Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 05, May-2021

Prediction of Cloud Server Job Failures using Machine Learning based KNN Classification and LSTM Modelling Methods

Bhushan Golani
Dept. of Mechanical Engineering
Indian Institute of Technology Kharagpur
Kharagpur, India

Joydeep Datta
Corporate Venturing & Innovations Group
Tata Communications
Mumbai, India

Gurdeep Singh
Corporate Venturing & Innovations Group
Tata Communications
New Delhi, India

***Abstract*—Cloud computing is the use of a network of remote servers hosted on the internet to store, manage and process data rather than a local server or a personal computer. The cloud computing industry has grown to a great extent in recent times, and it is important to make sure that the delivered service does not deviate from the correct intended service. So, to build a reliable cloud service platform, we need to understand and characterize failures. This work aims to understand the reasons for failure and predict failures that might occur in the future. To accomplish this, we use the Google cluster workload trace. Our analysis reveals that an overwhelming number of resources are used by jobs that eventually fail. For prediction, we use Long Short-Term Memory Network (LSTM) to forecast features on which failure depends. We then predict the termination status using the KNN classification model, which was trained on the existing data.**

Keywords—Cloud Computing, predict failures, forecast features, LSTM, KNN Classification model.

INTRODUCTION

Cloud refers to a network or internet. In other words, we can say that clouds are present at a remote location. Cloud can provide services over public and private networks, i.e., WAN, LAN, or VPN. Applications such as e-mail, web conferencing customer relationship management (CRM) execute on the cloud.

Cloud computing refers to manipulating, configuring, and accessing the hardware and software resources remotely. It offers online data storage, infrastructure, and application. Cloud computing offers platform independence, as the software is not required to be installed locally on the PC. Hence, cloud computing is making our business applications mobile and collaborative. The 2020 cloud computing results show that enterprises continue to embrace multi-cloud and hybrid cloud strategies and are already using more than two public and two private clouds on average. A majority of enterprises expect to increase cloud usage due to COVID-19. The executions of jobs in large-scale data centers are highly complex due to the underlying heterogeneous hardware, intricate dependencies among tasks, diversified job priorities, and sophisticated job scheduling [2]. Due to the complexity, dynamism, and openness of cloud computing, applications running on them are prone to failures that could affect many customers and could even lead to massive financial loss [1]. To reduce these losses and to make cloud computing more efficient, we need to prevent failures. Many prior studies on large-scale system reliability focus on hardware/software failures and their causes. While these are valuable, they do not provide much insight into failures experienced by end-users. Application failures have been analysed in popular systems such as Hadoop and distributed scientific workflows on Amazon EC2. However, these studies are limited to MapReduce or scientific computations and are difficult to extrapolate to generic clouds such as the Google cluster. In comparison, our work focuses on failure characteristics from the jobs' perspective and covers broader classes of jobs than MapReduce or scientific computations [3].

Our goal is to identify the reasons for cloud failure. By doing so, we understand the reasons for failures, and we can make suitable changes in the infrastructure. Also, we try to predict cloud failures in the future by using classification and time series models.

LITERATURE REVIEW

There has been a wide range of research to understand and predict cloud failures. There have been many studies on google Cluster data focusing on failure identification and prediction. Reiss et al. [21] present a comprehensive study of the Google cluster traces in recent times. They provide a broad characterization of the workload that focuses on issues such as cluster utilization, job and task properties affecting the scheduler, resource request versus actual usage, and the challenges in task scheduling. Their study identifies two important characteristics that may warrant new scheduler design. First, they observe some under-utilization of resources

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 05, May-2021

(i.e., significant mismatch between resource requests and actual resource usage). Second, they notice substantial delays in the scheduling of some requests that have constraints. Resource utilization and task implementation vary widely, this was studied by Reiss et al. [22].

Failure analysis of cloud centers was also done by many researchers such as Ford et al. [23] studied the impact of correlated failures on availability of distributed storage systems for Google clusters. Disk failures lead to loss in data and the major reason for the unavailability in the Google cloud storage system is transitory node failures. Analysis of physical machines and virtual machines highlighted their differences and similarities. Birke et al. [24] found that VMs have lower failure rates than PMs, and show a surprising trend that, in contrast to PMs, increasing the computation intensity by VM unit does not increase failure rate. Works which have characterized work load patterns by finding correlations between resource measures and failure are Fadishei et al. [18] which analyzes workload traces from the Grid Workload Archive project [25]. They discover correlations between job failures and performance metrics such as memory usage, CPU utilization, queue utilization, exit hour and migration of jobs, etc.

