Threat Analysis and Identification Using Map Reduce Hadoop Platform

Sumeet S. Vernekar
PG Student
Department of Computer Engineering
Pune Institute of Computer Technology, Pune, India

Amar Buchade
Assistant Professor
Department of Computer Engineering
Pune Institute of Computer Technology, Pune, India

Abstract

The area of security forensic has become important. More advance security attacks are growing day by day and the complexity of analyzing or identifying those persistent malicious program has grown. These malicious programs reside in our system as an innocent program and behave like normal program and are sometimes untraceable by the advance threat protection software such as antiviruses, but in the background either they are stealing data or they are creating some destructive programs. These threats can only be found out by proper analysis of the system's activity. Most system programs that reside in our computer system log each and every activity in the log files. Analyzing those log file help us in identifying the possible suspicious activity. The system presented in this paper tries to solve this problem by analyzing those log file using the most powerful processing framework "Hadoop".

Keywords: Event Correlation, Hadoop, Log File Analysis, Mapreduce, Threat Detection.

I. INTRODUCTION

In the recent year, cases of most advanced cyber security attack have been found. In most of the cases the malicious programs that were involved in these attacks, were designed using complex algorithms, which bypassed the analysis of advance security programs like antiviruses, next generation firewall and resided in the hosts machine like a normal system program, but in background they were either stealing important user information or they were creating destructive programs intending to destruct the end users system creating a disaster.

Most of these attacks were on educational institutes, business critical organizations or on national defence organizations, trying to stealing data or destruct them causing threat to these organizations or the nation's security. These threats were identified by cyber security forensic organization by backtracking the logs generated by the security devices such as next generation firewalls, antiviruses etc. and identifying the possible pattern of security compromise or any sign of possible. The system presented in this paper tries to solve this problem using most powerful processing framework "Hadoop", which works on the MapReduce programming paradigm [6]. The proposed approach uses the logs generated by the security devices such as the next generation firewall, antiviruses, system generated (operating system) logs etc. for the analysis and extract the necessary information from those logs identifying the possible pattern of security compromise, which helps the security analyst for taking decision on possible security attack. This paper is an implementation of paper [1]. The implementation is currently built to work with the system generated logs (operation system logs).

MapReduce is a popular distributed system implementation, which is implemented using the Hadoop framework [1]. It is a framework for distributed processing of the large data across the clusters of computers for various jobs. It uses the power of computer cluster for parallel processing of jobs, so that the job is been completed with a fast rate than that of the conventional system. Here the job means the log file to be analyzed. MapReduce Algorithm consists of the Map Phase and the Reduce Phase [7]. The input data is divided into several splits. These splits are then processed by the map function first. The instance of map function called as the mapper will generate the intermediate result in the form of Key-Value pair. Several values are associated with each key. Here the input is the log file to be analyzed. There is a reduce function associated with each key. The instance of the reduce function called as the reducer will further process the intermediate result to generate the final strip down result. The MapReduce algorithm will generate a log report, which will contain the contexts of events. This log report is further provided as an input for the Event Correlation function to identify the patterns and detect the problem or threats, which are then included in the Final Report. The Final Report can then be used by the administrator for the purpose of problem or threat resolution.

The paper is arranged as follows. The section II presents related work, the section III presents the Implementation, the section IV presents the Related Mathematical and the section V presents the results, followed by conclusion and reference.

II. RELATED WORK

Log file are widely used for the purpose of problem and security threat identification. These problems and threats are identified by detecting the suspicious pattern of events in the log file. The log files generated by the servers are very large in size, probably

in some gigabytes, as it records each and every event into the log file. Processing such a large log file requires proper algorithm and resources, so that the log file will be analyzed as early as possible. This paper presents a MapReduce algorithm for the purpose of log file analysis. MapReduce [6] is a popular distributed system algorithm, which uses clusters of computer as a resource. The most popular implementation of the MapReduce algorithm is the Hadoop [1] framework.

