A LOG TOOL FOR SOFTWARE SYSTEMS ANALYSES

Igor Karelin, Boris Lyubimov, Tatyana Gavrilova

Abstract: The article presents a new type of logs merging tool for multiple blade telecommunication systems based on the development of a new approach. The introduction of the new logs merging tool (the Log Merger) can help engineers to build a processes behavior timeline with a flexible system of information structuring used to assess the changes in the analyzed system. This logs merging system based on the experts experience and their analytical skills generates a knowledge base which could be advantageous in further decision-making expert system development. This paper proposes and discusses the design and implementation of the Log Merger, its architecture, multi-board analysis of capability and application areas. The paper also presents possible ways of further tool improvement e.g. - to extend its functionality and cover additional system platforms. The possibility to add an analysis module for further expert system development is also considered.

Keywords: Knowledge Base, Software Systems Analyses, Log Tool, Telecommunication System.

ACM Classification Keywords: B.8.0 Hardware – Performance and Reliability - General, D.2.5 Software – Software Engineering - Testing and Debugging, H.3.4 Information Systems - Information Storage and retrieval - Systems and Software.

Conference: The paper is selected from XIVth International Conference "Knowledge-Dialogue-Solution" KDS 2008, Varna, Bulgaria, June-July 2008

Introduction

Nowadays Telecommunication Infrastructure feels growing consumers' demand for high performance systems due to the changed character of traffic and increasing number of subscribers. The situation requires the particular level of system reliability and availability - both are the essential characteristics of a modern telecommunication system.

Such strict requirements should be supported by appropriate architectural solutions, such as redundancy of all the elements in the platform and mechanisms (both in hardware and software parts of the platform) providing rapid automatic recovery from failures.

But the growth of a system leads to the increase of data level required for comprehensive systems state analysis and supervision and life cycle description. The majority of this information is represented as log files – text files consisting of time-stamped status and error messages detailing the operational history of a given piece of software.

As it was already mentioned system state and lifecycle are described by the huge amount of jumbled data produced and distributed by multiple software units. These data represent the behavior of each unit on a long time scale.

The problem is that during system analysis (failure investigation for ex.) the search for information is time-consuming. Maintenance engineer should manually filter and sort data from all the log files to assess system state and its behavior. The situation can be more complicated in case of log files allocation in different network nodes (multi-board systems). The developed software tool helps the raw data to be automatically collected, analyzed, reordered and filtered according to engineer's needs in each particular case.

Further along in this paper, we will consider the existing methods and tools and share a new approach that has been successfully used by the authors in their work.

Overview of the existing methods for log capturing and analysis

Nowadays there is a great amount of systems, which are focused on the log capturing and analysis. Generally these programs are meant for solving this problem in definite fields. For example there is a great deal of such systems oriented on Web Log File Analysis. One of them FlashStats 2006 [4] analyzes web site's log files and provides comprehensive information about the web traffic. Another one is widely known site optimizing system Semonitor, which includes a proxy server logs analysis application ProxyInspector [5].

We can name a lot of systems like these, but every system type has its own advantages and disadvantages, as well as limitations, such as ability to work with local database only, or not enough accuracy and work speed.

All in all there is no common approach to the problems of information capturing, filtering and analysis from log files. It can be partly explained by the fact that every field of science or production has its own system architecture. It is precisely this fact that explains the necessity of a new tool creation.

So the article deals with the tool, focused on gathering, filtering and useful presentation of information from log files, found in the complex telecommunication system.

Architectural Approach

1. Experimental Approach

It is true to say that there is no value in simply collecting raw log data, the value comes in how it is presented, structured and how clear the problem is from that data.

What is the approach?

The system consists of several boards of different types to provide functionality with the required level of reliability, which works under Linux system. Each board contains particular set of tasks depending on its type, which supports required functionality.

According to this information it was decided to take next fields as primary keys for log events:

- Board name (type + slot number)
- Task name
- Timestamp

Setting up the filter parameter board name and task name user is able to choose boards and tasks from which logs are likely to be gathered. It is extremely important option as there are a lot of boards and tasks in the system. Also the time interval can be set in order to collect only needed information as well as white and black lists are assigned for the same purpose.

