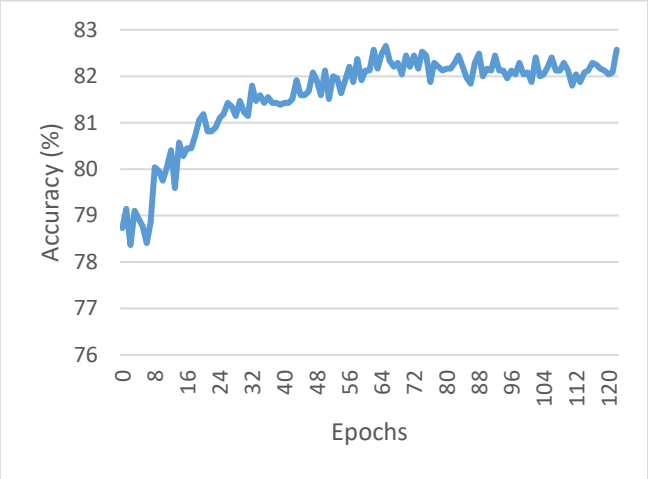

Fig. 14 Accuracy of undetermined

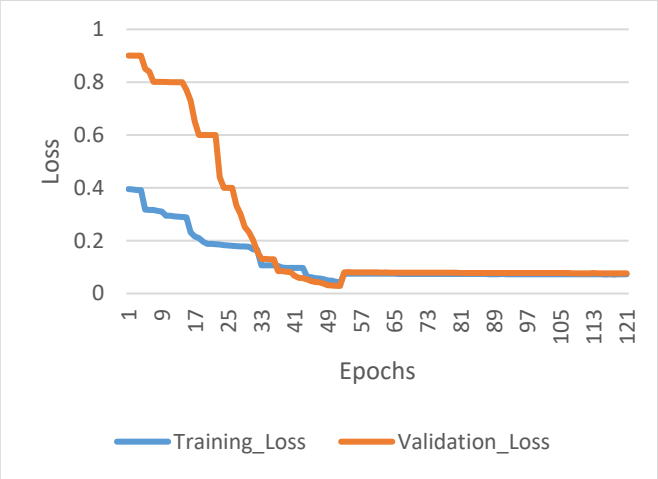

Fig. 15 Loss of undetermined

C. *Comparative analysis*

For the comparative analysis, we applied state of the art machine learning based techniques and provide prediction results. For this analysis, the LANL data set is divided into 8 segments based on node types. Five types of techniques: Gaussian-naïve bias; Support Vector Classifier; Decision Tree; K- nearest neighbors; Logistic regression and Random forest are applied on all 8 segments for failure prediction of all sub-types. The tables below provide a complete analysis of the applied algorithm's accuracy with respect to all failure types.

TABLE II
COMPARATIVE ANALYSIS OF ACCURACY FOR HARDWARE-BAASED FAILURES

Seg	Gaussian-NB	SVC	DT	KNN	LG	RF
1	0.5828	0.6771	0.7049	0.7297	0.5834	**0.7303**
2	0.7162	0.6621	0.6624	**0.7567**	0.6621	0.7702
3	0.5342	0.7671	0.6313	0.7123	**0.7945**	0.7260
4	0.8666	**0.8666**	0.8533	0.8484	0.8666	0.8412
5	0.5324	0.5861	0.6599	0.6040	0.6353	**0.6465**
6	0.7152	**0.7328**	0.6635	0.7182	0.7198	0.6950
7	0.6080	0.6649	**0.6907**	0.6185	0.5773	0.6752
8	0.6071	0.5714	0.4642	**0.6423**	0.5714	0.5357

As it can be seen from Table-2 that for hardware-based failure prediction for segment 1 random forest provided the highest accuracy of 73.03%, for segment-2 KNN performed better as 75.67% accuracy, for segment-3, logistic regression provided 79.45% accuracy. While for segment-4 SVC won the battle with 86.66%, for segment-5 random forest given 64.65% accuracy, on segment-6 SVC again stricken with 73.28% accuracy. Decision tree provided 69.07% accuracy and finally, KNN has provided 64.23% accuracy.

