IJIREEICE

ISSN (Online) 2321-2004
ISSN (Print) 2321-5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering

Vol. 8, Issue 5, May 2020

A Review of Logging Utility for Software Application

Amisha Singh[1], Shushrutha K.S. [2]

Department of Electronics and Communication Engineering, RV College of Engineering, Bangalore-59, India[1]

Associate Professor, Department of Electronics and Communication Engineering, RV College of Engineering, Bangalore-59, India[2]

Abstract: In recent years, software engineers are thriving to develop reliable and stable applications for embedded systems. Logging utility plays an important role in recognizing the errors and faster debugging process by being a back-burner activity. Software application logging is helpful however it has cost also like memory, battery life, etc. Moreover, previous examinations on logging show that optimal logging is challenging for software designers. The aim of this study is to orchestrate significant investigations on techniques used for solving various resource challenges while logging. One such technique is Variable rate logging (VRL) which stops the recording of the log when no change in state of activity is observed. The standstill and indoor/outdoor detection algorithm are used to achieve VRL. For memory management different storage techniques are used. Log records are stored in flash memory to keep a record of recent read/write operation to ease the debugging process. One more memory management technique used for applications interacting with database are in-memory write-ahead logging (IMWAL), this ensures no duplication of the data to be stored. In the end, various guideline to generate high quality logs are discussed.

Keywords: Logging techniques, Software applications, Maintenance and Debugging.

I. INTRODUCTION

Nowadays, usage of software application has increased enormously, and many important activities are carried out using these applications by personals and various commercial organizations. The failure of these application in any scenario could bring certain activities to halt causing inconvenience and leading to loss of time as well as monetary losses. As a procedure to accumulate the runtime information of structure direct, logging practice is progressively increasingly in present day programming improvement and upkeep. By having a logging utility with the source code, key information about the conduct of programming systems could be traced and recorded in logs. These logs expect a basic activity in researching, disappointment taking care of and structure recuperation, framework assessment, therefore on, which help a ton in the activity and upkeep of programming frameworks.

Logs are efficient and capable for finding and settling the issues the product frameworks are turning out to be increasingly productive in the development stage. Since logging practice is the starting stage to make logs, various experts are dedicating to create bound together logging details or apparatuses to control different designers doing logging or even help programmed logging. Logs are generated in substantial amount which would create various issues like memory problems, causing the slowdown of application and make it hard to extract or find the necessary information for debugging. The following section discusses algorithm for resource management for logging in various applications, followed by the logging techniques to generate high quality logs.

II. DISCUSSION

Logging is regularly consigned to a background action while architecting, structuring and creating applications or in any event, which upgrades the applications. Various organizations have different ways to store logs. Some might have dedicated logger devices whereas others may have servers and some cases might require the logs to be stored on the device where the application is running as in the case of embedded systems.

A. *Algorithms for Resource Management for Logging in Embedded systems*

This is the section describes various algorithm used for management of the resources in the embedded system. One of the most important aspect to consider while designing the logger utility is the amount of resources consumed in order to run that in the background. For devices like mobile and smart watches the resources are minimal such as battery life, memory etc., hence the designer would ensure the resources last for as long as possible. Logging for location is enabled in the devices which provide logs and information which many applications use for their purpose. Past research identified with recording and taking care of area logs neglected to truly consider substantial battery utilization.

To cut down the pace of logging, the gadget must utilize area sensors less much of the time and extend rest time of zone sensor. During the rest period of the area sensor, in any case, the gadget can't recover and record huge changes on client's area, for example, stops, moves, doorways, and flights. In this manner, as the rate of logging diminishes it broadens battery life, it is certain that the quality of area log statements, is harmed. This situation depicts battery life/resolution tradeoff. Hence Variable Rate Logging (VRL) [1] was introduced to overcome the battery consumption challenges. VRL can be implemented using Halt discovery algorithm or Indoor /Outdoor Detection Algorithm. This technique has shown improvement in battery life by 43 % than regular rate of logging.

