Proactive cloud service assurance framework for fault remediation in cloud environment

Jyoti Shetty, Sathish Babu B., Shobha G.
Department of Computer Science and Engineering, R.V. College of Engineering, India

Article Info

Article history:

Received Jan 6, 2019
Revised Sep 16, 2019
Accepted Sep 29, 2019

Keywords:

Cloud fault remediation analytical model
Cloud service assurance
Expert system
Proactive fault handling

ABSTRACT

Cloud resiliency is an important issue in successful implementation of cloud computing systems. Handling cloud faults proactively, with a suitable remediation technique having minimum cost is an important requirement for a fault management system. The selection of best applicable remediation technique is a decision making problem and considers parameters such as i) Impact of remediation technique ii) Overhead of remediation technique ii) Severity of fault and iv) Priority of the application. This manuscript proposes an analytical model to measure the effectiveness of a remediation technique for various categories of faults, further it demonstrates the implementation of an efficient fault remediation system using a rule-based expert system. The expert system is designed to compute a utility value for each remediation technique in a novel way and select the best remediation technique from its knowledgebase. A prototype is developed for experimentation purpose and the results shows improved availability with less overhead as compared to a reactive fault management system.

Corresponding Author:

Jyoti Shetty,
Department of Computer Science and Engineering,
R. V. College of Engineering,
Mysore Road, Bangalore, 560059, India.
Email: Sjyothi.12@gmail.com

1. INTRODUCTION

Resiliency in cloud computing environment is a challenge due to its shared and dynamic environment. It involves a complete process of monitoring, predicting, detecting, troubleshooting and finally remediation of faults [1]. Many works have been carried out in monitoring, predicting, detecting and troubleshooting faults in cloud but the fault remediation task remains relatively unexplored [2]. Endowing the cloud with a system which can automatically remediate the faults proactively is an interesting problem. Faults are remediated using techniques such as virtual machine (VM) migration, task migration, software rejuvenation, check pointing and rollback and others [3]. These techniques can be classified as preventive, reactive and hybrid approaches. A preventive technique is triggered before a fault occurs and it is designed to restore system to a normal state proactively, a reactive technique is triggered after a fault is detected to restore the system to normal a state, a hybrid approach is combination of reactive and proactive approach [3]. Also a fault can have multiple candidate remediation techniques, where each technique has associated cost and benefits, hence selection of an appropriate fault tolerance technique is a decision making problem.

This manuscript proposes to use an expert system for decision making i.e., selecting the best remediation technique based on the nature of fault occurred. Expert systems in artificial intelligence are knowledge based systems which mimics the decision making capability of human experts [4]. An expert system has a knowledge base or rule-base, a fact-base and an inference engine. The rule-base contains domain knowledge of experts for problem solving in form of IF-THEN rules. The facts base contains set of facts that are matched against the IF part of the rule. The inference engine deduces the results by matching the rule-base to facts base [5].

A Fault remediation process is a knowledge based problem, building upon the knowledge from previous fault handling instances. The rule-base for a fault remediation system captures faulty states of system and corresponding techniques, it can be constructed from various knowledge sources such as discussion forums, logs, manuals, Bugzilla and others [6, 7]. To best of our knowledge this is the first work using an expert system for cloud fault remediation. The main contributions of the work include: 1. A structured cost sensitive framework to rank a remediation technique based on multiple parameters, 2. Mathematical analysis to quantify a utility metric for each remediation technique, 3. A prototype of expert system to identify the fault type and apply the best possible remediation technique.

Formal Problem Definition, Consider that a cloud has M identical Virtual Machines (VMs) running on a Physical Machine (PM). Each VM has applications or services running on it, and the VMs and hosts are subjected to performance degradation or a fault. It is assumed that there exist a fault prediction system which acts as input to the expert system. Let there be N faults and set of P remedial techniques which can be used to handle the faults. Then the fault remediation system be represented as 5 tuple as in (1), where λ is the fault rate and α is the availability of the system given using (2) and (3) respectively. Each fault from a fault set F_N maps to a subset of remediation techniques R_P, the goal of fault remediation system is to select the best remediation technique from R_P such that it maximizes the availability α of the system.

$$E = \langle M, F_N, R_p, \alpha, \lambda \rangle \tag{1}$$

Where,

$$\alpha = \frac{MTBF}{MTBF+MTTR} \tag{2}$$

$$\lambda = \frac{1}{MTBF} \tag{3}$$

Where MTBF is Mean Time Between Failure and MTTR is Mean Time to Repair.