Failure prediction is an important field of research as it will prevent major losses and will help to improve the system. There have been many methods and models which have been used to achieve this task. The authors in [26] have made a good attempt to analyze the failure data of a large-scale production Cloud environment consisting of over 12,500 servers, which includes a study of failure and repair times and characteristics for both Cloud workloads and servers. Pitakrat et al. [27] proposed a hierarchical online failure prediction approach called Hora. Hora employed a combination of a failure propagation model and software system failure prediction techniques based on Bayesian networks. Zhang et al. [28] designed and implemented a new tool based on Random Forest (RF) called PreFix, for accurately predicting whether there will be a switch failure in the near future. However, machine learning approaches such as SVM have the same shortcomings just like statistical approaches, so they cannot handle the sequence data well in cloud data centers.

I. DATASET DESCRIPTION AND PREPARATION

The dataset we are using is the Google Borg Dataset. A Google cluster is a set of machines packed into physical enclosures (racks) and connected by a high-bandwidth cluster network.

Machines that share a common cluster management system form a cell. A single usage trace describes several days of the workload on a single Borg cell. We have been given data for eight cells. For each cell we have five tables named **Machine events Table, Machine attributes table, Collection Events Table, Instance Events Table, and Instance Usage Table**. For describing the dataset, we have used Google cluster-usage traces v3 [11]. We now describe each table:

Machine Events table has a (1) 'time' column, which has a timestamp in microseconds, (2) 'Machine id' column which is identifier for a machine, (3)'Type' Column which takes three values ADD, REMOVE and UPDATE which means a machine becoming available to the cluster, a machine removed from cluster and a machine has it resources changed is available to the cluster respectively. (4) 'Switch id' column, which is an identifier for the network switch that the machine is connected to. (5) 'Capacity' column, which shows the resources supplied to programs that run on the machine. (6) 'Platform id' column, showing the version of the chipset of the machine and (7) missing data reason column

Machine Attributes table has the following fields:(1) 'Time', (2) 'Machine id' (3) 'Name': It tells the attribute name. (4) 'Value': Machine attribute value (5) 'Deleted': It takes a Boolean value to indicate whether an attribute has been deleted or not.

Collection Events Table has the following fields:(1) 'time' (2) 'type': description of this column can be understood seeing Fig 1. (3) 'collection id': It is a representation of a collection. (4) 'scheduling class': It takes 1,2,3 values, where 1 denotes the least latency-sensitive task and 3 denotes the highest latency-sensitive task.
(5) 'missing type': It shows the type of missing data. (6) 'Collection type': This shows the type of collection, which takes two values 0 if the collection describes a job or 1 if the collection describes an alloc set.
(7) 'priority': A high value of this column means that the priority of the collection is high. (8) 'alloc collection id': identifier of the alloc set that hosts the job. (9) 'user': name of the user, it is obfuscated. (10) 'collection name' (11) 'collection logical name': collection name that reflect its purpose
(12) 'Parent collection id': It is an identifier of the parent collection. It takes a value of 0 if it has no parent collection.
(13) 'start after collection ids': zero or more unique identifiers of any collections that must finish before this one can start (14) 'max per machine': The number of instances which are preferred to be placed on a machine. (15) 'max per switch': The number of instances that are preferred to be placed on machines that have the same network switch. (16) 'vertical scaling': if enabled, the system determines how much CPU and RAM to request. (17) 'scheduler': the scheduler that was tasked with placing the thing.
Values are: ○ SCHEDULER DEFAULT: the default job scheduler ○ SCHEDULER BATCH: a secondary (batch) scheduler