TABLE III
COMPARATIVE ANALYSIS OF ACCURACY FOR UNDETERMINED FAILURES

Seg	Gaussian-NB	SVC	DT	KNN	LG	RF
1	0.7140	**0.7629**	0.4119	0.2840	0.7158	0.4276
2	0.7297	0.7162	0.7837	0.7297	**0.8108**	0.3243
3	0.6712	0.6882	**0.8356**	0.7671	0.5082	0.7954
4	0.4933	0.4193	**0.5833**	0.2866	0.3293	0.1893
5	0.7561	0.7964	0.7449	0.7651	**0.7919**	0.7718
6	0.2296	**0.7779**	0.6943	0.7643	0.7739	0.7603
7	0.4587	0.7731	**0.8092**	0.8092	0.7731	0.7989
8	0.4285	0.3292	**0.7385**	0.2928	0.6928	0.5871

Table-3 presents the analysis of accuracy for undetermined failures. It can be seen that the decision tree provided the highest accuracy values of 83.56%, 58.33%, 80.92% and 73.85% for segment-3, segment-4, segment-7, and segment-8 respectively. Similarly, for segment-2 and segment-5 logistic regression given the accuracies of 81.08 and 79.19%. SVC performed better for segment-1 and segment-6 with 76.29% and 77.79% respectively.

TABLE IV
COMPARATIVE ANALYSIS OF ACCURACY FOR SOFTWARE-BASED FAILURES

Seg	Gaussian-NB	SVC	DT	KNN	LG	RF
1	0.6458	0.6851	**0.8077**	0.5818	0.6857	0.4864
2	0.1864	0.4864	0.4959	**0.7864**	0.7219	0.7229
3	0.8631	0.1864	0.3159	0.5989	**0.8638**	0.1863
4	0.4733	**0.4933**	0.1333	0.7733	0.1273	0.4466
5	0.3489	0.7404	0.6577	0.6577	**0.7494**	0.7069
6	**0.8054**	0.5085	0.6860	0.0968	0.0985	0.4552
7	0.6907	0.7886	0.7268	0.7216	**0.8041**	0.7577
8	0.7142	0.7122	0.7142	0.7142	**0.7142**	0.7142

From Table-4 it can be visualized that logistic regression performed best with 86.38%, 74.94%, 80.41% and 71.42% accuracy for segment 3,5,7 and 8 respectively. While for 6th segment Gaussian performed well. Similarly, decision tree, KNN, and SVC provided accuracies of 80.77%, 78.64% and 80.54% accuracy for segment 1, 2 and 4 respectively.

TABLE V
COMPARATIVE ANALYSIS OF ACCURACY FOR HUMAN ERROR-BASED FAILURES

Seg	Gaussian-NB	SVC	DT	KNN	LG	RF
1	**0.7125**	0.6945	0.6753	0.1903	0.7112	0.7109
2	0.6987	0.4019	**0.7398**	0.6387	0.6908	0.3755
3	0.3982	0.3957	0.6897	0.5831	**0.7012**	0.1905
4	**0.7863**	0.4375	0.3951	0.1964	0.6708	0.6764
5	0.6821	0.6831	0.5930	0.6321	0.2964	**0.8648**
6	0.5639	**0.8705**	0.6014	0.3932	0.6931	0.5801

7	0.6745	0.7540	0.8301	**0.8494**	0.6301	0.7924
8	0.2788	0.6112	0.7190	0.7765	**0.7987**	0.7507