The paper [2] presents a bioinformatics approach for the identification or detection of subtle anomalies using Teiresias algorithm. This algorithm automates the classification of syslog message stream, which thereby increase the availability of the overall system.

An overview of syslog file usage for the purpose of customer problem identification and threat detection is presented in [3]. The paper also discusses the challenges in the log file analysis and also provides the possible solution over it.

The Iterative Partitioning Log Mining (IPLoM) approach is discussed in [4]. This approach says that the log files are divided into clusters and these clusters are then considered for the purpose of mining the appropriate patterns, so that proper alerts are generated by these patterns. The approach uses three step hierarchical partitioning process for the purpose of cluster generation. The fourth step is the generation of cluster description or line format for each of the cluster produced. A significant average F-Measure performance of 78% is obtained when the other algorithm achieves an F-Measure performance of 10%.

The paper [5] discusses an approach which uses data mining and statistical learning method for automatic monitoring and detection of abnormal behavior in the console log. It uses a two stage detection system, where in the first stage frequent pattern mining and distributed estimation technique are used to capture the dominate patterns. In the second stage, principal component analysis based anomaly detection methods are used for actual problem identification. It uses a real system data from a 203-node Hadoop cluster, which provides highly accurate and fast problem detection with better understanding of execution patterns in their systems.

In the paper [8], an approach which uses incremental algorithm that automatically infers the format of the system log file. The resulting description can then be used to generate a suite of data processing tools automatically. It also allows the analysts to modify inferred description as desired and incorporate those changes in future revisions.

An approach called as the Cluebox is discussed in [9], which uses machine learning techniques on the available performance logs to characterize workloads, predict performance and discover anomalous behavior. Using machine learning technique with historical performance observations, Cluebox was able to filter 2000 performance counters to 68 counters, which describes the running workload. Further, two scenarios are presented which demonstrates the effective troubleshooting, that adversely impacts application response time. The first scenario is unknown competing workload and the second scenario is after system consisting checker.

The paper [10] presents a framework for defect detection, which uses patterns of significant events represented as expressions of a specialized monitoring language to specify a particular threat model. The Viterbi algorithm is used to identify whether the system generated events fits the given pattern. This technique is been applied considering the threat models and monitoring policies in logs for the multi user based MS-Window system.

A cluster based detection system is presented in [11]. It considers the notion of inherent variability in the each line of the log file, consisting a combination of static message type field and variable parameter field, which are then separated to find correlation in repeating log event types. Each log line is abstracted to a unique ID or event type and a dynamic parameter value is extracted to give an insight of the current state of the system. This technique was implemented on the log file of the Virtual Computing Lab, which abstracted 727 unique event types.

Logsurfer log file analysis in presented in [14]. The main feature of Logsurfer is simple cluster maintenance, which helps in the identification and resolution of problems. It examines the messages in the log file and relates those messages with other messages in the log file for problem identification. It has a capability of modifying the results at run time, which allows us to detect complex patterns in log files intern helping in taking proper actions over the problems.

The paper [15] provides a methodology to mine rich source of information from console logs to automatically detect system runtime problems. It transforms the free text console logs into numerical features. These features are then analyzed using machine learning to detect the operation problems. It then shows the results in an operation friendly one-page decision tree showing the critical message associated with the problem detected.

In the paper [16] an approach called as the Logview is presented, which helps in visualizing the clusters generated using SCLT (Simple Log File Clustering Tool) in a treemap, showing the hierarchical structure of the clusters produced by SCLT. It speeds up the analysis of event data in order to detect the security issue on a given application.

III. IMPLEMENTATION

The system implements the MapReduce algorithm using the Hadoop technology. Hadoop uses the master slave architecture. For the implementation a Hadoop cluster of two slaves and one master is considered.

