RESEARCH ARTICLE OPEN ACCESS

THREAD BASED DEADLOCK DETECTION AND MANAGEMENT IN DISTRIBUTED DATABASE

J. Lethisia Nithiya[1], S. Sangeetha Priya[2]
Valliammai Engineering College, Chennai, India
Valliammai Engineering College, Chennai, India

Abstract --- In large scale multithreaded programs, deadlock plays an important role. It is considered as a hindrance in few large scale applications. Hence,therefore this work proceeds by using a new potential deadlock detection method. In this paper, we use lock dependencies that are categorized into thread specific partitions and equivalent lock dependencies. Any identical permutation that is being the same as another of the lock dependency chains are eliminated by the searches that are done over the set of lock dependency. To achieve the desired outcome this procedure involves repetition of steps and removes the lock dependencies before the deadlock localization occurs. It is validated by the large scale multithreaded programs. It is revealed that timestamps are primarily based on, start again policy.

Index Terms---Deadlock detection, Lock dependency, Large scale multithreaded programs.

Date of Submission: 10-02-2018 Date of acceptance: 28-02-2018

I. INTRODUCTION

A deadlock is an environment where more computer programs share similar resources. The accessing of resources from several programs leads to waiting unconditionally, without a proper solution. Previously developed operating system executes only one program at a time and they consist of all the resources. Lately developed OS executes multiple programs at a time.

A. *Deadlock impacts:*

Due to deadlock there are certain impacts.

- Time delay occurs in large scale applications.
- It won't have a feasible solution over a situation.
- System crashes due to deadlock.
- It waits unconditionally, hence the work will be pending.

Due to the issues involved in deadlock, it is mandatory for deadlock detection.We use Deadlock prevention algorithms for the recovery of deadlock. It is used in simultaneous programming when several processes must gain more than one shared resource. For the betterment of an application having no deadlocks, we can avoid deadlocks by certain techniques and algorithms. It can be done if certain information the system notices as when laying the request, it will go to an unsafe state,which means that particular state will suffer from deadlock.The system then grants requests that will lead to a safe state.

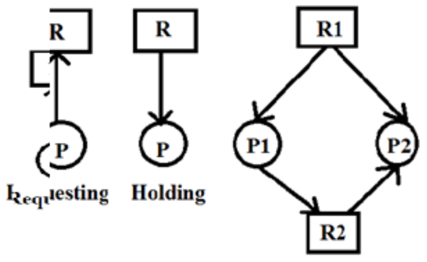


Fig. 1. Deadlock

B. *The main objective of our project is:*

- To assist locking services in inner deadlock prevention in local transactions, an efficient replica technique is issued.
- The appropriated stop aversion system using the former information on obliged assets by augmenting standard two-stage submits convention.

II. RELATED WORK

Bensalem, J.C. Fernandez, K. Havelund, and L. Mounier(2006) presented a framework in a multi-threaded program for conforming deadlock potentials discovered by runtime analysis of a single run. Also data mining topics of neural networks are also analyzed.To reveal deadlock potentials a lock graph is constructed in the form of cycles. Normally two kinds of deadlocks are there namely *resource deadlocks and communication deadlocks.* They focused only on *resource deadlocks, from now on referred to as deadlocks.* The advantages of the work revealed that detecting deadlocks come from the fact that concurrent programs several executions of the same program on the same input. Many effective algorithms and a system were described for detecting deadlock potentials in multi-threaded Java programs.But still they only concentrated on the detection of deadlocks and not in the transformation of the data's [12].

R. Agarwal et al. (2010) proposed techniques for detecting potential deadlocks.The basis is a well known algorithm where,when deadlocks are displayed in a nested order, its posses the order in a lock graph. Finally, they discussed the use of static analysis to mechanically reduce the deadlocks. The idea of utilizing segmentation was earlierly proposed into reducing the false positives in data race with the help of Eraser algorithm.It is easy to aggregate user interests from multiple sources. But it is not straightforward to build a general-level information for the given data.They can't carry out the building of large community-level information by adopting the approaches [6].

R. Raman, J.S Zhao, V. Sarkar, M. Vechev, and E.Yahav(2012) presented a new algorithm for structured parallel program called dynamic data race detection. It can make the program to process in parallel and gives an efficient execution. The algorithm is independent of the sequential portions of the language, meaning that one can apply it to any language where the parallelism is expressed using the async/finish constructs [4].

Z.F. Lai, S.C. Cheung, and W.K. Chan (2010) described a two-phase technique which can detect the violations of atomic-set effectively. They also implemented a technique which is known as prototype system ASSETFUZZER and involved it to a several number of programs for analyzing concurrency defect techniques. Verifying whether a process accepts this atomic-set correctness criterion is challenging. A set of problematic access patterns is described to detect the atomic-set correctness using runtime monitoring technique. They bring the concurrency bugs in sequential programs that do not exist. Hence the difficulty in this paper is to detect the concurrency bugs [7].

L. Lamport described the idea of temporal ordering of events. The distributed system can be done in a single system where the units and channels are in separate processes. In a scattered system, there will be many events occurring and it is impossible to say which event occurred first. The problem in this relation is only a half ordering of the events [16].

C.K. Luk et al. (2005) developed a new instrumentation system called Pin, to meet the needs such as bug detection, performance correction and profiling. The main aim is to give transparent, easy to use and efficient instrumentation. The pin tools are written using Pin's rich API in C or C++. As the result provides all the needs of an efficient instrumentation. But it provides a limited version to make modification in the program [13].

A. OBSTACLES IN THE RELATED WORK:

• The system can't avoid the deadlock and the process does not complete.
• A task that is executing should wait to induce the resources and it doesn't stop the deadlock efficiently.
• Sometimes all method stand still and the system can idle.

III. PROPOSED WORK

The projected design involves a Visa application. It includes three types of processors handled in a system and it will be monitored by Admin.1. A set of data of past crimes of which an individual has been sentenced 2. Passport, which proves the identity and nationality of the person for whom it was issued 3. The details of an individual passing or coming into a country for the purpose of permanent residence which is called as foreign immigration. In the scattered particularly grid surroundings, commercial exchanges are many times prepared and arranged on various sites. All exchanges, so what nearby some time worldwide, might approach a wide range of assets in the meantime. In this paper, we conjure neighborhood exchanges for stand out enormous settled exchange with a few scattered exchanges.

A. The issues in proposed system can be solved by:

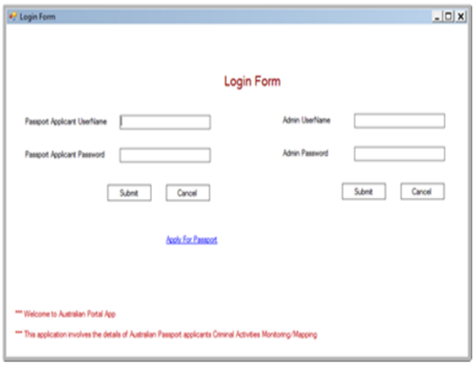
• Building an intuitive resource admin to spot an internal exchange collision.And set applicable locks for every transaction.Synchronization Parallel computing is used to avoid the deadlock occurrences in an application.

IV. METHODOLOGY

It is the structural area when the abstract model is changed into an acting structure.

A. System Access Module

The admin has given the type of authentication. It explains the connection between the individual who is a user and to the system. The users are permitted to make their evidences to login into the system. An admin should make sure the users design and login approval are correct, then the users will be allowed to use the application. Option of Security questions and answers were added to the system. It provides allowances to interact with the permission system inside the Application. It encloses a range of vision that makes it feasible to register and to turn on new users, password modifications, etc.