TABLE VI
COMPARATIVE ANALYSIS OF ACCURACY FOR NETWORK-BASED FAILURES

Seg	Gaussian-NB	SVC	DT	KNN	LG	RF
1	0.1919	0.5861	0.7178	0.6950	0.6824	**0.7391**
2	0.7143	0.6728	**0.7823**	0.6080	0.6628	0.3271
3	0.5833	0.5698	0.6482	**0.7635**	0.5444	0.1913
4	0.2976	0.1345	0.6589	0.5821	0.5739	**0.6791**
5	0.6943	0.6678	0.7953	0.5498	0.7539	**0.7679**
6	**0.7718**	0.7128	0.7530	0.4878	0.6854	0.5829
7	0.2569	0.7267	0.6184	0.6679	0.6712	**0.8374**
8	0.4387	0.4131	0.6159	0.6527	**0.7319**	0.6093

A similar type of mixed trends can be seen in Table-5 and Table-6 respectively. Due to the lack of pages, it is difficult to explain each table value. However, it is interesting to mention that random forest worked best for network failure for most of the segments i.e., segment-1, segment-4, segment-5, and segment-7.

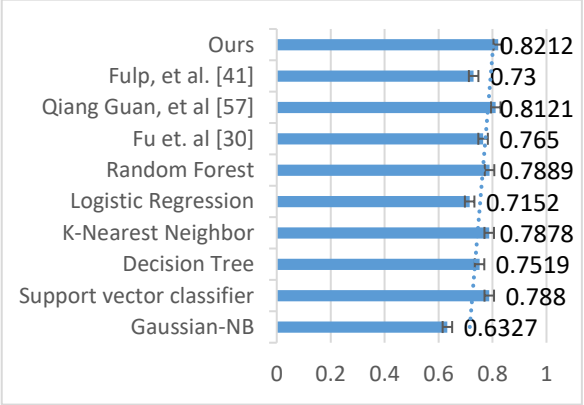

Fig. 16 Comparative analysis of different techniques with the proposed method

Fig. 16 provides a detailed analysis of available techniques with our proposed methodology. The overall accuracy of failures prediction for all five types of failures is 82.12%. It can be seen that our methods have outperformed the previous techniques with a significant margin.

VI. CONCLUSION

This paper presents a novel methodology for the failure prediction of high-performance computing clusters. The system is composed of two main components; an LSTM model for deep residual, spatiotemporal learning and a rainbow-reinforcement learning technique. We have tested this system on LANL dataset and performed detailed analysis of system behavior, for multiple clusters and all five failure types. The system is also tested on overall failure types and outperformed the previous models with 82.12% accuracy. However, there is still room for improvement in terms of accuracy, which we would target in our future research.

