IJCSE International Journal of Computer Sciences and Engineering Open Access

Research Paper Vol.-6, Issue-9, Sept. 2018 E-ISSN: 2347-2693

Clustering the Duplicate Open Crash Reports Based on Call Stack Traces of Crash Reports

Pushpalatha M N[1*], Mrunalini M[2]

[1]Dept. of Information Science and engineering, Ramaiah Institute of Technology, VTU, Bangalore, India
[2] Dept. of Computer Applications, Ramaiah Institute of Technology, VTU, Bangalore, India

[*]Corresponding Author: pushpalathamn1@gmail.com, Tel.: 9739012668

Available online at: www.ijcseonline.org

Accepted: 19/Sept/2018, Published: 30/Sept/2018

Abstract— A computer program such as software application that stops functioning properly is called software crash. Software crash is tedious problem in software development environment. Upon user permission, the crash report which contains the stack traces is sent to the developer or vendor. Software development team receives hundreds of crash reports from many deployment sites. There are many duplicate crash reports are generated, because many users submit the crash reports for the same problem. For analysing each crash reports, it may take more time. This motivates, to present the solution to analyse the crash reports and cluster the duplicate crash reports based on call stack similarities and store them into unique bucket, so that development resources can be optimized. In this paper, clustering the duplicate crash report of open source is proposed based on the similar information in the call stack. Hierarchical clustering technique is used to cluster the duplicate crash reports into unique bucket. Mozilla and Firefox open source crash reports are used for experiment and performance evaluation is done using purity determined the purity of clusters up to 80%. This method helps to increase the efficiency and reduce the number of developers along with an improved time to fix the bug.

Keywords- Crash reports, clustering technique

I. Introduction

Open and closed source software's incorporate built-in error reporting systems. For example, Window has window error reporting system and in a similar way, open source Firefox browser and Thunderbird email client have their own error or crash reporting system. Whenever, an application or software crashes or stops functioning normally, system ask for the user to submit that information to vendor. When user submits the crash information it is stored in the form of crash report in the crash reporting system.

Crash reports contains different frame and each frame contains different information like application name and version, application build date, module name and version, module build date and module offset which is crashed and its call stack traces. This information helps developer to understand the problem and fix the function that contains problem causing the crash. Fixing a crash means identifying the faulty functions which is time consuming and difficult task. Some of the crash reporting systems collects the crash reports automatically and organize them into multiple buckets. The crash reports collected by crash reporting system are used for debugging. When the crash reports caused by same bug are collected they are stored in more than one bucket by crash reporting systems. It is difficult for developer to analyse the same bug for multiple times and it is time consuming.

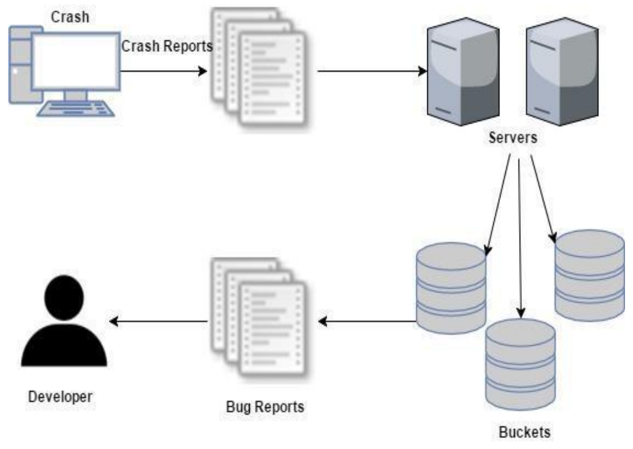

Figure 1: An overview of crash reporting system

Windows error reporting system receives a large number of crash reports from many deployment sites. The received crash reports are later grouped into different buckets automatically, by window error reporting system as shown in the Figure 1 [4]. In a similar way, open source crash reporting also receives a lot of crash reports from all over the world. All Mozilla related crash reports are stored in Mozilla crash reporting system.

In this paper, Crash reporting system is proposed, which analyse the crash reports and pre-processes the crash reports to retain the data which is required. Then similarity between crash reports is determined using position dependent model and prefix matching method proposed in [4]. Once duplicate crash reports are found we cluster the similar crash reports into one unique bucket. We are using data mining clustering technique to cluster the duplicate crash reports. This crash reporting system helps to reduce the debugging effort and time of the developers and also helps to prioritize which bugs needs to be fixed first.

The paper is organized as follows, Section I gives introduction, Section II gives related work on crash reports, Section III about Design and Implementation and Section IV gives conclusion of our work along future enhancement.