*** Welcome to Australian Portal App
*** This application involves the details of Australian Passport applicants Criminal Activities Monitoring/Mapping

Fig. 2. Login Form

The above snapshot is the basic login form for the entry of Australian visa application. Click the button that is used for entering into the passport application process and then it directs the user to the next screen.

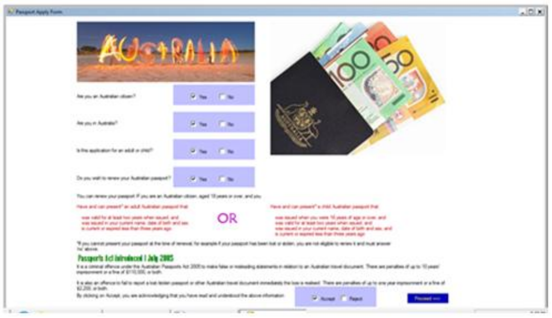


Fig. 3. Passport Apply Form

This page is more over like an acceptance of passport application. User needs to answer all the questions above and read the terms and finally need to accept it. Then continue with the process, it will automatically redirect to the next snapsho

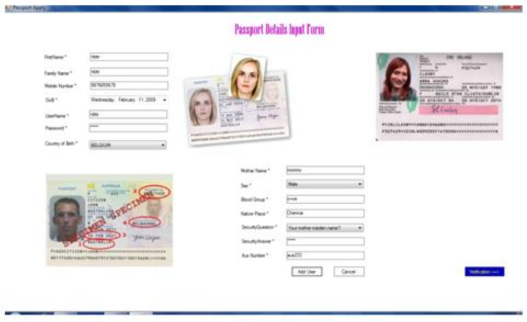
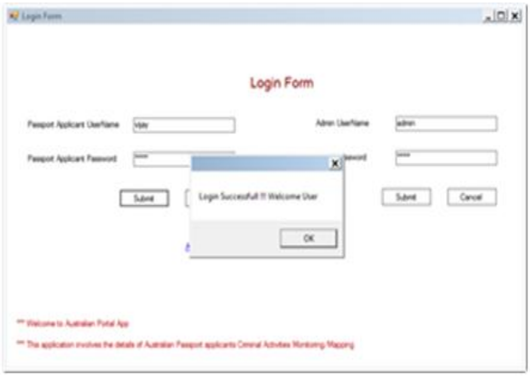


Fig. 4. Passport details input form

The above Figure is, passport details input form, where the applicant need to give his/her personal details. After filling all the required details, click on the button that adds the user details to the database and also to update the database. Then click on the button at right corner of the screen for verification process.



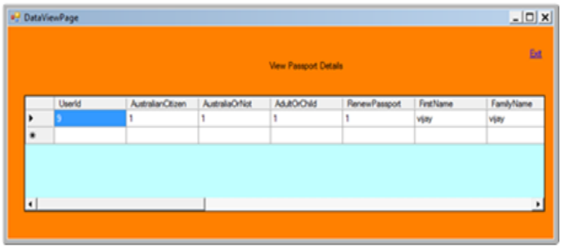
Passport Applicant UserName	vijay
Admin UserName	admin

Login Successfull !! Welcome User

*** Welcome to Australian Portal App
*** This application involves the details of Australian Passport applicants Criminal Activities Monitoring/Mapping

Fig. 5. User authentication

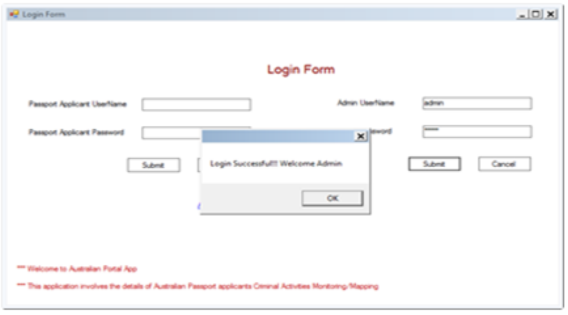
The above form is for user authentication, where users credential details are given. The user has to give his/her login and passport details input form for accessing the application. Click on the button for submitting. A pop up appears as login successful, now proceed with the acceptance. It redirects to the data view page.



UserId	AustralianCitizen	AustraliaOrNot	AdultOrChild	RenewPassport	FirstName	FamilyName
5	1	1	1	1	vijay	vijay

Fig. 6.Data View Page

This page is for user verification of his/her details which will get updated in the admins database. Now the user need to exit from the process, since his/her job is completed. Hence he/she get logged out of the system.



Admin UserName	admin

Login Successfull!! Welcome Admin

*** Welcome to Australian Portal App
*** This application involves the details of Australian Passport applicants Criminal Activities Monitoring/Mapping

Fig. 7. Admin's Credential

This form is for admins credential details to login into the system. When admin selects submit, he will be logged onto the system.

B. Criminal history Module

User's criminal history is verified by admin using multi-threading scenario towards each resource. Criminal history Data's are loaded into the database. Simultaneously resource will be modified and verified by admin. So, it makes deadlock occurs in the criminal history database and it is prevented by using the synchronization parallel computing technique. Each resource waits for another resource completion.

C. Passport validation Module

A user's passport is validated by admin using multi-threading scenario towards each resource. Passport Data's are loaded into the database. Simultaneously resource will be modified and verified by admin. A passport is validated before criminal history of particular user is also possible and updated in a System. Simultaneously, passport and criminal history process are handled at the same time. So, it makes deadlock occurs in Passport database and it is prevented by using the synchronization parallel computing technique.

D. Foreign immigration Process Module

User's is validated for foreign immigration by admin using multi-threading scenario towards each resource. Immigration Data's are loaded into the database. Simultaneously resource will be modified and verified by admin in this module also. Simultaneously, passport and criminal history process are handled at the same time. So, it makes deadlock occurs in immigration database and it is prevented by using the synchronization parallel computing technique.

E. Priority based Process Module

Priority wise Process is handled at simultaneously in an application and it is controlled by admin. In multi-threading Priority is given to particular resource based on allocation of time in an application. Updating of a particular resource is maintained and multiple processors are allowable. The multiple - request is rising within the application can be handled by each process in an application is monitored and priority wise it will be scheduled based on the algorithms.

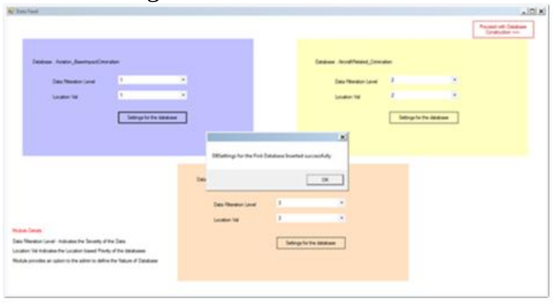


Fig. 8. Data Feed

The above screen shot explains three main types of criminal activities like:
a) Aviation_BaseImpactCriminalism
b) AircraftRelated_Criminalism
c) FireRelated_Criminalism.

Based on the severity of the criminalism, priority has to be set for each activity. Then click the button which is used to set the database. A popup appears and click the acceptance button. Then it redirects to the next step.

The below screenshot contains few possible criminal case history happened in different states at various time frames. It actually indicated the injection of cases in specific system's database. Now click the button that proceeds to the deadlock detection. Then it automatically redirects to the next screen.