REFERENCES

1. Foster, I., Gieraltowski, J., Gose, S., Maltsev, N., May, E., Rodriguez, A., ... & Adams, D. (2004, June). The grid2003 production grid: Principles and practice. In *Proceedings. 13th IEEE International Symposium on High performance Distributed Computing, 2004.* (pp. 236-245). IEEE.
2. Dumitrescu, C. L., Raicu, I., & Foster, I. (2005, November). Experiences in running workloads over grid3. In *International Conference on Grid and Cooperative Computing* (pp. 274-286). Springer, Berlin, Heidelberg.
3. Bolze, R., Cappello, F., Caron, E., Daydé, M., Desprez, F., Jeannot, E., ... & Mornet, G. (2006). Grid'5000: A large scale and highly reconfigurable experimental grid testbed. *The International Journal of High Performance Computing Applications*, *20*(4), 481-494.
4. Iosup, A., Jan, M., Sonmez, O., & Epema, D. H. (2007, September). On the dynamic resource availability in grids. In *2007 8th IEEE/ACM International Conference on Grid Computing* (pp. 26-33). IEEE.
5. Iosup, A., Epema, D., Couvares, P., Karp, A., & Livny, M. (2007, May). Build-and-test workloads for grid middleware: Problem, analysis, and applications. In *Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid'07)* (pp. 205-213). IEEE.
6. Casanova, H. (2007). Benefits and drawbacks of redundant batch requests. *Journal of Grid Computing*, *5*(2), 235-250.
7. Chen, Y., Zhang, B., & Chen, C. (2011, June). Modeling and performance analysis of P2P live streaming systems under flash crowds. In *2011 IEEE International Conference on Communications (ICC)* (pp. 1-5). IEEE.
8. BBC, Blackouts cause n America chaos, BBC News. [Online] Available: http://news.bbc.co.uk/2/hi/americas/3152451.stm (Aug 2003).
9. Kundur, P., Taylor, C., & Pourbeik, P. (2007). Blackout experiences and lessons, best practices for system dynamic performance, and the role of new technologies. *IEEE Task Force Report*.
10. Nygren, Erik, Ramesh K. Sitaraman, and Jennifer Sun. "The akamai network: a platform for high-performance internet applications." *ACM SIGOPS Operating Systems Review* 44.3 (2010): 2-19.
11. Javadi, B., Kondo, D., Iosup, A., & Epema, D. (2013). The Failure Trace Archive: Enabling the comparison of failure measurements and models of distributed systems. *Journal of Parallel and Distributed Computing*, *73*(8), 1208-1223.
12. Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M., & Bowman, M. (2003). Planetlab: an overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review*, *33*(3), 3-12.
13. Pertet, S., & Narasimhan, P. (2005). Causes of failure in web applications . Technical Report CMU-PDL (VOL.92)-05-109, Carnegie Mellon University.
14. Morshed, M. B., Dye, M., Ahmed, S. I., & Kumar, N. (2017). When the Internet Goes Down in Bangladesh. In *CSCW* (pp. 1591-1604).
15. West, D. M. (2016). Internet shutdowns cost countries $2.4 billion last year. *Center for Technological Innovation at Brookings, Washington, DC*.
16. https://www.thestar.com.my/tech/tech-news/2019/03/14/facebook-instagram-and-whatsapp-face-connectivity-issues/
17. Various sources, Microsoft windows live Hotmail, dec 30, 2010 to Jan. 2, 2011, More information: http://windowsteamblog.com/windows_live/b/windowslive/.archive/2011/01/03/hotmail-email-access-issue-now-resolved.aspx , http://www.infoworld.com/t/software-service/hotmail-fail-microsoft-lays-egg-in-the-cloud-616 (Dec 2010).
18. Reuters, Knight capital was hardly the first: Timeline of market glitches, Reuters report. [Online] Available: http://www.foxbusiness.com/industries/2012/08/02/knight-capital-was-hardly-first-timeline-market-glitches/ (Aug 2012).
19. Various sources, Amazon EC2 and RDS fail, april 21, 2011, from 12am to 12pm (12 hours), More information: http://www.zdnet.com/blog/projectfailures/cio-analysis-examining-amazons-cloud-failure/13152 , http://bits.blogs.nytimes.com/2011/04/21/amazon-cloud-failure-takes-down-web-sites/ , and http://www.economist.com/node/18620774/print . (Dec 2010).
20. The raw data and additional information are available at www.lanl.gov/projects/computerscience/data.
21. Schroeder, B., & Gibson, G. (2009). A large-scale study of failures in high-performance computing systems. *IEEE transactions on Dependable and Secure Computing*, 7(4), 337-350.
22. Zheng, W., Wang, Z., Huang, H., Meng, L., & Qiu, X. (2016, May). A prediction approach for correlated failures in distributed computing systems. In *2016 IEEE International Conference on Communications (ICC)* (pp. 1-6). IEEE.

