International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 11
July2016

Predictive Migration for Application High Availability

Prashanth Chillabatte
M.Tech, 4th Semester, Computer Engineering
Sri Jayachamarajendra College of Engineering, Mysuru
prashanthchillabatte@gmail.com

Abstract— **Customer applications and workloads are increasingly become complex today as business environments they serve are very dynamic, competitive and require IT systems and applications that are highly flexible to serve the changing needs of businesses. One such need is high availability of applications and services customer uses, to keep the continuity of business.**

Clustering of machines is one of the approaches used to achieve high availability. Traditional high availability clustering solutions involve restarting of business critical applications on the standby machines in the cluster when the primary machine goes down. The services rendered by the application would not be available causing the service downtime proportional to the complexity of the application.

The crux of the innovation is to use the log messages generated by the machine and techniques of machine learning such as xgboost to predict failure and time to failure of the machine, and before failure impacts application availability take action such as migration of applications to an healthier machine in the cluster. Migration doesn't require restart of applications and services, which drastically brings down the downtime of the application and the services rendered by them or completely eliminate it.

***Index Terms*—Business Continuity, Failure Prediction, High Availability (HA), Migration, Prediction, Predictive Migration**

I. INTRODUCTION

A common problem for high-end server systems or any IT infrastructure is that customers demand systems that never fail, but hardware components are inherently prone to failure causing software's running on them to fail as well. These failures have high costs for customers who may experience loss of service, in some situations, can mean millions of dollars in revenue loss.

The unplanned downtime even to a tune of few minutes or few occurrences with sub-minute level downtime in a year is unacceptable as it causes business disruption, loss of reputation and regulatory penalties. As Smartphone based e-commerce is emerging, the demand for highly available IT services is only increasing and any unplanned or unexpected

downtimes will reduce the competitiveness of business which in turn may force to windup the business operations. Similarly with emergence of consuming IT services in a new way such as Cloud, the cloud service provides and also private cloud

administrators has to ensure the IT infrastructure they provide are highly available.

Many IT firms like HP Enterprise, IBM etc., providing IT infrastructure with foundation layers such as Servers, Storage and Network have very efficiently proposed and implemented the idea of clustering as a solution for providing Highly Available IT infrastructure.

High availability clusters (also known as HA clusters) are groups of computers that support server applications that can be reliably utilized with a minimum of down time. They operate by using high availability software to harness redundant computers in cluster that provide continued service when system components fail. Without clustering, if a server running a particular application crashes, service rendered by application becomes unavailable until the crashed server is fixed. HA clusters handle this situation by immediately restarting the application on another system without requiring administrative intervention, a process known as failover.

One of the major problems with this classic HA clusters approach to handle an application or infrastructure failure is failover. Failover is a process initiated after the occurrence of the failure in the machine. Failover involves restarting of the application/service on standby machine in the cluster, which is a time consuming process and the time required for process grows exponentially with the complexity of the application under failover. As far business is concerned the restart time is considered to be downtime as application will not be able to render the service.

With ever-growing complexity and dynamicity of IT infrastructure, proactive failure management is an effective approach to enhance system dependability. Failure prediction is the key to such techniques. Failure prediction forecasts

International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 11
July2016

future failure occurrences in the IT infrastructure using runtime execution states of the system and the history information of observed failures.

Current work aims at building a prototype for predicting the failure of a machine by predicting time to failure, which may be caused due to the failure of any hardware components, based on the analysis of events log generated by hardware resources and built-in error analysis engine of the machine. The event log messages generated are lifeline for predictive analytics engine which will be part of any clustering solution. Once a failure is predicted, the cluster would initiate live migration of an application context to an alternate machine in the cluster.

Live migration is a process which checkpoints the applications on the primary machine, finds a healthier machine in the cluster and transfer the application's active memory and current execution state to another machine in real time and restores the application on the migrated machine from the point it was check pointed, which brings down the downtime of the application and services rendered by them to near zero or completely eliminate it. Figure 1 depicts how overall system looks like.

