Task Failure Prediction in Cloud Data Centers Using Deep Learning

Dr. Rafath Samrin[*1]
Professor[1]**,** Dept of Computer Science & Engineering[1]
Abdul Kareem Abid[2]**,Majed Mohd Taher**[3]**, Syed Abdul Rahman Ahmed**[4]
UG scholors[2,3,4],Dept of Computer Science & Engineering[2,3,4]
Deccan College of Engineering and Technology.Hyderabad, Telangana, India[1,2,3,4].
E-mail: rafathsamrin@deccancollege.ac.in[1], abdulkareemabid25@gmail.com[2] majedtaher786@gmail.com[3] ,syedabr9000@gmail.com[4].

ABSTRACT: A large-scale cloud data center must reduce the likelihood of failure while simultaneously providing high service dependability and availability. However, modern large-scale cloud data centers continue to experience high failure rates due to software and hardware flaws that frequently cause task and job failures. Such failures may have a significant negative impact on the dependability of cloud services and necessitate significant resource restoration. Task or work failures must be accurately predicted prior to their occurrence in order to reduce unexpected waste. Evaluation of previous system message logs and recognition of the relationship between the data and failures are two methods that have been published that use machine learning and deep learning to predict task or job failures. We present a cloud task and job failure prediction strategy based on multi-layer Bidirectional Long Short Term Memory (Bi-LSTM) to improve the accuracy of machine learning and deep learning-based failure prediction systems. The Bi-LSTM prediction algorithm determines the success of jobs and projects. With 93% and 87% accuracy for task failures, respectively, our method outperforms current cutting-edge prediction algorithms in trace-driven experiments.

Keywords *– deep learning and data centers in the cloud*

1. INTRODUCTION

Due to their on-demand services, resource savings, and high reliability, cloud computing services are gaining popularity. A wide range of user applications, including jobs, are supported by cloud data center processors, memory units, disc drives, networking devices, and other types of sensors. Users can use the cloud to fulfill their requests for data storage or the execution of applications. Physical machines (PMs) make up each cloud data center, and each PM can host a group of virtual machines (VMs). The tasks that users assign are handled by each VM. Many of the hundreds of thousands of computers in a massive cloud data center run hundreds of applications and receive daily work requests from people all over the world. A cloud data center may occasionally encounter a number of errors (such as disc, software, or hardware failures) due to the large variety of workloads and heterogeneity. Consider issues with software: In January of 2015, the Bing search engine owned by Microsoft and Yahoo Inc. went down for twenty minutes. In order to get them back online, it cost about $9000 per minute. Cloud service outages are frequently caused by hardware failure, particularly disc loss, according to previous research. Application execution failures will occur as a result of these various issues. As a result, the efficiency with which the failure is recovered and the program continues to function may be enhanced by accurate application failure prediction in advance.

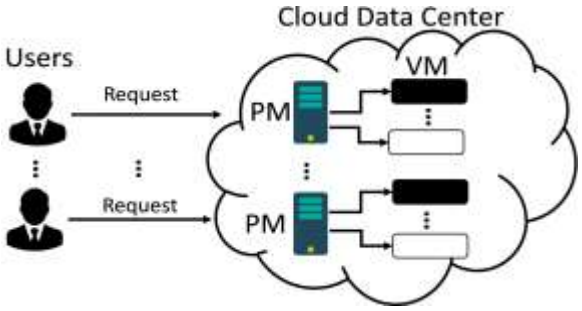

Fig.1: Example figure

A job is made up of one or more tasks, each of which needs its own set of resources. A job is terminated when one of its responsibilities fails. Previous research [3, 7-13] used the Hidden Semi-Markov Model (HSMM) and the Support Vector Machine (SVM) to anticipate task and job failures in cloud data centers. They take into account the utilization of CPU and memory, unmapped page cache, mean disc I/O time, disc utilization, and task or job failure as inputs. SVM and HSMM, on the other hand, are predicated on the assumption that each input is constant and distinct from the others, which is not the case in cloud data centers. They are therefore unable to manage sequence data or high-dimensional data, where distinct properties or time points of the data may be interdependent. In cloud server farms, the info attributes and uproarious information are variable in view of past occasions. HSMM and SVM cannot therefore anticipate cloud data center failures.

