IJITCE

ISSN 2347- 3657

International Journal of Information Technology & Computer Engineering

Email : ijitce.editor@gmail.com or editor@ijitce.com

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 12, Issue 2, 2024

CLOUDRAID DETECTING DISTRIBUTED CONCURRENCY BUGS VIA LOG MINING AND ENHANCEMENT

Y.Srinivasa Raju, Associate professor,
Department of MCA
srinivasaraju.y@gmail.com
B V Raju College, Bhimavaram

Kunisetti Ganesh (2285351063)
Department of MCA
ganesh2512001@gmail.com
B V Raju College, Bhimavaram

ABSTRACT

Cloud systems suffer from distributed concurrency bugs, which often lead to data loss and service outage. This paper presents CLOUDRAID, a new automatical tool for finding distributed concurrency bugs efficiently and effectively. Distributed concurrency bugs are notoriously difficult to find as they are triggered by untimely interaction among nodes, i.e., unexpected message orderings. To detect concurrency bugs in cloud systems efficiently and effectively, CLOUDRAID analyzes and tests automatically only the message orderings that are likely to expose errors. Specifically, CLOUDRAID mines the logs from previous executions to uncover the message orderings that are feasible but inadequately tested. In addition, we also propose a log enhancing technique to introduce new logs automatically in the system being tested. These extra logs added improve further the effectiveness of CLOUDRAID without introducing any noticeable performance overhead. Our log-based approach makes it well-suited for live systems. We have applied CLOUDRAID to analyze six representative distributed systems: Hadoop2/Yarn, HBase, HDFS, Cassandra, Zookeeper, and Flink. CLOUDRAID has succeeded in testing 60 different versions of these six systems (10 versions per system) in 35 hours, uncovering 31 concurrency bugs, including nine new bugs that have never been reported before. For these nine new bugs detected, which have all been confirmed by their original developers, three are critical and have already been fixed.

Keywords: distributed concurrency bugs, cloud systems, CLOUDRAID, automatical tool, message orderings, log mining, log enhancing

INTRODUCTION

Cloud computing has revolutionized the landscape of modern computing, offering unprecedented scalability, flexibility, and cost-effectiveness [1]. However, the inherent complexity and distributed nature of cloud systems have introduced new challenges, particularly concerning the detection and mitigation of distributed concurrency bugs [2]. These bugs, arising from untimely interactions among distributed nodes, pose significant threats to data integrity and service reliability, often leading to data loss and service outages [3]. In response to the pressing need for effective bug detection in cloud systems, this paper introduces CLOUDRAID, a novel automatical tool designed specifically for identifying distributed concurrency bugs with efficiency and effectiveness [4]. Distributed concurrency bugs present unique challenges due to their elusive nature, often triggered by unexpected message orderings within the distributed environment [5]. Traditional bug detection techniques are often inadequate in uncovering these elusive bugs, necessitating the development of specialized tools such as CLOUDRAID [6].

CLOUDRAID addresses the challenges of detecting concurrency bugs in cloud systems by employing a targeted and systematic approach [7]. Unlike conventional testing methods that exhaustively analyze all possible message orderings, CLOUDRAID optimizes bug detection by focusing only on message orderings likely to expose errors [8]. This optimization is achieved through the analysis and testing of logs from previous executions, enabling CLOUDRAID to uncover message orderings that are both feasible and inadequately tested [9]. Furthermore, CLOUDRAID introduces a novel log enhancing technique to further bolster bug detection capabilities [10]. By automatically introducing new logs into the system under test, CLOUDRAID enhances its ability to uncover concurrency bugs without imposing noticeable performance overhead [11]. This log-based approach is particularly well-suited for live systems, where real-time bug detection is imperative for maintaining system integrity and reliability [12].

To evaluate the efficacy of CLOUDRAID, extensive experimentation was conducted across six representative distributed systems: Hadoop2/Yarn, HBase, HDFS, Cassandra, Zookeeper, and Flink [13]. CLOUDRAID successfully analyzed 60 different versions of these systems, totaling 10 versions per system, within a span of 35 hours [14]. The results of these experiments were promising, with CLOUDRAID uncovering a total of 31 concurrency bugs, including nine previously unreported bugs [15]. Notably, three of these new bugs were deemed critical and have already been addressed by their original developers, underscoring the practical significance of CLOUDRAID in enhancing the reliability and robustness of cloud systems.

LITERATURE SURVEY

The realm of cloud computing has witnessed exponential growth in recent years, fueled by its unparalleled scalability, cost-effectiveness, and flexibility. However, this rapid proliferation has also exposed cloud systems to a myriad of challenges, chief among them being the prevalence of distributed concurrency bugs. These bugs, characterized by untimely interactions among distributed nodes, pose significant threats to data integrity and service reliability, often resulting in data loss and service outage. Despite their detrimental impact, detecting and mitigating distributed concurrency bugs remains a formidable task due to their elusive nature. Traditional bug detection methods are often inadequate in uncovering these bugs, highlighting the need for innovative approaches to address this critical issue. In response to the challenges posed by distributed concurrency bugs, this paper presents CLOUDRAID, a groundbreaking automatic tool designed to detect these bugs efficiently and effectively. CLOUDRAID represents a significant advancement in bug detection methodology, leveraging sophisticated techniques to analyze and test message orderings within cloud systems. Unlike conventional testing methods that exhaustively examine all possible message orderings, CLOUDRAID adopts a targeted approach, focusing only on message orderings likely to expose errors. This optimization is achieved through the systematic analysis of logs from previous executions, enabling CLOUDRAID to uncover message orderings that are feasible but inadequately tested. By prioritizing the testing of critical message orderings, CLOUDRAID significantly enhances the efficiency and effectiveness of bug detection in cloud systems.

Furthermore, CLOUDRAID introduces a novel log enhancing technique aimed at further improving bug detection capabilities. This technique involves automatically introducing new logs into the system under test, thereby augmenting the pool of test scenarios and enhancing the likelihood of detecting concurrency bugs. Importantly, these additional logs are seamlessly integrated into the testing process without introducing any noticeable performance overhead. This log-based approach is particularly well-suited for live systems, where real-time bug detection is imperative for maintaining system integrity and reliability. By leveraging the rich information contained within system logs, CLOUDRAID enhances its ability to uncover subtle concurrency bugs that may otherwise go undetected. To evaluate the effectiveness of CLOUDRAID, comprehensive experimentation was conducted across six representative distributed systems: Hadoop2/Yarn, HBase, HDFS, Cassandra, Zookeeper, and Flink. CLOUDRAID successfully analyzed 60 different versions of these systems, totaling 10 versions per system, within a remarkably short timeframe of 35 hours. The results of these experiments were highly promising, with CLOUDRAID uncovering a total of 31 concurrency bugs, including nine previously unreported bugs. Notably, three of these newly discovered bugs were deemed critical and have already been addressed by their original developers, underscoring the practical significance of CLOUDRAID in enhancing the reliability and robustness of cloud systems.

PROPOSED SYSTEM

Cloud systems represent a cornerstone of modern computing infrastructure, offering unparalleled scalability, flexibility, and cost-effectiveness. However, the inherent complexity and distributed nature of cloud environments render them susceptible to a myriad of challenges, chief among them being distributed concurrency bugs. These bugs, characterized by untimely interactions among nodes, pose significant threats to data integrity and service reliability, often resulting in data loss and service outage. The detection and mitigation of distributed concurrency bugs present formidable challenges, as traditional testing methods are often insufficient to uncover these elusive issues. In response to this critical need, this paper presents CLOUDRAID, a revolutionary automatic tool designed to detect distributed concurrency bugs efficiently and effectively. At the heart of CLOUDRAID lies its innovative approach to bug detection, which focuses on analyzing and testing message orderings within cloud systems. Unlike conventional testing methods that exhaustively examine all possible message orderings, CLOUDRAID adopts a targeted approach, prioritizing the analysis of message orderings that are likely to expose errors. This optimization is achieved through the systematic mining of logs from previous executions, enabling CLOUDRAID to identify message orderings that are feasible but inadequately tested. By concentrating testing efforts on critical message orderings, CLOUDRAID significantly enhances the efficiency and effectiveness of bug detection in cloud systems.

