Research Article

International Journal of Advanced Computer Research, Vol 6(23)
ISSN (Print): 2249-7277 ISSN (Online): 2277-7970
http://dx.doi.org/10.19101/IJACR.2016.623015

Transformation of LOG file using LIPT technique

Amit Kumar Sinha* **and Vinay Singh**
Usha Martin Academy, Ranchi, India

Received: 15-January-2016; Revised: 23-March-2016; Accepted: 28-March-2016

Abstract
Log files in complex systems quickly grow into huge sizes. Often, they must be kept for a long period of time. For the convenience and storage economy, log files should be compressed. However, most of the available log file compression tools use a general-purpose algorithm, which do not take advantage of redundancy specific log files. The objective of this paper is to use Length Index Preserving Transformation (LIPT) technique to transform the log files and reduces the significant amount of redundant characters. The applied transformed file in data compression tool will effectively reduce storage space. It has been observed that LIPT is an effective way to transform log files for size reduction and the file size get reduced to the average of 44% before the compression. During the process of transformation the redundant character gets reduced.

Keywords
Compression, LIPT, Reduction, Transformation.

1.Introduction
Now we are living in a data world. