International Journal For Advanced Research In Science & Technology
A peer reviewed international journal www.ijarst.in
ISSN: 2457-0362

DISTRIBUTED CONCURRENCY BUGS DETECTION SYSTEM USING IMPROVED LOG MINING

R SUMA

Master of Computer Applications (MCA),

SVKP & Dr. K.S Raju Arts & Science College(A),

Penugonda, W.G.Dt., A.P, India

Sumaramba3@gmail.com

K LAKSHMAN REDDY

Associate Professor in Computer Science,

SVKP & Dr.K.S Raju Arts & Science College(A),

Penugonda, W.G.Dt., A.P, India

Klreddy29@yahoo.co.in

ABSTRACT:

Distributed concurrency problems can cause data loss and service outages in cloud systems. This work introduces CLOUDRAID, a novel automatic method for quickly and precisely finding distributed concurrency issues. Due to the fact that they are caused by unanticipated message orderings and untimely interaction among nodes, distributed concurrency problems are notoriously difficult to identify**.** Only the message orderings that are likely to reveal flaws are examined and tested automatically by CLOUDRAID in order to swiftly and effectively find concurrent bugs in cloud systems. For example, CLOUDRAID searches through the logs of earlier runs to find message orderings that are technically possible but haven't been thoroughly verified. We also provide a method for automatically adding new logs to the system under test called log boosting. These additional logs enhance CLOUDRAID's functionality while adding no appreciable performance overhead. Our log-based methodology makes it suitable for use in live systems. Six exemplary distributed systems—Hadoop2/Yarn, HBase, HDFS, Cassandra, Zookeeper, and Flink—were analysed using CLOUDRAID. In 35 hours, CLOUDRAID was able to test 60 different iterations of these six systems—ten iterations for each system—discovering 31 concurrent problems, including nine newly reported ones. Three of the nine newly discovered issues are critical and have already been addressed, according to their original developers, who have all confirmed them.

INTRODUCTION

The core building blocks of contemporary cloud applications are distributed systems, such as scale-out computing frameworks [1], [2], distributed key-value stores [3], [4], scalable file systems [3], [4], and cluster management services [2]. High stability of their underlying distributed systems becomes essential as cloud applications offer users round-the-clock online services. Distributed systems, however, are notoriously challenging to implement correctly. Real-world distributed systems commonly have software vulnerabilities that frequently result in data loss and cloud outages and cost service providers millions of dollars per incident [5], [6]

Distributed concurrency issues are among the most problematic distributed systems bugs [7], [8]. Complex message inter leavings, or unexpected orderings of communication events, set off these problems. Programmers find it challenging to handle and properly analyse concurrent executions

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal www.ijarst.in
ISSN: 2457-0362

across several machines. This feature has spurred a significant amount of research on distributed system model checkers [9], [10], [11], [12], which rigorously test all potential message orderings in order to uncover faults that are difficult to find. When performing the same workload that was previously certified, these model checks should theoretically ensure reliability. State-space explosion is a challenge that distributed system model checkers must deal with, though [9]. It is still challenging to scale recent advancements to numerous significant real-world applications [9]. For instance, 5,495 messages are involved in our tests using the Word Count workload on Hadoop2/Yarn. It becomes impractical to test every conceivable message ordering in a timely manner, even in such a straightforward instance.

LITERATURE SURVEY

We go over earlier investigations into distributed concurrent bug detection, log analysis, and log improvement. 5.1 Detection of distributed concurrency bugs Distributed system model checkers have received a lot of attention in the literature [9], [10], [11], [12], and [32]. These checkers intercept communications in real time and thoroughly shuffle their orderings. Despite being strong, they have a state space explosion issue. Recent technologies [9], [12] utilise state reduction approaches to solve this issue, however they may still not be scalable for huge state spaces [9]. In order to identify race circumstances in distributed systems, Liu et al. [33] recently enhanced race detection methods for multi-threaded programs [34], [35], [36], [37], [38], and [39]. In order to gather runtime traces at runtime, their method instruments memory accesses and communication events in a system. Using a happen before model tailored to distributed systems, an offline analysis is carried out to examine the happen-before link between memory accesses. Accesses to concurrent memory that could result in exceptions are viewed as damaging data races. To more thoroughly confirm the discovered race conditions, a trigger is used. According to [40], its method limits itself to message orderings involving just two messages and mines logs to retrieve runtime traces without instrumentation. With the help of two important extensions, we have increased the efficacy of this earlier strategy in this paper. First, we provide a novel log improvement method that enables us to find flaws that would otherwise go undetected. We can now identify issues that appear in message orderings with any number of messages, which is the second advancement.