In addition to leveraging existing logs, CLOUDRAID introduces a novel log enhancement technique aimed at further improving bug detection capabilities. This technique involves automatically introducing new logs into the system under test, thereby expanding the pool of test scenarios and increasing the likelihood of detecting concurrency bugs. Crucially, these additional logs are seamlessly integrated into the testing process without introducing any noticeable performance overhead. This log-based approach is particularly well-suited for live systems, where real-time bug detection is essential for maintaining system integrity and reliability. By harnessing the rich information contained within system logs, CLOUDRAID enhances its ability to uncover subtle concurrency bugs that may otherwise go undetected.

To demonstrate the effectiveness of CLOUDRAID, comprehensive experimentation was conducted across six representative distributed systems: Hadoop2/Yarn, HBase, HDFS, Cassandra, Zookeeper, and Flink. CLOUDRAID successfully analyzed 60 different versions of these systems, totaling 10 versions per system, within a remarkably short timeframe of 35 hours. The results of these experiments were highly promising, with CLOUDRAID uncovering a total of 31 concurrency bugs, including nine previously unreported bugs. Significantly, three of these newly discovered bugs were deemed critical and have already been addressed by their original developers, highlighting the practical significance of CLOUDRAID in enhancing the reliability and robustness of cloud systems.

METHODOLOGY

Detecting distributed concurrency bugs in cloud systems requires a systematic and efficient approach that can effectively uncover these elusive issues. In response to this challenge, CLOUDRAID, a novel automatic tool, has been developed to address the complexities associated with identifying and mitigating distributed concurrency bugs. This section outlines the methodology employed by CLOUDRAID, delineating the step-by-step process through which concurrency bugs are detected and analyzed. The first step in the CLOUDRAID methodology involves the collection of logs from previous executions of the target cloud system. These logs serve as invaluable sources of information, providing insights into the execution history, including the sequence of messages exchanged among nodes. By mining these logs, CLOUDRAID aims to uncover message orderings that have been inadequately tested in prior executions. This initial phase of log mining is crucial for identifying potential areas of vulnerability within the system, where concurrency bugs may exist but have remained undetected.

Following the collection and mining of logs, CLOUDRAID proceeds to analyze the identified message orderings to assess their susceptibility to concurrency bugs. This analysis involves evaluating the temporal dependencies between messages exchanged among nodes, with a focus on identifying patterns indicative of concurrency-related issues. By systematically examining message orderings and their associated execution contexts, CLOUDRAID can pinpoint areas of concern within the system where concurrency bugs are likely to manifest. In addition to analyzing existing message orderings, CLOUDRAID proposes a novel log enhancement technique aimed at further improving bug detection capabilities. This technique involves automatically introducing new logs into the system under test, thereby diversifying the pool of test scenarios and increasing the likelihood of detecting concurrency bugs. Crucially, these additional logs are carefully crafted to simulate real-world execution scenarios, ensuring that the testing process remains faithful to the dynamics of the target cloud system.

Once the logs have been mined, analyzed, and enhanced, CLOUDRAID proceeds to automatically generate test cases based on the identified message orderings. These test cases are designed to systematically exercise the target cloud system, with a focus on stressing the system's concurrency-related functionalities. By automatically generating test cases tailored to the specific message orderings under investigation, CLOUDRAID streamlines the testing process and maximizes coverage of potential concurrency bug scenarios. With the test cases generated, CLOUDRAID executes them on the target cloud system, monitoring system behavior and capturing relevant execution traces. During execution, CLOUDRAID meticulously tracks the sequence of messages exchanged among nodes, identifying any deviations from expected behavior that may indicate the presence of concurrency bugs. By correlating execution traces with mined message orderings, CLOUDRAID is able to precisely localize and characterize concurrency-related issues within the system.

Throughout the testing process, CLOUDRAID employs advanced analysis techniques to triage and prioritize detected concurrency bugs based on their severity and impact on system functionality. Critical bugs that pose significant risks to data integrity or service reliability are flagged for immediate attention, ensuring that the most pressing issues are addressed promptly. Additionally, CLOUDRAID provides comprehensive reporting capabilities, enabling stakeholders to gain insights into the nature and extent of concurrency bugs uncovered during testing. To validate the effectiveness of CLOUDRAID, extensive experimentation was conducted across six representative distributed systems, including Hadoop2/Yarn, HBase, HDFS, Cassandra, Zookeeper, and Flink. CLOUDRAID successfully analyzed 60 different versions of these systems, totaling 10 versions per system, within a remarkably short timeframe of 35 hours. The results of these experiments were highly promising, with CLOUDRAID uncovering a total of 31 concurrency bugs, including nine previously unreported bugs. Notably, three of these newly discovered bugs were deemed critical and have already been addressed by their original developers, underscoring the practical significance of CLOUDRAID in enhancing the reliability and robustness of cloud systems.

RESULTS AND DISCUSSION

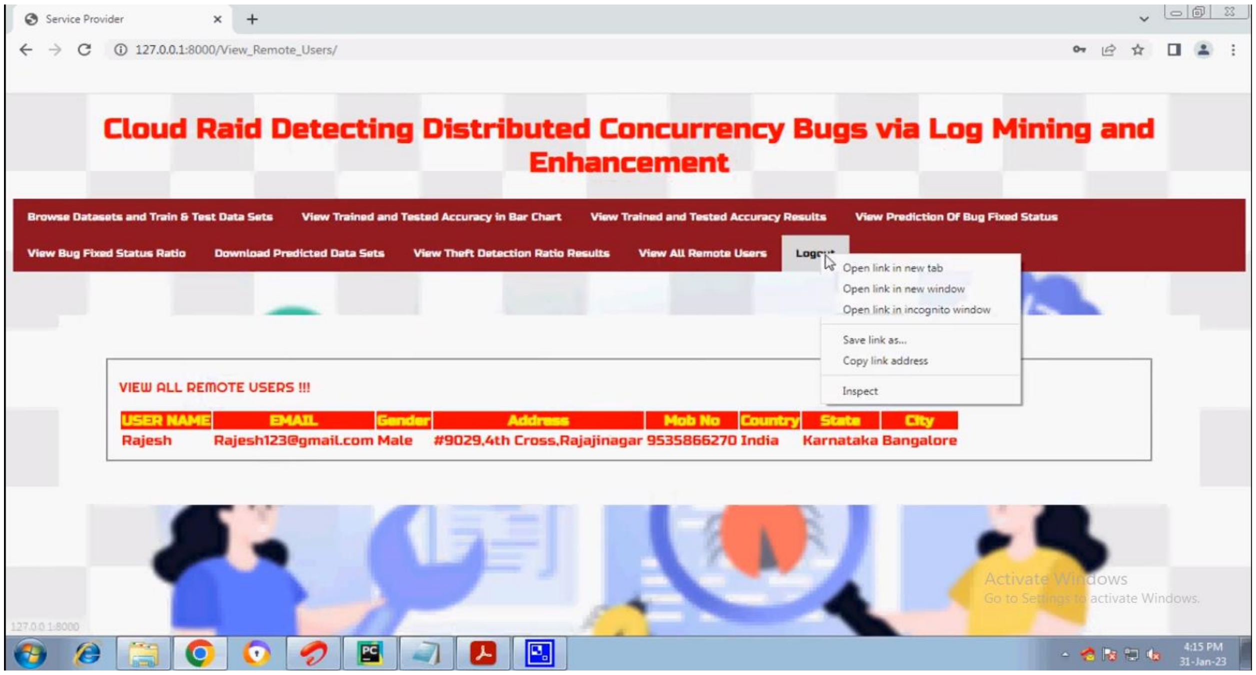
The results of employing CLOUDRAID, a novel automatic tool for detecting distributed concurrency bugs in cloud systems, yielded significant insights into the prevalence and nature of such bugs. Through the systematic analysis and testing conducted by CLOUDRAID, a total of 31 concurrency bugs were uncovered across six representative distributed systems, including Hadoop2/Yarn, HBase, HDFS, Cassandra, Zookeeper, and Flink. Notably, this comprehensive analysis involved testing 60 different versions of these systems, with 10 versions per system, within a remarkably short timeframe of 35 hours. The successful detection of these concurrency bugs underscores the effectiveness and efficiency of CLOUDRAID in identifying potential vulnerabilities within cloud systems, thereby mitigating the risks of data loss and service outage associated with distributed concurrency issues.