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 05, May-2021

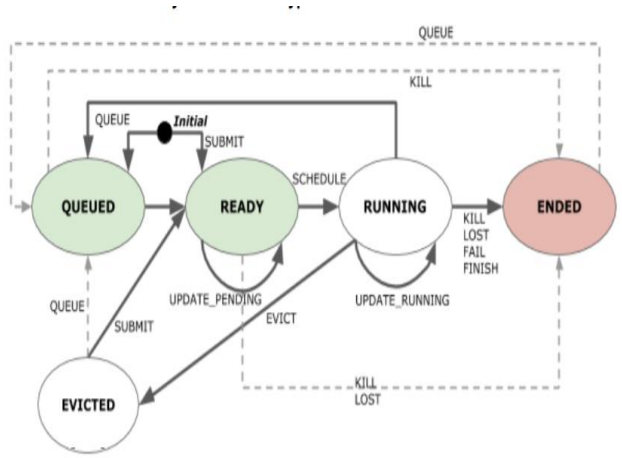


Fig. 1

Instance Events Table: (1) 'Time' (2) 'Type': type of collection in which this instance is present (3) collection id: identifier of the collection this instance is part of. (4) 'scheduling class' (5) 'missing type' (6) 'collection type'. (7) 'priority' (8) 'alloc collection id' (9) instance index: position within the collection. (10) 'machine id': Machine on which instance was planned to run. (11) 'Alloc instance index': index of the alloc in which the instance is running. (12)'resource request': The resources requested for the instance.

Instance Usage Table: (1) 'start time': the starting time of the measurement period. (2) 'end time': the end time of the measurement period. (3) collection id (4) instance index (5) machine id (6) alloc collection id (7) 'alloc instance index' (8) 'collection type' – job or alloc set (9) average usage – the average consumption of resources in the window (10) maximum usage – maximum observed usage during the window. (11) random sampled usage – the observed usage during a randomly-selected 1s sample in the window, CPU data only (a Resources struct). (12) 'assigned memory' – the upper bound of memory for this instance given to the OS kernel by the Borglet (13)
'page cache memory' – the average memory used for the instance's file page cache by the OS kernel (14) 'cycles per instruction' – the CPU cycles used and dividing by the number of instructions executed. (15) 'memory accesses per instruction' – the memory accesses used divided by the number of instructions executed. (16) 'sample rate' – the number of samples taken per second during the window. (17) 'cpu usage distribution' – 11 coarsely-spaced percentiles of the observed CPU usage during different samples: 0%ile (minimum), 10%ile, 20%ile, 30%ile, 40%ile, 50%ile, 60%ile, 70%ile, 80%ile, 90%ile, 100%ile (maximum) (18). tail cpu usage distribution – 9 finely-spaced percentiles of the observed CPU usage during different samples: 91%ile, 92%ile, 93%ile, 94%ile, 95%ile, 96%ile, 97%ile, 98%ile, 99%ile

DATA PREPARATION:

- We focused on two tables which were the Instance events table and Instance Usage table. We merged the Instance events table and Instance Usage Table for hundred machines. The primary keys used were machine id, collection id, and instance index.
- After merging the two tables based on these primary keys, we then use the time column in the instance event table and the start time and end time column in the instance usage table to take only those rows in which the time in the instance events table lies in between start time and end time of the instance usage table.
- After this, we sort the dataset with respect to the time column.
- Two new fields are created, one named percentage memory, which would be equal to average memory/ assign memory, and class average, which takes discrete values such as 1 for percentage memory lying between 0 to 10, 2 for percentage memory lying between 10 to 20 and so on. These two new features were formed to get an insight into the dependence of failure on the ratio of average memory by assigned memory.
- A column called 'failure' is formed, which takes three values, '0' if the instance is successfully finished, '1' is the event fails, '-1' is for other events.
    - '1' is assigned to instances where the instances have their type value as FAIL and LOST.
    - '0' is assigned to instances where the instances have their type value as FINISH.
    - '-1' is assigned to instances which take type value as SUBMIT, QUEUE, ENABLE, SCHEDULE, EVICT, KILL, UPDATE PENDING AND UPDATE RUNNING.
- We use the up-sampling technique to get a balanced dataset for classification.