White list can be used to gather only logs which contain particular word(s). For example, if you want to analyze logs where the word “error” appears you should write it down in the white list. On the other hand, if you do not want some words to be contained in logs you should write it down in the black list.

2. Development tools selection

From the very beginning it was decided that an execution should be supported not only on a target platform, but also on local machines with a provided log folder or an archival file (for more info see section 3). These platforms work under different Unix/Linux systems. So it was decided to take Perl which obvious advantage is that there is no need to compile different executables for various platforms. Perl script can be easily modified. Perl is good for fast development in this area and is obviously aimed at work with text structures. Also Perl Tk allows easily create required GUI.

3. Overview of Log Merger Architecture

The simplified system architecture is presented in Figure 1.

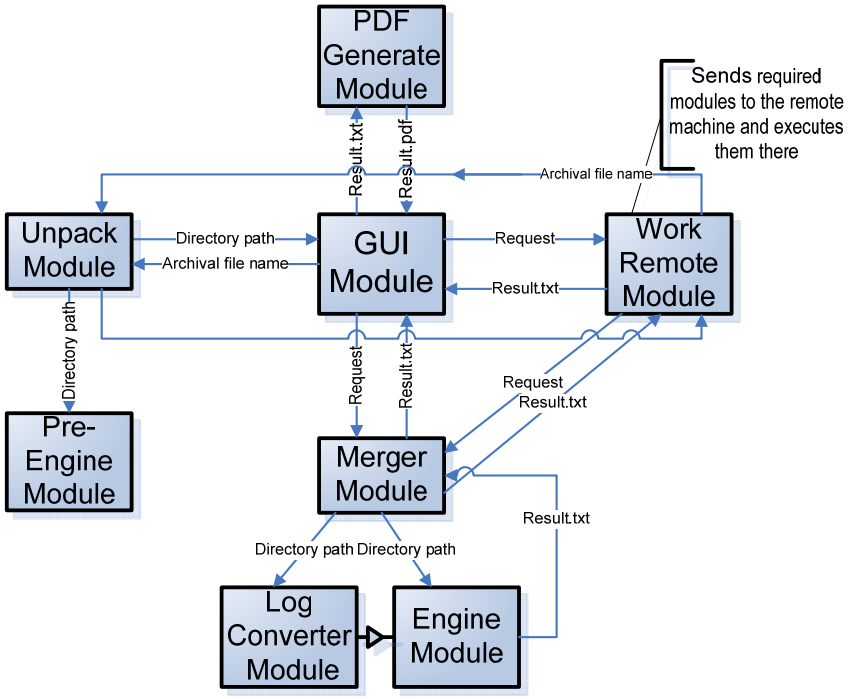

Figure 1. Log Merger architecture

1) GUI module

GUI collects the user information, creates filters and executes Pre-Engine and Merge Engine scripts to receive merge file. After this file is received it is represented in text window and user can browse and modify it. Also there is no need to run remerge process if user wants to reduce number of observed tasks and boards. GUI allows switching on/off events from tasks and boards of initial filter, which was used during merge procedure previously, without rerunning engine again. At the end user can save a final file in PDF format with different colors for various types of events (board + task) – this functionality is provided by the PDF Generate module.