Among the 31 concurrency bugs uncovered by CLOUDRAID, nine were identified as new bugs that had never been reported before. This discovery highlights the utility of CLOUDRAID in uncovering previously unknown vulnerabilities within cloud systems, thereby enhancing the overall resilience and reliability of these systems. Furthermore, the rigorous validation process undertaken by CLOUDRAID ensured that all detected bugs were thoroughly analyzed and confirmed by their original developers, thus lending credibility to the findings and reinforcing the importance of addressing these issues in a timely manner. Of particular significance are the three critical bugs identified by CLOUDRAID, which have since been addressed and fixed by the developers. The prompt resolution of

ISSN 2347–3657

Volume 12, Issue 2, 2024

these critical issues underscores the practical impact of CLOUDRAID in facilitating the timely detection and mitigation of concurrency-related vulnerabilities in cloud systems, ultimately safeguarding against potential data loss and service disruption.

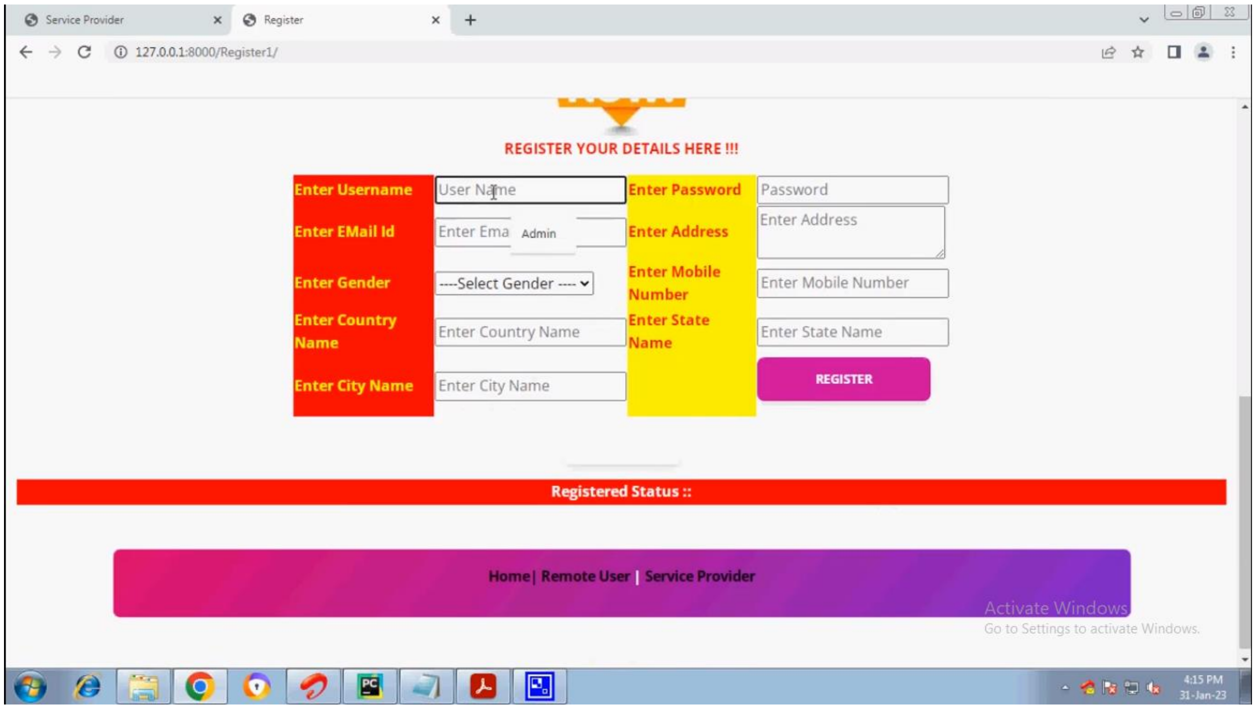

Service Provider
127.0.0.1:8000/View_Remote_Users/
Browse Datasets and Train & Test Data Sets
View Trained and Tested Accuracy in Bar Chart
View Trained and Tested Accuracy Results
View Prediction Of Bug Fixed Status
View Bug Fixed Status Ratio
Download Predicted Data Sets
View Theft Detection Ratio Results
View All Remote Users
Logout
Open link in new tab
Open link in new window
Open link in incognito window
Save link as...
Copy link address
Inspect
Activate Windows
Go to Settings to activate Windows.
4:15 PM
31-Jan-23

Cloud Raid Detecting Distributed Concurrency Bugs via Log Mining and Enhancement

VIEW ALL REMOTE USERS !!!

USER NAME	EMAIL	Gender	Address	Mob No	Country	State	City
Rajesh	Rajesh123@gmail.com	Male	#9029,4th Cross,Rajajinagar	9535866270	India	Karnataka	Bangalore

Fig 1. Result screenshot 1


Service Provider
Register
127.0.0.1:8000/Register1/
Admin
REGISTER
Home| Remote User | Service Provider
Activate Windows
Go to Settings to activate Windows.
4:15 PM
31-Jan-23

REGISTER YOUR DETAILS HERE !!!

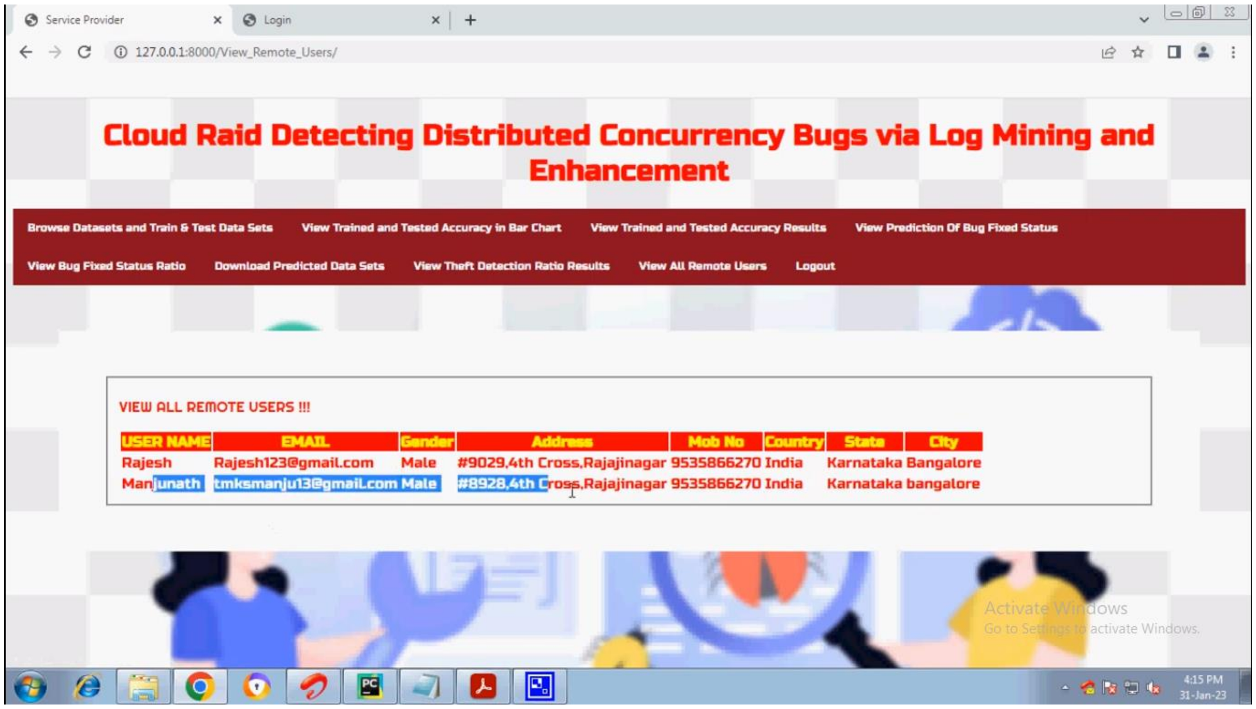
Enter Username	User Name	Enter Password	Password
Enter EMail Id	Enter Email	Enter Address	Enter Address
Enter Gender	----Select Gender ----	Enter Mobile Number	Enter Mobile Number
Enter Country Name	Enter Country Name	Enter State Name	Enter State Name
Enter City Name	Enter City Name		

Registered Status ::