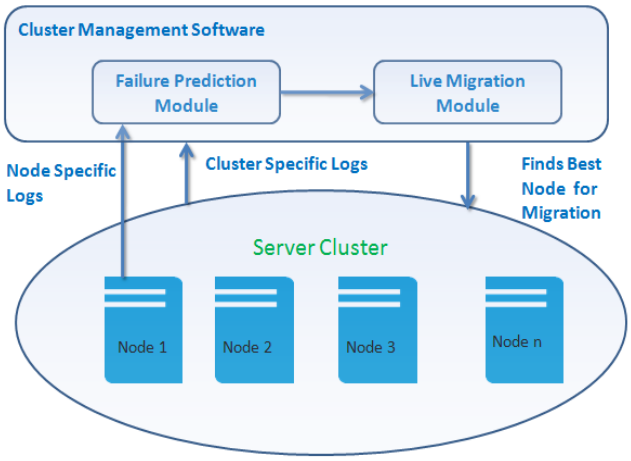

Figure 1: Cluster with Predictive Migration Capability

II. Related Concepts & Work

A. High Availability

High availability [17] refers to a system or component that is continuously operational for a desirably long length of time. Availability can be measured relative to "100% operational" or "never failing." A widely-held but difficult-to-achieve standard of availability for a system or product is known as "five 9s" (99.999 percent) availability. [15] In 1998, HP management committed to a new vision for HA in open systems: 99.999% availability, with no more than five minutes of downtime per year.

It is [1] paradoxical that the larger a system is, the more critical is its availability, and the more difficult it is to make it highly-available. Process control, production control, and transaction processing applications are the principal consumers of high-availability systems. Telephone networks, airports, hospitals, factories, and stock exchanges cannot afford to stop because of a computer outage. Any loss of service, whether planned or unplanned, is known as an outage. Down time is the duration of an outage measured in units of time (e.g., minutes or hours).

High Availability as Requirement: In the current business climate, HA computing is a requirement, not a luxury. HA is a form of insurance against the loss of business due to computer downtime.

High Availability as Opportunity: Highly available computing provides a business opportunity, since there is an increasing demand for "around-the-clock" computerized services in areas as diverse as banking, financial market operations, communications, resource management, e-commerce and etc.

B. System Log Files

System log files are important for managing computer systems since they provide a history or audit trail of events [16]. Any change in system status is termed as an event. Given the log file information it may be possible to determine causes of events that have occurred. It is also possible to use the information contained in system log files for predicting events.

System log files are typically text files that consist of messages sent by applications to the logging service. Syslog is the configurable general purpose logging application available for different Unix platforms [16]. Applications send information to the syslog process, which stores this information in a text file in the order that they arrive.

Syslog is primarily responsible for managing the log file while the message content is largely created by the application.

C. Failure Prediction & Migration

Failure Prediction is about getting information in advance on any abnormal behavior of a system parameter and component which can lead to the failure of the entire system. Predictive analytics techniques enable to do so.

Predictive Analytics is a practice of extracting information from existing raw data to determine the patterns and predict future outcomes and trends. Predictive analytics does not tell what will happen in the future; It forecasts what might happen in the future with an acceptable level of reliability and includes what-if scenarios & risk assessment.

Failure: Each module has an ideal specified behavior and an observed actual behavior. A failure occurs when the actual behavior deviates from the specified behavior. The failure occurred because of an error - a defect in the module. The cause of the error is a fault. The time between the occurrence of the error and the resulting failure is the error latency. When

International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 11
July2016

the error causes a failure, it becomes effective. Error latency is also termed as time to failure (TTF) or time to live (TTL). Figure 2 depicts the relationship between fault, error and failure.

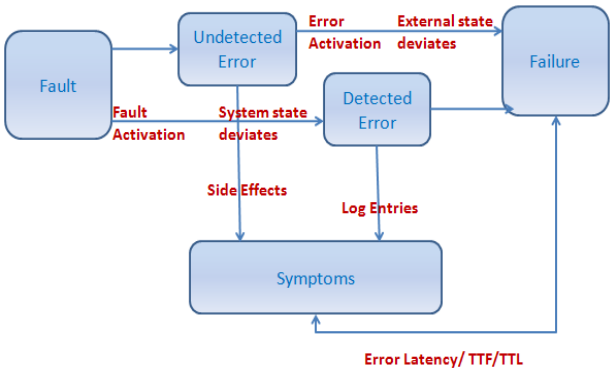

Figure 2: Relations between Faults, Error and Failure

Predicting the near term future is more clever and frequently more successful than attempting long term predictions [2]. Short term predictions are especially helpful to prevent potential disasters or to limit the damage caused by computer system failures. Online failure prediction incorporates measurements of actual system parameters during runtime in order to assess the probability of failure occurrence in the near future in terms of seconds or minutes.

Migration [18] refers to the process of moving an application between different physical machines without disconnecting the client or application. Memory, storage, and network connectivity of the application are transferred from the original guest machine to the destination.

D. XGBoost

XGBoost is short for "Extreme Gradient Boosting", where the term "Gradient Boosting" is proposed in the paper Greedy Function Approximation: A Gradient Boosting Machine, by Friedman [3]. XGBoost is based on this [3] model. It's a supervised learning model. This model is often described as a blackbox, meaning it works well but it is not trivial to understand how [19].

XGBoost is known for its fast speed and accurate predictive power; it also has various functions to help you understand the model, like assessing the importance of each feature used in building the model, displaying the trees built to understand the splits and the interactions between features. Xgboost implicitly have features like cross validation.

XGBoost has both linear model solver and tree learning algorithms. Its capacity to do parallel computation on a single machine makes xgboost at least 10 times faster than existing gradient boosting implementations. It supports various objective functions, including regression, classification and ranking [20].

E. Related Work

A significant body of work has been published in the area of failure prediction research. Gordon Hughes and Joseph Murray analyze failure in hard disk drives [4] [5]. Their general framework is to detect anomalies, or variations from "normal" behavior, using a rank-sum nullhypothesistest. This work is limited only to one component of the system. Greg Hamerly and Charles Elkan also examine failures in disk drives [6], but use different statistical tests based on naive Bayesian classifiers.

Erinn Fulp [7] et.al describes new spectrum-kernel support vector machine approach to predict failure events based on system log files. There have been several approaches for predicting system failure using system log files [8-12]. System log files consist of messages created by the different processes executing on the system. The information recorded varies from general messages concerning user logins to more critical warnings about program failures. Prediction methods include standard machine learning techniques such as Bayes networks, Hidden Markov Models (HMM), and Partially Observable Markov Decision Process (POMDP) [11].

The use of time-series analysis is common among these methods since a system message in isolation has been shown to be insufficient for predicting failure [12, 13]. We also believe that certain sequences of log messages may provide sufficient information to predict failure. But the large amount of information available in system log files makes finding the right pattern(s) difficult.

Significant amount of research is already done in the field of predicting failure of system. However providing accurate with sufficient lead time remains a challenging problem. A piece of information that is lacking in machine predictive maintenance is a good estimation of Time to Failure [14].

III. PREDICTION MODULE

Prediction module makes the machine learn about normal functioning conditions of the system and the conditions leading to the failure of the machine. This module continuously monitors the machine and detects the conditions which may lead to the failure of the machine.

Prediction model has 2 parts, first is to build a prediction model called training the model and second is evaluating the model called testing. Figure 3 shows the various steps involved in building prediction model.

International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 11
July2016

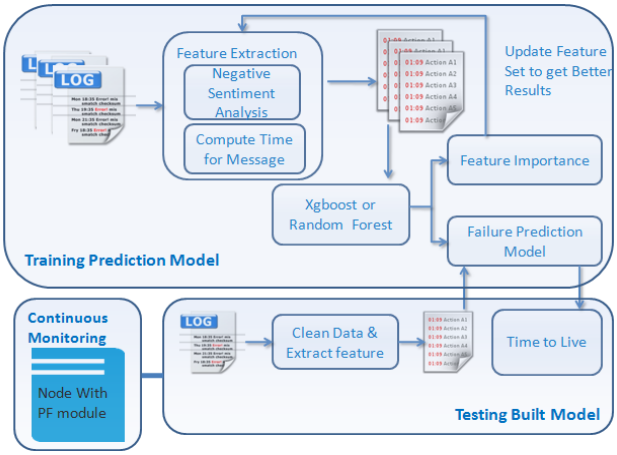

Figure 3: Prediction Model

Building failure prediction model is not a one-shot process of building a data set and running a learner, but rather an iterative process of running the learner, analyzing the results, modifying the data and/or the learner, and repeating.

A. *Labeling Data Set*

All machine learning models are based on general principal of learning from the past experiences. Failure data should be gathered for training a prediction model. In this process extracting of instances i.e. data items from software archives and labeling (TRUE or FALSE) is done. Our prediction model learns from the log files of various different server machines, of different types. Log files contain the records for event resulting in system state change. This log files have records of events or chain of events that has led to the failure of machines in the past along with the timing of the events.

Log records do have a specific format as seen in figure 4 consisting of the four fields. First field in the log file is time object, the time field is the time the message was recorded by the syslog facility. Second field is the hostname of the machine sending the message. Third field is the source of the message, for example kernel or user space and last field are the actual messages. Message is the text portion of the entry that describes the event that has occurred.

Log files are searched for entries causing shutdown of the system. Shutdown entries in the log files might be the results of planned or unplanned machine shutdown. Log entries before the shutdown are parsed for containing negative sentiments indicating critical future events. All matched shutdown entries are labeled as planned or unplanned based on the previous messages that have occurred 30 minutes before the machine shutdown.

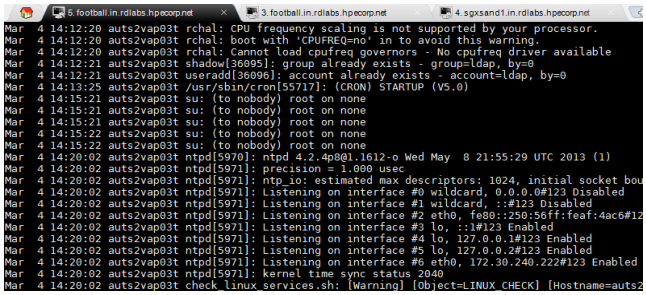

Figure 4: Sample HPE Server Log File

Figure 5 shows the shutdown log entry along with entries containing negative sentiments marked in red box. We label it as unplanned shutdown.

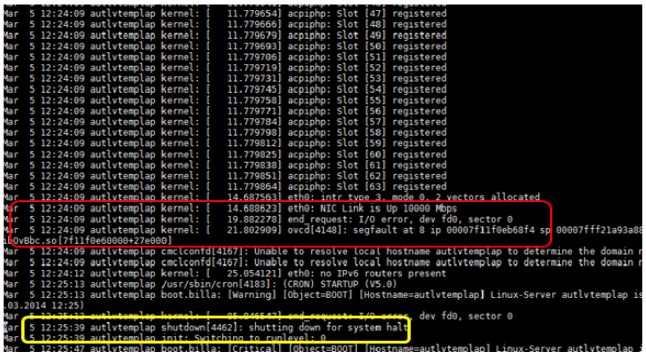

Figure 5: Labeled Unplanned System Shutdown Log Entry

B. *Feature Extraction*

Feature extraction is the most important factor in building any prediction model. This is typically where most of the effort in a project goes.

Labeled log files generated in the previous step becomes input for feature selection and extraction. Our prediction model generates 3 features from every log message entry.

- Sequence Number
- Time to Live/ Time to Failure
- Message

Sequence number is a numerical value assigned to each message based on its order of entry in the labeled log file. For each file numbering starts from 1 and incremented with unit of 1 till the last message in the log file is numbered. Sequence number is unique for each message only within the respective labeled log file.