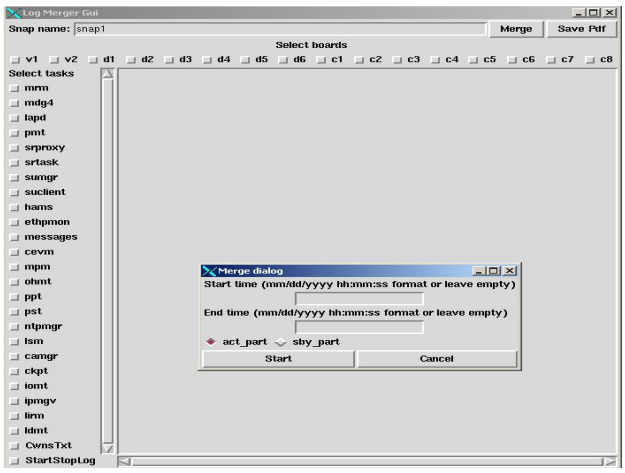

Figure 3: Graphic User Interface

2) Unpack, Pre-Engine and Log Converter modules

One of the most important problems for this type of tool is reusability. Different systems have different folders and logs structures. Moreover, in some systems log files are dumped into one archival file. So it was decided to separate out the following functionalities – Unpack module which unpacks archive file and returns the name of the directory with a specially organized file set and Pre-Engine and Log Converter modules which will convert different files to one input format for Engine module. It was also decided to implement a rule engine which will process files according to the rules. At the moment these rules are processed one after another, and the possibility to use Rete algorithm [1][2][3] is reasonable to be investigated in further studies.

3) Engine

There are two possible sorting modes:

- Ordering by timestamps
- Ordering by tasks, boards and then ordering by timestamps

For sorting by timestamps we used heuristic algorithm (See Figure 4) based on the statement that each log file is time ordered inside itself. So we receive the list of time ordered log files and should merge them according to the rules.