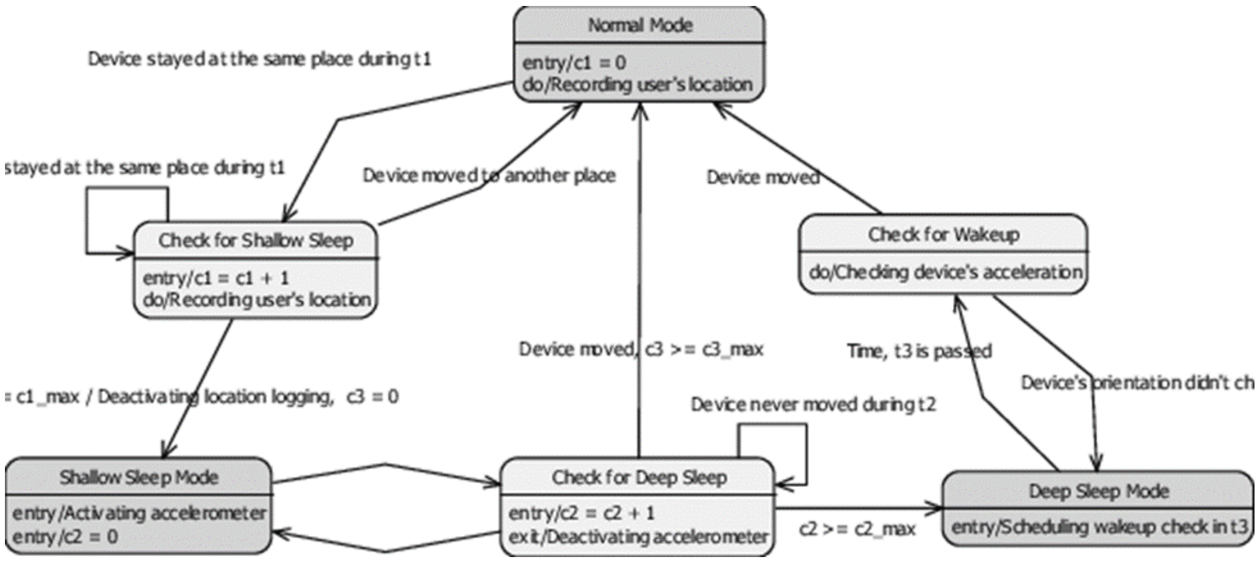
Normal Mode
entry/c1 = 0
do/Recording user's location
Device stayed at the same place during t1
stayed at the same place during t1
Device moved to another place
Device moved
Check for Shallow Sleep
entry/c1 = c1 + 1
do/Recording user's location
Check for Wakeup
do/Checking device's acceleration
Device moved, c3 >= c3_max
Time, t3 is passed
Device's orientation didn't ch
= c1_max / Deactivating location logging, c3 = 0
Device never moved during t2
Shallow Sleep Mode
entry/Activating accelerometer
entry/c2 = 0
Check for Deep Sleep
entry/c2 = c2 + 1
exit/Deactivating accelerometer
c2 >= c2_max
Deep Sleep Mode
entry/Scheduling wakeup check in t3

Fig. 1 Standstill Detection Algorithm [1]

