IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

COLLEGE LECTURE NOTES MANAGEMENT SYSTEM

***(1)* Dr.K.SAILAJA MCA,M.TECH,phD (2)CHANDRAHASREDDY**

PROFFESOR & HOD Student

DepartmentofComputer Applications Department of Computer Applications

Chadalwadaramanamma engineering colleagechadalawadaramanammaengineerincolleage

Abstract – As the world is being developed with the new technologies, discovering and manipulating new ideas and concepts of taking everything online are rapidly changing. It is difficult for teacher's to circulate their notes to each and every student whom is he/she teaching. College Notes Gallery provide an easy approach for both students and teachers to circulate the notes whether of any kind like lecture notes, assignment questions, question papers and all the important documents. The teachers and students can upload the documents from anywhere and students can download it. Overall it is managed by the admin. Existing System and its Limitations Mostly the notes are circulated on WhatsApp or any kind so it gets very difficult to manage the important notes at the time of need. Need of Proposed System My system will provide an easy approach to share the documents for studying purpose. Multiple users can work simultaneously on the system. It will be easy for the teachers to circulate the notes to each and every students.

Keywords- Admin panel, Notes, Student, Teacher, College notes, Gallery

1.INTRODUCTION

Distributed systems, such as scale-out computing frameworks distributed key-value stores scalable file systems and cluster management servicesare the fundamental building blocks of moderncloud applications. As cloud applications provide 24/7online services to users, high reliability of their underlyingdistributed systems becomes crucial.

However, distributedsystems are notoriously difficult to get right. There are widelyexisting software bugs in real-world distributed systems,which often cause data loss and cloud outage, costing serviceproviders millions of dollars per outrage.

Among all types of bugs in distributed systems, distributedconcurrency bugs are among the most troublesome. These bugs are triggered by complex interleavingsof messages, i.e., unexpected orderings of communicationevents. It is difficult for programmers to correctlyreason about and handle concurrent executions on multiplemachines. This fact has motivated a large body of research ondistributed system model checkers whichdetect hard-to-find bugs by exercising all possible messageorderings systematically. Theoretically, these model checkerscan guarantee reliability when running the same workloadverified earlier. However, distributed system model checkers face the state-space explosion problem. Despite recentadvancesit is still difficult to scale them to many largereal-world applications. For example, in our experimentsfor running the WordCount workload on Hadoop2/Yarn,5,495 messages are involved. Even in such a simple case, itbecomes impractical to test exhaustively all possible messageorderings in a timely manner.

2. ITERATURE SURVEY

2.1DIFFERENTAUTHORSDISCUSSION:X u et al. mine console logs from a system and apply machine learning techniques to detect anomaly executions. Mined information such as logged values and logging frequencies is visualized to help users diagnose anomaly behaviors. DISTALYZER compares logs from abnormal and normal executions to infer the strongest association between system components and performance. Iprof extracts request IDs and timing information from logs to profile request latency. Stitch [60] organizes log instances into tasks and sub-tasks, by analyzing relations among the logged ID variables to profile different components in the entire distributed software stack.

2.2 DOMAIN DESCRIPTION

Fault injection techniques are commonly used to test the resilience of distributed systems. However, they focus on how to inject faults at different system states to expose bugs in the fault handlers. CLOUDRAID can be applied together to detect fault-related concurrency bugs more effectively.

3. PROBLEM STATEMENT

3.1.EXISTING SYSTEM

Liu et al. have recently extended race detection techniques for multi-threaded programs to detect race conditions in distributed systems. Their approach instruments memory accesses and communication events in a system to collect runtime traces at run time. An offline analysis is performed to analyze the happen-before relation among the emory accesses, by using a happen before model customized to distributed systems. Concurrent memory accesses that may trigger exceptions are regarded as harmful data races.A trigger is employed to further verify the detected race conditions. In, its approach mines logs to recover runtime traces without instrumentation, by restricting itself to message orderings involving only two messages. In this paper, we have improved the effectiveness of this earlier approach with two significant extensions.

3.2DISADVANTAGES OF EXISTING SYSTEM

An existing methodology doesn't implement a novel strategy for detecting distributed concurrency bugs.The system is not aiming at CLOUDRAID leverages the run-time logs of live systems and avoids unnecessary repetitive tests.

4. PROPOSED SYSTEM

4.1 PROPOSED SYSTEM

We propose a new approach, CLOUDRAID, for detecting concurrency bugs in distributed systems efficiently and effectively. CLOUDRAID leverages the run-time logs of live systems and avoids unnecessary repetitive tests, thereby drastically improving the efficiency and effectiveness of our approach.We describe a new log enhancing technique for improving log quality automatically. This enables us to log key communication events in a system automatically without introducing any noticeable performance penalty. The enhanced logs can further improve the overall effectiveness of our approach.