The system presented in this paper has four main components:

(1) Log Collection
(2) Context Generation
(3) Event Correlation
(4) Alert generation

For the implementation purpose only the logs from the Linux systems are considered. The typical log format is as given below,

A. DATETIME LOGGING_HOST PROCESS MESSAGE

Whenever an activity is done the log message for the same is written in the corresponding log file. Initially the DATETIME at which the activity as taken place is written, then the host which is logging its IP Address or Host name is written (LOGGING_HOST), followed by the PROCESS which is responsible for that activity and the MESSAGE indicating what activity is performed by that process. Below is a sample log line.

B. Nov 15 19:49:33 test sshd[9554]: error: PAM: Authentication failure for rootsuser from 192.168.10.1

Form the above log line it can be seen that the activity tool place on "**Nov 15 19:49:33**". This activity was performed by the host "test" (hostname) and the process responsible is "sshd" and then the message, which indicate that the host "192.168.10.1" was trying of gain "rootsuser" access, but could not succeed.

Following is the description of each of the component:

1) *Log Collection*

This component is responsible for the collection of logs. Each system generate two types of logs

(1) Message
(2) Secure

The location of those log files are /var/logs/message and /var/logs/secure. But for some Linux system there are other files where the logging takes place.

This component is basically responsible for collecting logs from these log locations and places them in the master system. For the log collection user have to provide the details to the system form the logs have to be collected. The details include hostname or IP address, root password and the log file location. The prerequisite for the logs collection is that the ssh service should be enable of the system from where the logs have to be read. Once the logs are read then the next component context generation comes into picture.

2) *Context Generation:*

This component is responsible for generating context based on the hostname or the IP address. This helps in identifying the logs from different systems. This is basically a MapReduce program, which generates the context. The inputs to this component are the logs that are collected in the log collection phase. The output will be the logs with context (hostname or IP address) attached.

3) *Event Correlation:*

This component is responsible for categorizing the logs based upon the categories. The logs are categorized in three main categories :

(1) Host Intrusion
(2) Firewall
(3) System

Based on the message in the logs, they are categorized in the respected category. The categorization is done based upon the keywords. This is also a MapReduce program.

Each process in the Linux system has some significance. So based on activities performed by the processes they are categorized into above three categories. Basically the message for each process is considered for the categorization.
Following description provides the categorization of log line:

1) *Host Intrusion Categorization:*

For the categorization of the logs in host intrusion category the following Linux processes are considered.

(1) sshd - secure shell daemon, responsible for remote connection.
(2) su, sudo - for super user access.
(3) gpasswd, passwd - for configuring the password.
(4) groupadd, groupdel, useradd, userdel,
(5) usermod - user and group management activity.
(6) kernel - process that monitors the kernel activity.
(7) ftpd - process monitoring the ftp access.
(8) cron - process that maintains the Linux processes.

Each of the above process log messages of the activities they perform. Based upon the message that these process log they are categorized into host intrusion category.

Following is an example that shows the categorization of a log in host intrusion category.

C. Nov 15 19:49:33 test sshd[9554]: error: PAM: Authentication failure for rootsuser from 192.168.10.1

The above log line is categorized into host intrusion category. It can be observed that the user from the host "192.168.10.1" is trying to get access of the user "rootsuser" and fails. This is an indication of security attack, but may not be attack as the user might be a ligimate user and have forgot the password. But is the same activity is done numerous numbers of times, and then it can be a brute force attack. So further, if we can monitor the count of the same activity, which will help the security analyst to identify the security breach.

Similarly for each such above processes, based on the message they are categorized into host intrusion category.

2) *Firewall Categorization:*

The logs with below process are categorized into firewall category based on the message.

(1) Firewall – process that monitors the firewall activity.

Consider the below example for the firewall log categorization,

D. Nov 8 20:43:01 test Firewall[61]: Stealth Mode connection attempt to TCP 192.168.1.120:139 from 192.168.1.117:13005

The above log line determines the firewall activity. The above log line indicates that there is a connection attempt in TCP mode from the host "192.168.1.120:139" port "139" to "192.168.1.117" port "13005".