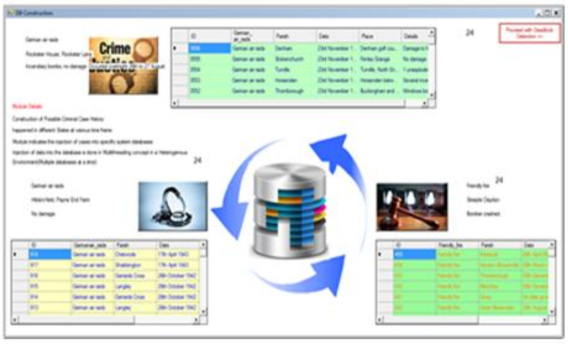
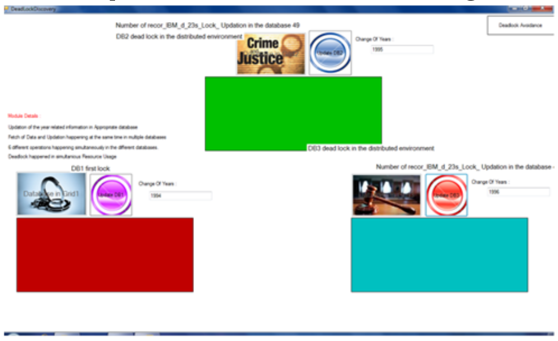


Fig. 9.DB Construction

F. Deadlock detection Module

In this module, we are going to detect the deadlock in the distributed application. Possible distributed deadlock will be created due to coincidental information access among the Process. One or two processes with high priority will be decided. Deadlock is prevented by using algorithms and techniques involved with multi-threading.



DeadLockDiscovery
Number of recor_IBM_d_23s_Lock_ Updation in the database 49
DB2 dead lock in the distributed environment
Change Of Years : 1995
Deadlock Avoidance
Module Details :
Updation of the year related information in Appropriate database
Fetch of Data and Updation happening at the same time in multiple databases
6 different operations happening simultaneously in the different databases.
Deadlock happened in simultanious Resource Usage
DB3 dead lock in the distributed environment
DB1 first lock
Change Of Years : 1994
Number of recor_IBM_d_23s_Lock_ Updation in the database -
Change Of Years : 1996

Fig. 10. Deadlock Discovery

This particular screenshot deals with the detection of deadlock that is particularly in which phase there is a lock. We need to give the year specification to update the database according to the particular year. Click the update buttons to start its process of finding where the user has been encountering a problem. Then the process finds to which phase the user details are matched and alters that the deadlock has been occurring in a particular criminal. Click the avoidance option to avoid deadlock.

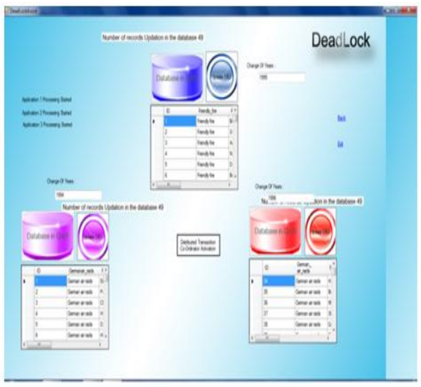


Fig. 11.Deadlock Avoidance

By using Synchronization Parallel Compute technique and Priority Scheduling algorithm, the deadlock can be avoided in the particular database. This is the final snapshot of our work.

G. BENEFITS OF THE FORTHPUT WORK

• Providing a productive replication instrument to backing for locking administrations of counteracting internal halt in nearby exchanges.

• Designing a timestamp in basically based victimized person determination rules can shatter a stop cycle when worldwide halt is identi.

V. CONCLUSION

Hence by this project exhibits deadlock detection and avoidance with the help of a synchronization parallel computing algorithm. This work presents a novel distributed deadlock blocking mechanism, utilizing the prior knowledge of needed resources by extending two phase commit protocol. This work gives greater efficiency compared to previous works. Likewise, this can be improved for future enhancements.

REFERENCES

[1] Sheng, Wubin, J. Yu, Z. Weng, and D. Tang. "A distributed and reusable workflow-based auto test framework for distributed integrated modular avionics," In AUTOTESTCON 2017 IEEE, pp. 1-8, 2017.

[2] Y. Cai and W.K. Chan, "Lock Trace Reduction for Multithreaded Programs,"IEEE Trans. Parallel and Distributed Systems, vol. 24, no. 12, pp. 2407-2417, Dec. 2013.

