Software Analysis Using Code Metrics

Darren Blanckensee — 1147279

***Abstract*—Due to the rising popularity in the area of Internet of Things (IoT) there has been significant research done on how to implement edge and fog computing in order to improve the speed and efficiency of any communication between edge devices, the fog gateway and the cloud if necessary. Edge and fog computing involve data transmission whether it is between edge devices, between edge devices and fog nodes or between fog gateways and the cloud. All of these forms of transmission can be made more efficient and faster using certain filtering and compression techniques. Using filtering and compression techniques will speed up response time and use less bandwidth which will not only improve user experience but make the edge and fog computing processes more efficient in terms of space, time and energy. This would have direct effects on the efficiency of IoT systems.**

I. Introduction

SOFTWARE is being developed at an ever increasing rate. It is important when writing programs to follow good programming practices and to hold the code one writes to a certain standard so as to ensure a high quality product. Often when software projects are large with many different classes and files with thousands of lines to keep track of it becomes increasingly difficult to maintain a standard as it becomes impossible for developers to read all the lines of code and make sure that the quality requirements are met. One way to address this issue is to make use of software metrics. Software metrics are standards of measurement that provide developers with quantifiable measurements of various characteristics of the code they have developed.

One of the most simple metrics often used as an example in software is the lines of code metric (LOC) which measures how many lines of code exist in a program. This metric alone does not however provide the developer with useful information relating to the quality of the software and whether or not good programming practices are being followed. As So and so once said Puthequoteherepleasedarren. Other more useful metrics exist and the use of these in analysing software projects is the purpose of this report. All code in this project (including the metrics tool used) is written in Python.

The Chosen Metrics section details which metrics have been selected to allow for sound analysis of a code base that provides developers with useful information that could be used to maintain a standard of coding. Along with the selected metrics, explanations of each metric will be provided so as to explain the importance of each of the chosen metrics and how they should be used. The Code Base Analysis section covers the analysis of the authors own code base along with three major releases of the open source software, Freevo Media Library.

II. Chosen Metrics

The metrics being used in this project are the following:

- Cyclomatic Complexity
- Maintainability Index
- Coverage
- Halstead Metrics
- Dependencies

III. Code Base Analysis

IV. Experimental Setup and Computational Model

The code will be stored on a public repository on Github as will any papers or files relating to the project that are used to aid the authors. Anybody who wishes to validate the results of this project will then be able to download the source code and required files and will be able to test it on their own edge devices.

V. Preliminary Results

whoops

VI. Explanation of the Rest of the Work to be Accomplished

There are a number of tasks that still need to be done. Firstly the authors need to aqquire the smartphones and tablets that will be used as edge devices, root them and install ubuntu on them. Secondly the data that will be used to simulate the data that is produced by the sensors needs to be aqquired. Two types of data are needed data that is generated in realtime so as to mirror a device that monitors real time information such as weather stations that transmit data as it is generated

and data that is generated and is only trasnmitted after a certain amount of time or after a certain amount of data has been collected for example weekly electrical load profiles for a smart house where data is collected throughout the week allof which is only transmitted at the end of each week.

After programming the Android devices via the SD cards, they are to be tested accordingly in order to determine whether they perform the required tasks. The authors will also research and implement the optimum techniques for accelerated computation.

More research into what various compression algortihms and filtering techniques exist in the areaa of edge and fog computing is to be done. The top performing algortihms and techniques are then to be implemented on the smartphones and tablets (edge devices). Actually using the compression and filtering techniques when transmitting the data between devices and the fog gateway should be at the discretion of the authors so that when testing takes place comparisons are made easy. Additionally, tests need to be developed and for this multiple types of queries have to be identified as the basis for the tests. At the time of writing the types of queries that are to be used are select queries (with conditions), mathematical select queries that return the sum or the aggregate of the data in the selection, join queries and possibly insert and delete queries.

VII. Methods for Validations of Expected Results and Exceptions

At the time of writing this proposal it was unclear as to which languages and libraries would be used and therefore the most accurate timing method could not be determined. Once this information is known , research will be done to find the most accurate timing procedure. The time will be started when the query takes place and will stop when the result is returned. It is expected that when compression and filtering is implemented the time will be significantly shorter than when there is neither filtering nor compression. It is unclear how much faster it will be with the filtering and compression however that is part of the projects goal, to find out how much faster edge and fog computing queries can be run.

Along with the time measurement there need also be another measurement that takes place to determine whether the query has returned the correct result. Accelerating the process of edge and fog computing is important however so is the integrity of the data as it is not helpful reducing the response time if this incurrs substantial errors. To validate the query results the result of the query when using compression and filtering will be compared with the result of the query when compression and filtering are not used. If identical then both results were succesfully generated and if not then a closer look will need to be taken at the data returned by the compression and filtering technique. It is important to note that bloom filters when used do, although not very often, return false positives. False positives would mean returning data that should not be there however without the filtering there would be significantly more data that need not be there so it is assumed that false positives are not fatal as overally the data will still have been significantly compressed and therefore the response time reduced.

VIII. Risk Management

An immediate risk that is facing the project is damaging the devices used, which could possibly compromise the reliablilty of the data collected and/or used.

IX. Literature Review

According to [1], there exist a number of compression algorithms in edge computing. Some of the more popular types of edge compression are various forms of time series compression whether it be sampling or representing a time period by the average of all the values within that time period [2]. This is a valid form of compression however it introduces a level of fuzziness as the values are not true values and the value at any given time can never be exact. Depending on the time period over which the average is calculated these averages'accuracy will vary. The more accurate the average the less compressed the data, this method therefore is not ideal.

Another method suggested by [4] is using perceptually important points to represent the data. This method also introduces fuzziness and like the previous method depending on the number of perceptually important points the representations accuracy varies in a manner that is inversely proportional to the compression rate. Furthermore these compression methods are suited for data sets as opposed to data streams as stated in [1].

bbbbbbeeeeeeeeeeeffffffffffooooooooooorrrrrrrrrreeeeeeeeeee

The suggested method for this project is the use of a Bloom filter which is a way to test if a piece of data exists in a set or not.

X. Schedule and Time-line

XI. Summary of Proposal and Planned Additional Work to Complete

References

[1] Apostolos Papageorgiou, Bin Cheng, Erno Kovacs, Real-Time Data Reduction at the Network Edge of Internet-of-Things Systems. NEC Laboratories Europe Heidelberg, Germany, 2015.

[2] B.-K. Yi and C. Faloutsos. Fast Time Sequence Indexing for Arbitrary Lp Norms. In Proceedings of the 26th International Conference on Very Large Data Bases, VLDB 00, pages 385394. Morgan Kaufmann Publishers Inc., 2000.

[3] F. Chung, T. Fu, R. Luk, and V. Ng. Flexible time series pattern matching based on Perceptually Important Points. In International Joint Conference on Artificial Intelligence, Workshop on Learning from Temporal and Spatial Data, pages 17, 2001.

[4] Weisong Shi, Fellow, IEEE, Jie Cao, Student Member, IEEE, Quan Zhang, Student Member, IEEE, Youhuizi Li, and Lanyu Xu. Edge Computing: Vision and Challenges.