Fault Category and Remediation Techniques, Broadly faults in cloud computing can be categorized as: VM resource insufficiency, hardware fault, PM resource insufficiency, application fault, software fault, system fault [8]. The VM/PM resource insufficiency refers to VM/PM having resource crunch with respect to memory, storage and CPU. The application faults are like synchronization deadlocks, buffer overflow, application performance degradation and many others. The software faults here are operating system or system fault, and the hardware faults is malfunctioning of hardware components and network fault is failure to communicate with other PMs or VMs. Each fault can be rectified in multiple ways, e.g., a VM resource insufficiency fault can be addressed using remediation techniques such as dynamic resource scaling of the VM, migration of few processes to another VM, migrating the VM to PM with sufficient resource, or scaling out more VMs for load balancing. The summary of mapping of fault category to remediation technique is as shown in Table 1.

Table 1. Fault mapping to remediation technique

Fault Category	Remediation Technique	Fault Category	Remediation Technique
VM Resource insufficiency	- Scaleup VM resources, Process Migration, Scale out more VMs, VM Migration to PM with sufficient resources	Software/System fault	- Software rejuvenation, Process Migration, Preventive checkpoint and roll back, VM restart
PM Resource insufficiency	- Process Migration, VM Migration to PM with sufficient resources	Hardware fault	- VM Migration
Application faults	- Check point and Roll back, Application / service restart, VM restart, Software rejuvenation	Network fault	- Software rejuvenation - Application / service restart - VM restart
Unknown fault	- Create checkpoint and alert system administrator		

Organizations usually maintain a knowledge base of faults and remediation techniques [6, 7], which can be programmatically extracted to develop rules and knowledgebase of an expert system. The remediation techniques are available as both preventive and reactive techniques. A preventive technique is applied before a fault occurs e.g., scaling VM resources after a fault is predicted, a reactive technique is applied after a fault occurs e.g., VM restart after a fault is detected. A preventive technique is preferred over reactive technique, but in case of a false negatives i.e., when prediction system fails to identify a fault, and fault occurs then a reactive technique is applied. Each remediation technique has different impact in rectifying a fault. Accessing the impact of each remediation technique will aid in selection of the best suitable remediation technique.

2. PROPOSED FRAMEWORK AND IMPLEMENTATION

The proposed fault remediation can be a component of complete cloud service assurance system as show in Figure 1. A cloud service assurance system is a performance management system to monitor and manage performance and faults.

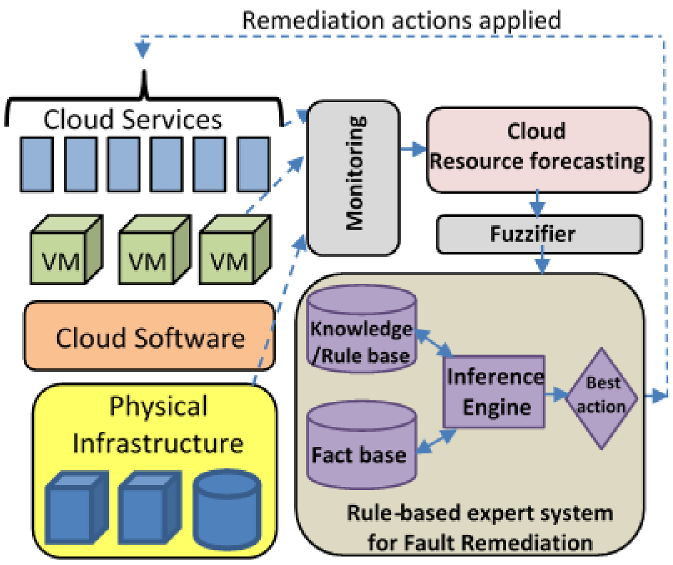

Figure 1. Proactive cloud service assurance framework

The components of the proposed cloud service assurance framework are as following:

- Monitoring: Cloud monitoring component measures the key performance indicators(KPIs) such as CPU utilization, memory utilization, disk utilization of physical machine, VM and applications running on VMs. Various tools and frameworks are available to monitor KPIs on cloud [9], which are stored in a database for analysis, forecasting and prediction using machine learning models.
- Cloud resource forecasting: The KPIs obtained from the monitoring module can be used as historical data to train machine learning models to forecast future values of KPIs [10]. The forecasted values can be used to predict faulty state of system and initiate proactive actions.
- Fuzzifier: The resource usage values or KPIs from the forecasting module is given as input to fuzzifier module. The fuzzifier does the process of transforming crisp values of KPIs into linguistic terms such as high, low, medium [11]. This mapping of continuous values to categorical values simplifies expert system decision making rules. The KPI values are mapped to different membership functions to check which membership function fits best. It was observed that a trapezoidal membership function fits best to the data.
- Fault Remediation (an Expert System): The fuzzifier module provides discrete values to the remediation module, which is an expert system responsible to identify the category of fault and select the best remediation technique. The various other system parameters like hardware error, application error, system error from the system logs, prediction modules are provided as input to the expert system. The fault remediation expert system rules are like traditional IF-THEN rules, which has antecedent and consequent part. The antecedent part consists of one or more input parameters with comparative values, the consequent part of rule consists of the category of fault. Once the category of fault is identified a utility value is calculated for each candidate remediation technique using (4).

The knowledge base/rule-base of the proposed expert system consists of rules like-

$$storage\ is\ high\ \vee CPU\ is\ high \vee\ network\ load\ is\ high\ \vee memory\ is\ high \rightarrow Resource\ insufficiency$$

$$Resource\ insufficiency\ \rightarrow best\_action(\text{Scaleup VM resources, Scale out more VMs}, Process\ Migration, VM\ Migration)$$

An expert system is implemented using PyKE release 1.1. PyKE is python knowledge engine, which can be used to build expert system for decision making where each problem has multiple solutions [5]. PyKE supports multiple fact-base, rule-base and an inference engine. The fact-base contains set of universally true statements. The rule-base consists of backward chaining rules. The inference engine applies rules to facts to generate additional facts to prove the goals through backward-chaining. The expert system uses heap datastructure to implement a priority queue of remediation techniques based on the utility values calculated using (4).

2.1. Analytical model for utility

This section discusses an analytical approach used to select the best remediation technique for a fault. The proposed approach defines a utility for each technique such that higher the utility value the better is the technique. The utility value is based on parameters such as i) Impact of remediation technique ii) Overhead of remediation technique ii) Severity of fault and iv) Priority of an application, as given in (4).

$$U(r_i) = \frac{P^i_{impact} * Severity_f}{overhead_i * application\_priority} \tag{4}$$

Where,
P^i_{impact} - The posterior probability of system being in a stable state after applying remediation technique
$Overhead_i$ - The overhead of remediation technique interms of availability
$Severity_f$ - Severity of fault is degree to which the fault can negatively impact the system. It is determined by the system administrator/expert. More the severity value the more critical the fault is. In this study the severity values for faults are defined as low (0.3), medium (0.5) and critical (0.7). Further the hardware faults, VM/PM resource insufficiency are assigned critical value (0.7). Application fault and Software fault are assigned medium (0.5) and network fault is assigned low value (0.3).
$application\_priority$ - The priority of an application is defined based on nice value in Linux based system. It ranges from 0 to 19, lower the value, higher is the priority of application.

The P^i_{impact} is posterior probability of achieving a stable system state after remediation technique r_i is applied and is computed using Baye's rule as follows [12].

$$P^i_{impact} = P(Normal\_state | r_i, \text{fault\_type}) = \\ = \frac{P(r_i\ | Normal\_state\ ) * P(fault\_type\ | Normal\_state\ ) * P(Normal\_state)}{P(r_i) * P(fault\_type)} \tag{5}$$

Where r_i is the ith remedial technique, Normal_state is stable state of system where fault is rectified. $P(r_i\ | Normal\_state\ )$ is the likelihood i.e., probability of remedial technique r_i lead to normal state. $P(fault\_type\ | Normal\_state\ )$ is likelihood i.e., probability of the system going to normal state after fault type fault occurs, $P(Normal\_state)$ is the prior probability of normal system, $P(r_i)$ is the prior probability of remedial technique, $P(fault\_type)$ is the prior probability of fault type, the likelihood and prior probabilities are initialized from the observation of the system in past.

The overhead of remediation technique i is defined as ratio of expected availability time by the actual useful computation time, calculated using (6).

$$Overhead_i = \frac{t^i_{av}}{expected\_availibility} \tag{6}$$

Where $expected\_availibility$ is the duration the system is expected to be available as per Service Level agreement (SLA), e.g., availability of 0.9999. The t^i_{av} is useful computation time of the system i.e., the time system spends only doing useful computation, which is given as in (7) and illustrated using Figure 2. It shows execution time line of system, where t is system uptime, d is downtime during which the system is unavailable and w is fault preparation time i.e., w is a fraction amount of t that the system spends in preparing the system to handle faults, e.g, for checkpointing technique it involves installing checkpoint, for migration it would be copy time from source to destination etc.