II. DATA INSIGHTS AND HYPOTHESIS

A. RELATION BETWEEN ASSIGN MEMORY & MAX MEMORY/ AVERAGE MEMORY FOR FAILURE

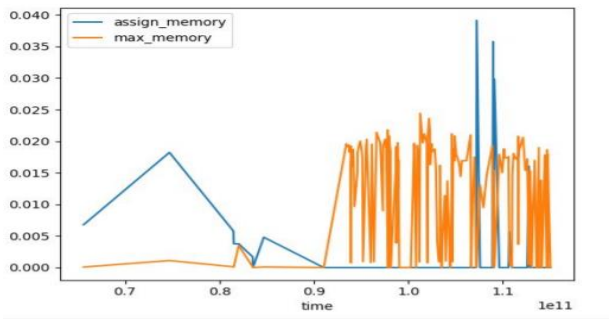


Fig. 2: Assign memory and max memory plotted for failure points

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 05, May-2021

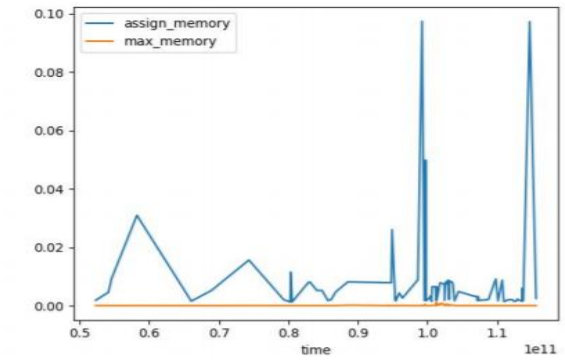


Fig. 3: Assign memory and max memory plotted for non-failure points

We see that one of the reasons for failure is when the value of maximum memory is more than the value of assigned memory as seen from Fig 2.

- When further investigated we see that out of 88,794 points which have failed, there are 2 points for which average memory is more than assigned memory and there are 88792 points for which average memory is less than assigned memory.
- For 36417 points which have failed there are 4976 points for which average memory is more than assigned memory and there are 31441 points for which average memory is less than assigned memory.
- If assigned memory is less than average memory it will surely fail but when it is higher it may fail due to other reasons.

B. *RELATION BETWEEN AVERAGE CPU CONSUMPTION & MAXIMUM MEMORY FOR FAILURE*

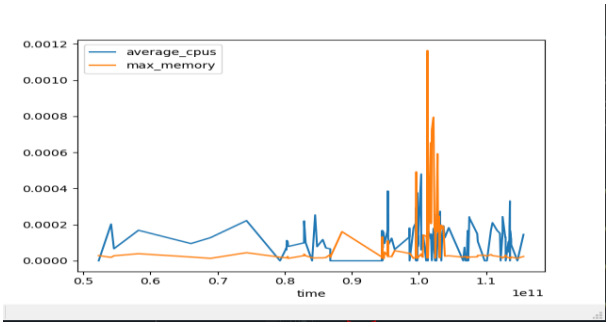


Fig 4: Average CPU & Maximum Memory plotted for failure points.

When the average CPU consumption surpasses the maximum memory, failure will occur. When the CPU consumption is less than the maximum memory failure may happen due to other reasons.

C. *RELATION OF PERCENTAGE AVERAGE WITH FAILURE*

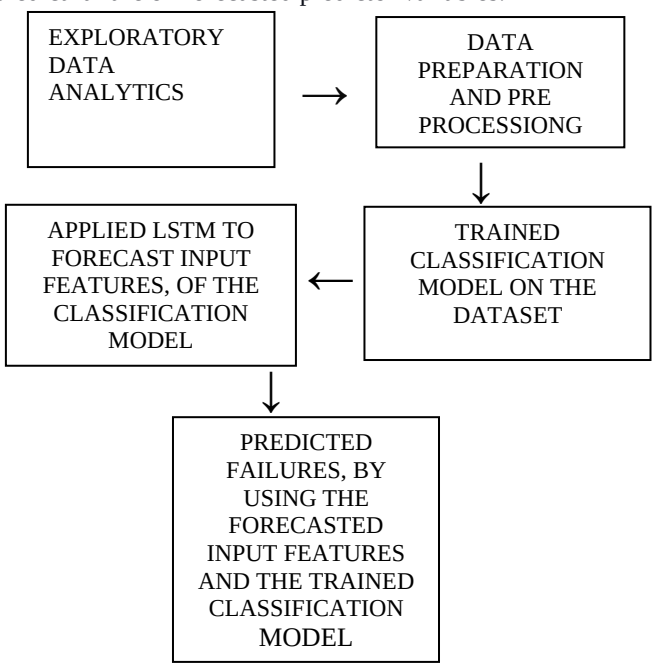
- We see that the ratio of frequency of failure points to non-failure points increases with increase in percentage average value which is the ratio of average memory and assign memory. So, by this observation we can say that the probability that a point fails increases when the assign memory/average memory value increases.

Sr. No	*Percentage average value range (count of points)*	*Count of failure points*	*Count of finished points*
1	0-10 (6,33,792)	27,968	83,838
2	10-20 (1,29, 971)	1,618	4,232
3	20-30 (3,04,498)	253	533
4	30-40 (3,23,007)	120	94
5	40-50 (1,18,892)	119	84
6	50-60 (2,68,410)	295	2
7	60-70 (149752)	328	2
8	70 - 80 (1,27,628)	307	0
9	80-90 (2,35,118)	107	0
10	90-100 (23,038)	7	0

III. MODELLING AND RESULTS

MODEL OVERVIEW

After pre-processing and merging the instance usage and instance events table, we apply the KNN classification model. The failure column formed acts as our target variable. The values of the predictor variables are forecasted using the LSTM model. Now the trained classification model is used to predict failure on forecasted predictor variables.



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 05, May-2021

A. KNN CLASSIFICATION

- This is a supervised Machine Learning method. This algorithm stores all the data, and whenever a new data point has to be classified it classifies that data to a category based on similarity.
- We use 'time', 'assign memory', 'average memory', 'average CPU', 'max CPU', 'max memory', 'page cache memory', 'cpi', 'mpi', 'percentage average', 'class average' as our predictor variables and the Failure column our target variable. The failure column has a very low number of failure points as compared to Finished and other events. So, we up sample the dataset to get a balanced dataset. We then apply KNN model on seventy percent of data and use the rest thirty percent of data for validation.
- The Table below shows precision, recall and F1-score for failure events (denoted by 1), finished events (denoted by 0) and other events (denoted by -1)

	Precision	Recall	F1-score
-1	1.00	0.94	0.97
0	0.69	1.00	0.82
1	0.28	0.99	0.44
accuracy			0.94
Macro avg	0.66	0.97	0.74
Weighted avg	0.98	0.94	0.95

- Precision is the ratio between the True Positives and all the Positives
- Precision is 1 for 'other events' (submit, queue etc.) which means all these events has no false positives. 0.28 Precision for events which fail means 28% of times the generated alert is true.
- The recall is the measure of our model correctly identifying True Positives.
- We have attained a Recall of 1 for events which finish successfully, and a recall of 0.99 for failure points. So out of 9176 failure points 9085 values are identified correctly.
- The $\mathbf{F_1}$ score is the harmonic mean of the precision and recall. The highest possible value of an F-score is 1.0, indicating perfect precision and recall, and the lowest possible value is 0, if either the precision or the recall is zero. We have attained a F1 score of 0.82 for points which have finished successfully and a F1 score of 0.44 for failure points.
- We have attained an overall accuracy of 94%.

B. LONG SHORT-TERM MEMORY

The Long Short-Term Memory (LSTM) model was proposed by Hochreiter et al. [10]. It is a type of recurrent neural network which can have long term dependencies. The term long short-term memory comes from the following intuition. The LSTM model introduces an intermediate type of storage called memory cell. Unlike the standard RNN with a hidden layer, in LSTM, each ordinary node in the hidden layer is replaced by a memory cell. Each memory cell is a composite unit built from simpler nodes that are connected through some multiplicative nodes A standard memory cell of a LSTM consists of the following elements [8]: Input node: Input node, denoted as gc, takes activation from the input data point x(t) at the current time step and from the hidden layer at the previous time step h(t−1). Usually, the summed weighted input is run through a tanh activation. An LSTM memory cell with a forget gate. [8] Input gate: Input gate, denoted as ic, takes activation from the current input data point x(t) as well as from the previous hidden layer data. A gate is so-called because its value is used to multiply the value of another node. It is a gate in the sense that if its value is zero, then flow from the other node is cut off. If the value of the gate is one, all flows pass through. The value of the input gate ic multiplies the value of the input node gc. Internal state: The internal state sc is the central part of an LSTM memory cell that has a self-connected recurrent edge with fixed unit weight. This edge spans adjacent time steps with constant weight. Therefore, error can flow across time steps without vanishing or exploding. This edge is often called the constant error carousel. Forget gate: These gates provide a method of "learning to forget". Output gate: The output gate is denoted as oc, which is multiplied by the internal state sc to produce the final value vc of a memory cell.

DATA PREPARATION FOR LSTM

LSTM is a recurrent neural network. The input to every LSTM layer must be three-dimensional. The three dimensions of this input are:

- **Samples**. One sequence is one sample. A batch is comprised of one or more samples.
- **Time Steps**. One time step is one point of observation in the sample.
- **Features**. One feature is one observation at a time step

We have taken ten previous timesteps of all the input features which are going to be fed in the classification model to predict the eleventh value

of each of them. To predict the twelfth value, it takes 2nd to 11th timestep and so on.
Refer Fig 4 for better understanding. It shows an example of how the data is reshaped for one of the features i.e., Average memory.