II. RELATED WORK

Related work on the crash reports is analyzed in [1]. Functional Frequency, cyclometric complexity, Inverse Bucket Frequency, Average Distance to crash point for calculating the scores and later ranked according to score [3]. Along above four metrics [3] in [2] used Number of times where function is referred in static call graph. (NC) for finding the scores and ranked the functions. The function which is having highest score is given more importance for debugging than lower. The fault will be checked on the top functions. [4] Duplicate crash reports are clustered together using the hierarchical clustering algorithms for the crash reports of windows error reporting system. In our work, we used the concept from [4] for clustering the open source crash reports. In [6] used natural language for predicting the duplicated bug reports, i.e., is used only summary and description information in bug report for finding whether the new bug report is duplicate of already available bug reports or not. In [7] used textual information of bug report along with stack traces for predicting the duplicate bug reports. [8] Experiment is done on the Thunderbird and Firefox crash reports and found that only top crashes around 10 to 20 will give large majority of crash reports. [8] Predicted the top crashes of new releases by training on the features of top crashes of past releases using machine learning techniques. This will help to increase the software quality by quickly fixing the top crashes first, which gives good user experience.

III. DESIGN AND IMPLEMENTAION

3.1 Design model for clustering duplicate crash reports

Crash reports are loaded to the software. After pre-processing is done on crash reports. In pre-processing the information which is required to find the similarities are retained. These pre-processed crash reports are helpful in determining duplicate crash reports. Then duplicate crash reports are determined using Position Dependent Model method and Prefix matching method [4]. Once the matching is done we get the percentage of similarities between the crash reports which helps us to cluster the duplicate crash reports. The Duplicate crash reports are clustered into one unique bucket. This helps to reduce the time for fixing the bug and also helps to prioritize the bugs to be fixed. After clustering the Duplicate crash reports the purity of cluster is determined to evaluate its performance. Once the duplicate crash reports are organized into unique bucket the developers can fix the bug without much effort and time.

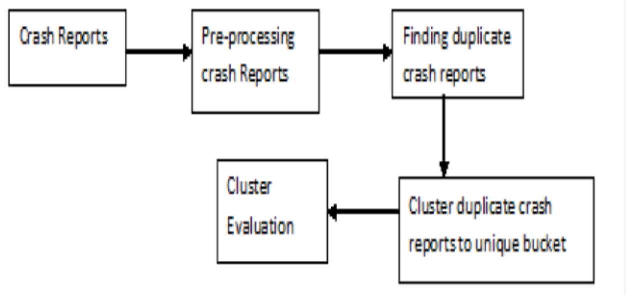
Crash Reports
Pre-processing crash Reports
Finding duplicate crash reports
Cluster Evaluation
Cluster duplicate crash reports to unique bucket

Figure 2: Design model for clustering duplicate crash reports

Figure 2 shows design model for clustering duplicate crash reports. The steps involved are discussed below.

The system has following steps
1. Crash reports: Open source crash reports are taken from the Mozilla crash reporting system
2. Pre-processing crash reports: This pre-processes the crash reports to remove the information which is not required.
3. Finding duplicate crash reports: Matching call stacks similarities to determine the duplicate crash reports.
4. Cluster duplicate crash reports to unique bucket: This step clusters the duplicate crash reports based on the call stack similarities into unique bucket.
5. Cluster evaluation: this step determines the purity of clusters which helps to evaluate the Clusters.

3.2 Steps Involved in Implementation Phase

Implementation phase of crash reporting system has following steps:

1. Loading crash reports into source code and Pre-processing the crash reports.
2. Finding the similarities between the call stacks of crash reports.
3. Clustering the duplicate crash reports based on similarities.
4. Evaluation of clusters.

3.2.1 Pre-processing

In pre-processing crash reports are taken as input. These crash reports are taken from Mozilla crash reporting system. Crash reports include crash info, crashing thread, modules, system info, and threads. In our work, considered the frames of crash reports which are present in crashing thread information. Each frame contains different information like application name and version, application build date, module name and version, module build date and module offset which is crashed and its call stack traces.

These crash reports contain more information which is not required. To remove information which is not required we are performing pre-processing step. In pre-processing the crash reports are analysed and pre-processed

Implementation steps involved in pre-processing are:
Step 1: Taking crash reports as input.
Step 2: Analysing each crash reports and identifying the information which in not required.
Step 3: Pre-processing the crash reports to retain information which is required.

3.2.2 Call Stack Similarities

Once the crash reports are pre-processed next step is to find the call stack similarities to find the duplicate crash reports. Once the crash reports are pre-processed it contains information about frames which is required to find duplicate crash reports. In our projects we are determining the duplicate crash reports by considering the function which is crashed by using prefix matching method and position dependent model method.

In Position dependent model more weight should be given to top frames which are close to top frame because the frame which cause for crash occurs at the top frames of the stack trace. The alignment offset between the matched crash reports should be very small. In our work, we are keeping offset threshold as three and we are considering only seven frames from the top of the stack trace. Then we are using prefix matching to find duplicate crash reports. In prefix matching method we are comparing function attribute of one frame with function attribute of other frames in remaining crash reports. Once the matching is done the matching percentage is displayed.

Implementation steps involved in call stack similarities are:
Step 1: set distance of frame from top frame as 7.
Step 2: set the alignment offset between matched frames among the crash reports into 3.
Step 3: set matching threshold as 50%.
Step 4: perform the prefix matching method to determine the duplicate crash reports.
Step 5: display the matching percentages of the crash reports

3.2.3 Clustering

Once the matching of frames in crash reports is done we group the duplicate crash reports by clustering them. We are keeping matching threshold as 50. If the matching percentage between the frames in crash reports is 50 or above 50 we group those crash reports into one cluster. In the similar way we are grouping all the matched crash reports into respective clusters. For clustering we are using the idea of hierarchical clustering. After finding the duplicate crash reports we are merging the similar crash reports into one cluster.

