Cognitive Neural Network Classifier for Fault Management in Cloud Data Center

S.Indirani[1]
Research Scholar
Department of Computer Science
Mother Teresa Women's University, Kodaikanal-624102

Dr.C.Jothi Venkateswaran[2]
Former Associate Professor and Head
PG and Research Department of Computer Science
Presidency College (Autonomous), Chennai-600 005

***Abstract*—Pro-actively handling the fault in data center is a means to allocate the VM to Host before failures, so that SLA meets for the tasks running in the data center. Existing solution [1] on fault prediction in datacenter is based on a single parameter of temperature and the fault tolerance is implemented as a reactive solution in terms of VM replication. Different from these works, a proactive fault tolerance with fault prediction based on deep learning with multiple parameters is proposed in this work. In this work Cognitive Neural Network (CNN) is used to predict the failure of hosts and initiate migration or avoid allocation to the hosts which has high probability of failures. Hosts in the data center are scored on failure probability (FP-Score) based on parameters collected at various levels using CNN. VM placement and migration policies are fine-tuned using FP-Score to manage the failure proactively.**

***Keywords*—Deep learning; Cognitive Neural Network (CNN); FP-Score; fault tolerance; VM allocation; VM migration**

I. Introduction

Cloud data center has become a low cost solution for hosting of users computation and storage due to its unlimited resources on demand and pay as go model. Increasing number of users and enterprises are migrating their storage and computations to cloud data centers. Host failures happen in data center due to various reasons like memory exhaustion, hardware failures, software failures etc. Even in case of host failures ensuring the continuity of user's tasks with full or partial recovery in least latency is an important fault tolerance characteristic. Without it the quality of service of the data center is reduced and it will impact the business of data center provider.

Proactive fault tolerance is a way to reduce the down time and ensure higher QOS for the data center. It involves prediction of VM or host failures in advance and corrective actions to avoid the failure or in case of failure, reduce the impact of failure. Proactive fault tolerance has the overhead of increased resource consumption but considering reactive fault tolerance, it reduces the downtime and for the increased QOS provided, the overhead is reasonable. Deep Learning models are the latest trend in machine learning. They are used for various predictions in different domains of economic, health care, weather forecast and manufacturing etc. Deep learning models have the capability to learn the high quality features and semantic relations automatically from the structured and unstructured information compared to previous machine learning models where features are selected manually.

In this work host fault is modeled using CNN. The features collected across all of the OSI and OS layer at each sampling period is used as input for the prediction model. From the error logs collected over a sampling period from Cloud and OS layer a fuzzy failure probability score (FP-Score) is calculated. Training set is created labeling the OSI and OS layer features with FP-Score. CNN model is trained to predict the FP-Score from the Cloud and OS metrics. The VM Placement and VM migration policies in data center are modified in accordance with FP-Score. Dynamic replication strategy is also proposed for VM in certain cases to reduce the impact of failure.

II. Related Work

In [1] fault tolerant scheduling to enhance the reliability of tasks is proposed. The sub reliability requirements of tasks are calculated and from it the VM reliability is calculated. For critical task VM, reliability is ensured with replication.CPU temperature based failure prediction is proposed in [2]. The prediction function model for CPU temperature in the data center is modeled as

$$f(t|A,\omega,t_i,t_{i+1}) = \begin{cases} e^{t}, & 0 \le t \le t_i \\ e^{t_i}, & t_i \le t \le t_{i+1} \\ A\sin(\omega t - \omega t_{i+1}) + e^{t_i}, & t_{i+1} \le t \le t_{i+2} \end{cases}$$

Based on the temperature model, the deteriorating physical machines are identified and the VMs are migrated from the rest of the physical machines selected using PSO based optimization. The time to recover from failure is very short in case of temperature based failure prediction. In [3] ranking based framework is proposed to locate significant components in cloud applications and rank them to provide fault tolerance for those components. The cloud application is modeled as weighted directed graph with each component as vertex and the component invocation as edges. The weight value of the edge is calculated based on invocation frequency.

$$W(e_{ij}) = \frac{frq_{ij}}{\sum_{j=1}^{n} frq_{ij}}$$

The significance value of the component (or node) is calculated as

$$V(C_i) = \frac{1-d}{n} + d \sum_{k \epsilon N(c_i)} V(C_k)\, W(e_{ki})$$

Where n is the number of components, $N(c_i)$ is the set of components invoking C_i. The significance value above a threshold is determined as component which requires reliability and those components are replicated to ensure reliability. Fault prediction based on Byzantine fault detection is proposed in [4]. The fault model is constructed using service resource usage model and message transmission model of the cloud applications. Byzantine fault tolerance techniques are widely employed for building reliable systems but the parameters for modeling based on resource usage alone is not sufficient in data centers. The reliability of each host is modeled in terms of host ability to complete the task successfully in [5]. The task migration from low reliability node to high reliability is done for proactive fault tolerance. In [6] VM health status is monitored frequently and compared against QOS parameters to detect violations. Based on the violations, the risk level of VM is calculated. Load is shifted from high risk VM to low risk to provide increased reliability. A online predictive model for job failure using statistical learning techniques in proposed in [7]. Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), and Logistic Regression (LR) are applied for statistical learning. Time varying patterns of job failures and their system dependency is used to create a failure prediction model of tasks. The model is only used for aborting tasks which are predicted to fail and avoid the resource wastage. But the idea of predicting failures from statistical modeling of job attributes and system attributes can be used for migration decisions. The use of logs for error diagnosis is explored in [7]. Failure of system can be predicted based on the error logs. In [8] simple machine learning based on sparse coding is used to identify anomaly conditions. The model is trained for normal class data alone instead of both normal and abnormal class data. Normal class data captures run time behavior of a job that has not experienced a performance fault. The features deviation from normal class data is diagnosed as anomaly. The work in [9] proposed a anomaly prediction model for hosts based on parameters like CPU consumption, memory usage, input/output data rate, buffer queue length. Totally 20 features are collected in every 10 seconds to create a measurement stream. Triple state stream classifier was designed to classify each measurement sample to normal, alert or anomaly state. Prediction model triggers alarm whenever an alert state precedes anomaly state. An unsupervised non-intrusive domain independent framework for predicting latent faults in host is proposed in [10]. The main idea behind this method is to compare the machines performing the same task at same time. A machine deviating from normal behavior is tagged as anomalous. The performance counter values for machine m over period t is denoted as $x(m,t)$. To tag the machine m as suspicious, its value must be measured against other machines x values that run the same task at same time. To predict latent fault three tests sign test, turkey test and LOF test is proposed. Sign test is based on measuring $v(m)$ and termed as genuine when $v(m)$ is close to empirical mean of all machines.

$$v(m) = \frac{1}{T(M-1)} \sum_{t \in T} \sum_{m' \in M} \frac{x(m,t) - x(m',t)}{||x(m,t) - x(m',t)||}$$

Turkey test is based on calculation of Turkish depth function which gives high scores to points which are at center positions in the sample and low scores to points which are in the perimeter.LOF test is based on Local Outlier Factor outlier detection algorithm. The LOF function tries to find outliers by only looking at local neighborhoods. The assumption is that on different areas, the density of the sample might be different and this does not imply that points in less dense areas are outliers. The score of the LOF function is not calibrated. The greater the LOF score is, the more suspicious the point is. Based on this the suspicious hosts are identified. This approach is applicable only for speculative executive environment where resource consumption is not importance and only reliability is a concern. But in the proposed solution for fault tolerance resource consumption is also important. In [11] critical events identifying faults are determined and ranked based on their impact on fault diagnosis. The sequence of events in their order of occurrence that leads to fault is modeled in terms of a detection graph. Edge ranking algorithm is used the select the events critical to a particular fault and an event fault pattern is created. Critical events are selected based on three factors of affinity, weight and time decay of event. Affinity describes the involvement of event. Weight describes the discriminative power of the event and time decay how recent is the event to fault. The rank is calculated as

$$Rank(e) = \sum U_e \times W_e \times D_e$$

A sample event detection graph for three faults f1, f2, and f3 is given below in Fig. 1.

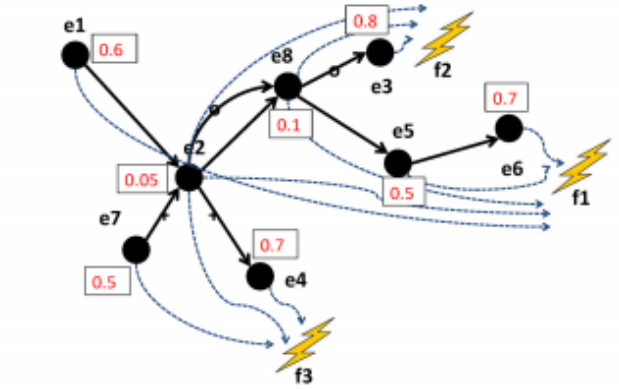

Fig. 1. Fault Graph [11].

The approach is adaptive to learn new fault patterns. But in typical datacenter, the number of events is very large and modeling the relationship is a cumbersome process. Redundant and irrelevant event removal can be done to reduce the modeling complexity. An automatic classification and identification of crisis in datacenter using efficient representation of datacenter state called finger print is proposed in [12].The finger print is learnt using statistical selection and summarization of the hundreds of performance metrics in datacenter. Performance metric in a particular epoch across all the application servers is summarized by computing the quantiles of the measured values (such as the median of CPU utilization on all servers). Based on the past

observation, the values are characterized as hot, cold or normal. From it a summary vector containing one element per quantile per tracked metric, indicating whether the value of that quantile is cold, normal, or hot during that epoch is created. The summary vector is epoch finger print. Consecutive epoch finger prints are summarized as crisis fingerprint which characterizes the state of the datacenter. By matching to known crisis fingerprint, the state of datacenter can be predicted. The take way from this approach is that finger print can be done for host state and based on it task migration policy can be designed. An approach for fault localization in host by analyzing the dependencies among anomalous key performance indicators is proposed in [13]. A model is trained to capture the normal system behavior and this model is used to pinpoint the faulty resources that are likely to be responsible for possible failure. System execution under normal conditions in the form of relations among Key Performance Indicators (monitored KPIs), which are metrics collected on a regular basis from target resources of the system at different abstraction levels is used to train the model. A KPI base line model is created for normal executions. The deviation from base line model is measured frequently for tasks in hosts and task deviating greater than threshold is predicted to fail.

Unsupervised Behavior Learning (UBL) system for predicting failures in cloud is proposed in [14].UBL leverages an unsupervised learning method called the Self Organizing Map (SOM) which is able to capture complex system behavior with less expensive computational needs. System level behavior captured at OS and cloud layers are used to train SVM to classify normal behaviors. Deviation from normal system behaviors are predicted as anomalies. The continuous state of system from pre-failures to failure is modeled using SOM.UBL can raise an advanced alarm when the system leaves the normal state but has not yet entered the failure state. But for large datacenter, the application of UBL is more resource intensive. In [15] an integrated online anomaly prediction and virtualization-based prevention technique is proposed for virtualized cloud systems. Statistical learning is applied over system level metrics (cpu, memory, io statistics) for advance anomaly prediction and coarse grained anomaly cause inference in terms of identification of faulty VMs. An anomaly prediction model is created for each.

VM applies Tree Augmented Naïve Bayesian Network. It is an extension of the naive Bayesian model with consideration of dependencies among attributes into account. It can classify a system state into normal or abnormal and give a ranked list of metrics that are mostly related to the anomaly. The problem with this method is that momentary surge in VM load is also detected as fault without consideration for next state of VM.