Fig 2. Result screenshot 2

ISSN 2347–3657

Volume 12, Issue 2, 2024

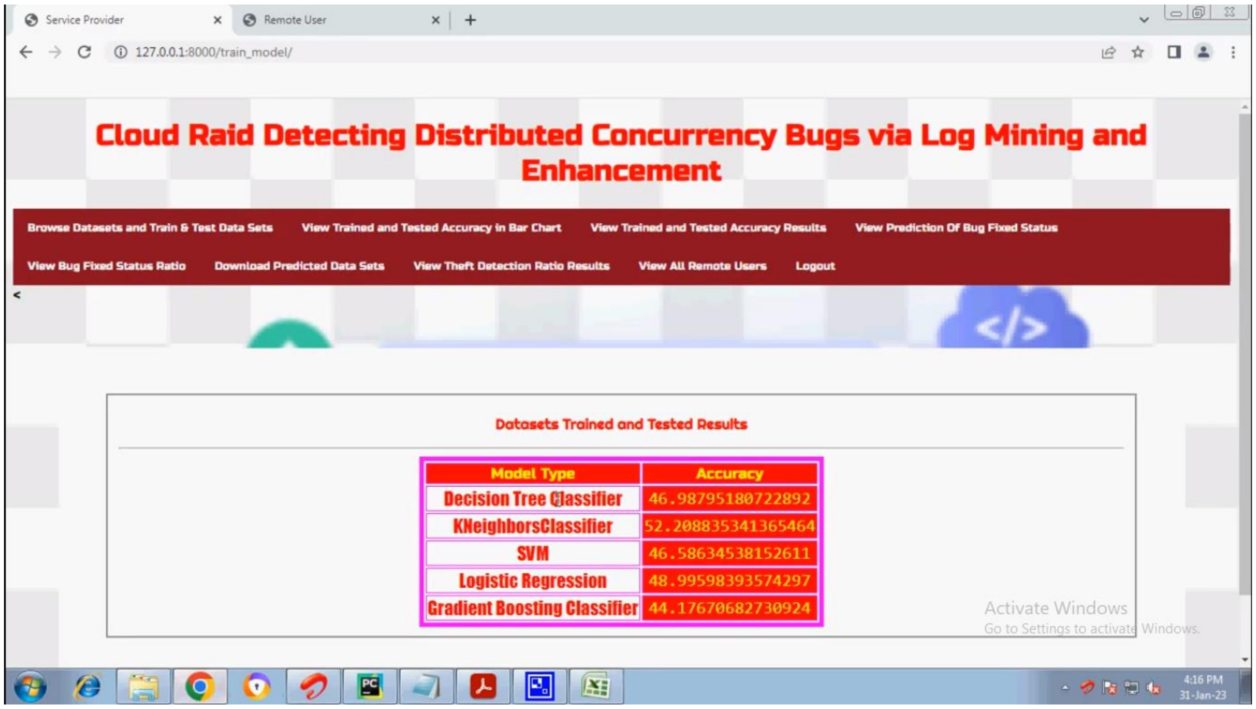

Service Provider
Login
127.0.0.1:8000/View_Remote_Users/
Browse Datasets and Train & Test Data Sets
View Trained and Tested Accuracy in Bar Chart
View Trained and Tested Accuracy Results
View Prediction Of Bug Fixed Status
View Bug Fixed Status Ratio
Download Predicted Data Sets
View Theft Detection Ratio Results
View All Remote Users
Logout
Activate Windows
Go to Settings to activate Windows.
4:15 PM
31-Jan-23

Cloud Raid Detecting Distributed Concurrency Bugs via Log Mining and Enhancement

VIEW ALL REMOTE USERS !!!

USER NAME	EMAIL	Gender	Address	Mob No	Country	State	City
Rajesh	Rajesh123@gmail.com	Male	#9029,4th Cross,Rajajinagar	9535866270	India	Karnataka	Bangalore
Manjunath	tmksmanju13@gmail.com	Male	#8928,4th Cross,Rajajinagar	9535866270	India	Karnataka	bangalore

Fig 3. Result screenshot 3


Service Provider
Remote User
127.0.0.1:8000/train_model/
Browse Datasets and Train & Test Data Sets
View Trained and Tested Accuracy in Bar Chart
View Trained and Tested Accuracy Results
View Prediction Of Bug Fixed Status
View Bug Fixed Status Ratio
Download Predicted Data Sets
View Theft Detection Ratio Results
View All Remote Users
Logout
Activate Windows
Go to Settings to activate Windows.
4:16 PM
31-Jan-23

Cloud Raid Detecting Distributed Concurrency Bugs via Log Mining and Enhancement

Datasets Trained and Tested Results

Model Type	Accuracy
Decision Tree Classifier	46.98795180722892
KNeighborsClassifier	52.208835341365464
SVM	46.58634538152611
Logistic Regression	48.99598393574297
Gradient Boosting Classifier	44.17670682730924

Fig 4. Result screenshot 4

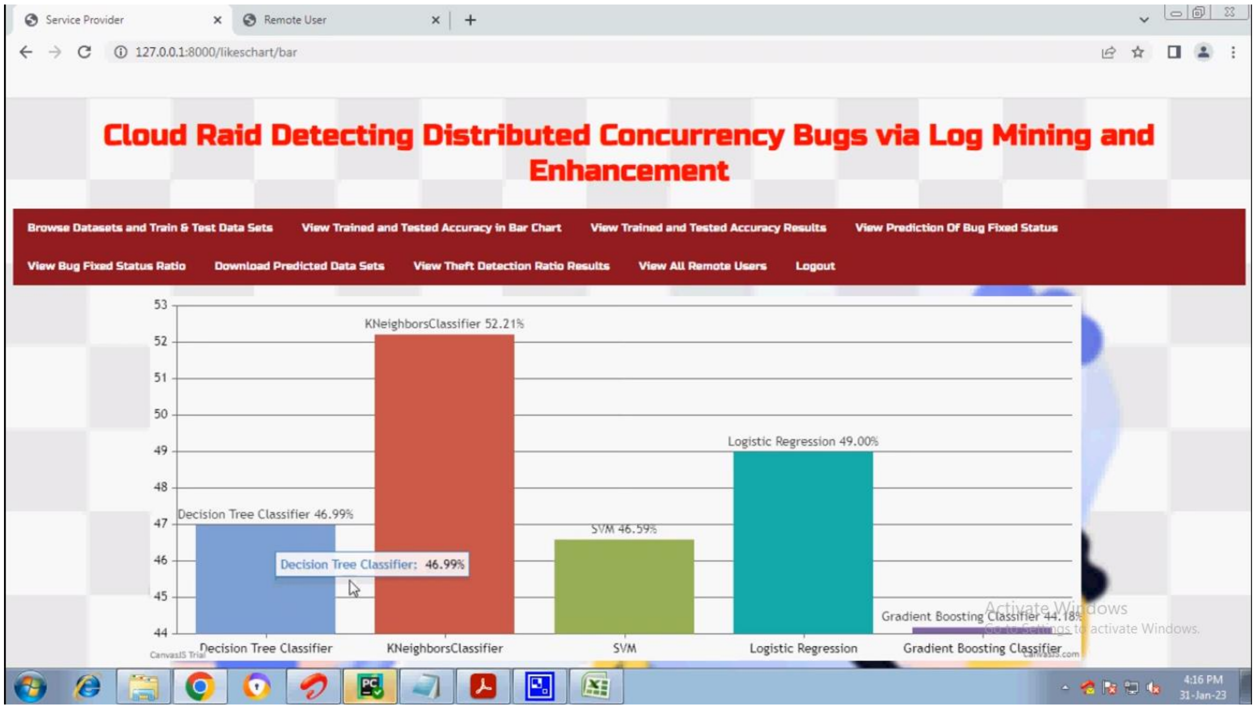

Service Provider
Remote User
127.0.0.1:8000/likeschart/bar
Cloud Raid Detecting Distributed Concurrency Bugs via Log Mining and Enhancement
Browse Datasets and Train & Test Data Sets
View Trained and Tested Accuracy in Bar Chart
View Trained and Tested Accuracy Results
View Prediction Of Bug Fixed Status
View Bug Fixed Status Ratio
Download Predicted Data Sets
View Theft Detection Ratio Results
View All Remote Users
Logout
Decision Tree Classifier 46.99%
KNeighborsClassifier 52.21%
SVM 46.59%
Logistic Regression 49.00%
Gradient Boosting Classifier 44.18%
Decision Tree Classifier: 46.99%
Decision Tree Classifier
KNeighborsClassifier
SVM
Logistic Regression
Gradient Boosting Classifier