Implementation steps involved in clustering are:
Step 1: finding the crash reports with matching similarity as 50 or above 50 %.
Step 2: group the duplicate crash reports into respective cluster.
Step 3: display the clusters with similar crash reports

3.2.4 Cluster Evaluation

When all the duplicate crash reports are clustered we perform the evaluation of clusters. In evaluation we are finding whether all the duplicate crash reports are clustered into respective group or not. To perform the evaluation we are determining the purity of the cluster. This purity helps us to determine whether all duplicate crash reports are grouped into respective cluster or not. We determine the purity using the following formula 1.

$$Purity = \frac{1}{N}\sum_{k=1}^{n} \max_{c} |y_i \cap z_c| -----[1]$$

Where N is the number of crash reports, n is the number of clusters, y_i is cluster in y and z_c is the maximum number of crash reports correctly classified for that cluster y_i. Purity gives the accuracy of clustering algorithm. It sum ups the correctly classified class per cluster over the total number of crash reports.

Implementation steps involved in evaluation of clusters are:
Step 1: determine the total number of clusters.

Step 2 : find count of crash reports which are not grouped into respective cluster.
Step 3 : determine the purity of clusters.
Step 4 : display the purity.

IV. Conclusion and Future Scope

4.1 Conclusion

Crash reporting system is used for clustering the duplicate crash reports based on the similarities between the call stack traces. This system helps to solve the debugging efforts, increases the efficiency and reduces the number of developers with an improved time to fix the bug. In the proposed crash reporting system, based on the analysis of the crash reports taken from Mozilla and Firefox crash dumps, the crash reports are pre-processed to retain the required information and call stack similarities are determined to know the matching similarities between the crash reports and duplicate crash reports are found. The duplicate crash reports are grouped into respective clusters using hierarchical clustering technique. The clusters are evaluated to know whether all the duplicate crash reports are grouped into respective clusters or not. Cluster evaluation is done by finding the purity of clusters; the results have shown upto 80% for open source crash reports. The literature shows [4] 88% of cluster purity for crash reports of windows error reporting system.

4.2 Scope for further enhancement

The crash reporting system to cluster the duplicate crash reports for Mozilla crash reports can be enhanced to find the duplicate crash reports and cluster them for the projects of other organizations and other open source software's.

Refereneces

[1] Asha Ramaraddi Belahunashi, Pushpalatha M N," A Survey on analysing the crash reports of software applications", International Research Journal of Engineering and Technology , Volume 4, Issue 6, pp.1014-1017, June 2017.

[2] Divya R S, Pushpalatha M N, "Software CrashLocator: Locating the Faulty Functions by Analyzing the Crash Stack Information in Crash Reports", International Journal of Advanced Engineering, Management and Science (IJAEMS), Vol-2, Issue-5, pp.269-273, May- 2016

[3] Rongxin Wu, Hongyu Zhang, Shing-Chi Cheung, and Sunghun Kim, "CrashLocator: Locating Crashing Faults Based on Crash Stacks", ISSTA 2014 Proceedings of the 2014 International Symposium on Software Testing and Analysis, Pages 2014-214, 2014

[4] Yingnong Dang, Rongxin Wu, Hongyu Zhang, Dongmei Zhang, and Peter Nobel, "Rebucket: a method for clustering duplicate crash reports based on call stack similarity". In Proceedings of the 34th International Conference on Software Engineering, pages 1084– 1093. IEEE Press, 2012.

[5] P. Runeson, M. Alexandersson, and O. Nyholm, "Detection of Duplicate Defect Reports Using Natural Language Processing", in Proc. ICSE 2007, Minneapolis,USA, pp. 499-510, May 2007.

[6] X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun, "An approach to detecting duplicate bug reports using natural language and execution information", in Proc. ICSE'08, Leipzig, Germany, pp. 461-470, 2008

[7] D. Kim, X. Wang, S. Kim, A. Zeller, S. Cheung, and S. Park, "Which crashes should i fix first? Predicting top crashes at an early stage to prioritize debugging efforts", IEEE Transactions on Software Engineering, pp. 430-447, 2011.

Authors Profile

Ms. Pushpalatha M N pursed Bachelor of Engineering from University BDT college of Engineering, Kuvempu University in 2003 and Master of Technology from Ramiah Institute of Technology, Visvesarya Technological University in year 2006. she is currently pursuing Ph.D. and currently working as an Assistant Professor in Department of Information Science and Engineering, Ramaiah Institute of Technology since 2006. Her area of research is in Data Mining and Software engineering.

Dr.M.Mrunalini, completed her PhD in Computer Applications from Visvesvaraya Technological University, Belgaum, in 2015. She is currently working as an Assistant Professor in Department of Computer Applications. Her areas of interests are Software Engineering, Data Warehouse-ETL tools , Software Security and Big data.