4.2ADVANTAGES OF PROPOSED SYSTEM

The proposed approach focuses on detecting the bugs caused by order violation, i.e., the bugs which manifest themselves whenever a message arrives at a wrong order with respect to another event. The majority of these bugs can be exposed by reordering a pair of messages, as suggested previously.However, relatively few but critical bugs still occur when more than two messages are involved. These bugs can only be exposed under special timing conditions, involving, for example, some specific messages or events (e.g., node crashes or reboots). To detect such errors, we have empowered our approach with the capability of reordering an arbitrary number of messages for an application.

5.IMPLEMENTATION

5.1 Admin

In this module, the Service Provider has to login by using valid user name and password. After login successful he can do some operations such as View All Users and Authorize, View All Datasets,View All Bug Report Datasets By Chain,View All Severity Category Results,ViewAll Bug Fixed Results,View All Bug Resolved Results.

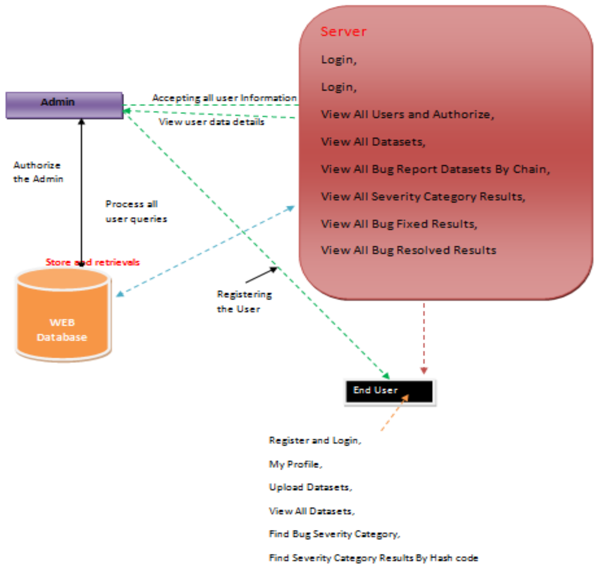
5.2 View and Authorize Users

Inthismodule,facultyregisterandlogintothesystem.Heallowsuploadingmaterials, events, attendance, marks in the system. He can view theirstudent'sattendancedetails, marksdetails, and updatehisprofile.

5.3 End User

In this module, there are n numbers of users are present. User should register before doing any operations. Once user registers, their details will be stored to the database. After registration successful, he has to login by using authorized user name and password. Once Login is successful user will do some operations like MyProfile,UploadDatasets,View All Datasets,Find Bug Severity Category,Find Severity Category Results By Hashcode.

6.SYSTEM ARCHITECTURE



7.CONCLUSION

We present CLOUDRAID, a simple yet effective tool for detectingdistributed concurrency bugs. CLOUDRAID achieves itsefficiency and effectiveness by analyzing message orderingsthat are likely to expose errors from existing logs. Ourevaluation shows that CLOUDRAID is simple to deploy andeffective in detecting bugs. In particular, CLOUDRAID cantest 60 versions of six representative systems in 35 hours,finding successfully 31 bugs, including 9 new bugs that have never been reported before.

7.1 FUTURE ENHANCEMENT

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.

8.REFERENCES

[[1] J. Dean and S. Ghemawat, “Mapreduce: Simplified data
processing on large clusters,” Commun. ACM, vol. 51,
no. 1, pp. 107–113, Jan. 2008. [Online]. Available: http:
//doi.acm.org/10.1145/1327452.1327492

[2] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar,
R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino,
O. O’Malley, S. Radia, B. Reed, and E. Baldeschwieler, “Apache
hadoop yarn: Yet another resource negotiator,” in Proceedings of
the 4th Annual Symposium on Cloud Computing, ser. SOCC ’13.
New York, NY, USA: ACM, 2013, pp. 5:1–5:16. [Online]. Available:
http://doi.acm.org/10.1145/2523616.2523633

[3] L. George, HBase: the definitive guide: random access to your planet-size
data. " O’Reilly Media, Inc.", 2011.

[4] A. Lakshman and P. Malik, “Cassandra: a decentralized structured
storage system,” ACM SIGOPS Operating Systems Review, vol. 44,
no. 2, pp. 35–40, 2010.