Time to Live or Time to Failure is calculated for every message entry in the log file. TTL or TTF is the time difference calculated between the shutdown log time and the message log entry time. TTL or TTF is expressed in units of seconds.

Message is the plain text portion of the log entry, describing the event that has occurred due to which system state has changed.

International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 11
July2016

Extracted features are transformed into a CSV file format, which are used to build the prediction model using XGBoost.

C. *Training Prediction Model: XGBoost Lerner*

A black box machine learner "XGBoost" is used to build prediction model using training data set generated in previous step.

Model of xgboost: Tree Ensembles. Tree ensemble is a set of classification and regression trees (CART). Usually, a single tree is not strong enough to be used; xgboost constructs n numbers of trees with an "Additive Training" strategy i.e. fix what is learned and add one new tree at a time such that adding new tree results in better prediction accuracy. Better prediction accuracy comes with reduction in training loss. Prediction from multiple trees is added.

Tree boosting i.e. learning tree takes general principal of defining an objective function and optimizing it. Objective function consists of two parts: training loss and regularization as denoted in equation 1.

$$Obj(\theta) = L(\theta) + \Omega(\theta) \quad (1)$$

L is training loss function and **Ω** is regularization term. The training loss measures how predictive our model is on training data. Root mean squared error (RMSE) is used as training loss function. Regularization term controls the complexity of the model, which helps avoid overfitting of the model. We define complexity as shown in equation 2.

$$\Omega(f) = \Upsilon T + (1/2)\lambda \sum_{j=1}^{T} \omega_j^2 \quad (2)$$

T is the number of leaves in the tree**, f** denotes tree, **ω** is the vector of scores on leaves, and Υ and λ are constants.

Xgboost validates the addition of trees using cross validation. Divides the training data in nfold parts; xgboost retains the first part to use it as test data and constructs tree from rest data folds. It reintegrates the first part to the training dataset and retains the second part, do training and so on.

Xgboost only works with numeric data types convert all categorical features to numeric type. Assign values to default parameters and run the learner on training data set. Xgboost returns the trained prediction model along with feature importance object and tress built.

D. *Prediction & Assessment*

Training phase builds the prediction model, which is evaluated by testing with different test data log files. The same process of feature extraction is done for test log files also. Extracted feature instance is applied on the built prediction model. Prediction model gives us the predicted time to live/time to failure for every message in test data set.

IV. MIGRATION

Migration is responsible for live migration of applications from one machine to another in the cluster. Whenever predicted time to failure of the machine for a log message is less than threshold, migration is triggered.

Migration involves dumping and restoring processes running on two different machines in the cluster. Dumping process runs on machine triggering migration called dumping node, restoring process running on the machine where applications are migrated to called restoring node.

Migration is a synchronized process between dumping and restoring nodes. Both the process mounts temporary file system on the respective nodes.

Dumping process triggers launching a page server on the restoring node. Page Server accepts pages from dumping node and puts them into tmpfs mount on restoring node. Dumping node does not store any images.

Migrating application is identified on dumping node, copy application running context, memory images to the restoring node. On successful restore of the migrating application, temporary file systems are unmounted and applications are killed.

Restoring process responds to the dumping process by launching page server to accept application images. After copying images files, applications migrated are restored and temporary file systems are unmounted.

On successful completion of restore process, application/packages on dumping node are stopped. Figure 6 gives the pictorial representation of the migration module.