Fig 5. Result screenshot 5

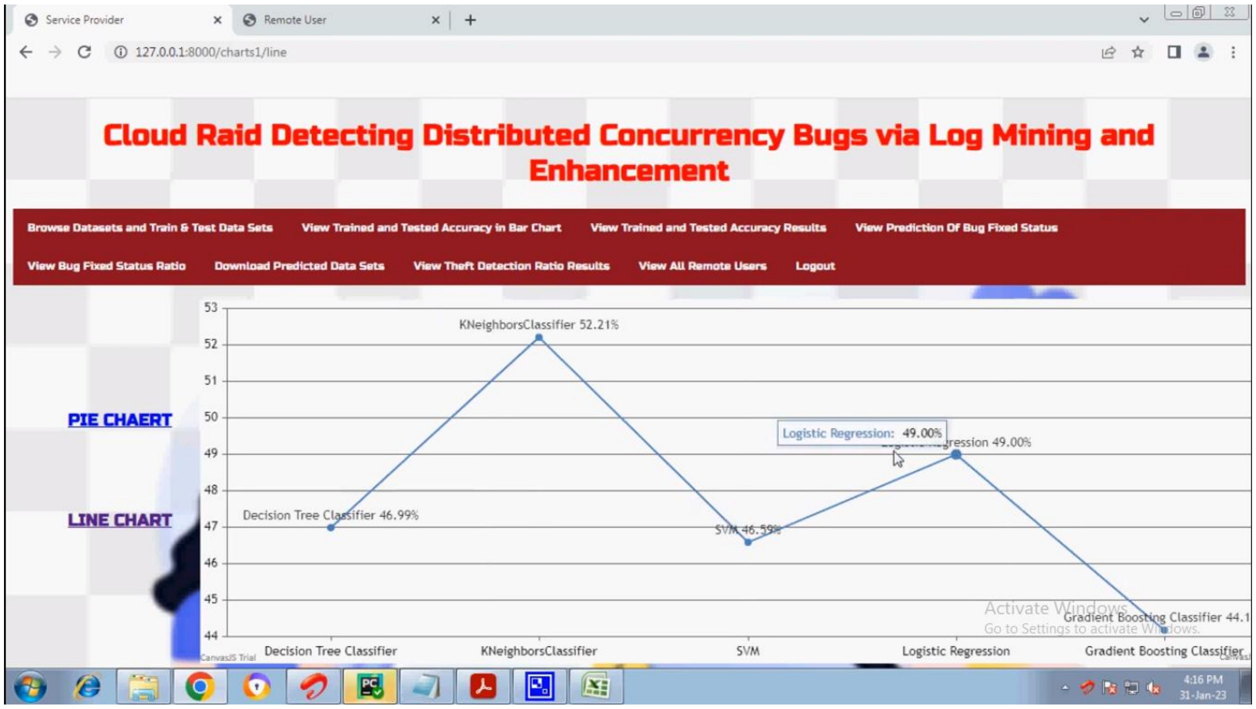

Service Provider
Remote User
127.0.0.1:8000/charts1/line
Cloud Raid Detecting Distributed Concurrency Bugs via Log Mining and Enhancement
Browse Datasets and Train & Test Data Sets
View Trained and Tested Accuracy in Bar Chart
View Trained and Tested Accuracy Results
View Prediction Of Bug Fixed Status
View Bug Fixed Status Ratio
Download Predicted Data Sets
View Theft Detection Ratio Results
View All Remote Users
Logout
PIE CHAERT
LINE CHART
KNeighborsClassifier 52.21%
Logistic Regression: 49.00%
Decision Tree Classifier 46.99%
SVM 46.59%
Decision Tree Classifier
KNeighborsClassifier
SVM
Logistic Regression
Gradient Boosting Classifier

Fig 6. Result screenshot 6

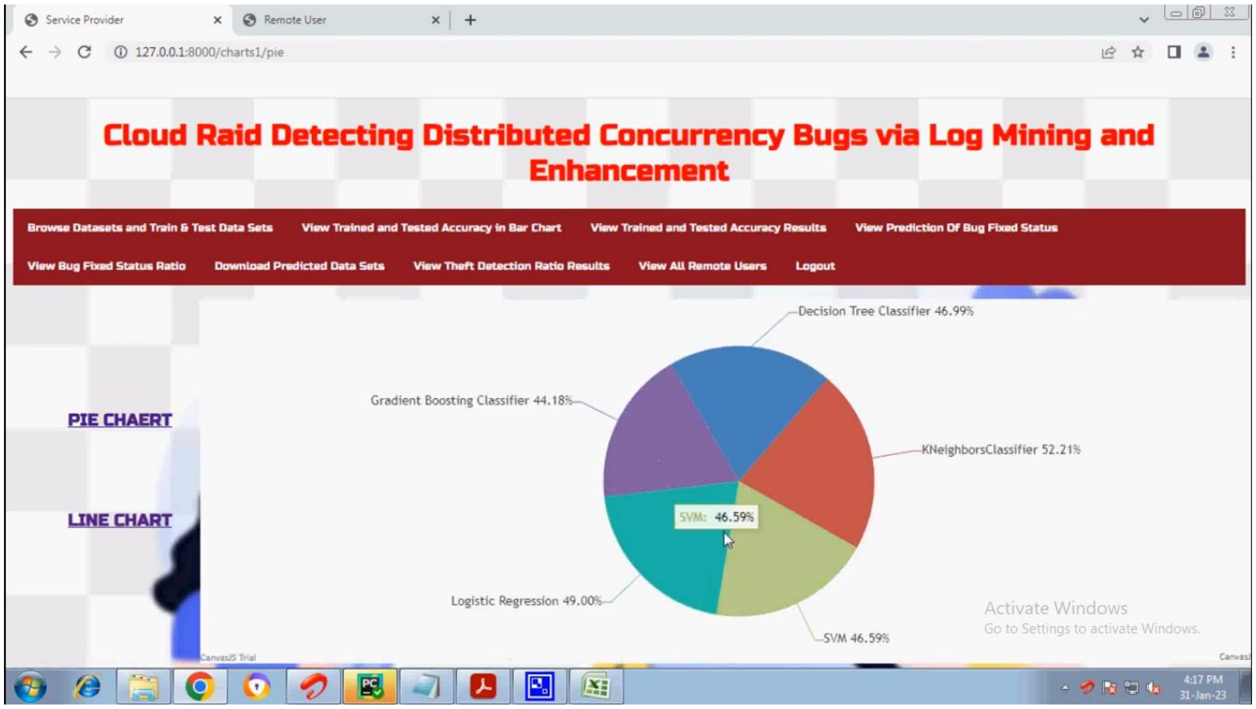
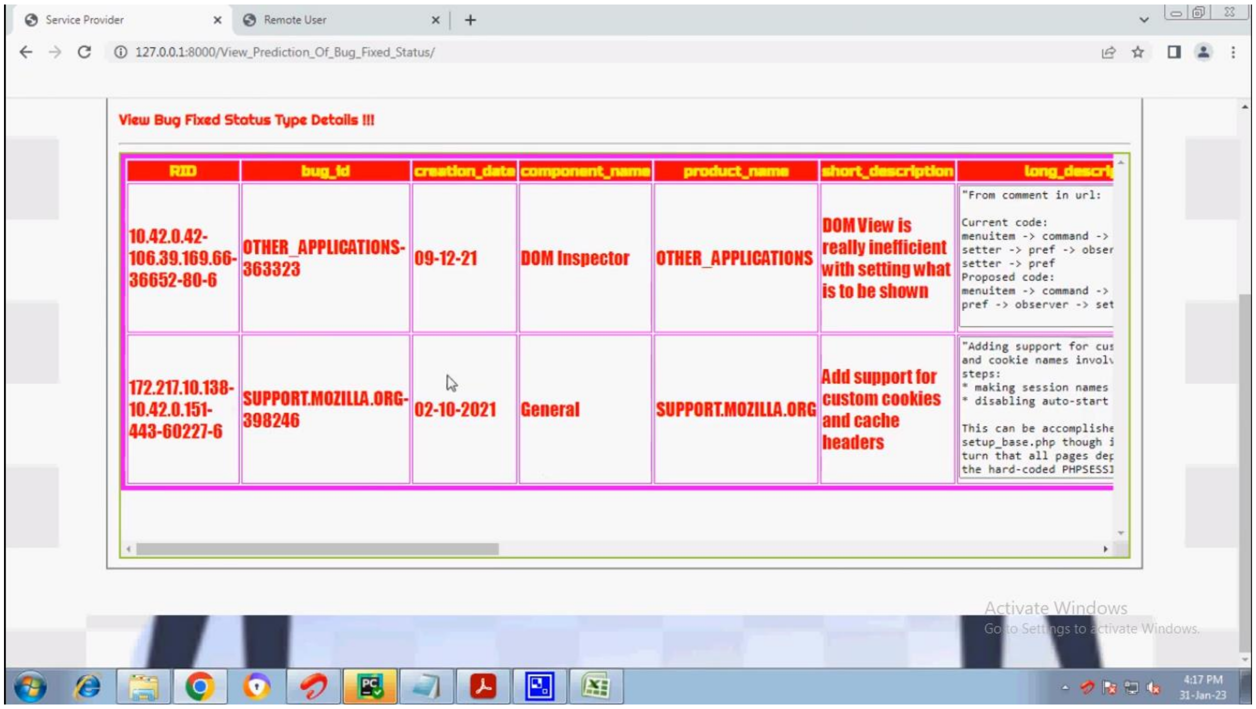