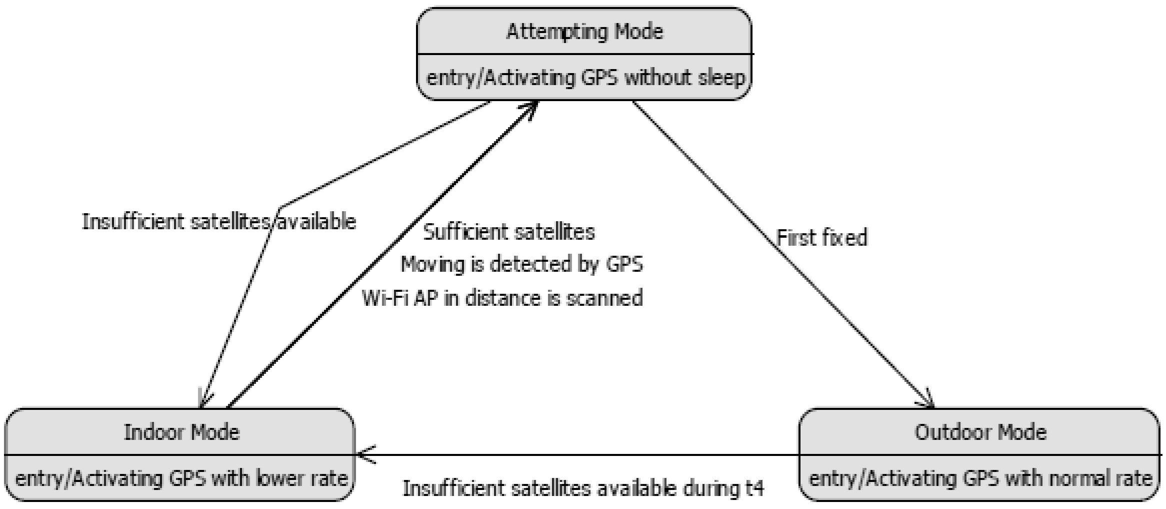
Attempting Mode
entry/Activating GPS without sleep
Insufficient satellites available
Sufficient satellites
Moving is detected by GPS
Wi-Fi AP in distance is scanned
First fixed
Indoor Mode
entry/Activating GPS with lower rate
Insufficient satellites available during t4
Outdoor Mode
entry/Activating GPS with normal rate

Fig. 2 Indoor or Outdoor Detection Algorithm [1]

Memory is one the most important resource in all the application and devices, especially in embedded systems. Embedded system is employed to do various jobs with real-time computing constraints. The storage devices used in these systems also play an important role in management of resources. In the recent years, flash memory has replaced hard disk which was used in many devices due to non-volatility, shock resistance, low-power consumption and huge-capacity. Flash memory has proven improved performance such as execution time of the embedded devices.[14] proposes a method for maintaining a LOG record that holds the history of recent write and update along with the information for the flow of the application. The log record holds the data which show the development of the application by a team of developers, this helps to efficiently merge, delete and review the various changes done in the product development stage.

Many applications interact with the database, each time a information is to be added to the database they first written to the log record and then the actual database. This is done using write-ahead logging to provide atomicity (with UNDO info) and durability (with REDO info). This results into a lot of data overhead and utilize a lot of memory resources. An efficient in-memory write-ahead logging (IMWAL) is introduced in [15]. IMWAL ensures that new data for a write operation for embedded database is written to the Non-volatile RAM as write-ahead data log. This is used again to update the embedded database file by remapping the memory address and therefore it reduces the additional memory storage requirement and writes. The proposed architecture is shown in Figure 2.3.

Hence, the VRL results in the improvement of battery life and various storing techniques in the memory bring in efficient usage of the memory available. This enhances the overall performance of the embedded system.

B. *Logging practices used to generate High quality logs:*

The quality of the logs recorded play an important role in debugging as well as in overcoming the resource challenges. The next challenge for designing a logger utility is the output latency. Many applications are developed on a distributed system and involve dependencies on the network. To record the activities or the logs of the application, there are 2 known methods 1) Message logging and 2) coordinated checkpointing. Despite what might be expected, [2] present various examinations demonstrating that for figure concentrated applications executing in equivalent on lots of workstations, message logging has higher disillusionment free overhead than coordinated checkpointing.