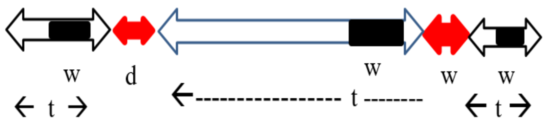

Figure 2. Execution time line of a system

The useful computation time t^i_{av} over the entire lifetime of a system can be calculated as - availability time minus the preparation time with probability density function integral over period(w, ∞). The lower bound of the integral is w as it is the time system spends preparing for fault doing no useful computation work and upper bound of integral is over entire lifetime of system i.e., ∞.

$$t^i_{av} = \int_w^\infty \frac{t-w}{t+d}\, p(t)dt \tag{7}$$

The integral in (7) cannot be solved analytically because any integrals from 1 to ∞ of (1/t) cannot be expressed interms of polynomials [13]. Hence using first order approximation by approximating the denominator t with MTBF, d with D total downtime of the system [14], the following equation is derived.

$$t^i_{av} = \frac{1}{MTBF+D} \int_w^\infty t - w\ p(t)dt$$

Considering exponential probability distribution for fault rate [15] with probability distribution function p(t)=λ $e^{-\lambda t}$, where λ = $\frac{1}{MTBF}$ the equation becomes

$$t^i_{av} = \frac{1}{MTBF+D} \int_w^\infty (t-w)\ \lambda\, e^{-\lambda t}\ dt$$

Solving the above equation

$$t^i_{av} = \frac{MTBF}{MTBF+D}\ e^{\frac{-w}{MTBF}} \tag{8}$$

Thus (8) gives actual available time excluding the time system spends in preparing for fault handling. Where w is the preparation time for different remediation techniques as given in Table 2, the preparation time is actually time wasted as it is not contributing to the system throughput. Thus given t^i_{av} and $expected\_availability$ the overhead is then calculated using (6). Using the overhead, application priority, fault severity and P^i_{impact} the utility of the remedial technique is computed using (4).

3. EXPERIMENT AND PERFORMANCE EVALUATION

The measurements for downtime, MTBF, availability and remediation preparation time given in Table 2 are based on the work done in [16-18] and are used as base values for the experiments in this work. The experimental setup consists of an Intel Xeon Server with dual 6 core 2.4 GHz Processor, 128GB RAM, configured with Openstack Pike [19] cloud platform and hosting three VMs. The resource insufficiency anomalies like crunch CPU, memory and disk [20] were inserted by running workloads. The Nagios monitoring software is configured to monitor the CPU Utilization, memory utilization and disk utilization metrics of the three virtual machine. An ensemble forecasting module is developed to forecast the future resource usage values so that the errors are identified before they actually occur, so as to initiate proactive actions. Further these values are converted to high, low and medium values by the fuzzy module. An expert system is configured and designed using Pyke 1.1, which receives the fault information from the each VM, computes the utility value and triggers corresponding remedial action.

Table 2. Scenario values

Scenario					
Downtime time: D = 4 minute, MTBF = 262800 minutes, Expected availability = 0.98					
Preventive Remediation Techniques	Wasted time/ Preparation time(w)	Preventive Remediation Techniques	Wasted time/ Preparation time(w)	Preventive Remediation Techniques	Wasted time/ Preparation time(w)
Checkpointing and rollback	2 min	process migration	30 sec	VM restart	5 min
Scaleup VM resources	2 min	VM migration	1 min	Roll back to previous check point	1 min
Scale out more VMs	2 min	Software rejuvenation	3 min	Application / service restart	1 min
Task resubmission	1 min				

The Table 3 shows the utility value calculations by the expert system for all the candidate remedial techniques for VM resource insufficiency fault and same is represented in Figure 3. It is observed that process migration technique with highest utility value is selected as best remedial technique by the expert system. A process migration involves migrating one or more processes to other VM to reduce the resource crunch on VM experiencing the fault. The utility value calculations by the proposed expert system for all the candidate remediation techniques for system fault is as shown in Table 4 and is represented using Figure 4. It is observed that VM restart technique with highest utility value is selected as best remedial technique by the expert system. Restarting the VM will restore the system to stable state. The Table 5 shows the utility value calculations by the expert system for all candidate remedial techniques for an application fault and same is represented in Figure 5. It is observed that software rejuvenation technique with highest utility value is selected as best remedial technique by the expert system.