23. Hacker, T. J., Romero, F., & Carothers, C. D. (2009). An analysis of clustered failures on large supercomputing systems. *Journal of Parallel and Distributed Computing*, *69*(7), 652-665.
24. Stephan, Michael, and Jutta Docter. "JUQUEEN: IBM Blue Gene/Q® Supercomputer system at the Jülich supercomputing centre." *Journal of large-scale research facilities JLSRF* 1 (2015): 1.
25. Ford, D., Labelle, F., Popovici, F., Stokely, M., Truong, V. A., Barroso, L., ... & Quinlan, S. (2010). Availability in globally distributed storage systems.
26. Song, H., Leangsuksun, C., Nassar, R., Gottumukkala, N. R., & Scott, S. (2006, April). Availability modeling and analysis on high performance cluster computing systems. In *First International Conference on Availability, Reliability and Security (ARES'06)* (pp. 8-pp). IEEE.
27. Beiersdorfer, P., Brown, G.V., Hildebrandt, L., Wong, K.L. and Ali, R., 2001. Multiparameter data acquisition system for spectroscopy. *Review of Scientific Instruments*, *72*(1), pp.508-512.
28. Yalagandula, P., Nath, S., Yu, H., Gibbons, P. B., & Seshan, S. (2004, December). Beyond Availability: Towards a Deeper Understanding of Machine Failure Characteristics in Large Distributed Systems. In *WORLDS*.
29. Drako, D. (2013). *U.S. Patent No. 8,447,856*. Washington, DC: U.S. Patent and Trademark Office.
30. Fu, S., & Xu, C. Z. (2007, November). Exploring event correlation for failure prediction in coalitions of clusters. In Proceedings of the 2007 ACM/IEEE conference on Supercomputing (p. 41). ACM.
31. Bailey, D. H. (2011). NAS parallel benchmarks. *Encyclopedia of Parallel Computing*, 1254-1259.
32. Liu, T., & Song, H. (2003, December). Availability prediction and modeling of high mobility OSCAR cluster. In *2003 Proceedings IEEE International Conference on Cluster Computing* (pp. 380-386). IEEE.
33. Muppala, J., Ciardo, G., & Trivedi, K. S. (1994). Stochastic reward nets for reliability prediction. *Communications in reliability, maintainability and serviceability*, *1*(2), 9-20.
34. Liang, Y., Zhang, Y., Xiong, H., & Sahoo, R. (2007, October). Failure prediction in ibm bluegene/l event logs. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*(pp. 583-588). IEEE.
35. Gunn, S. R. (1998). Support vector machines for classification and regression. *ISIS technical report*, *14*(1), 5-16.
36. Chawla, N. V., Lazarevic, A., Hall, L. O., & Bowyer, K. W. (2003, September). SMOTEBoost: Improving prediction of the minority class in boosting. In *European conference on principles of data mining and knowledge discovery* (pp. 107-119). Springer, Berlin, Heidelberg.
37. Liao, Y., & Vemuri, V. R. (2002). Use of k-nearest neighbor classifier for intrusion detection. *Computers & security*, *21*(5), 439-448.
38. Gara, A., Blumrich, M. A., Chen, D., Chiu, G. T., Coteus, P., Giampapa, M. E., ... & Liebsch, T. A. (2005). Overview of the Blue Gene/L system architecture. *IBM Journal of research and development*, *49*(2.3), 195-212.
39. Li, Y., & Lan, Z. (2006, May). Exploit failure prediction for adaptive fault-tolerance in cluster computing. In *Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06)* (Vol. 1, pp. 8-pp). IEEE.
40. Katz, E. D., Butler, M., & McGrath, R. (1994). A scalable HTTP server: The NCSA prototype. *Computer Networks and ISDN systems*, *27*(2), 155-164.
41. Fulp, E. W., Fink, G. A., & Haack, J. N. (2008). Predicting Computer System Failures Using Support Vector Machines. *WASL*, *8*, 5-5.
42. Gu, J., Zheng, Z., Lan, Z., White, J., Hocks, E., & Park, B. H. (2008, September). Dynamic meta-learning for failure prediction in large-scale systems: A case study. In *2008 37th International Conference on Parallel Processing* (pp. 157-164). IEEE.
43. The TOP500 Supercomputer Sites. www.top500.org
44. Oliner, A. J., Sahoo, R. K., Moreira, J. E., Gupta, M., & Sivasubramaniam, A. (2004, April). Fault-aware job scheduling for bluegene/l systems. In *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.* (p. 64). IEEE.