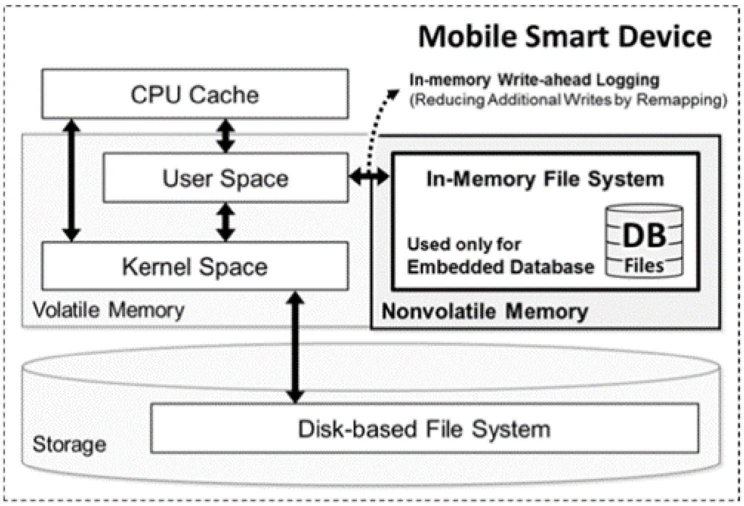

Figure 3 Prosed architecture for in-memory write ahead

Message logging conventions, many cases, bring about a lot shorter yield inertness than facilitated checkpointing. Consequently, message logging ought to be utilized for applications including significant interaction with the clients.

Certain software application might be developed for on-site monitoring. For example, in an oil drilling field, many sensors might to employed to monitor and do the work required. Software application written would instruction as to what to do under certain conditions. These applications provide logs which are transferred through a network to the storage facility. Hence, the results from the on-site drilling monitoring and comprehensive logging instrument [3] can be used to guide proper drilling operations and to gain information of the reservoir. The logs are transferred via GPRS or CDMA network to the server, to realize an intranet network as shown in the below figure 2.3.

To get only valuable logs various filters can be designed while developing the logging utility for the application. These filters are called log levels. There are various log levels defined for storing information on the bases of the events that occur during the running of the applications. Defines log levels are ALL, DEEBUG, INFO, WARNING, ERROR, FATAL and NONE. Every log statement to be stored in a log file should have important information, including an error message, source of origination, and a time stamp to record the time of occurrence, also a logging level. Logging proclamations ought to have no or little effect on the application's conduct.

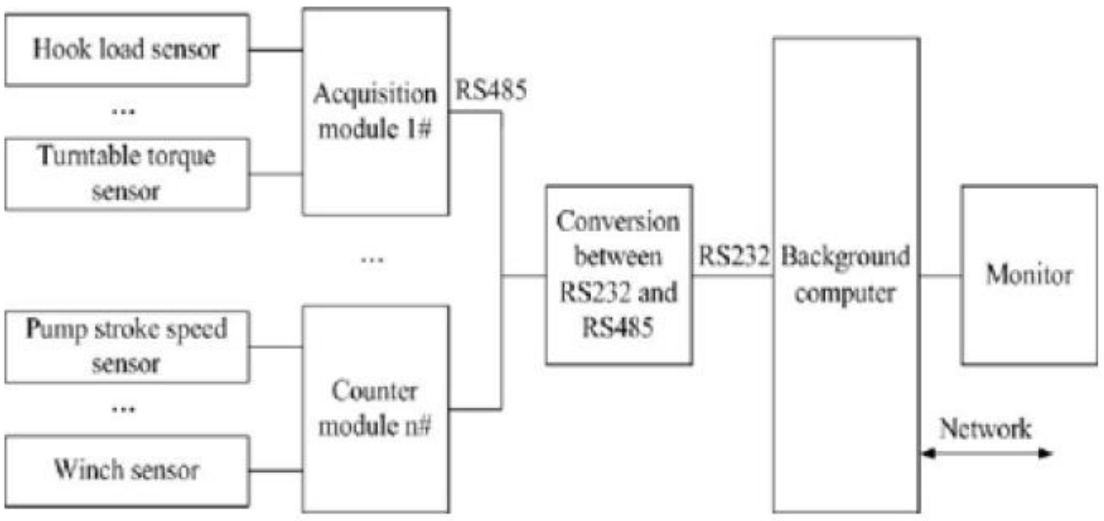

Fig. 4 The Logging Data Acquisition System [3]

Standard workday today incorporates utilization of programming applications that empower quicker undertaking preparing and can do as such for a few assignments in equal for us. All things considered, additionally the finest applications can behave abnormally or may not perform task accurately. For examining the abnormal behavior of a product app, logging has proven to be significant. By looking at app log statements, programming designers are able discover the reason and obtain a solution or fix with fixed bugs. Then again, logging can proffer a huge measure of information which might result in memory flooding. This is one of the significant reasons for not logging every information available. The choice regarding what information to be logged is usually decided during the product advancement stage.