Table 3. Utility values for VM resource insufficiency

VM Resource insufficiency, severity = 0.7, Priority = 15			
Remedial technique (r_i)	P^i_{impact}	Overhead$_i$	U(r_i)
Scaleup VM resources	0.30	1.0203	3.087
Scale out more VMs	0.40	1.0203	4.116
Process migration	0.50	1.0204	5.145
VM migration	0.45	1.0203	4.630

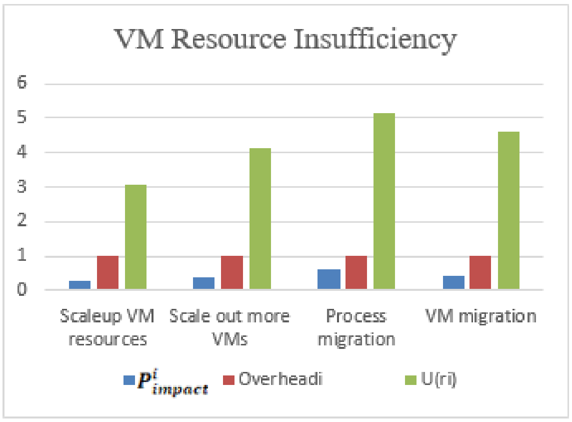

Figure 3. Utility values for VM resource insufficiency

Table 4. Utility values for system fault

Software/System fault, severity = 0.5, Application Priority= 15			
Remedial technique (r_i)	P^i_{impact}	Overhead$_i$	U(r_i)
Software rejuvenation	0.60	1.0203	4.410
process migration	0.60	1.0204	4.410
Preventive checkpointing and rollback	0.50	1.0203	3.675
VM Restart	0.65	1.0203	4.777

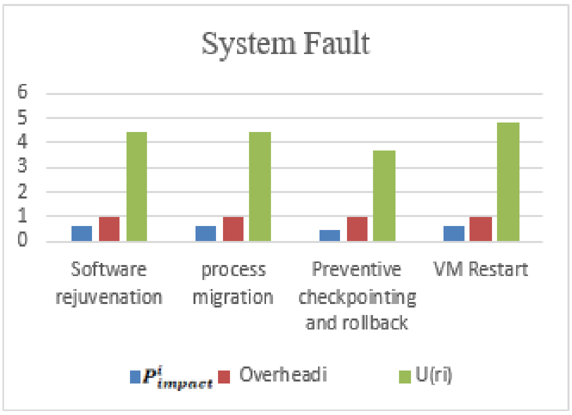

Figure 4. Utility values for system fault

Table 5. Utility values for application fault

Application fault, severity = 0.5, application_priority = 15			
Remedial technique (r_i)	P^i_{impact}	Overhead$_i$	U(r_i)
Preventive checkpointing and rollback	0.50	1.0203	3.675
Application/Service restart	0.55	1.0204	4.04
Software rejuvenation	0.65	1.0204	4.777
VM Restart	0.5	1.0203	3.675

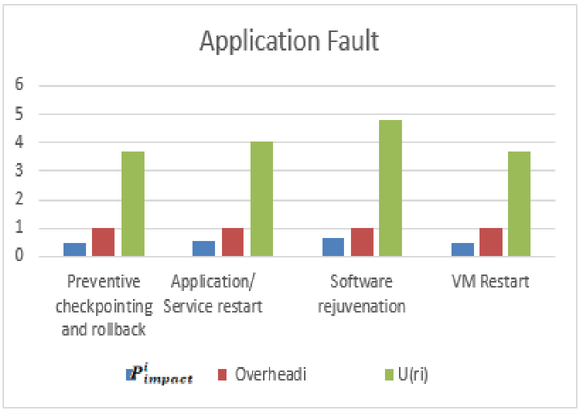

Figure 5. Utility values for application fault

Handling faults proactively by the remediation system is expected to improve the availability α of the system as it reduces the number of failure by handling the faults proactively, thus decreasing the failure rate λ and increasing MTBF. The improved availability of system can be given as in (9).

$$\alpha' = \frac{MTBF'}{MTBF' + MTTR'} \tag{9}$$

Where

$$MTBF' = \frac{1}{\lambda'}$$

Where $\alpha', MTBF', MTTR', \lambda'$ are availability, MTBF, MTTR and failure rate for the proposed proactive fault remediation system. The efficiency of a system is defined as percentage of faults it was able to handle proactively. For example a 10% efficient system will be able to prevent or handle proactively 10% of faults occurred. In such 10% efficient system the number of faults are reduced by 10% thus reducing the failure rate, increasing MTBF accordingly.

In this study to analyse the effectiveness of the proposed system, a exponentially distributed random fault rate is generated and the useful computation time given by (8) is calculated for system with no fault handling mechanism i.e a reactive fault handling system, system with 10% efficiency, system with 40 % efficiency and system with 60% efficiency. The useful computation time is plotted against the fault rate as shown in Figure 6. It is observed that a proactive system has more useful computation time compared to reactive fault management system and as efficiency of proactive system increases the useful computation time increases, and the system is able to handle faults proactively.

Similarly the overhead given by the (6) is calculated for exponentially distributed random fault rate for the reactive and proactive system with different efficiencies. The overhead values are plotted against the fault rate as shown in Figure 7. It is observed that as fault rate increases the overhead increases and overhead of reactive fault management system is more than proactive system, also as efficiency of system increases the overhead incurred is less as system is able to prevent the faults from occurring. Finally the MTBF and availability of the system are calculated using (9) for exponentially distributed random fault rate. The availability values are plotted against increasing fault rate as shown in Figure 8. It is observed that the availability of the proactive system is more than reactive fault management system and availability of the system increases with increase in efficiency of the system, as efficiency of expert system improves the availability increases. Based on the experiments and performance evaluation discussed, it is observed that the proposed expert system improves the availability of the system over a reactive fault management system.

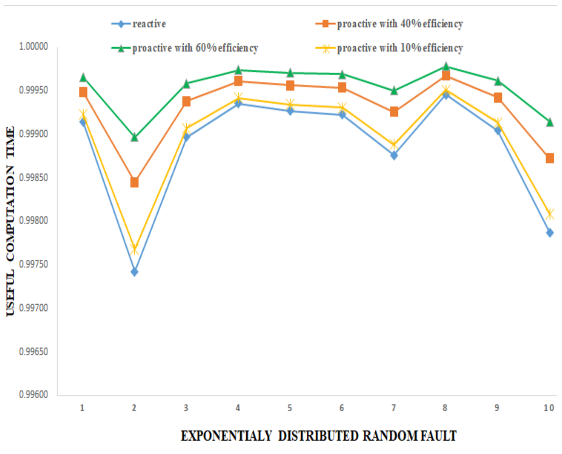

Figure 6. Fault rate versus useful computation time

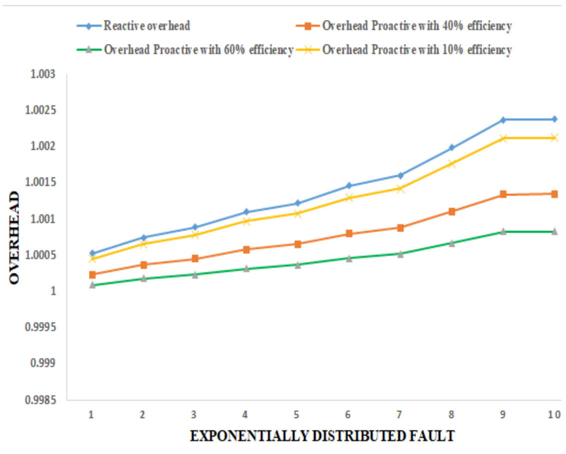

Figure 7. Fault rate versus overhead

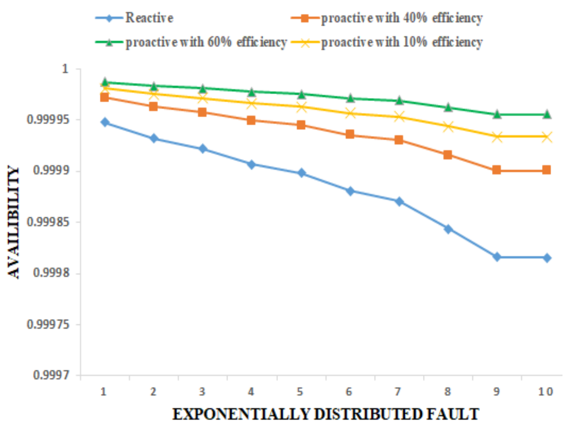

Figure 8. Fault rate versus availability

4. RELATED WORK

This section provides summary of previous related work. A knowledge management system [21] was designed for preventing SLA violations. At first the approach [21] formulates behavior of knowledge management system to prevent SLA violation and then evaluate different techniques to model the behavior such as situation calculus and case based reasoning. Case based reasoning was found to be effective and was designed and tested. In case based reasoning for each case corresponding remediation approach is initiated. An approach combining SLA prediction and cross layer adaptation was designed to prevent SLA violation [22]. The architecture for the proposed concept is given but it does not provide any details on implementation techniques used or experimental results.