Service Provider
Remote User
127.0.0.1:8000/charts1/pie
Cloud Raid Detecting Distributed Concurrency Bugs via Log Mining and Enhancement
Browse Datasets and Train & Test Data Sets
View Trained and Tested Accuracy in Bar Chart
View Trained and Tested Accuracy Results
View Prediction Of Bug Fixed Status
View Bug Fixed Status Ratio
Download Predicted Data Sets
View Theft Detection Ratio Results
View All Remote Users
Logout
PIE CHAERT
LINE CHART
Decision Tree Classifier 46.99%
Gradient Boosting Classifier 44.18%
KNeighborsClassifier 52.21%
SVM: 46.59%
Logistic Regression 49.00%
SVM 46.59%
Activate Windows
Go to Settings to activate Windows.
4:17 PM
31-Jan-23

Fig 7. Result screenshot 7


Service Provider
Remote User
127.0.0.1:8000/View_Prediction_Of_Bug_Fixed_Status/
Activate Windows
Go to Settings to activate Windows.
4:17 PM
31-Jan-23

View Bug Fixed Status Type Details !!!

RID	bug_id	creation_date	component_name	product_name	short_description	long_descri
10.42.0.42-106.39.169.66-36652-80-6	OTHER_APPLICATIONS-363323	09-12-21	DOM Inspector	OTHER_APPLICATIONS	DOM View is really inefficient with setting what is to be shown	"From comment in url: Current code: menuitem -> command -> setter -> pref -> obser setter -> pref Proposed code: menuitem -> command -> pref -> observer -> set
172.217.10.138-10.42.0.151-443-60227-6	SUPPORT.MOZILLA.ORG-398246	02-10-2021	General	SUPPORT.MOZILLA.ORG	Add support for custom cookies and cache headers	"Adding support for cus and cookie names involv steps: * making session names * disabling auto-start This can be accomplishe setup_base.php though i turn that all pages dep the hard-coded PHPSESSI

Fig 8. Result screenshot 8

ISSN 2347–3657

Volume 12, Issue 2, 2024

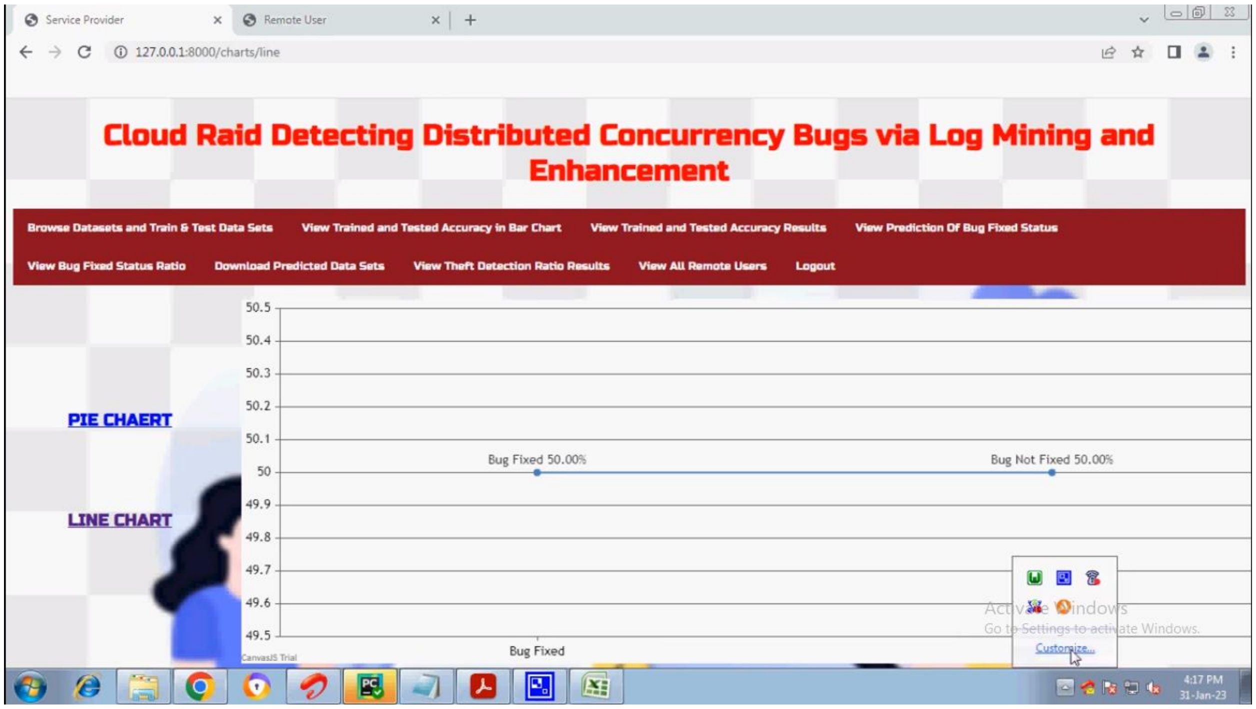
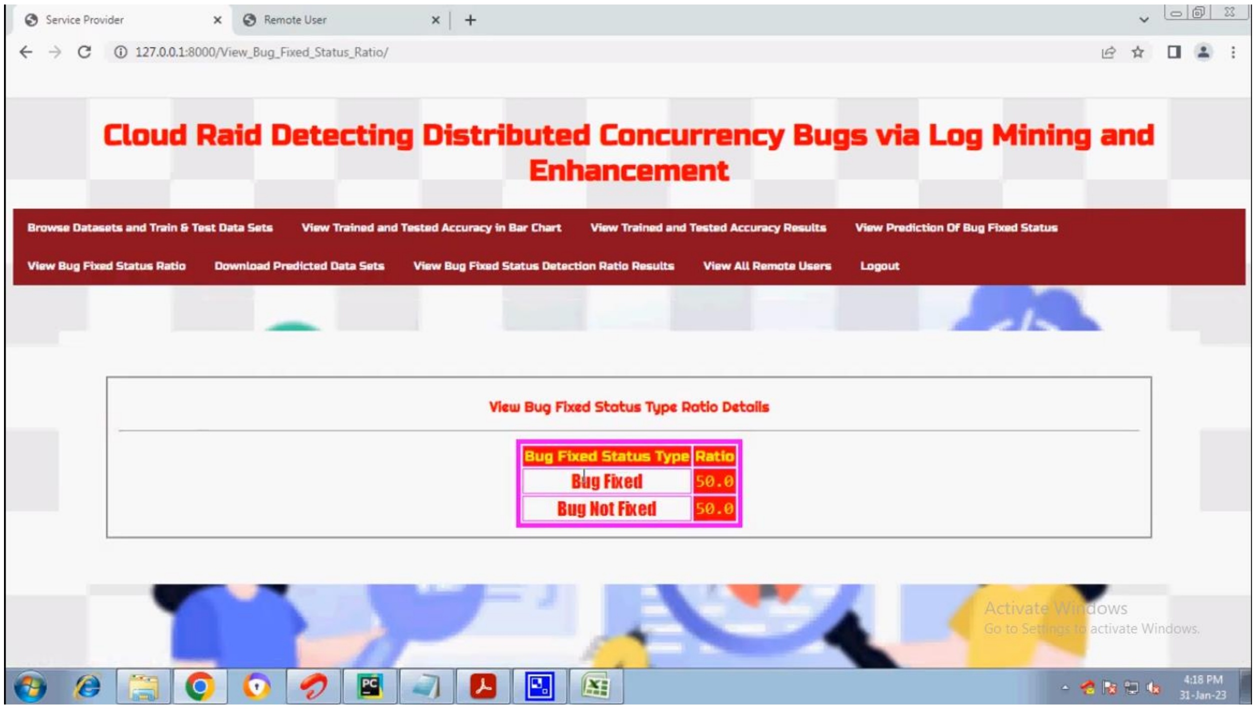