[3] Y. Cai, K. Zhai, S.R. Wu, and W.K. Chan, "TeamWork: Synchronizing Threads Globally to Detect Real Deadlocks for Multithreaded Programs,"Proc. 18th ACM SIGPLAN Symp. Principles and Practice of Parallel Programming (PPoPP), pp. 311-312, 2013.

[4] R. Raman, J.S Zhao, V. Sarkar, M. Vechev, and E.Yahav, "Scalable and Precise Dynamic Datarace Detection for Structured Paralelism," Proc. 33rd ACM SIGPLAN Conf. Programming Language Design and Implementation (PLDI), pp. 531-542, 2012.

[5] Y. Cai and W.K. Chan, "LOFT: Redundant Synchronization Event Removal for Data Race Detection,"Proc. IEEE 22nd Int'l Symp. Software Reliability Eng. (ISSRE), pp. 160-169, 2011.

[6] R. Agarwal, S. Bensalem, E. Farchi, K. Havelund, Y. Nir-Buchbinder, S.D. Stoller, S. Ur, and L. Wang, "Detection of Deadlock Potentials in Multithreaded Programs,"IBM J. Research and Development, vol. 54, no. 5, pp. 520-534, Sept. 2010.

[7] Z.F. Lai, S.C. Cheung, and W.K. Chan, "Detecting Atomic-Set Serializability Violations for Concurrent Programs through Active Randomized testing," Proc. ACM/IEEE Int'1 Conf. Software Eng. (ICSE), PP.235-244,2010.

[8] S. Burckhardt, P. Kothari, M. Musuvathi, and S. Nagarakatte, "A Randomized Scheduler with Probabilistic Guarantees of Finding Bugs,"Proc. 15th ASPLOS on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pp. 167-178, 2010.

[9] M.D. Bond, K.E. Coons, and K.S. Mckinley, "PACER: Proportional Detection of Data Races,"Proc. ACM SIGPLAN Conf. Programming Language Design and Implementation (PLDI), pp. 255-268, 2010.

[10] Marino, M. Musuvathi, and S. Narayanasamy, "LiteRace: Effective Sampling for Lightweight Dta-Race Detection," Proc. ACM SIGPLAN Conf. Programming Language Design and Implementation (PLDI), pp. 134-143, 2009.

[11] Y. Wang, T. Kelly, M. Kudlur, S. Lafortune, and S. Mahlke, "Gadara: Dynamic Deadlock Avoidance for Multithreaded Programs," Proc. Eighth USENIX Conf. Operating Systems Design and Implementation (OSDI), PP. 281-294, 2008.

[12] Bensalem, J.C. Fernandez, K. Havelund, and L. Mounier, "Confirmation of Deadlock Potential Detected by Runtime Analysis," Proc. Workshop Parallel and Distributed Systems: Testing and Debugging (PADTAD), pp. 41-50, 2006.

[13] C. K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V.J. Reddi, and K. Hazelwood, "Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation," Proc. ACM SIGPLAN Conf. Programming Language Design and Implementation (PLDI), pp. 191-200, 2005.

[14] S. Bensalem and K. Havelund, "Scalable Dynamic Deadlock Analysis of Multi - Threaded Programs," Proc.Parallel and

Distributed Systems:Testing and Debugging (Padtad-3), IBM Verification Conf.(PADTAD), 2005.

[15] E. Knapp, “Deadlock Detection in Distributed Database Systems,” ACM Computing Surveys, vol. 19, no. 4,pp. 303-328, 1987.

[16] L. Lamport, “Time, Clocks, and the Ordering of Events in a Distributed System, “Comm. ACM, vol. 21, no.7, pp.558-565, 1978.

J. Lethisia Nithiya "THREAD BASED DEADLOCK DETECTION AND MANAGEMENT IN DISTRIBUTED DATABASE "International Journal of Engineering Research and Applications (IJERA) , vol. 8, no. 02, 2018, pp. 56-61