2. LITERATURE REVIEW

Diehard: reliable scheduling to survive correlated failures in cloud data centers:

A single failure in a large data centre might bring about associated disappointments of a few actual PCs and the projects running on them simultaneously. Such connected disappointments may considerably risk a help's or alternately movement's unwavering quality. This study models the effect of irregular and related disappointments on work constancy in a server farm. We center around associated disappointments brought about by blackouts or organization part disappointments, as well as projects that lead a huge number of a similar activity. We give a measurable dependability model and a guess method for assessing work unwavering quality in the circumstance of related disappointments. We likewise check out at the subject of booking work with unwavering quality imperatives. The booking issue is organized as an advancement issue, with the goal of acquiring the expressed dependability with the least additional errands. We give a booking approach that approximates the base measure of occupations expected, as well as a situation to accomplish the ideal work steadfastness. We break down the effectiveness of our methodology utilizing an insightful procedure and by recreating a bunch with various disappointment reasons and unwavering quality levels. The outcomes demonstrate the way that the calculation can accurately anticipate the most modest number of additional undertakings expected to guarantee work constancy.

Failure prediction of data centers using time series and fault tree analysis

This study proposes a methodology for anticipating server farm disappointments on the web. Due to the large number of servers and components, a server farm typically has a high rate of failure. At these kinds of foundations, arduous tasks and projects that last a long time are also common. The framework's presentation depends on the machines' accessibility, which might be effortlessly endangered on the off chance that disappointment isn't dealt with smoothly. The principal reason for this exposition is to make a dependable equipment disappointment expectation model. The precision of expectation might increment in general framework execution. In this review, we

utilize two methodologies: Issue Tree Examination and Auto Regressive Moving Average (ARMA). The tests were then conducted on a virtual group created using Simi's foundation. The outcomes show that the figure precision is 97%. As a result, we accept our plan is possible and that it very well may be extended for later use in server farms.

Partial-parallel-repair (ppr): a distributed technique for repairing erasure coded storage

With the multiplication of information in applications surrounding us, deletion coded capacity has arisen as a convincing option in contrast to replication in light of the fact that, despite the fact that having a far lower capacity above, it conveys more noteworthy information misfortune opposition. The Reed-Solomon code is the most generally utilized cancellation code since versatile to the coding settings decide the reachable unflinching quality and gives the most significant level of constancy for a given store above. In any case, in view of organization restricts, the recreation time for unavailable information turns out to be unsuitably lengthy. A few proposed arrangements either require extra capacity or cutoff how much coding boundaries that might be utilized. In this paper, we offer Partial Parallel Repair (PPR), an original conveyed remaking procedure that isolates the reproduction cycle into discrete halfway tasks and timetables them on countless hubs previously engaged with the information recreation. Then, at that point, to diminish network clog, a circulated convention continuously consolidates these incomplete disclosures to reestablish the missing information blocks. In a nutshell, rather than the k time required by a (k, m) Reed-Solomon code, our method could complete the organization transmission in (log2(k + 1) time. According to our findings, PPR essentially reduces disabled read and fix times. Furthermore, we need not bother with any additional stockpiling above because our solution works with existing deletion codes. To save essentially additional time during the game, we show this by superimposing PPR on top of two past moves close, the Local Revamping Code and the Turned Reed-Solomon code.

Approaches for resilience against cascading failures in cloud datacenters

In a cutting edge cloud datacenter, a fountain disappointment will bring about many Service Level Objective (SLO) breaks. At the point when a bunch of physical machines (PMs) in one disappointment space come up short, their jobs are moved to PMs in another disappointment space. In any case, because of the cloud's affinity to oversubscribe assets, the new space getting expanded jobs might become overpowered, bringing about area disappointments and resulting trouble shift to different spaces. This technique is gone on until the outpouring comes up short. Be that as it may, not many past systems have been displayed to successfully control flowing disappointments. We propose a Cascading Failure Resilience System (CFRS) that consolidates three ways to deal with address this issue: over-burden evasion, over-burden versatility, and over-burden strength. Dynamic Oversubscription Ratio Adjustment (OAVR), VM Reassignment (VMset), and VM Reinforcement Set Arrangement (VMset) are choices . With regards to area disappointments, bombed PMs, and SLO infringement, the discoveries of the follow driven recreation preliminaries show that CFRS outflanks other examination philosophies.

Proactive incast congestion control in a datacenter serving web applications

Network dormancy is turning out to be progressively essential to the client experience because of the quick development of web applications in datacenters. At the point when an enormous number of solicitations show up at the front-end server simultaneously, incast stop up

decisively increments network dormancy. Previous incast issue arrangements were inefficient at preventing incast blockage because they typically dealt with information transmission directly between information servers and front-end servers. In this work, we present a Proactive Incast Congestion Control (PICC) technique to increment reasonability fundamentally. PICC confines the quantity of data servers that can be related with the front-end server all the while to decrease incast blockage through data circumstance on the grounds that each affiliation has an exchange speed limit. Particularly, the front-end server consolidates well-known pieces of information, also known as items that are frequently mentioned, into as few information servers as is posible while avoiding overloading the system. It likewise redistributes information things that are planned to be questioned in a similar server simultaneously or successively. Thus, PICC limits the quantity of information servers associated with the front-end server simultaneously (to forestall incast clog) as well as the quantity of association establishments (which lessens the organization dormancy). Since the chose information servers frequently have huge information move lines, PICC contains a lining defer decrease calculation that focuses on information things with more modest sizes and longer holding up periods. The exploratory outcomes from reproduction and a genuine bunch utilizing a benchmark show that PICC beats past incast blockage issue arrangements.