This helps the security analyst of identify the security traffic flowing in and out of the system.

3) *System Categorization:*

The logs with below process are categorized into system category based on the message.

(1) shutdown, reboot - monitors the shutdown and reboot activity
(2) auditd - monitor the audit activity
(3) inetd, xinetd - monitors the initialization process
(4) syslogd, rsyslogd - monitors the syslog logging service.

Consider the below example for the system log categorization,

E. Aug 12 11:38:38 corsair reboot: [ID 662345 auth.crit] rebooted by root

The above log line determines the system activity. The above indicate that the system is rebooted by the root user. This helps the analyst of monitor the system activities.

4) *Alert Generation:*

This component is responsible for the generation of alerts from the categories formed in the event correlation phase. Here the important fields mentioned below are populated from the logs, which make it easy to the analyst to get more information about the particular activity. Following is the list of fields which are considered for the alert generation.

(1) Logging date – Date when the event was logged.
(2) Event date – Date when the event generated.
(3) Logging Device IP/Hostname – IP/Hostname of the system which logged the event.
(4) Service Name – Service Name for which the event was logged.
(5) Process ID – Process ID of the service.
(6) Process Name – Process Name of the service.
(7) Network Protocol – Network Protocol used.
(8) Source IP/Hostname – Source IP/Hostname present in the log line.
(9) Source Port – Source port present in the log line.
(10) Destination IP/Hostname – Destination IP/Hostname present in the log line.
(11) Destination Port – Destination port present in the log line.

The above fields give more information about the activities performed in the system by the services to the security analyst.

IV. RELATED MATHEMATICS

- Input: Log File to be analyzed.
- Output: Final report for alert generation.
- System:

$S = \{ I, O, LR, E, M, R, EC, A \}$

where,

I = Input = SL = Syslog file.

where,

$SL = \{ e_1, e_2, e_3, \ldots, e_n \}$

where, e_i is an event occurred.

O = Output = FR = Final report for alert generation.

where,
FR = { C_ie_j } i, j = 1 to n
where,
C_ie_j is an event from the context C_i
E = Event Repository = { W, P, ρ }
where,
W = Keyword list
P = Policies to be applied
ρ = Severity
M = Map function.
R = Reduce function.
EC = Event Correlation function.
A = Alert generation function.

- Functions :
  - Map function :
    M (DN, D) ∈ K → V
    where,
    DN = Document name
    D = Document content
    K = Key = machine name
    V = value = (date, time, message) triplet.
    It specifics a map function M, which maps a key K to value V
  - Reduce function :
    R (K, V) ∈ C → (D, T, EM)
    where,
    K = Key
    V = Value
    C = Context
    D = Date
    T = Time
    EM = Event message
    It specifics a reduce function R, which reduces and produces a context C containing date D, time T and event message EM of each event belonging to a particular context.
  - Event Correlation function :
    (1) For keywords identification :
    if
    EC (C^i_{ej}) ∈ EM → W
    Then enter C_ie_j event with severity to the final report FR.
    For Policy application :
    R → M
    Each rule R is associated with a message M to be displayed.
    if
    P (R) ∈ R → C
    Then enter a message belonging to rule R with severity into the final report FR.
    (2) Display function :
    D (FR) ∈ FR → I
    D is a display function which displays the final report FR on the interface I.

V. RESULTS

This system was tested with the input of varying log file and its behavior was noted. Below are the results for the same.
The logs files with following sizes were provided as input.

(1) First Log File: 9.58 MB
(2) Second Log File: 153 MB
(3) Third Log File: 321.6MB
(4) Fourth Log File: 643.1MB
(5) Fifth Log File: 1.3 GB

Table - 1
Results With Varying Log File Size.