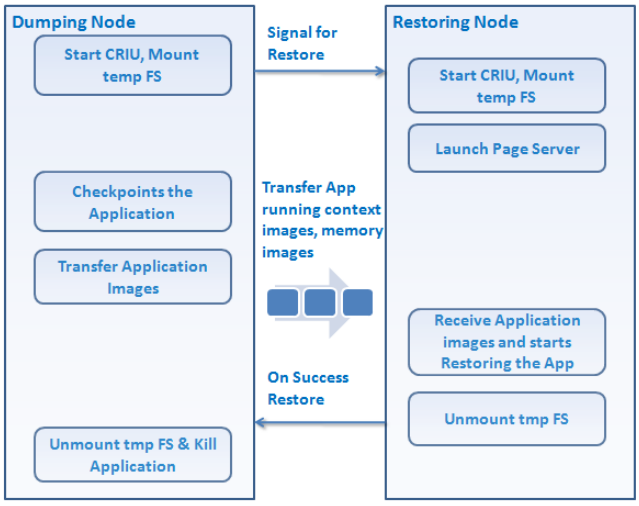

Figure 6: Steps Involved in Migration Process

CRIU an open community tool is used for migration of application in user-space. Checkpoint/Restore In User-space is a software tool for Linux operating system. Using this tool, running application can be migrated in user-space.

International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 11
July2016

V. RESULTS

The evaluation of a prediction model requires a testing data set besides a training data set. The labels of instances in the testing set are predicted and the prediction model is evaluated by comparing the prediction and real labels. Labeled log files obtained in first step is divided into training and testing data set. 70% of the labeled log files are used as training data set and remaining 30% is used for testing the built prediction model. Failure prediction model is tested for different testing data sets, results of two such data sets is given in figure 7 and 8. Figure shows predicted value of TTL with actual value of TTL. Predicted TTL deviates not more than 1minute or 60 seconds from actual TTL value.

Data Editor

File Edit Help

	seqno	Timetolive	message	predictedTtl
129	4	174	Membership: membership at is HALTED (coordinator >	230.843
130	5	169	Service cmlogd terminated due to a	192.3064
131	6	169	The cluster daemon aborted our connection	172.367
132	7	169	The cluster daemon aborted our connection	203.1791
133	8	168	Service assistant daemon halted.	211.7341
134	1	132	autlvtemplap boot.billa: [Critical] [Object=BOOT]>	135.9059
135	2	132	autlvtemplap I/O error occurred while writing; er>	135.5022
136	3	132	autlvtemplap Received signal terminating.	175.8598
137	4	128	autlvtemplap rpcbind: rpcbind terminating on sign>	172.5223
138	5	128	autlvtemplap kernel: Kernel log daemon terminatin>	159.6181
139	6	128	autlvtemplap Termination requested via signal te>	198.7148
140	7	93	bind failed (Address already in use (errno = serv>	130.167

Figure 7: Test Results of Data Set 1 in Excel Format

RStudio

File Edit Code View Plots Session Build Debug Tools Help

myxgb.R × mytest ×

Filter

	seqno	Timetolive	message	predictedTtl
1	1	126	kernel: [end_request: I/O error dev sector	100.5027
2	2	115	kernel: [end_request: I/O error dev sector	136.7119
3	1	810	kernel: Memory error not recovered	801.8119
4	2	810	kernel: [Hardware Error] Machine check events logged	801.5334
5	3	810	kernel: Uncorrected hardware memory error in user-a...	776.2235

Figure 8: Test Results of Data Set 2 in Console

Migration process is tested for migration of a running application and memory updates by the migrated application on migrated machine.

Note the process id(PID) of the application before migrating on source machine. Figure 9 highlights the PID of migrating application with yellow box and source machine name is highlighted with maroon box.

After migration, look for PID of the migrated application on destination machine. Figure 10 highlights the PID of migrated process on destination machine highlighted with maroon box.

Presence of PID of migrated application on destination machine marks the successful completion of application migration.

Current implementation migrates an application that updates a text file regularly with the system time instance. Figure 11 shows the file under updation and few entries of the same on source node before migration. Figure 12 shows the presence of same file on destination node after migration.