3. METHODOLOGY

Notwithstanding, there are a couple of issues in the LSTM-based forecast frameworks. To start, the methods simply dissect central processor use, memory use, reserve memory utilization, mean circle I/O time, and plate use as info attributes. More information highlights might additionally increment forecast exactness. Second, the LSTM-based expectation model utilized multi-facet development in addition to single-layer LSTM development, which is incapable of managing a large number of information qualities. Thirdly, there is a strong connection between input factors in the cloud server farm, such as how much memory is used and how much processor is used. The LSTM-based expectation model typically assigns higher loads to information closer to the time for a given expectation time and lower loads to information further away from the time. This is because of the way that data further away from the time an affects the measure. In any case, such circumstances cannot actually demonstrate the degree of impact because other factors may in fact have a greater impact on the disappointment (such as disappointments in long-term occupations). Assuming that the loads of information things are determined utilizing the genuine information follow, execution might increment. To increment figure precision, another expectation model for disappointment expectation in the cloud server farm should be laid out.

Disadvantages:

1. Existing enormous scope cloud server farms, then again, keep on having significant failure rates because of a scope of factors, including equipment and programming defects, which frequently bring about undertaking and occupation failures.
2. Such failures might fundamentally affect the dependability of cloud administrations while likewise requiring a lot of assets to reestablish the help.

To settle these issues, we depict in this paper a failure forecast model called Bi-LSTM, which depends on a multi-facet Bidirectional LSTM. To start, Bi-LSTM consolidates extra info highlights than past frameworks, like work prioritization, task resubmissions, and planning

delays. Second, the multi-facet construction of Bi-LSTM empowers it to deal with a large number of info qualities with better precision. A multi-facet design might decrease the quantity of boundaries in computing capabilities while keeping similar number of neurons, bringing about a more limited estimation time. Third, instead of just giving higher loads to information things nearer to the given time for expectation than to information things farther away from the time, Bi-LSTM might produce information thing loads relying upon their actual impact on the failure. We embrace a follow driven failure expectation concentrate on utilizing Google group follow and contrast Bi-execution LSTM's with that of other state of the art forecast calculations.

Advantages:

1. Our technique accurately detects task and job failures.
2. We also observe that the time cost overhead for Bi-LSTM is similar to that of RNN and LSTM, meaning that Bi-LSTM may provide enhanced prediction performance at no extra time cost.

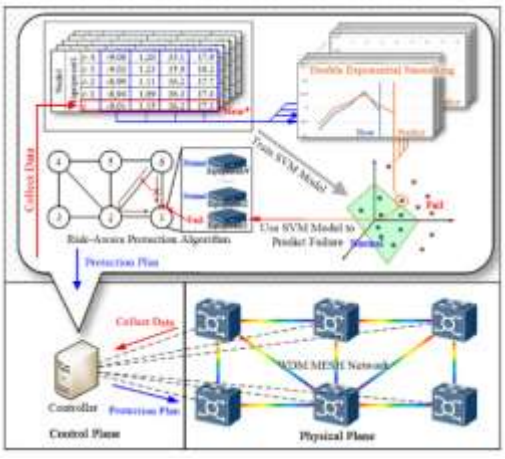

Fig.2: System architecture

MODULES:

We designed the modules indicated below to carry out the aforementioned project.

- Data exploration: we will use this module to enter data into the system.
- This module will be used to read data for processing.
- Data splitting into train and test: Using this module, data will be split into train and test.
- Model generation methods include Random Forest, Decision Tree, KMM, Support Vector Machine, Voting Classifier, CNN, CNN+LSTM, LSTM, BiLSTM, RNN, and CNN with KFoldVaildation.
- User registration and login: Using this module will need you to register and login.
- User input: Using this module will offer prediction input.
- The ultimate predicted value will be displayed as a prediction.

4. IMPLEMENTATION

ALGORITHMS:

Random Forest: A Directed machine learning(Ml) Calculation is a sort of ML calculation that is in many cases utilized in order and relapse applications. It constructs decision trees from a few examples, utilizing the greater part vote in favor of characterization and the normal for relapse.

Decision Tree: To choose whether or not to part a hub into at least two sub-hubs, decision trees utilize different systems. The homogeneity of the sub-hubs is improved by the rise of sub-hubs. The hub becomes more tidy as it approaches the objective variable at the end of the day.