III. Proposed Solution

Different from previous solutions, the proposed solution formulates a deep learning based failure prediction model. The model is able to predict the failure in advance such that there is enough time for migration and mitigate the failures. With cognitive neural learning model based on features extracted across all layers, the fault prediction capability is increased in the proposed solution. Each host is ranked in terms of FP-Score (values 1 to 10). Highest value denotes host is approaching failure sooner than lower value hosts. Features extracted across three categories care used for deep learning. The features collected at various layers are given in Table 1. The features are sampled at every window interval of time. Every t samples is aggregated in form of a 2-D matrix.

$$M = \begin{pmatrix} f11 & \cdots & f1N \\ \vdots & \ddots & \vdots \\ ft1 & \cdots & ftN \end{pmatrix}$$

TABLE. I. Features

Category	Name	Description
Cloud Layer	Average VM Memory utilization	The average memory utilization of VMs running in the host
	Average VM Disk Utilization	The average disk utilization of VMs running in the host
	Average VM Bandwidth Utilization	The average bandwidth utilization of VMs running in the host
	Normalized Cloudlet arrival rate	The task arrival rate at host
	CloudletSatisfaction Ratio	The rate at which incoming tasks are translated to cloudlet on a VM in the host
OS Layer	Host CPU Utilization	The utilization percentage of CPU in the host
	Host Memory Utilization	The utilization percentage of memory in the host
	Host Disk Utilization	The utilization percentage of Disk in the host
	Host Network Utilization	The utilization percentage of network bandwidth in the host
	Host temperature	The temperature of the host
	Host Resource Depletion Ratio Vector	The resource depletion ratio in terms of CPU and memory and disk.
Error Features	Cloudlet Error Histogram	The distribution count of cloudlet errors in log over various error levels.
	VM Error Histogram	The distribution count of VM errors in log over various error levels.
	OS Error Histogram	The distribution count of OS errors in log over various error levels.

Where N features collected over t sample intervals.

The matrix is labeled with a Failure Probability Score (FP).

A Loop controlled Cognitive Neural Network is trained to obtain the label for the input matrix M. Deep learning classifier is shown in Fig. 2. The architecture of the model is shown in Fig. 3. The cognitive network is a three layered feed forward radial bias function network. The transformation applied in each layer is given below in Table II.

The prediction output of the neural network classifier with K hidden layer neuron is given as

$$y_j^i = \propto_{j0} + \sum_{k=1}^{K} \propto_{jk} \Phi_k(X^i), j = 1,2 \dots n$$

Where the details of parameters are in Table 3.

The parameter for fine tuning the performance of the cognitive neural network is done by Loop control.

A CNN classifier is trained to predict the output label (FP-Score) for any input sample of Cloud, OS, and Error features expressed in form of 2-D matrix. With this classifier, the FP-Score of all hosts in data center is calculated periodically. The VM allocation and migration policies in the data center are modified based on the FP-Score. FP-VM allocation and FP-VM migration are the two policies proposed in this work integrating FP score of host with allocation and migration decisions.

FP-VM allocation does not create all VMs at once. Instead VMs are created in a progressive manner. Task are allocated to VM with best fit and in case a match is not found, task are kept pending in queue. The hosts in the data center are sorted in ascending order of FP-Score and the hosts are categorized to three levels as in Fig. 4.