Factors	*Components.*						
	Context	*Host Intrusion Category*	*Host Intrusion Alerts*	*Firewall Category*	*Firewall Alerts*	*System Category*	*System Alerts*
First File (Size: 9.58 MB)							
Input File	*9.58MB (75328 events)*	*8.4MB (54390 events)*	*8.6MB (53512 events)*	*8.4MB (54390 events)*	*6.4MB (39932 events)*	*8.4MB (54390 events)*	*4.0MB (24507 events)*
Output File	*8.4MB (54390 events)*	*8.6MB (53512 events)*	*20.1MB (33430 events)*	*6.4MB (39932 events)*	*94.2KB (134 events)*	*4.0MB (24507 events)*	*92.5KB (208 events)*
Time	*22 sec*	*29 sec*	*26 sec*	*49 sec*	*24 sec*	*113 sec*	*24 sec*
Second File (Size: 153 MB)							
Input File	*153MB (1205248 events)*	*134.8MB (870240 events)*	*137.4MB (856288 events)*	*134.8MB (870240 events)*	*102.7MB (638912 events)*	*134.8MB (870240 events)*	*63.3MB (392112 events)*
Output File	*134.8MB (870240 events)*	*137.4MB (856288 events)*	*322.2MB (534880 events)*	*102.7MB (638912 events)*	*1.5MB (2144 events)*	*63.3MB (392112 events)*	*1.4MB (3328 events)*
Time	*50 sec*	*86 sec*	*100 sec*	*265 sec*	*25 sec*	*898 sec*	*28 sec*
Third File (Size: 321.6 MB)							
Input File	*321.6MB (2410496 events)*	*269.6MB (1740480 events)*	*274.8MB (1712576 events)*	*269.6MB (1740480 events)*	*205.4MB (1277824 events)*	*269.6MB (1740480 events)*	*126.6MB (784224 events)*
Output File	*269.6MB (1740480 events)*	*274.8MB (1712576 events)*	*644.5MB (1069760 events)*	*205.4MB (1277824 events)*	*2.9MB (4288 events)*	*126.6MB (784224 events)*	*2.6MB (6656 events)*
Time	*72 sec*	*119 sec*	*197 sec*	*491 sec*	*28 sec*	*1099 sec*	*28 sec*
Fourth File (Size: 643.1 MB)							
Input File	*643.1MB (4820993 events)*	*539.2MB (3480960 events)*	*549.5MB (3425152 events)*	*539.2MB (3480960 events)*	*410.7MB (2555648 events)*	*539.2MB (3480960 events)*	*253.3MB (1568448 events)*
Output File	*539.2MB (3480960 events)*	*549.5MB (3425152 events)*	*1.3GB (2139520 events)*	*410.7MB (2555648 events)*	*5.9MB (8576 events)*	*253.3MB (1568448 events)*	*5.8MB (13312 events)*
Time	*120 sec*	*211 sec*	*279 sec*	*855 sec*	*24 sec*	*1563 sec*	*28 sec*
Fifth File (Size: 1.3 GB)							

Input File	*1.3GB (9641984 events)*	*1.1GB (6961920 events)*	*1.1GB (6850304 events)*	*1.1GB (6961920 events)*	*821.5MB (5111296 events)*	*1.1GB (6961920 events)*	*481.1MB (3136481 events)*
Output File	*1.1GB (6961920 events)*	*1.1GB (6850304 events)*	*2.5GB (4279040 events)*	*821.5MB (5111296 events)*	*11.8MB (17152 events)*	*481.1MB (3136481 events)*	*11.01MB (26620 events)*
Time	*216 sec*	*326 sec*	*594 sec*	*1224 sec*	*35 sec*	*1920 sec*	*28 sec*