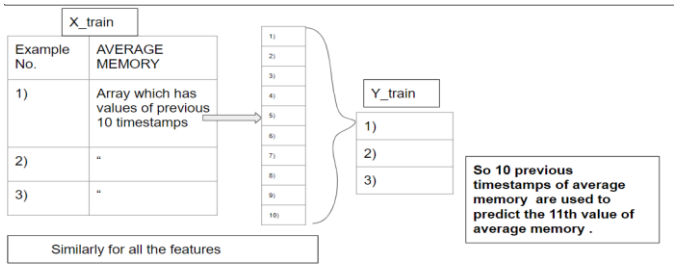


Fig. 4

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 05, May-2021

LSTM ARCHITECTURE

The input structure looks like (number of examples, 10 timestamps, 8 features).
The output structure looks like (number of examples, 8)
The LSTM architecture is explained using Fig 5.

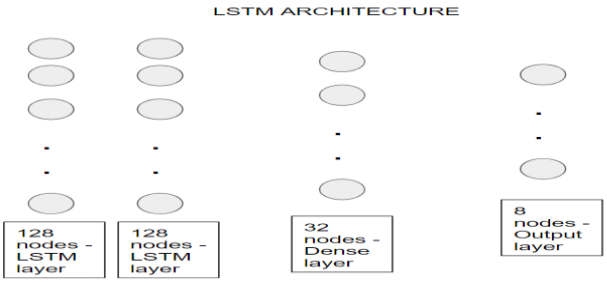


Fig. 5

LSTM RESULTS

The Accuracy obtained is 74.11% and loss is 1.6866e-04.

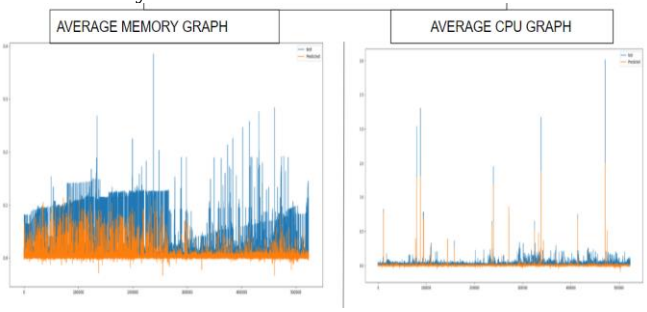


Fig 6.

The Blue lines represent the actual values and the orange lines represent the predicted value.

IV CONCLUSION

With the increasing cloud usage, it becomes important to understand when faults might occur and try to prevent them from happening. The data available about such workloads and traces is less. I have worked on the google borg dataset and tried to understand that why failures occur and how we could prevent them from happening. By plotting some graphs and making some tables we got some insights about the data. We first built a hypothesis that when the average memory consumed surpasses the assigned memory, failure occurs. We proved this by plotting the graphs and saw that for events which finish successfully, at no time the average memory would surpass the assigned memory but for failure events this is not true. For events which fail but do not have their average memory surpassing the assigned memory might fail due to other reasons, we also found some strongly correlated columns and dropped some of them. We then presented a classification model, KNN which was able to correctly identify 9085 points out of 9176 failure points. The model also correctly identified other events like submit, queue etc.

For predicting events which may fail in the future we used Long Short-Term Memory Model which predicted the input features to the classification model like the average memory consumption, assigned memory consumption etc. Then the trained classification model was to be used to predict whether the forecasted input features would fail or not. We achieved a good overall accuracy of 94% with a recall of 99% for the classification model and accuracy of around 75% for the LSTM model for the attributes of average memory consumption, assigned memory consumption. The two models when connected as a ML pipeline would help in predicting the task/job failure before time that will help the respective team to take pro-active actions to prevent the same. This work would help in improving the cloud server infrastructure, monitoring & management and we hope that we contribute in improving it by our work.