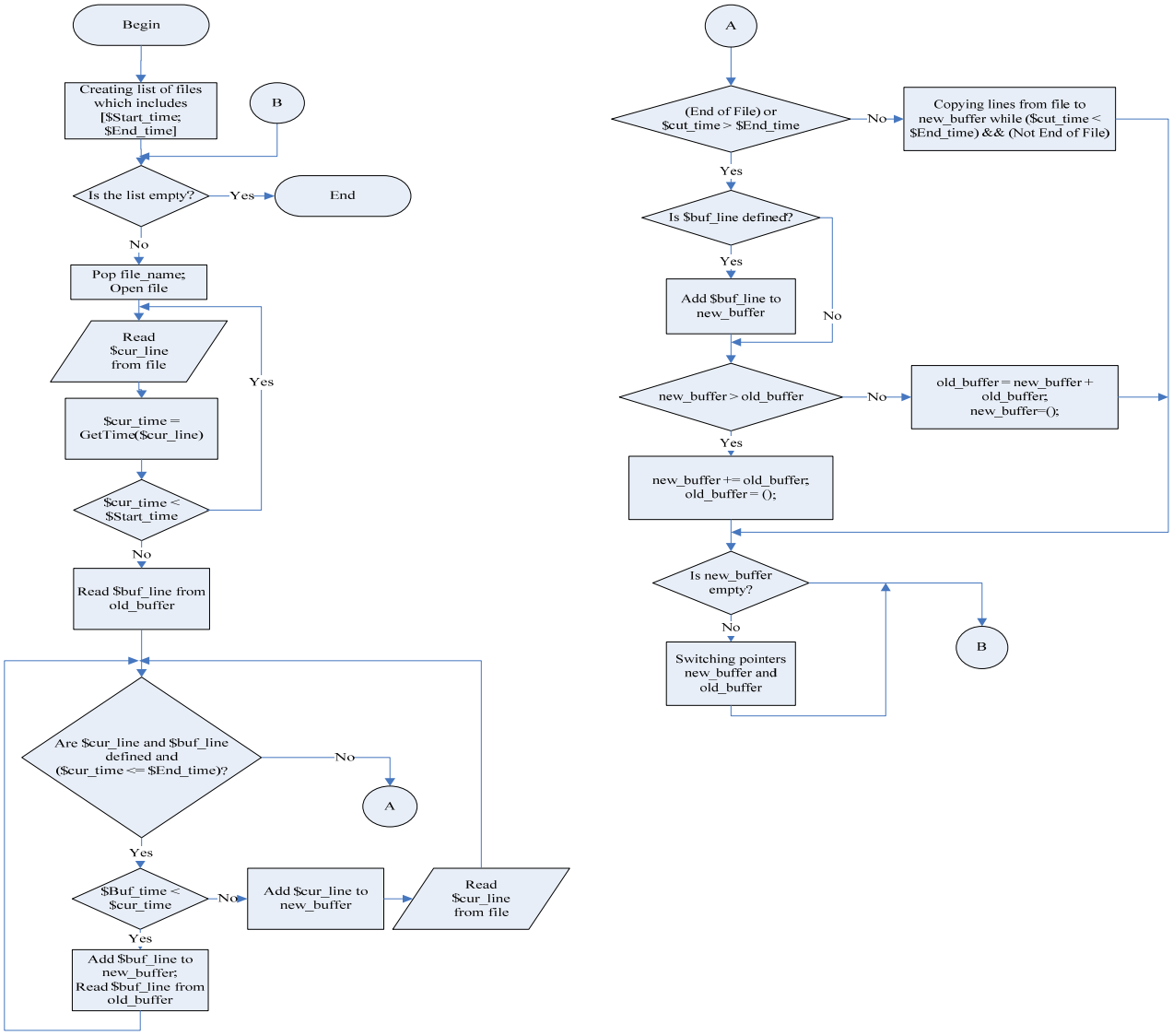

Figure 4: Timestamp sort

4) Work Remote module - Network Support

One of the most important features of the system is the network mode support which is provided by the Work Remote module.

Our application is working on the local machine connected to the target platform through several network gateways (Figure 5).

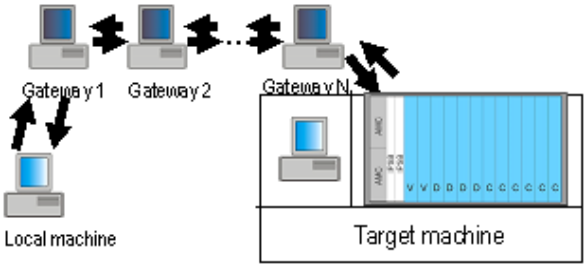

Figure 5: Work process in network mode

The module connects to the target machine and sends required scripts there. When it is necessary the application starts gathering of the required logs on the target machine and then starts merging process. After execution it removes script files and sends the result file to the local machine.

5) Introduction of analysis module as a further development

As a further development analysis module based on the knowledge and analytical skills of system experts should be added to the application to form an expert decision-making system together with the knowledge base generated by Log Merger. Rule engines (e.g. inference rule [6]) can be used for such purpose. Such system will automatically collect and investigate information from log files and then simply give the results freeing maintenance engineers of the necessity even to look through log files.

Conclusion

So we have developed a new type of system backed up by easy to perform common techniques which can be used on almost every platform both through the graphical user interface (GUI) and by the command line so that the user who does not possess libraries required for the GUI work can work with this system by calling scripts – the help file is given. Two run modes – local and network – enable user to gather required information from log files allocated either on the local machine or on remote machines without copying all log files to the local machine. As a result the structured information from different boards is collected in one file which could be defined as a knowledge base and can be used to develop an expert analyzing system.

Bibliography

[1] Charles Forgy, "A network match routine for production systems." Working Paper, 1974.

[2] Charles Forgy, "On the efficient implementation of production systems." Ph.D. Thesis, Carnegie-Mellon University, 1979.

[3] Charles Forgy, "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem", *Artificial Intelligence*, 19, pp 17-37, 1982

[4] FlashStats - http://www.maximized.com/products/flashstats2006/

[5] LogInspector - http://www.semonitor.ru/log_analyzer.html

[6] Proceedings By Luc De Raedt And Peter Flach (Paperback - Oct 2 2001) «Machine Learning»: ECML 2001: 12th European Conference On Machine Learning, Freiburg, Germany, September 5-7, 2001.

Authors' Information

Igor Karelin – Student of Saint Petersburg State Polytechnic University (fourth year), Software Engineer, Motorola Software Group – Russia, Saint-Petersburg Software center, T4 Business House, 12 Sedova str., 192019, St. Petersburg, Russia; e-mail: ngv743@motorola.com, ikarus47@mail.ru

Tatyana Gavrilova - Scientific adviser, Professor, Doctor of Science, Full professor in St. Petersburg State Technical University, department of Intelligent Computer Technologies, St. Petersburg, Russia; e-mail: tgavrilova@gmail.com

Boris Lyubimov – Graduate student of Saint Petersburg State Polytechnic University, Software Engineer, Motorola Software Group – Russia, Saint-Petersburg Software center, T4 Business House, 12 Sedova str., 192019, St. Petersburg, Russia; e-mail: abl100c@motorola.com