The guidelines [4] decisions on information to be logged for application will result in enhancement in the quality of logging. When considering logging for the product application advancement, it is critical to settle on what information to log and step by step instructions to log. Each log statement should, record what happened, when it happened, what triggered the event, and why it happened. As a result of these guidelines, a statement used for logging an event should be able answer queries in the figure 2.4, it also shows the approaches used to answer the questions.

The engineers were recording the client ID of the logged client once, at start of the application, class name with a process-ID, the time stamp, also storing special case when occurred. The recorded logs contained very basic and essential information for business (e.g. user authorized, access available, new projects created etc.). The data recorded included a lot of info which was recorded using the logging utilities by the app itself rather than by the engineers. This was causing the documents getting incredibly enormous in barely any time. The size of the largest files was available inside XML settings and as soon as the limit was attained, log statements were stored in new record automatically, the product engineers would not mediate this process. Hence, above mentioned technique helped in keeping the number of log files less.

Table 2.1 Necessary information each log statement should contain [4]

Question to Answer in the Logging	Approach used in investigated java projects
What did happen?	Stack trace of an exception logged Direct log messages informing that a method was successfully finished
When did it happen?	Server time stamp
Where did it happen?	Java class name, called method
Who triggered the event?	User name
Who was involved?	User name
Why did it happen?	Stack trace of an exception logged Direct log messages informing that a method was successfully finished
What caused it to happen?	Stack trace of an exception legged

IJIREEICE

ISSN (Online) 2321-2004
ISSN (Print) 2321-5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering

Vol. 8, Issue 5, May 2020

DevOps alludes to a huge number of practices committed to quickening present day programming building process. It breaks the boundaries between programming improvement and IT activities and means to deliver and keep up top notch programming frameworks. Software logging is generally utilized in DevOps. However, it faces four major challenges:

1. No existing rules on creating top notch logging code.
2. Trouble in keeping up and advancing source code for logging.
3. Limited component for quality input.
4. Heterogeneous and complex telemetry information.

Certain automated approaches are proposed [5] to enhance programming of logging utility in DevOps. This is done by utilizing different sorts of programming repositories (e.g., historical repositories, communication repositories, bug repositories, and runtime repositories) as Figure 2.4 shows.

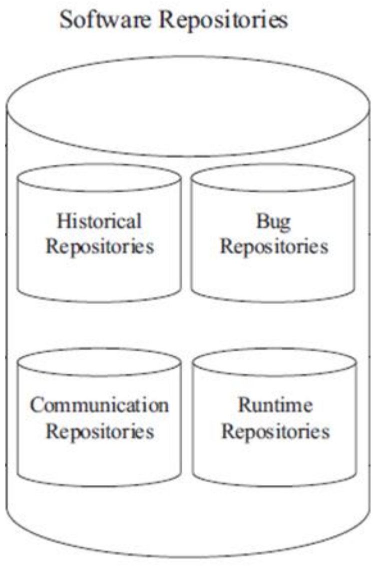

Figure 2.5 Various repository proposed for Challenges in DevOps [5]

Historical repository helps in establishing the guidelines for logging utility that produces high quality logs and helps in maintaining and improving the logging code. Bug repository studies the issues in logging code and evaluating various approaches to solve the issues. Communication repository provides feedback on the quality of the logs produced and provide a room for improvement in the logging approach currently used. Runtime repository estimates the code coverage by the execution logs and helps in corelating the complex data for problem diagnosis.

Hence, the quality of logs plays an crucial role in debugging and reporting the correct source of failure. As the quality of logs increase the amount of log generated decrease which in turn reduce the consumption of resources.