TABLE. II. TRANSFORMATIONS FUNCTIONS

Input Layer	No transformation
Hidden Layer	Gaussian Activation function
Output Layer	Linear Activation function

TABLE. III. NEURAL PARAMETERS

N	number of output layer neurons
$\propto_{j0}$	basis to the j output neuron
$\propto_{jk}$	Weight connecting j to k th neuron
$\Phi_k(X^i)$	The response of k th hidden neuron to input X^i modeled as $\Phi_k(X^i) = exp\left(-\frac{\|\|x^i - \mu_k\|\|^2}{{\sigma_k}^2}\right)$ μ_k, σ_k represent the center and width of k[th] hidden neuron

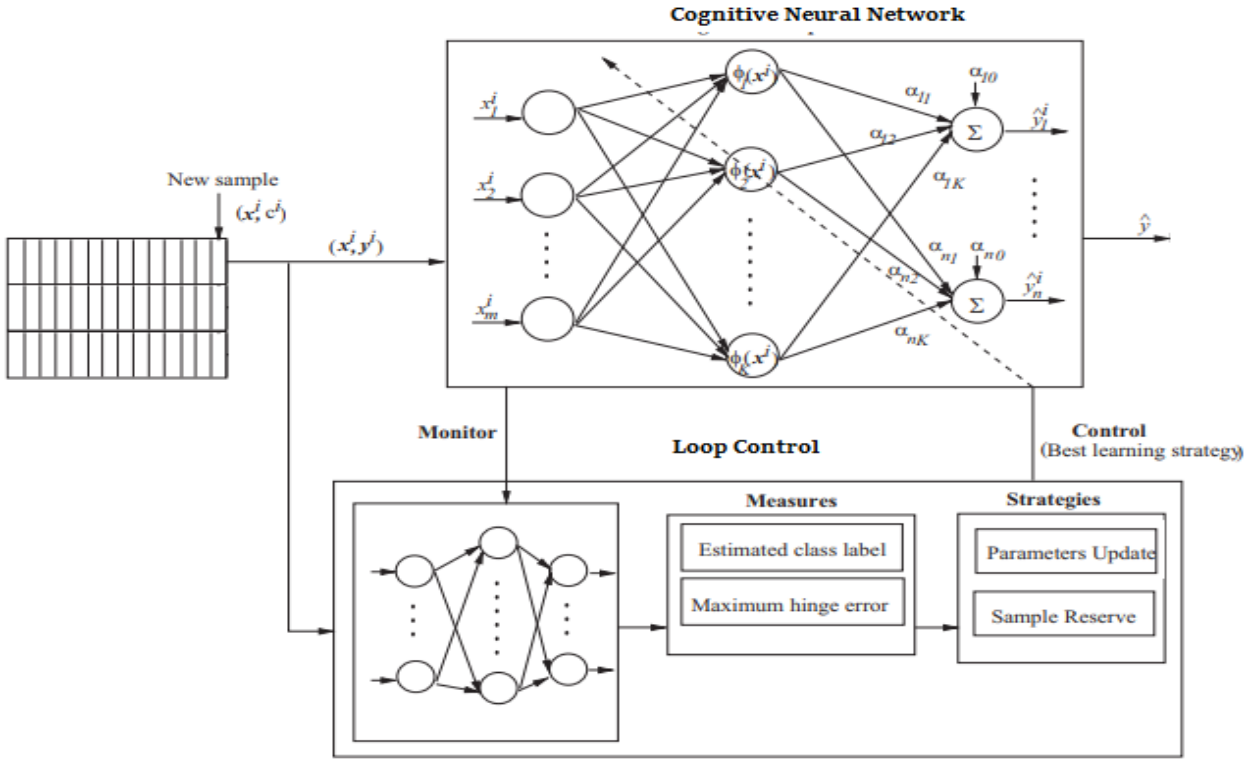

Fig. 2. Deep Learning Classifier.