A study [23] on techniques to be taken upon SLA violation defines five escalation levels: VM reconfiguration, migrate application, migrate VM, physical machine ON/OFF, outsourcing to other cloud provider. Depending on level of SLA violation suitable technique is initiated starting from the lowest level to highest. A middle layer [24] is defined to achieve fault tolerance which can tolerate node failure. It uses techniques such as replication and checkpointing for fault tolerance. CloudPD [25] is cloud problem determination and diagnosis system which uses an online learning approach to deal with frequent reconfiguration and other high tendency faults. It diagnoses anomalies based on pre-computed fault signatures and allows remediation to be integrated in an automated fashion. A proof-of-concept fuzzy control module for scaling up and scaling down of the resources without violation of SLA is proposed [26]. It uses imprecise information form the system administrator in form non numeric linguistic variable for example moderate, high and low. These fuzzy inputs are used in fuzzy control system that uses expert knowledge to inference fuzzy output. After defuzzifying the output is crisp value which controls overall scaling of the system.

PREPARE [27] uses Pearson correlation to detect anomalous attributes and 2-dependent Markov chain model with tree augmented Bayesian networks model for prediction of VM performance anomalies. Further it uses prevention schemes such as elastic resource scaling and VM live migration. In [28] selects the remediation technique based on wasted time. In this work they combine the remediation techniques like preventive checkpoint and rollback. Regular checkpoint and rollback, preventive migration with regular checkpointing, preventive rejuvenation with regular check pointing. Further they develop performance models for calculating wasted time and use it to select the remedial technique with least value for wasted time. The model does not assume any failure distribution. However proposed work complements these approaches by developing an analytical model to select the best technique by considering overhead and impact of each remediation technique.

5. CONCLUSION

Fault remediation in cloud environment is important but relatively less explored research. The proposed fault remediation system in cloud uses an analytical approach to rank the remediation techniques based on overhead and impact parameters. An expert system is then implemented that maps faults to remedies and selects the best remediation technique based on analytical model developed. The simulation results show that system is able to rank and select the best technique for given scenarios. Further the efficiency of system is analyzed for overhead and availability parameters. It shows that availability of the proposed system is better than reactive fault management system while overhead is less. Thus the fault remediation system improves availability and reduce downtime by handling the faults proactively.

REFERENCES

[1] O. Ibidunmoye, “Performance problem diagnosis in cloud infrastructures,” Licentiate dissertation, Department of Computing Science, Umeå University, Umeå, 2016.

[2] H. S. Gunawi, et al., “FATE and DESTINI: A Framework for Cloud Recovery Testing,” *NSDI'11 Proceedings of the 8th USENIX conference on Networked systems design and implementation*, pp. 238-252, 2014.

[3] Ganesh, et al., “A study on fault tolerance methods in Cloud Computing,” *2014 IEEE International Advance Computing Conference (IACC)*, Gurgaon, pp. 844-849, 2014.

[4] M. Shabdiz, et al., “Designing expert system for detecting faults in cloud environment,” *Advances in Computer Science: an International Journal*, vol. 2, 2013.

[5] Pyke, “A Python Knowledge Engine,” 2018, [Online], Available: http://pyke.sourceforge.net/

[6] “VMware Knowledgebase,” 2018, [Online], Available: https://kb.vmware.com/s/

[7] “Microsoft Knowledgebase,” 2018, [Online], Available: http://kbupdate.info/

[8] X. Chen, et al., “Failure Analysis of Jobs in Compute Clouds: A Google Cluster Case Study,” *IEEE 25th International Symposium on Software Reliability Engineering*, Naples, pp. 167-177, 2014.

[9] Shetty J., et al., “Monitoring OpenStack Services Using Nagios,” in Shetty N., et al., (eds), *Emerging Research in Computing, Information, Communication and Applications*, Springer, New Delhi, 2015.