With these two additions, we have demonstrated experimentally that our framework is more capable of identifying flaws in fresh programs. To assess the robustness of distributed systems, fault injection techniques [41], [42], [43], [44], [45], [46], [47], [48], [49] are frequently utilised. They concentrate instead on how to introduce faults at various system states to reveal flaws in the fault handlers. Combining CLOUDRAID can improve the efficiency of fault-related concurrent problem detection. Log Analysis (5.2) Numerous studies [25], [45], [50], [51], [52], [53], [54], [55], [56], [57], [58] have mined logs for a variety of information, including temporal invariants [51], [53], user request flow [50], [52], system architecture [25], and timing information [56]. The information that has been mined can then be used to improve monitoring, understanding, and analysis of complex distributed systems. Machine learning

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal www.ijarst.in
ISSN: 2457-0362

methods are used by Xu et al. [17] to analyse console logs from a system and find anomalous executions. To aid users in identifying anomalous behaviours, mined data is visualised, including logged values and logging frequencies. To determine the strongest correlation between system components and performance, DISTALYZER [59] compares logs from abnormal and normal executions.

To profile request latency, Iprof [18] pulls request IDs and timing data from logs. By examining the relationships between the logged ID variables to profile various components across the full distributed software stack, Stitch [60] groups log instances into tasks and sub-tasks. However, CLOUDRAID effectively detects concurrent problems by sifting through logs to find message orderings that haven't been used enough. Similar log analysis is used by CRASHTUNER [61] to infer some system meta-information, such as the running nodes and tasks/resources connected to each node. The meta-info is used by this tool to identify crash-recovery problems, which are brought on by crashing a node where the meta-info for that node is being accessed.

CLOUDRAID, on the other hand, uses log analysis to reveal the orderings between communication events in order to find distributed concurrency problems. 5.3 Enhancement of Logs A number of log improving approaches [26], [27], [28], [29], [62], [63] exist to assist developers in more efficiently identifying the main causes in online systems. For instance, Log Enhancer [26] employs dependence analysis to identify factors that have an effect on particular conditional branches and incorporates these variables into the existing logs. In order to add more logging statements to the exception handlers, Log Advisor [27] analyses unlogged exceptions in the source code. Our log enhancer attempts to effectively find more crucial signals that could result in concurrency issues, as opposed to some earlier log improving solutions. As a result, our log enhancer adds log statements for unlogged messages where local ID variables are present.

PROBLEM STATEMENT

To assess the robustness of distributed systems, fault injection techniques [41], [42], [43], [44], [45], [46], [47], [48], [49] are frequently utilised. They concentrate instead on how to introduce faults at various system states to reveal flaws in the fault handlers. Combining CLOUDRAID can improve the efficiency of fault-related concurrent problem detection.

Machine learning methods are used by Xu et al. [17] to analyse console logs from a system and find anomalous executions. To aid users in identifying anomalous behaviours, mined data is visualised, including logged values and logging frequencies. To determine the strongest correlation between system components and performance, DISTALYZER [59] compares logs from abnormal and normal executions. To profile request latency, Iprof [18] pulls request IDs and timing data from logs. By examining the relationships between the logged ID variables to profile various components across the full distributed software stack, Stitch [60] groups log instances into tasks and sub-tasks. However, CLOUDRAID effectively detects concurrent problems by sifting through logs to find message orderings that haven't been used enough.

Similar log analysis is used by CRASHTUNER [61] to infer some system meta-information, such as the running nodes and tasks/resources connected to each node. The meta-info is used by this tool to

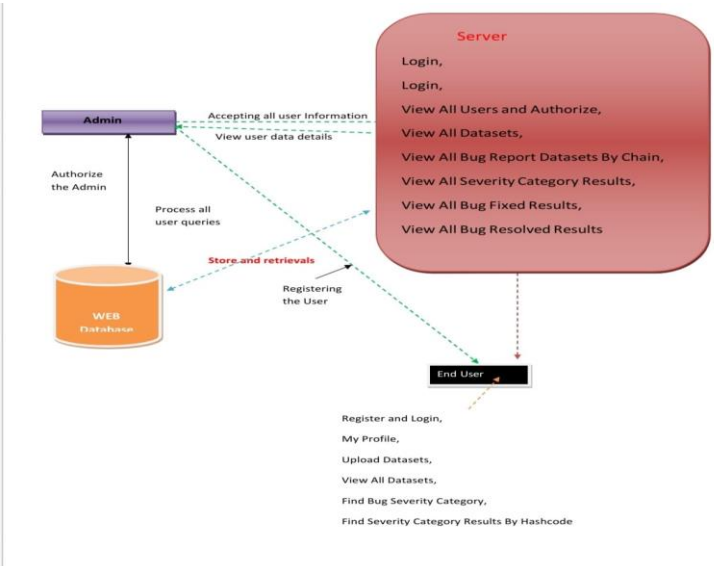
International Journal For Advanced Research In Science & Technology
A peer reviewed international journal www.ijarst.in
ISSN: 2457-0362

identify crash-recovery problems, which are brought on by crashing a node where the meta-info for that node is being accessed. CLOUDRAID, on the other hand, uses log analysis to reveal the orderings between communication events in order to find distributed concurrency problems.An existing methodology does not use a cutting-edge technique to identify distributed concurrency problems. The system doesn't strive for CLOUDRAID uses the run-time logs of active systems and stays away from pointless repeating checks.

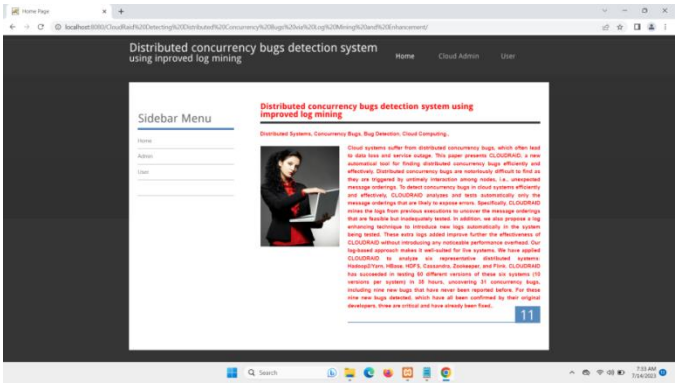
METHODODLOGY

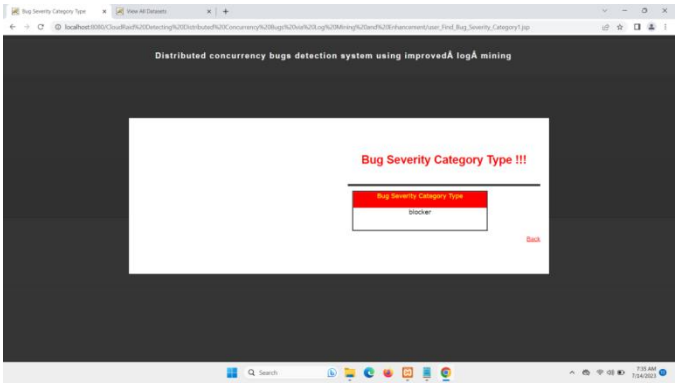
The suggested method focuses on finding flaws that result from order violations, or bugs that appear anytime a message comes in the incorrect order relative to another event. Rearranging a pair of messages will most likely reveal the majority of these issues, as was previously stated. However, when there are more than two messages involved, a small number of serious errors still happen. Only under precise temporal circumstances, such as those involving some particular messages or events (such as node crashes or reboots), can these flaws be made public. We have given our method the ability to randomly reorganise any amount of messages for an application in order to discover such issues.

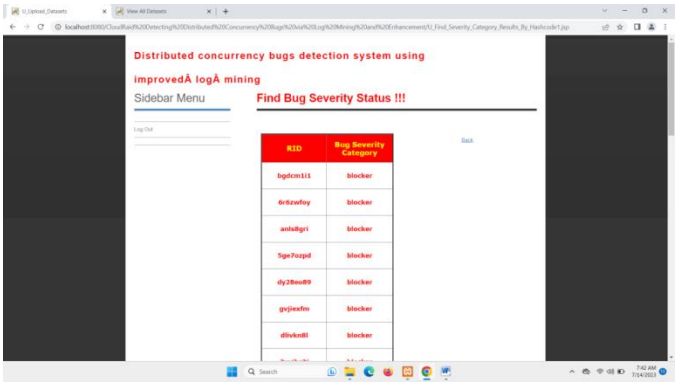
ARCHITECHTURE:



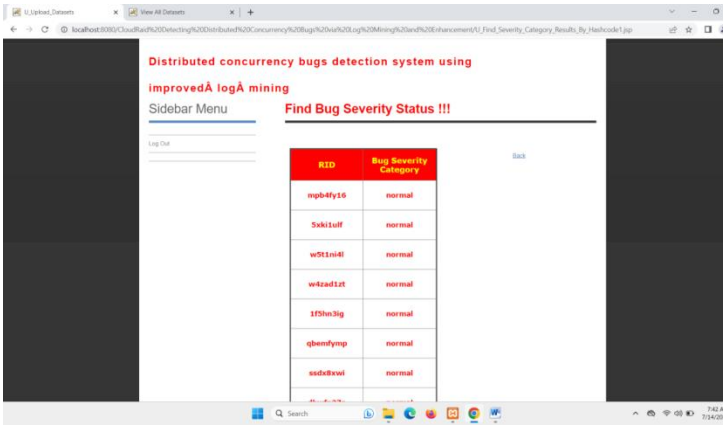
RESULTS:

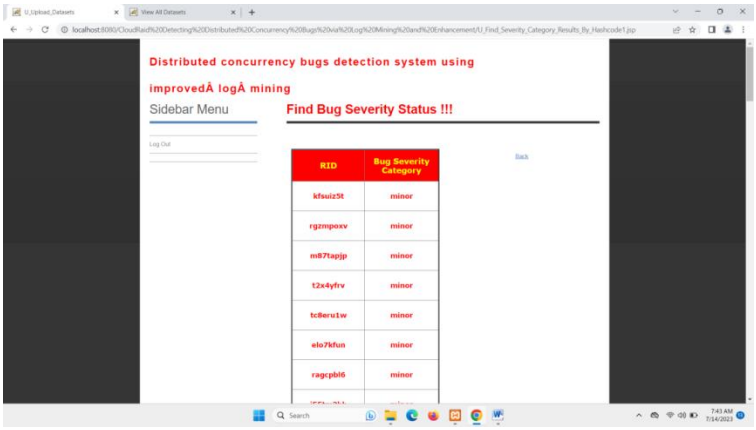






International Journal For Advanced Research In Science & Technology
A peer reviewed international journal www.ijarst.in
ISSN: 2457-0362





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. X. Zhao, Y. Zhang, D. Lion, M. F. Ullah, Y. Luo, D. Yuan, and M. Stumm, “lprof: A non-intrusive request flow profiler for distributed systems.” in OSDI, vol. 14, 2014, pp. 629–644.

2. L. Li, C. Cifuentes, and N. Keynes, “Boosting the performance of flow-sensitive points-to analysis using value flow,” in Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering, ser. ESEC/FSE ’11. New York, NY, USA: ACM, 2011, pp. 343–353. [Online]. Available: http://doi.acm.org/10.1145/2025113.2025160

3. “Precise and scalable context-sensitive pointer analysis via value flow graph,” in Proceedings of the 2013 International Symposium on Memory Management, ser. ISMM ’13. New York, NY, USA: ACM, 2013, pp. 85–96. [Online]. Available: http://doi.acm.org/10.1145/2464157.2466483

4. T. Tan, Y. Li, and J. Xue, “Efficient and precise points-to analysis: Modeling the heap by merging

equivalent automata," in Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation, ser. PLDI 2017. New York, NY, USA: ACM, 2017, pp. 278–291. [Online].Available: http://doi.acm.org/10.1145/3062341.3062360

5. Y. Sui and J. Xue, "On-demand strong update analysis via valueflow refinement," in Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, ser. FSE 2016. New York, NY, USA: ACM, 2016, pp. 460–473. [Online]. Available: http://doi.acm.org/10.1145/2950290.2950296

ABOUT AUTHORS

R.SUMA

Currently pursuing MCA in SVKP&DR.K.S.RAJU Arts &Science college affiliated to Adikavi Nannaya University, Rajamahendravaram. Her research interests include Data Structures,Web Technologies,Operating SystemData Science and Artificial Inteligence.

K LAKSHMANA REDDY

Working as Associate Professor in SVKP &Dr K S Raju Arts & Science College(A), Penugonda , West Godavari District, A.P. He received Master's Degree in Computer Applications from Andhra University'C' level from DOEACC, New Delhi and M.Tech from Acharya Nagarjuna University, A.P. He attended and presented papers in conferences and seminars. He has done online certifications in several courses from NPTEL. His areas of interests include Computer Networks, Network Security and Cryptography, Formal Languages and Automata Theory and Object Oriented programming languages.

.