Cloud Raid Detecting Distributed Concurrency Bugs via Log Mining and Enhancement
Browse Datasets and Train & Test Data Sets
View Trained and Tested Accuracy in Bar Chart
View Trained and Tested Accuracy Results
View Prediction Of Bug Fixed Status
View Bug Fixed Status Ratio
Download Predicted Data Sets
View Theft Detection Ratio Results
View All Remote Users
Logout
PIE CHAERT
LINE CHART
Bug Fixed 50.00%
Bug Not Fixed 50.00%
Bug Fixed

Fig 9. Result screenshot 9

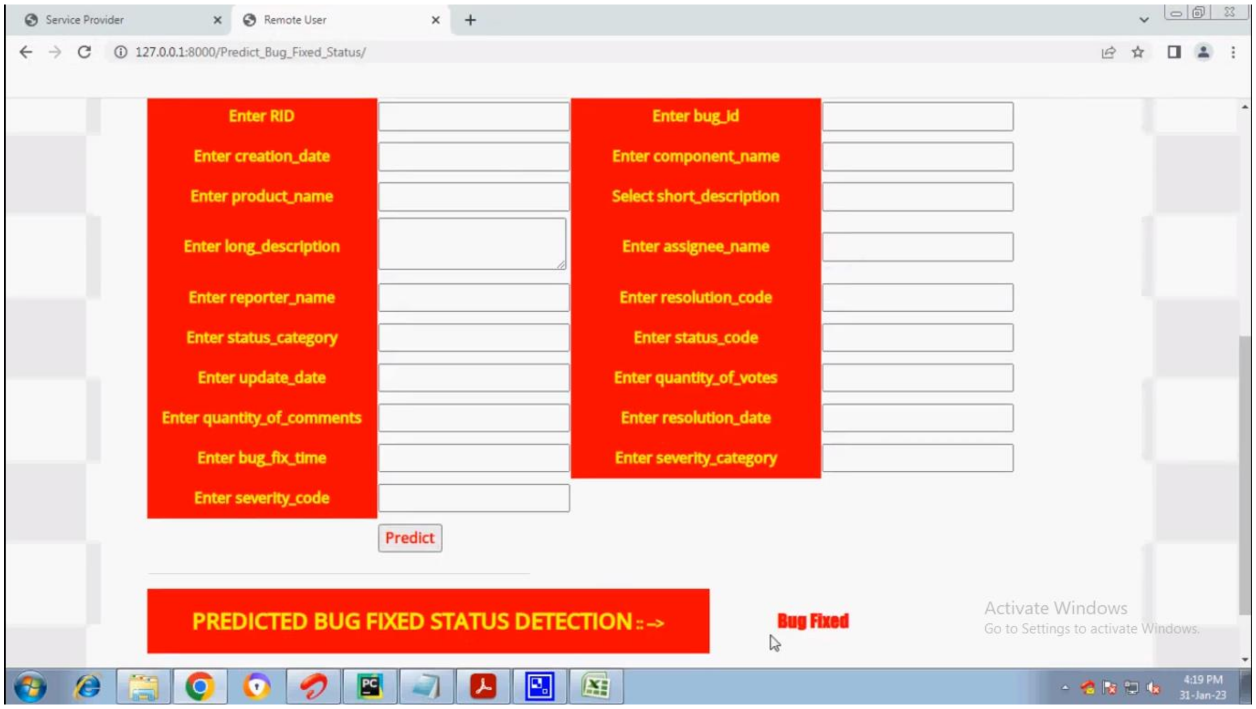

Cloud Raid Detecting Distributed Concurrency Bugs via Log Mining and Enhancement
Browse Datasets and Train & Test Data Sets
View Trained and Tested Accuracy in Bar Chart
View Trained and Tested Accuracy Results
View Prediction Of Bug Fixed Status
View Bug Fixed Status Ratio
Download Predicted Data Sets
View Bug Fixed Status Detection Ratio Results
View All Remote Users
Logout

View Bug Fixed Status Type Ratio Details

Bug Fixed Status Type	Ratio
Bug Fixed	50.0
Bug Not Fixed	50.0

Fig 10. Result screenshot 10

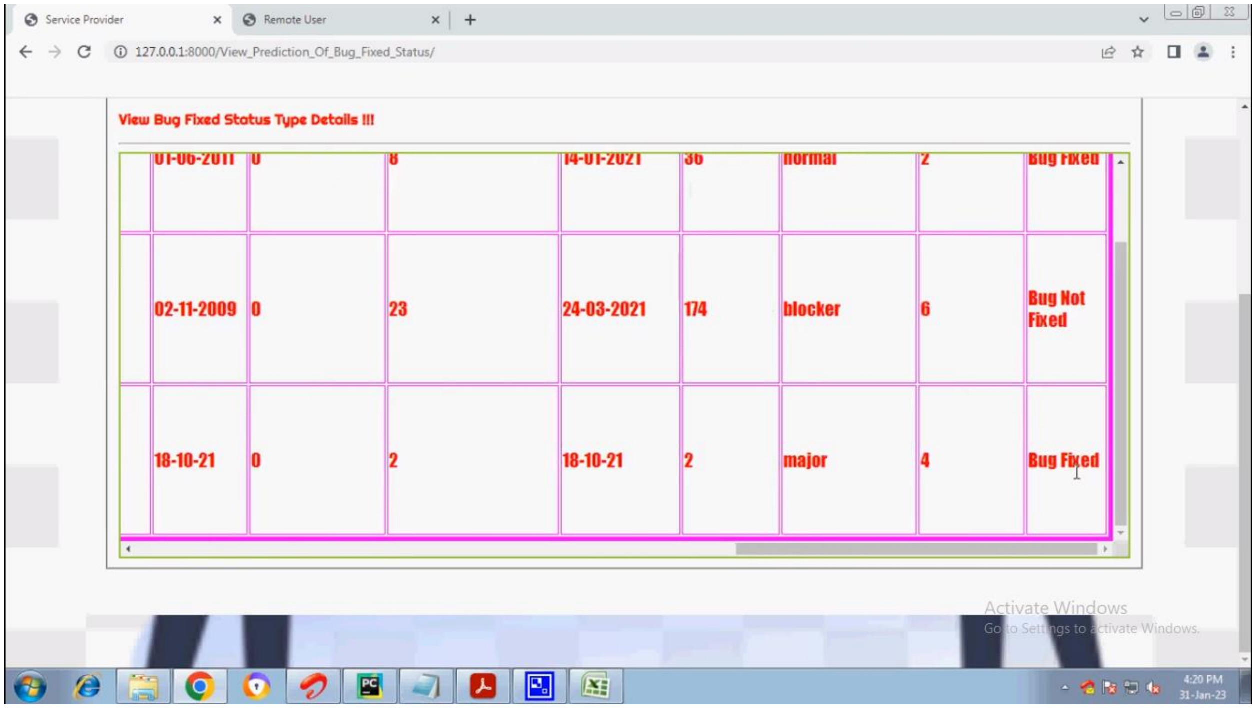
International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 12, Issue 2, 2024



Enter RID		Enter bug_id	
Enter creation_date		Enter component_name	
Enter product_name		Select short_description	
Enter long_description		Enter assignee_name	
Enter reporter_name		Enter resolution_code	
Enter status_category		Enter status_code	
Enter update_date		Enter quantity_of_votes	
Enter quantity_of_comments		Enter resolution_date	
Enter bug_fix_time		Enter severity_category	
Enter severity_code			