[10] J. Shetty and G. Shobha, “An ensemble of automatic algorithms for forecasting resource utilization in cloud,” *2016 Future Technologies Conference (FTC)*, San Francisco, CA, pp. 301-306, 2016.
[11] V. Persico, et al., “A Fuzzy Approach Based on Heterogeneous Metrics for Scaling Out Public Clouds,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, pp. 2117-2130, 2017.
[12] S. Zhai, et al., “Bayesian networks application inmulti-state system reliability analysis,” *Applied Mechanics and Materials*, vol. 347, pp. 2590-2595, 2013.
[13] “The Math forum,” 2018, [Online], Available: http://mathforum.org/library/drmath/view/65407.html
[14] Ahmad, et al., *A Textbook on Ordinary Differential Equations*, Springer International Publishing, 2014.
[15] M. Paun, et al., “HPC Application in Cloud Environment,” *Romanian Journal of Information Science and Technology*, vol. 18, pp. 109-125, 2015.
[16] L. B. Gomez, et al., “FTI: high performance fault tolerance interface for hybrid systems,” *IEEE/ACM SC*, 2011.
[17] M. S. Bouguerra, et al., “Improving the computing efficiency of HPC systems using a combination of proactive and preventive checkpointing,” *IEEE/ACM IPDPS*, 2013.
[18] Gainaru, et al., “Fault prediction under the microscope: a closer look into HPC systems,” *ACM/IEEE Supercomputing Conference (SC)*, 2012.
[19] https://docs.openstack.org/pike/
[20] J. Shetty, et al., “An Empirical Performance Evaluation of Docker Container, Openstack Virtual Machine and Bare Metal Server,” *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 7, pp. 205-213, 2017.
[21] V. C. Emeakaroha, et al., “Towards Knowledge Management in Self-adaptable Clouds,” *Proceedings of 6th World Congress on Services*, Miami, pp. 527-534, 2010.
[22] E. Schmieders, et al., “Combining SLA Prediction and Cross Layer Adaption for Preventing SLA Violations,” *Proceedings of 2nd Workshop on Software Services: Cloud Computing and Applications based on Software Services*, Timisoara, Romania, 2011.
[23] M. Maurer, et al., “Enacting SLAs in Clouds using Rules,” *Proceedings of the 17th International Conference on parallel processing, Bordeaux*, France, 2011.
[24] L. Arockiam and G. Francis E., “FTM- A middle layer Architecture for Fault Tolerance in Cloud Computing,” *International Journal of computer Applications*, *ICNICT*, pp. 12-16, 2012.
[25] B. Sharma, et al., “CloudPD: Problem Determination and Diagnosis in Shared Dynamic Clouds,” *Proceedings of 43rd Annual IFIP/IEEE International Conference on Dependable Systems and Networks*, Budapset, pp. 1-12, 2013.
[26] S. Frey, et al., “Fuzzy Controlled QoS for Scalable Cloud Services,” *Proceedings of fourth International Conference on Cloud Computing*, *GRIDs and Virtualization*, pp. 150-155, 2013.
[27] Y. Tan, et al., “PREPARE: Predictive Performance Anomaly Prevention for Virtualized Cloud Systems,” *32nd IEEE International Conference on Distributed Computing Systems*, pp. 285-294, 2012.
[28] S. Huang, et al., “Differentiated Failure Remediation with Action Selection for Resilient Computing,” *2015 IEEE 21st Pacific Rim International Symposium on Dependable Computing (PRDC)*, Zhangjiajie, China, pp. 199-208, 2015.

BIOGRAPHIES OF AUTHORS

Jyoti Shetty ia working as assistant professor at CSE department R V College of Engineering, Bengaluru, since 2009. She has received her master’s degree from form VTU and persuing her Ph.D (CSE) from VTU. Her research areas of interest are Cloud Computing and Datamining

B.Sathish Babu received his Ph.D degree in Electrical Communication Engineering from Indian Institute of Science, Bangalore, India. His research interest includes Cognitive agents based control solutions for Networks, Grid Computing, Cloud Computing Scheduling and Security Issues, Context -aware Trust issues in Ubiquitous Computing , High performance computing, Privacy issues in WSN, and Opportunistic Computing.

Shobha G. is the head and Prof., who CSE is associated with R.V. College of Engineering, Bangalore, since 1995. She has received her master’s degree from BITS, Pilani and Ph.D (CSE) from Mangalore University. Her research areas of interest are database management systems, data mining, data warehousing, image processing and information and network security.