45. Chakravorty, S., Mendes, C., & Kalé, L. (2005, February). Proactive fault tolerance in large systems. In *HPCRI Workshop in conjunction with HPCA* (Vol. 2005, pp. 1-7).
46. Hoffmann, G. A., Salfner, F., & Malek, M. (2011). Advanced failure prediction in complex software systems. Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II, Institut für Informatik.
47. Schulz, M., Bronevetsky, G., Fernandes, R., Marques, D., Pingali, K., & Stodghill, P. (2004, November). Implementation and evaluation of a scalable application-level checkpoint-recovery scheme for MPI programs. In *Proceedings of the 2004 ACM/IEEE conference on Supercomputing* (p. 38). IEEE Computer Society.
48. Sahoo, R. K., Oliner, A. J., Rish, I., Gupta, M., Moreira, J. E., Ma, S., ... & Sivasubramaniam, A. (2003, August). Critical event prediction for proactive management in large-scale computer clusters. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 426-435). ACM.
49. Gers, F. A., Schmidhuber, J., & Cummins, F. (1999). Learning to forget: Continual prediction with LSTM.
50. Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., ... & Silver, D. (2018, April). Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
51. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
52. Ciardo, G., & Trivedi, K. S. (1993). A decomposition approach for stochastic reward net models.
53. Sak, H., Senior, A., & Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth annual conference of the international speech communication association*.
54. Mao, J., Xu, W., Yang, Y., Wang, J., Huang, Z., & Yuille, A. (2014). Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv preprint arXiv:1412.6632*.
55. Yin, X., Goudriaan, J. A. N., Lantinga, E. A., Vos, J. A. N., & Spiertz, H. J. (2003). A flexible sigmoid function of determinate growth. *Annals of botany*, *91*(3), 361-371.
56. Kalman, B. L., & Kwasny, S. C. (1992, June). Why tanh: choosing a sigmoidal function. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks* (Vol. 4, pp. 578-581). IEEE.
57. Guan, Q., Zhang, Z., & Fu, S. (2012). Ensemble of bayesian predictors and decision trees for proactive failure management in cloud computing systems. *Journal of Communications*, *7*(1), 52-61.
58. Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., & De Freitas, N. (2015). Dueling network architectures for deep reinforcement learning. arXiv preprint arXiv:1511.06581
59. Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *6*(02), 107-116.
60. Bardi, M., & Capuzzo-Dolcetta, I. (2008). Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations. Springer Science & Business Media.
61. Fu, S. (2010). Failure-aware resource management for high-availability computing clusters with distributed virtual machines. *Journal of Parallel and Distributed Computing*, *70*(4), 384-393.

Author Profile

Kamaljit Kaur was born in Kapurthala, Punjab, India.She completed B.Tech. with Distinction from PunjabTechnical University, and M.Tech. with Gold Medalfrom Guru Nanak Dev University, Amritsar. She ispursuing PhD in the field of Cloud Computing fromGuru Nanak Dev University, Amritsar. From 2007 to2008 she was employed with Dr. B.R.AmbedkarNIT, Jalandhar as Lecturer and from 2010 to 2012, she worked as an Assistant Professor at LovelyProfessional University, Jalandhar. She joined Guru Nanak Dev University,Amritsar in July 2012 where she is currently working as an AssistantProfessor. Her research interests are in Resource Provisioning in CloudComputing, Resiliency in Cloud Computing and Distributed Systems, BigData, IoT. Kamaljit Kaur has published and presented more than 45 papersin scientific journals and international conferences.

Kuljit Kaur Chahal is an assistant professor in theDepartment of Computer Science, Guru Nanak DevUniversity, India. She received her PhD degree incomputer science from Guru Nanak Dev University,India. Her research interests are in distributedcomputing, Web services security, and open-sourcesoftware