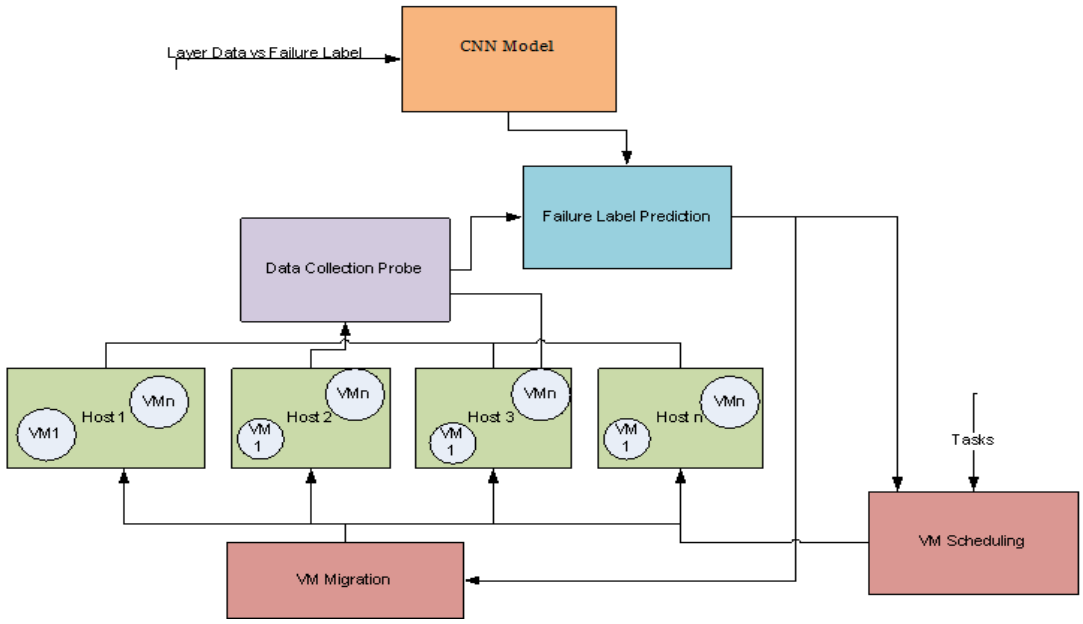

Fig. 3. Solution Architecture.

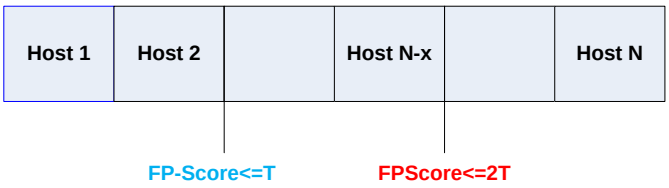

Fig. 4. FP-Score Categorization.

VM's are allocated to host who FP-Score is less than the threshold (T) in the order of first fit. In case VM cannot be allocated to host with FP-Score less than T, the host with score less than 2T is considered with replication in at least 2 hosts. The hosts with FP score greater than T and less than 2T has large chances of failing, so the VM is allocated in redundant manner in this case to handle the failure proactively. The VM are never allocated to host with FP-Score greater than 2T as they has highest changes of failing soon and are in process of migrating their load to reduce their FP-Score.

FP-VM migration policy migrates the VM in host based on their FP-Score. For the host whose FP-Score is greater than 2T, VMs in it are migrated in a progressive manner to other hosts whose score is less than 2T and checked if the FP-Score is reducing , VMs in host are migrated till the FP-Score reaches less than T. During migration, if a favorable host less than T cannot be found, VM is migrated to hosts <=2T and the VM is tried for replication too. If the replication fails, then VM is check pointed. Even in case of VM cannot be migrated due to hosts unavailable, the VM is check pointed. Check pointing is done as the last decision, so that in case of host fails, the VM can be restored till the last check point. Check pointing is a resource consuming task and it is done only as the last resort. For a check pointed VM, if favorable host less than T becomes available later, the check pointing process is immediately abandoned. The pseudo code for VM allocation and migration in the proposed solution is given in Fig. 5 and 6.

Algorithm : FP-VM Allocate
Input: VM, T

Sort Hosts based on FP-Score in ascending order
Split Hosts to three ranges ≤T, ≤2T,>2T
R=Allocate VM to Hosts ≤T
If R is success return;
R=Allocate VM to 2 different Host in T<FPScore≥2T
If R is success return;
Skip the VM allocation till some time

Fig. 5. VM Allocation.

Algorithm : FP-VM Migrate
Input: VM, T

Sort Hosts based on FP-Score in ascending order
Split Hosts to three ranges ≤T, ≤2T,>2T
R=Migrate VM in host >2T to <T
If R is success return;
R=Migrate VM to Host in T<FPScore≥2T
If R is success migrate VM to T<FPScore≥2T
R2== Replicate VM in T<FPScore≥2T
If R2== fail , checkpoint VM
If R is fail , check point VM

Fig. 6. VM Migration.

IV. Results

Performance evaluation of the proposed solution is done on Cloud-Sim [16] test bed for various simulation configurations. The cloudlet, VM, Host configurations for simulation test bed is imported from Google cluster trace. The performance is measured in terms of

1) Response time
2) Deadline Miss Ratio
3) VMs Allocated

The performance of the proposed solution is compared with QFEC solution proposed in [1].

The response time for completion of task is measured for varied number of tasks and the result is below.

From the results in Fig. 7, it can be seen that the response time is comparatively less in the proposed FP-CNN model. It is due to reduction in the amount of replication and wastage of resources consumed by replicated tasks. Deadline miss ratio is measured for varied number of tasks and the result is below.

From the results in Fig. 8, it can be seen in the proposed system, the deadline miss ratio is less than 20% as against 55% in case of QFEC method. The number of VM's allocated for replication is measured for varying number of tasks and the result is below.

From the results in Fig. 9, The number of VM allocated for replication is less in the proposed method compared to QFEC. The number of instances of intolerance to failure is measured for varied number of tasks and given below.

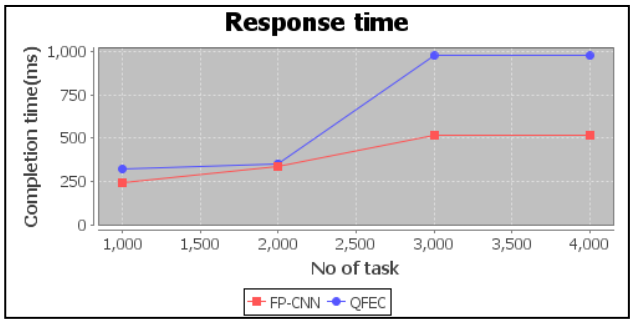

Fig. 7. Response Time.

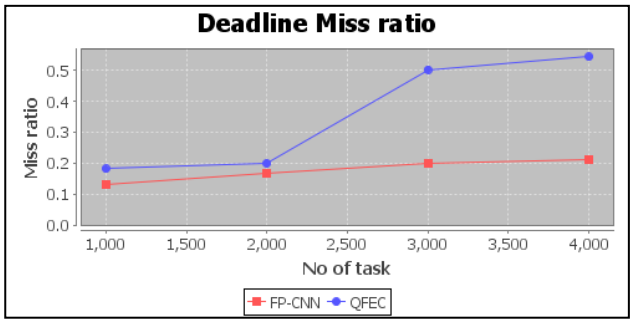

Fig. 8. Deadline Miss Ratio.

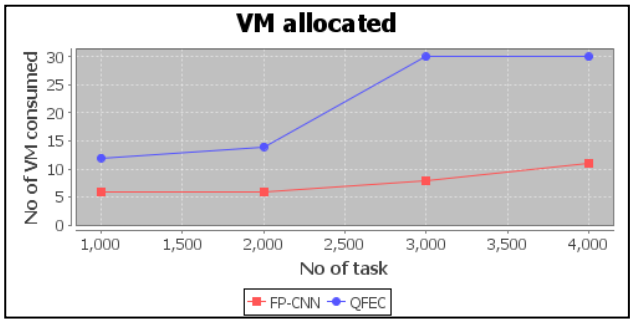

Fig. 9. VM Allocated.

V. Conclusion

A Deep learning based proactive fault tolerance solution called FP-CNN is proposed. Each host in datacenter is assigned to a FP-Score which indicates the failure probability of host. The FP-Score allocation is based on features extracted from various layers and deep learning on those features. Two novel solutions for VM allocation and VM migration is proposed integrating the FP-Score value of hosts. Performance was evaluated in Cloud-Sim platform against Google cluster dataset and from the results, the effectiveness of the proposed solution in providing a proactive fault tolerance is proved.

References

[1] Guoqi Xie,Gang Zeng "Quantitative Fault-Tolerance for Reliable Workflows on Heterogeneous IaaS Cloud",IEEE TRANSACTIONS ON CLOUD COMPUTING 2017.

[2] Jialei Liu,Shangguang Wang,Ao Zhou "Using Proactive Fault-Tolerance Approach to Enhance Cloud Service Reliability",IEEE Transactions on Cloud Computing May 2016.

[3] Z. Zheng, T. Zhou, M. Lyu, and I. King, "Component ranking for faulttolerant cloud applications," IEEE Transactions on Services Computing, vol.5, no.4, pp. 540-550, 2012.

[4] Fan G, Yu H, Chen L, Liu D. Editors. Model based byzantine fault detection technique for cloud computing. 2012 IEEE Asia-Pacific Services Computing Conference (APSCC);Guilin. 2012 Dec 6-8. p. 249–56.

[5] E. AbdElfattah, M. Elkawkagy, and A. El-Sisi, "A reactive fault tolerance approach for cloud computing," in 2017 13th International Computer Engineering Conference (ICENCO), 2017, pp. 190– 194.

[6] M. K. Gokhroo, M. C. Govil, and E. S. Pilli,"Detecting and mitigating faults in cloud computing environment," 3rd IEEE Int. Conf. , 2017.

[7] D. Yuan, D. Park, P. Huang, Y. Liu, M. Lee, Y. Zhou, and S. Savage, "Be Conservative: Enhancing Failure Diagnosis with Proactive Logging," in USENIX OSDI, 2012, pp. 293–306.

[8] S. Kadirvel, J. Ho, and J.AB Fortes, Fault management in map-reduce through early detection of anomalous nodes, in10th International Conference on Autonomic Computing (ICAC 13), California, USA, Jun.2013, pp.235-245.

[9] Y. Tan, X. Gu, and H. Wang,Adaptive system anomaly prediction for large-scale hosting infrastructures,in 29th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, Zurich, Switzerland, Jul. 2010, ACM, pp. 173-182.

[10] M. Gabel, R. G. Bachrach, N. Bjorner, and A. Schuster, Latent fault detection in cloud services,Microsoft Research, Tech. Rep. MSR-TR2011-83, 2011.

[11] Q. Zhu, T. Tung, and Q. Xie,Automatic fault diagnosis in cloud infrastructure,in IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom), Bristol, UK, Dec.2013, vol. 1, pp. 467-474.

[12] P. Bodik, M. Goldszmidt, A. Fox, D. Woodard, and H. Andersen, "Fingerprinting the datacenter: automated classification of performance crises," in Proceedings of the 5th European conference on Computer systems (EuroSys2010), Paris, France, 2010, pp. 111–124.

[13] Leonardo Mariani,Cristina Monni,Mauro Pezzé "Localizing Faults in Cloud Systems",IEEE 11th International Conference on Software Testing, Verification and Validation 2018.

[14] Daniel Joseph Dean, Hiep Nguyen, and Xiaohui Gu. "UBL: Unsupervised Behavior Learning for Predicting Performance Anomalies in Virtualized Cloud Systems". In: Proceedings of the International Conference on Autonomic Computing. ICAC '12. ACM, 2012, pp. 191–200.

[15] Z. Li and L. Itti, "Saliency and gist features for target detection in satellite images," IEEE Trans. Image Process., vol. 20, no. 7, pp. 2017–2029, Jul. 2011.

[16] CloudSim:http://www.cloudbus.org/cloudsim/.