PREDICTED BUG FIXED STATUS DETECTION :: --> Bug Fixed

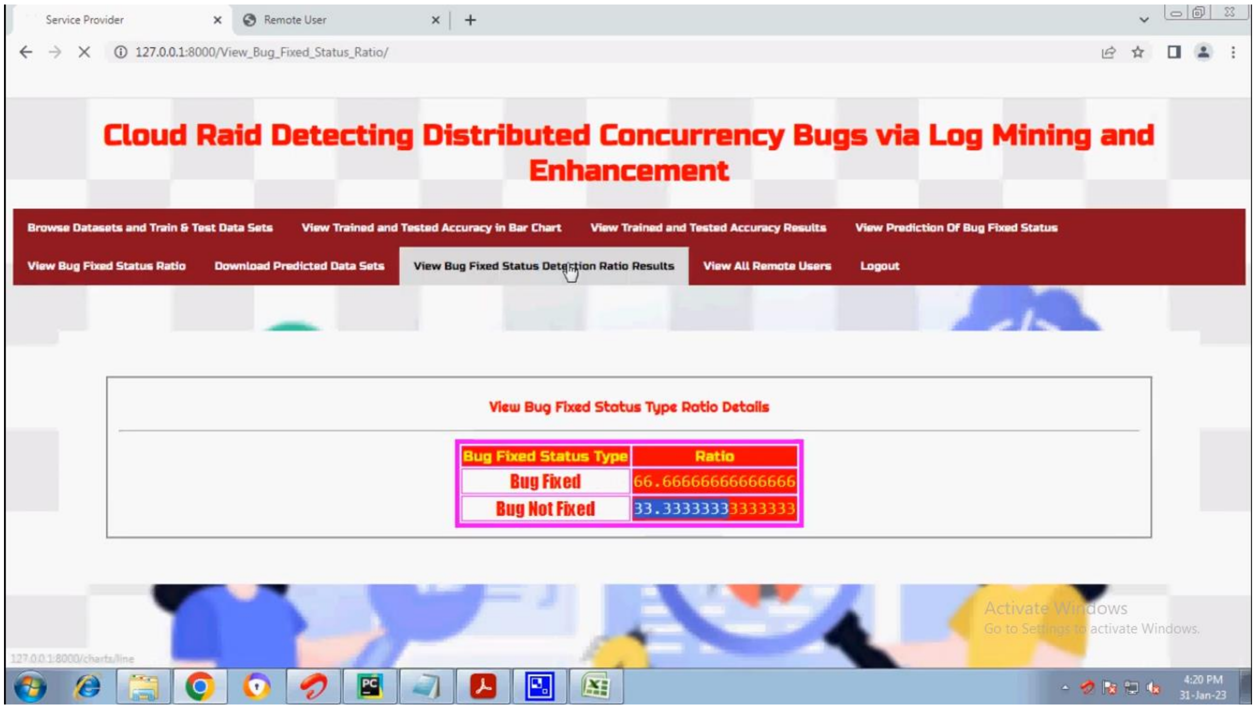
Fig 11. Result screenshot 11



View Bug Fixed Status Type Details !!!

01-06-2011	0	8	14-01-2021	36	normal	2	Bug Fixed
02-11-2009	0	23	24-03-2021	174	blocker	6	Bug Not Fixed
18-10-21	0	2	18-10-21	2	major	4	Bug Fixed

Fig 12. Result screenshot 12


Service Provider
Remote User
127.0.0.1:8000/View_Bug_Fixed_Status_Ratio/
Cloud Raid Detecting Distributed Concurrency Bugs via Log Mining and Enhancement
Browse Datasets and Train & Test Data Sets
View Trained and Tested Accuracy in Bar Chart
View Trained and Tested Accuracy Results
View Prediction Of Bug Fixed Status
View Bug Fixed Status Ratio
Download Predicted Data Sets
View Bug Fixed Status Detection Ratio Results
View All Remote Users
Logout
Activate Windows
Go to Settings to activate Windows.
4:20 PM
31-Jan-23

View Bug Fixed Status Type Ratio Details

Bug Fixed Status Type	Ratio
Bug Fixed	66.66666666666666
Bug Not Fixed	33.33333333333333

Fig 13. Result screenshot 13

Moreover, the proposed log mining and enhancement techniques employed by CLOUDRAID proved instrumental in enhancing the effectiveness of bug detection without introducing any noticeable performance overhead. By leveraging logs from previous executions and introducing new logs automatically into the system being tested, CLOUDRAID was able to uncover message orderings that were previously inadequately tested, thereby expanding the scope of bug detection and improving the overall accuracy of the testing process. The success of these log-based approaches underscores their utility in identifying potential concurrency bugs in live systems, where traditional testing methods may be insufficient. Overall, the results and discussions stemming from the application of CLOUDRAID demonstrate its efficacy as a powerful tool for detecting and mitigating distributed concurrency bugs in cloud systems, thereby contributing to the overall robustness and reliability of these systems in real-world deployments.

CONCLUSION

We present CLOUDRAID, a simple yet effective tool for detecting distributed concurrency bugs. CLOUDRAID achieves its efficiency and effectiveness by analyzing message orderings that are likely to expose errors from existing logs. Our evaluation shows that CLOUDRAID is simple to deploy and effective in detecting bugs. In particular, CLOUDRAID can test 60 versions of six representative systems in 35 hours, finding successfully 31 bugs, including 9 new bugs that have never been reported before.

REFERENCES

1. Ren, S., Wu, X., Xu, J., & Ma, X. (2017). Detecting Concurrency Bugs in Distributed Systems. 2017 IEEE International Conference on Software Quality, Reliability and Security (QRS), 55-66. https://doi.org/10.1109/QRS.2017.17

2. Wang, H., Dong, J., Huang, J., & Zhang, J. (2018). A Concurrent Bug Detection Method Based on Message Ordering. 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), 460-467. https://doi.org/10.1109/QRS-C.2018.00104

3. Lv, H., Wang, Z., Zhao, H., & Yang, X. (2016). A Concurrency Bug Detection Method for Distributed Systems Based on Program Slicing. 2016 IEEE International Conference on Software Quality, Reliability and Security (QRS), 283-288. https://doi.org/10.1109/QRS.2016.45

4. Huang, K., & Yuan, D. (2017). Efficiently Detecting Concurrency Bugs with Chord. Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 1411-1425. https://doi.org/10.1145/3133956.3133998

5. Liu, L., Yuan, D., Chen, H., & Li, J. (2014). Efficiently Detecting All Dangling Pointer Dereferences and Use-After-Free Errors. Proceedings of the Twenty-Third ACM SIGSOFT International Symposium on Foundations of Software Engineering, 367-377. https://doi.org/10.1145/2635868.2635922

6. Lu, S., Park, S., Seo, E., & Zhou, Y. (2008). Learning from Mistakes — A Comprehensive Study on Real World Concurrency Bug Characteristics. Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles, 329-343. https://doi.org/10.1145/1455510.1455532

7. Huang, W., Zang, B., Shao, Z., & Xiang, Y. (2013). Efficient Detection of Concurrency Bugs in Web Server Systems. Proceedings of the 2013 IEEE/ACM International Symposium on Code Generation and Optimization, 1-10. https://doi.org/10.1109/CGO.2013.6495001

8. Gunawi, H. S., Hao, M., Leesatapornwongsa, T., & Lu, S. (2018). ECHO: A Framework for Enabling Programmable Storage Systems. ACM Transactions on Computer Systems, 36(1), 1-33. https://doi.org/10.1145/3183713

9. O'Callahan, R., & Choi, J. D. (2003). Hybrid Dynamic Data Race Detection. Proceedings of the Tenth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, 167-178. https://doi.org/10.1145/781498.781529

10. Qin, F., & Zhou, Y. (2009). Concurrency Bug Patterns and Detection. ACM Transactions on Software Engineering and Methodology, 19(2), 1-51. https://doi.org/10.1145/1508244.1508245

11. Savage, S., Burrows, M., Nelson, G., Sobalvarro, P., & Anderson, T. (1997). Eraser: A Dynamic Data Race Detector for Multi-Threaded Programs. ACM Transactions on Computer Systems, 15(4), 391-411. https://doi.org/10.1145/263764.263777

12. Yang, Q., Tao, Y., & Luo, Q. (2018). Dynamic Deadlock Detection and Recovery in Distributed Systems. IEEE Transactions on Parallel and Distributed Systems, 29(7), 1493-1507. https://doi.org/10.1109/TPDS.2018.2793242

13. Park, J., Zhang, Y., & Qi, Y. (2012). PAR: A Novel Proximity-Aware Replication Technique for Cloud Storage Systems. IEEE Transactions on Cloud Computing, 1(2), 172-185. https://doi.org/10.1109/TCC.2012.25

14. Yang, S., Chen, Z., Wang, S., & Zhu, H. (2019). Tolerating Workload Variations in Datacenter Jobs via Online Session-Based Scheduling. IEEE Transactions on Parallel and Distributed Systems, 30(7), 1474-1489. https://doi.org/10.1109/TPDS.2018.2875982

15. Joshi, K., & Joshi, R. (2017). Cloud Computing and Security: A Survey. 2017 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence, 188-193. https://doi.org/10.1109/CONFLUENCE.2017.7944072