KNN: KNN is a basic calculation that keeps generally past occasions and characterizes new information or cases utilizing a closeness measure. It is frequently used to order an information point in view of its neighbors' characterizations.

SVM: SVM is a managed ML calculation that might be utilized for both characterization and relapse. However we term them relapse

concerns, they are better grouped. Finding a hyperplane in an N-layered space that clearly orders the information focuses is the goal of the SVM method.

Voting classifier: A voting classifier is an ML assessor that trains and predicts relying upon the results of many base models or assessors. For every assessor yield, amassing measures may be matched democratic choices.

CNN: A CNN is a deep learning network engineering that is typically used in applications that handle pixel information and recognize images. Deep learning utilizes a few kinds of brain organizations, however CNNs are the favored organization engineering for endlessly perceiving objects.

LSTM: A kind of fake brain network called long short-term memory (LSTM) is utilized in electronic thinking and deep learning. LSTM highlights criticism associations, dissimilar to standard feedforward brain organizations. This sort of recurrent neural network (RNN) can break down single pieces of information (like pictures), yet in addition entire information groupings (like discourse or video).

BiLSTM: BiLSTM is a truncation for Bidirectional Long Short-Term Memory (BiLSTM) In time series handling, LSTM frequently overlooks future data. BiLSTM utilizes LSTM to handle series information in both forward and in reverse directions, associating the two secret layers.

RNN: A recurrent neural network (RNN) is a sort of counterfeit brain network in which hub associations might make a cycle, permitting one hub's result to influence ensuing contribution to that equivalent hub. This permits it to show worldly powerful way of behaving. Utilizing their interior state, RNNs, which are worked from feedforward brain organizations, can oversee variable length groupings of sources of info (memory). As an outcome, they might perform undertakings like unsegmented, connected penmanship acknowledgment or discourse acknowledgment. On a basic level, repetitive brain networks are Turing finished and can perform erratic calculations to deal with inconsistent information successions.

5. EXPERIMENTAL RESULTS

Fig.3: Home screen

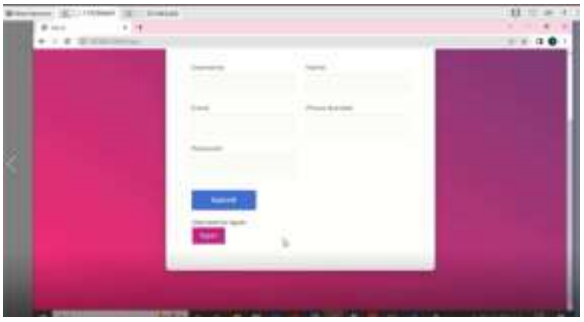
Fig.4: User registration

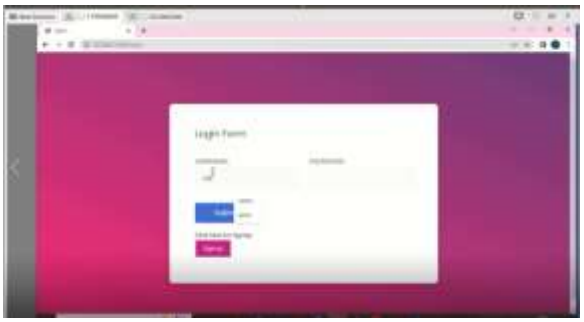
Fig.5: user login


Fig.6: Main screen

Fig.7: User input

Fig.8: Prediction result

6. CONCLUSION

In cloud server farms, high help steadfastness and accessibility are critical for application QoS. We introduced a multi-facet Bidirectional LSTM disappointment forecast model in this review (called Bi-LSTM). Utilizing Google bunch follow, Bi-LSTM can all the more accurately expect the end situations with assignments and occupations than past strategies. In our procedure, we initial info the information into forward and in reverse states to change the heaviness of both nearer and far off input highlights. We then discover that extra information properties are vital for accomplishing high expectation precision. Second, in the tests, we look into Bi-LSTM with an assortment of assessment techniques, for example, quantifiable, machine learning, and deep learning-based strategies, and we assess execution utilizing three measurements: precision and F1 score, recipient working brand name, and time cost previously. As indicated by the information, we anticipated task disappointment with 93% exactness and occupation disappointment with 87% precision. We likewise got a 86% F1 in work disappointment expectation and a 92% F1 in task disappointment forecast. The low FPR of our forecast method Bi-LSTM shows that proactive disappointment the executives in light of expectation results is turning out to be more successful. Additionally, we observe that Bi-LSTM's time cost is comparable to that of RNN and LSTM, suggesting that Bi-LSTM may provide enhanced expectation execution at no additional cost.

REFERENCES

[1]“https://techcrunch.com/2015/01/02/following-bing-coms-brief-outagesearch-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. Okanovic, 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.