REFERENCES

[1] Predicting Application Failure in Cloud: A Machine Learning Approach
[2] Predicting and Mitigating Jobs Failures in Big Data Clusters
[3] Failure Analysis of Jobs in Compute Clouds: A Google Cluster Case Study
[4] N. El-Sayed and B. Schroeder, “Reading between the lines of failure logs: Understanding how HPC systems fail,” in Proc. of the 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), June 2013, pp. 1–12
[5] K. V. Vishwanath and N. Nagappan, “Characterizing cloud computing hardware reliability,” in Proc. of the 1st ACM Symposium on Cloud Computing, June 2010, pp. 193–204.
[6] S. Kavulya, J. Tan, R. Gandhi, and P. Narasimhan, “An Analysis of Traces from a Production MapReduce Cluster,” in Proc. of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, May 2012, pp. 94–103.
[7] K. Ren, Y. Kwon, M. Balazinska, and B. Howe, “Hadoop’s Adolescence,” Proceedings of the VLDB Endowment, vol. 6, no. 10, pp. 853– 864, Aug. 2013.
[8] Z. C. Lipton, “A Critical Review of Recurrent Neural Networks for Sequence Learning,” The Computing Research Repository (CoRR), vol. abs/1506.00019, June 2015.
[9] F. Gers, J. Schmidhuber, and F. Cummins, “Learning to Forget: Continual Prediction with LSTM,” Neural Computation, vol. 12, no. 10, pp. 2451–2471, Jan. 2000.
[10] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” Neural Computation, vol. 9, no. 8, pp. 1735–1780, Jan. 1997.
[11] Google cluster-usage traces v3 John Wilkes, with much help from Charles Reiss, Nan Deng and Muhammad Tirmazi Original version 2020-04-01, updated 2020-04-22
[12] Characterizing Machines and Workloads on a Google Cluster Zitao Liu, Sangyeun Cho
[13] Failure analysis of distributed scientific workflows executing in the cloud T. Samak, D. Gunter
[14] Job failures in high performance systems: A large scale empirical study
[15] Towards understanding heterogeneous clouds at scale: Google trace analysis, Charles Reiss
[16] Characterizing Cloud Applications on a Google Data Center, 2013 42nd International Conference on Parallel Processing
[17] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. A. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zhang, “TensorFlow: A system for large-scale machine learning,” The Computing
[18] H. Fadishei, H. Saadatfar, and H. Deldari, “Job failure prediction in grid environment based on workload characteristics,” in Proc. of the 14th International CSI Computer Conference, July 2009, pp. 329–334
[19] H. Pannu, J. Liu, and S. Fu, “A self-evolving anomaly detection framework for developing highly dependable utility clouds,” in Proc. of the IEEE Global Communications Conference, Dec. 2012, pp. 1605– 1610

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 05, May-2021

[20] A. Khan, X. Yan, S. Tao, and N. Anerousis, "Workload characterization and prediction in the cloud: A multiple time series approach," in Proc. of the IEEE Network Operations and Management Symposium, Apr. 2012, pp. 1287–1294.
[21] C. Reiss, A. Tumanov, G. Ganger, R. Katz, and M. Kozuch, "Towards understanding heterogeneous clouds at scale: Google trace analysis," Intel Science and Technology Center for Cloud Computing, Pittsburgh, PA, USA, Tech. Rep., 2012.
[22] C. Reiss, A. Tumanov, G. Ganger, R. Katz, and M. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in Proc. of the 3rd ACM Symposium on Cloud Computing, 2012.
[23] D. Ford, F. Labelle, F. Popovici, M. Stokely, L. Truong, V., C. Grimes, and S. Quinlan, "Availability in globally distributed storage systems," Procdings of USENIX Symposium on Operating Systems Design and Implementation, 2010
[24] R. Birke, I. Giurgiu, L. Chen, D. Wiesmann, and T. Engbersen, "Failure analysis of virtual and physical machines: patterns, causes and characteristics," in 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2014.
[25] A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, and D. H. J. Epema, "The Grid Workloads Archive," Future Generation Comp. Syst., vol. 24, no. 7, pp. 672–686, Jan. 2008.
[26] G. Bontempi, S. Ben Taieb, and Y. A. Le Borgne, Machine learning strategies for time series forecasting, Lect. Notes Bus. Inf. Process., vol. 138 LNBIP, pp. 6277, 2013.
[27] T. Pitakrat, D. Okanovic, A. van Hoorn, and L. Grunske, "Hora: ´ Architecture-aware online failure prediction," Journal of Systems and Software, 2018
[28] 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

(This work is licensed under a Creative Commons Attribution 4.0 International License.)