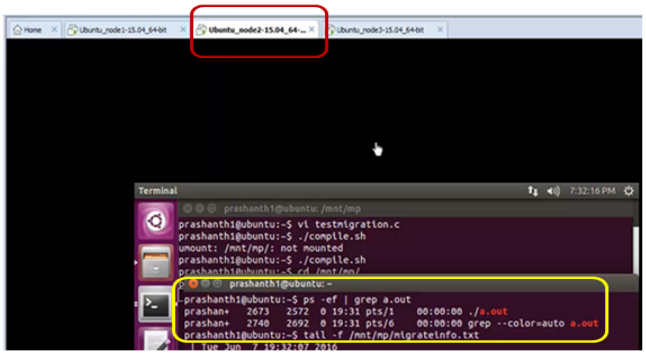

Figure 9: Applications PID on Source Node

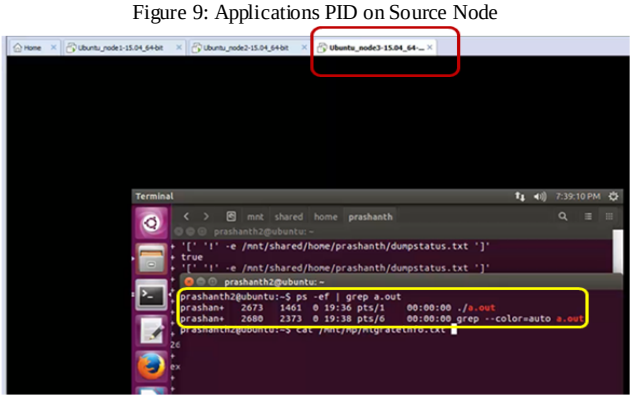

Figure 10: Migrated Applications PID on Destination Node

Presence of text file to which time instance are written on destination node marks the successful migration of application memory and run time updating to the same text file concludes the running state of application after migration. Running application is migrated i.e. live migration.

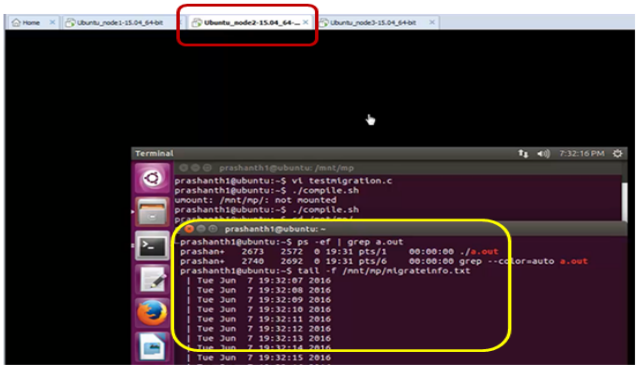

Figure 11: Application File Write on Source Node before Migration

International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 11
July2016

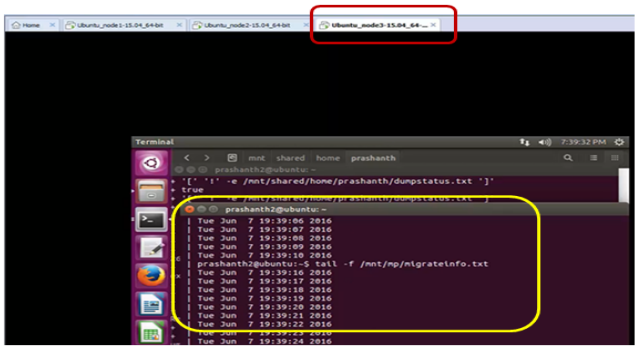

Figure 12: Application Writes on Destination Node after Migration

VI. Conclusion

Large-scale and complex IT infrastructure centers are susceptible to software and hardware failures, which significantly affect the system performance and management. In this work, we present a failure prediction and live migration mechanism for achieving high availability.

Log files typically contain useful information about system failures. These files record the history of the system's state which provides information to determine the causes of critical events. Although log file analysis has been primarily performed after an event has occurred, increasingly this information is being used to predict events. We propose to use tree ensemble based XGBoost learner model to build failure prediction model based on the information contained in log files.

In this work we implement a prototype of live migration of applications using open software CRIU in user-space. Experimental results show that our proposed model can forecast failure dynamics with high accuracy and migrates applications in user-space.

Both proposed prototype models need more enhancements and fine tuning to deploy in complex and large scale IT infrastructures to achieve HA. We need several ten thousand or lakhs of failure event logs with system monitoring daemons running to build more generic and scalable prediction model.

Acknowledgment

Gratitude takes three forms-"A feeling from heart, an expression in words and a giving in return". I take this opportunity to express my heart-felt feelings.

I feel great to express my gratefulness to my guide Dr. Roopamala T D, Associate Professor, department of CS&E, for her guidance and for being source of motivation and support throughout this work.

I feel great to express my gratefulness to Mr. Bhakthavatsala Naidu, Master Architect, SETL lab, HP Enterprise, Bangalore, for his guidance and for being source of motivation and support throughout this work.

I feel great to express my gratefulness to Mr. Arun Ramachandran, SETL lab, HP Enterprise, Bangalore, for his guidance and support throughout this work.

I would like to thank all those people who have directly or indirectly helped me successfully complete this work.

References

[1] Jim Gray and Daniel P. Siewiorek. "High Availability Computer Systems", High Availability Paper for IEEE Computer Magazine Draft.

[2] Felix Salfner, Maren Lenk, and Miroslaw Malek. "A Survey of Online Failure Prediction Methods", ACM Journal Name, 2005.

[3] Jerome H. Friedman. "Greedy Function Approximation: A Gradient Boosting Machine", IMS 1999 Reitz Lecture, February 24 1999.

[4] Hughes G, Murray J, Kreutz-Delgado K. and Elkan C. "Improved disk-drive failure warnings", In IEEE Transactions on Reliability, vol. 51, no. 3, September 2002.

[5] Murray J, Hughes G, and Kreutz-Delgado K."Hard drive failure prediction using non-parametric statistical methods", In Proc. ICANN/ICONIP, June 2003.

[6] Hamerly G, and Elkan C. "Bayesian approaches to failure prediction for disk drives", In Proceedings of the Eighteenth International Conference on Machine Learning, 2001.

[7] Errin W. Fulp, Glenn A.Fink and Jereme N.Haack. "Predicting Computer system Failure using Support Vector Machine".

[8] FU S, and XU C.-Z. "Exploring event correlation for failure prediction in coalitions of clusters", In Proceedings of the IEEE/ACM International Conference on High Performance Computing, Networking, Storage and Analysis, 2007.

[9] Liang Y, Zhang Y, Xiong H, and Sahoo R. "Failure prediction in ibm bluegene/l event logs", In Proceedings of the IEEE International Conference on Data Mining, 2007.

[10] Stearley J, and Oliner A. J. "Bad words: Finding faults in Spirit's syslogs", In Proceedings of the 8th IEEE International Symposium on Cluster Computing and the Grid, 2008.

[11] Xue Z, Dong X, MA S, and Dong W. "A survey on failure prediction of large-scale server clusters", In Proceedings of the International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007.

[12] Yamanishi K, and Maruyama Y. "Dynamic syslog mining for network failure monitoring", In Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, 2005.

[13] Pinheiro E, Weber W.-D, and Barroso L. A. "Failure trends in a large disk drive population", In Proceedings of the USENIX Conference on File and Storage Technologies, 2007.

[14] Stearley J, and Oliner A. J. "Bad words: Finding faults in Spirit's syslogs", In Proceedings of the 8th IEEE International Symposium on Cluster Computing and the Grid, 2008.

[15] "Clusters for High Availability", 2nd edition, Pearson Education 2001.

[16] Garfinkel, S. "Practical UNIX and Internet Security", O'Reilly, 2003.

[17] http://searchdatacenter.techtarget.com/definition/high-availability

[18] https://en.wikipedia.org/wiki/Live_migration

[19] https://www.kaggle.com/tqchen/otto-group-product-classification-challenge/understanding-xgboost-model-on-otto-data/notebook

[20] http://www.analyticsvidhya.com/blog/2016/01/xgboost-algorithm-easy-steps/