[5] Z. Guo, S. McDirmid, M. Yang, L. Zhuang, P. Zhang,
Y. Luo, T. Bergan, P. Bodik, M. Musuvathi, Z. Zhang, and
L. Zhou, "Failure recovery: When the cure is worse than
the disease," in Proceedings of the 14th USENIX Conference on
Hot Topics in Operating Systems, ser. HotOS'13. Berkeley, CA,
USA: USENIX Association, 2013, pp. 8–8. [Online]. Available:
http://dl.acm.org/citation.cfm?id=2490483.2490491

[6] D. Yuan, Y. Luo, X. Zhuang, G. R. Rodrigues, X. Zhao, Y. Zhang,
P. U. Jain, and M. Stumm, "Simple testing can prevent most critical
failures: An analysis of production failures in distributed dataintensive
systems," in Proceedings of the 11th USENIX Conference on
Operating Systems Design and Implementation, ser. OSDI'14. Berkeley,
CA, USA: USENIX Association, 2014, pp. 249–265. [Online].
Available:
http://dl.acm.org/citation.cfm?id=2685048.2685068

[7] H. S. Gunawi, M. Hao, T. Leesatapornwongsa, T. Patana-anake,
T. Do, J. Adityatama, K. J. Eliazar, A. Laksono, J. F. Lukman,
V. Martin, and A. D. Satria, "What bugs live in the cloud?
a study of 3000+ issues in cloud systems," in Proceedings of
the ACM Symposium on Cloud Computing, ser. SOCC '14. New
York, NY, USA: ACM, 2014, pp. 7:1–7:14. [Online]. Available:
http://doi.acm.org/10.1145/2670979.2670986

[8] T. Leesatapornwongsa, J. F. Lukman, S. Lu, and H. S. Gunawi,
"Taxdc: A taxonomy of non-deterministic concurrency bugs in
datacenter distributed systems," in Proceedings of the Twenty-First
International Conference on Architectural Support for Programming
Languages and Operating Systems, ser. ASPLOS '16. New
York, NY, USA: ACM, 2016, pp. 517–530. [Online]. Available:
http://doi.acm.org/10.1145/2872362.2872374

[9] T. Leesatapornwongsa, M. Hao, P. Joshi, J. F. Lukman, and H. S.
Gunawi, "Samc: Semantic-aware model checking for fast discovery
of deep bugs in cloud systems." in OSDI, 2014, pp. 399–414.

[10] H. Lin, M. Yang, F. Long, L. Zhang, and L. Zhou, "Modist: Transparent
model checking of unmodified distributed systems," in 6th
USENIX Symposium on Networked Systems Design & Implementation
(NSDI), 2009.

[11] J. Simsa, R. E. Bryant, and G. Gibson, "dbug: systematic evaluation

of distributed systems." USENIX, 2010.

[12] H. Guo, M. Wu, L. Zhou, G. Hu, J. Yang, and L. Zhang, "Practical software model checking via dynamic interface reduction," in Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles. ACM, 2011, pp. 265–278.

[13] D. Borthakur et al., "Hdfs architecture guide," Hadoop Apache Project, vol. 53, 2008.

[14] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed, "Zookeeper: Waitfree coordination for internet-scale systems." in USENIX annual technical conference, vol. 8, no. 9, 2010.

[15] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, "Apache flink: Stream and batch processing in a single engine," Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, vol. 36, no. 4, 2015.

[16] (2018) Wala home page. [Online]. Available: http://wala. sourceforge.net/wiki/index.php/Main_Page/.

[17] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles. ACM, 2009, pp. 117–132.

[18] 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.

[19] 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

[20] ——, "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

[21] 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

[22] 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
[23] (2018) Google protocol buffer. [Online]. Available: https:
//developers.google.com/protocol-buffers/.
[24] E. Gamma, Design patterns: elements of reusable object-oriented software.
Pearson Education India, 1995.
[25] J.-G. Lou, Q. Fu, Y. Wang, and J. Li, "Mining dependency in
distributed systems through unstructured logs analysis," ACM
SIGOPS Operating Systems Review, vol. 44, no. 1, pp. 91–96, 2010.
[26] D. Yuan, J. Zheng, S. Park, Y. Zhou, and S. Savage, "Improving
software diagnosability via log enhancement," ACM Transactions
on Computer Systems (TOCS), vol. 30, no. 1, pp. 1–28, 2012.
[27] J. Zhu, P. He, Q. Fu, H. Zhang, M. R. Lyu, and D. Zhang, "Learning
to log: Helping developers make informed logging decisions,"
in 2015 IEEE/ACM 37th IEEE International Conference on Software
Engineering, vol. 1. IEEE, 2015, pp. 415–425.