III.CONCLUSIONS

Logging practice assumes an imperative job in present day programming improvement furthermore, upkeep. This paper presents an orderly audit on currently available researches about logging techniques in programming designing. There are many challenges faced in designing of the logging code such as system quality attributes (battery, memory, execution speed, security, etc.). First challenge of battery life in embedded devices, continuous running of logging utility in the background consumes a lot of battery life. To improve the battery life of the devices variable rate logging is implemented using standstill and indoor/outdoor algorithm. Logging utility design also face the challenge of memory usage, to tackle this flash memories are used with log record and faster a read and write to improve performance of the device. On more technique used to solve the memory usages is in-memory write-ahead logs, this help reduce write overheads. Various applications require to transfer the log data through a network for various analysis purposes, hence high quality and useful information must be sent over the network rather than enormous number of useless logs. To produce high quality logs several questions, need to be answered while developing the logging utility. The most important contents to log an event are when did it happen, location of the program that triggered the event, information message stating the event. Since there no strict guidelines for designing the logging utilities various software repositories are used efficient and high-quality logging utilities. These repositories provide quality feedback which helps in evolving and maintaining the logging code, help in problem diagnostics and evaluating code coverages.

IJIREEICE

ISSN (Online) 2321-2004
ISSN (Print) 2321-5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering

Vol. 8, Issue 5, May 2020

REFERENCES

[1] C. Lee, M. Lee and D. Han, "Energy-Efficient Location Logging for Mobile Device," 2010 10th IEEE/IPSJ International Symposium on Applications and the Internet, Seoul, 2010, pp. 84-90.
[2] E. N. Elnozahy and W. Zwaenepoel, "On the use and implementation of message logging," Proceedings of IEEE 24th International Symposium on Fault- Tolerant Computing, Austin, TX, USA, 1994, pp. 298-307.
[3] Li Peng, "The development of modern comprehensive logging instrument and integrated applications of logging information," IEEE Conference Anthology, China, 2013, pp. 1-3.
[4] G. Rong, Q. Zhang, X. Liu and S. Gu, "A Systematic Review of Logging Practice in Software Engineering," 2017 24th Asia-Pacific Software Engineering Conference (APSEC), Nanjing, 2017, pp. 534-539.
[5] B. Chen, "Improving the Software Logging Practices in DevOps," 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), Montreal, QC, Canada, 2019, pp. 194-197.
[6] G. C. Richard and M. Singhal, "Using logging and asynchronous checkpointing to implement recoverable distributed shared memory," Proceedings of 1993 IEEE 12th Symposium on Reliable Distributed Systems, Princeton, NJ, USA, 1993, pp. 58-67.
[7] S. Lal, N. Sardana and A. Sureka, "LogOptPlus: Learning to Optimize Logging in Catch and If Programming Constructs," 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), Atlanta, GA, 2016, pp. 215-220.
[8] M. Dávideková and M. Gregu Ml, "Software Application Logging: Aspects to Consider by Implementing Knowledge Management," 2016 2nd International Conference on Open and Big Data (OBD), Vienna, 2016, pp. 102-107.
[9] J. Jeong, C. H. Park, J. Huh and S. Maeng, "Efficient Hardware-Assisted Logging with Asynchronous and Direct-Update for Persistent Memory," 2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Fukuoka, 2018, pp. 520-532.
[10] M. Nagappan and M. A. Vouk, "Abstracting log lines to log event types for mining software system logs," 2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010), Cape Town, 2010, pp. 114-117.
[11] A. Razavi and K. Kontogiannis, "Pattern and Policy Driven Log Analysis for Software Monitoring," 2008 32nd Annual IEEE International Computer Software and Applications Conference, Turku, 2008, pp. 108-111.
[12] N. Vermaak, N. Gurusinghe, T. Ariyarathna and R. Gouws, "Data logger and companion application for time-of-use electricity," 2018 IEEE International Conference on Industrial Electronics for Sustainable Energy Systems (IESES), Hamilton, 2018, pp. 465-470.
[13] Yanke Hu, Xiwang Zhang and Liangmin Hu, "A fast approach to enable mobile apps with Geo-Location logging and reporting," 2015 23rd International Conference on Geoinformatics, Wuhan, 2015, pp. 1-4.
[14] S. Yang and C. Wu, "A Low-Memory Management for Log-Based File Systems on Flash Memory," 2009 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, Beijing, 2009, pp. 219-227.
[15] S. Ryu, K. Lee and H. Han, "In-memory write-ahead logging for mobile smart devices with NVRAM," in IEEE Transactions on Consumer Electronics, vol. 61, no. 1, pp. 39-46, February 2015.