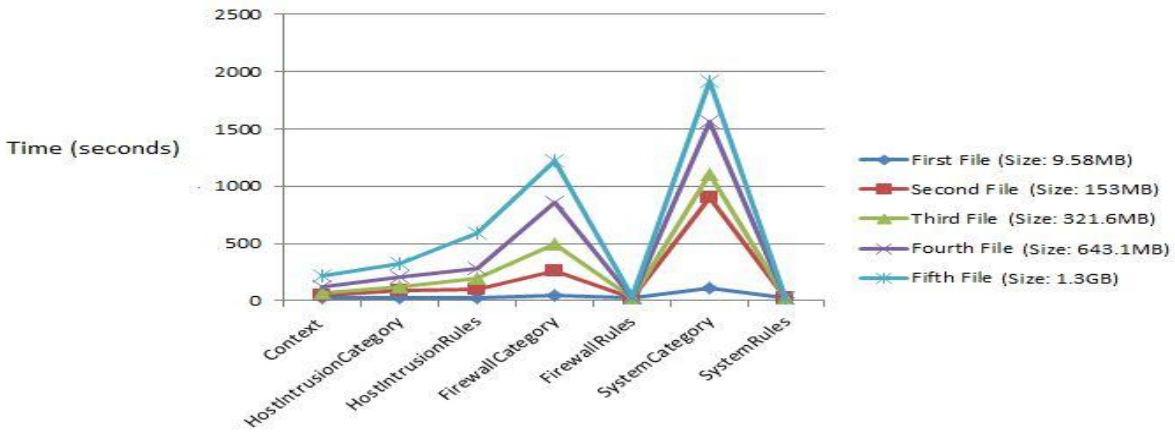

Fig. 1: Time Vs File Size Graph

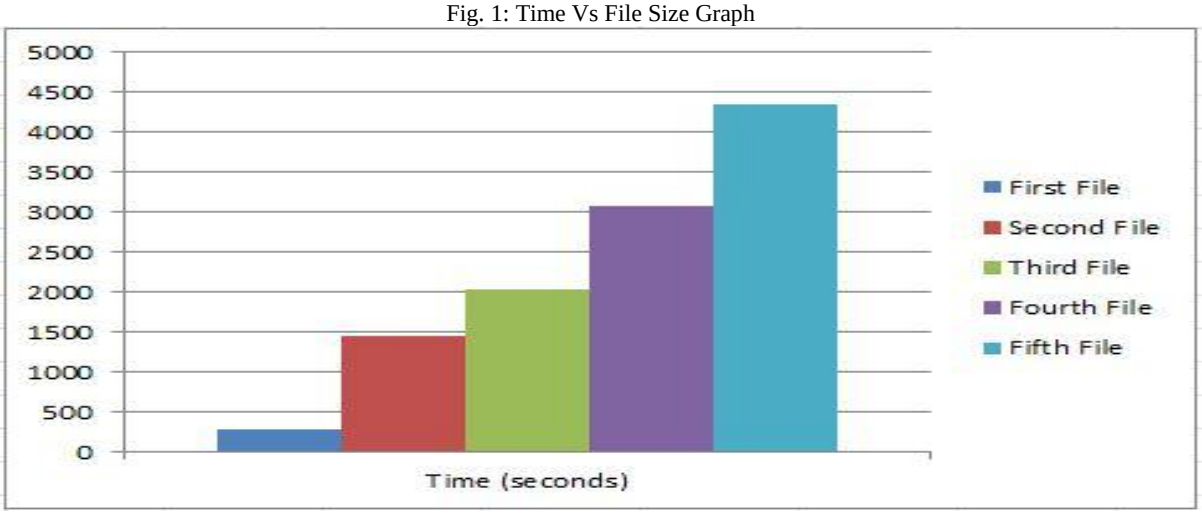

Fig. 2: Total Time For Each File

First File Size: 9.58 MB, Second File Size: 153 MB, Third File Size: 321.6 MB, Fourth File Size: 643.1 MB and Fifth File Size: 1.3 GB (Fig. 2.)

It can be derived from the graph (Fig. 1. and Fig. 2.), that as the logs size increases, the time required in processing the logs increases linearly.

VI. CONCLUSION

A final conclusion can be drawn from the result that, the system performs linear with respect to the varying log file input. The system is able to process logs efficiently and categorize them in appropriate categories. Thus proposed system provides an efficient way of log collection and correlation to identify the system threats and problems and helps the security analyst to identify the threats and problem in the system more efficiently, helping them to take appropriate action on them.

ACKNOWLEDGMENT

I would like to acknowledge Prof. Amar Buchade Department of Computer Engineering PICT Pune for providing his valuable time and guidance.

REFERENCES

[1] S. S. Vernekar, A.R. Buchade, "MapReduce based Log File Analysis for System Threats and Problem Identification". In the Proceeding of 3rd IEEE International Advance Computing Conference (IACC), Feb 2013, Ghaziabad, India. 2013 978-1-4763-4523-3/12.
[2] Hadoop Document : http://hadoop.apache.org.
[3] J. Stearley. "Towards Informatic Analysis of Syslogs". In the Proceeding of CLUSTER '04 Proceedings of the 2004 IEEE International Conference on Cluster Computing.
[4] W. Jiang, C. Hu, S. Pasupathy, A. Kanevsky, Z. Li, Y. Zhou. "Understanding Customer Problem Troubleshooting from Storage System Logs". In the Proceeding 7th USENIX Conference on File and Storage Technologies 2009.
[5] A. Makanju, A. N. Zincir-Heywood, E. E. Milios. "Clustering Event Logs Using Iterative Partitioning". In the Proceeding of KDD'09, June 28–July 1, 2009, Paris, France. 2009 ACM 978-1-60558-495-9/09/06.
[6] W. Xu, L. Huang, A. Fox, D. Patterson, M. Jordan. "Online System Problem Detection by Mining Patterns of Console Logs". In the Proceeding of ICDM '09 Proceedings of the 2009 Ninth IEEE International Conference on Data Mining.
[7] J. Dean and S. Ghemawat. "MapReduce: Simplified Data processing on Large Clusters". In the Proceeding of 6th Conference on Symposium on Operating Systems Design and Implementation, 2004.
[8] J. Dean and S. Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters". In the Proceeding of Communications of the ACM, 51(1), 2008.
[9] K. Fisher, D. Walker, K. Q. Zhu. "Incremental Learning of System Log Formats". In the Proceeding of ACM SIGOPS Operating Systems Review Volume 44 Issue 1, January 2010.
[10] S. R. Sandeep, M. Swapna, T. Niranjan, S. Susarla, S. Nandi. "CLUEBOX: A Performance Log Analyzer for Automated Troubleshooting". In the Proceeding of WASL'08 Proceedings of the First USENIX conference on Analysis of system logs.
[11] A. Razavi, K. Kontogiannis. "Pattern and Policy Driven Log Analysis for Software Monitoring". In the Proceeding of Annual IEEE International Computer Software and Applications Conference 2008.
[12] M. Nagappan, M. A. Vouk. "Abstracting Log Lines to Log Event Types for Mining Software System Logs". In the Proceeding of MSR, 2010, pp.114-117.
[13] S. E. Hansen, E. T.Atkins. "Automated System Monitoring and Notification With Swatch". In the Proceeding of LISA – November 1-5, 1993 – Monterey, CA.
[14] R. Vaarandi. "SEC – a Lightweight Event Correlation Tool". In the Proceeding of 2002 IEEE Workshop on IP Operations and Management.
[15] J. E. Prewett. "Analyzing cluster log files using Logsurfer". In the Proceeding of Annual Conf. on Linux Clusters 2003.
[16] W. Xu, L. Huang, A. Fox, D. Patterson, M. I. Jorda. "Detecting Large-Scale System Problems by Mining Console Logs". In the Proceeding of 26th International Confer- ence on Machine Learning, Haifa, Israel, 2010.
[17] A. Makanju, S. Brooks, A. N. Zincir-Heywood, E. E. Milios. "LogView: Visualizing Event Log Clusters". In the Proceeding of PST '08 Proceedings of the 2008 Sixth Annual Conference on Privacy, Security and Trust.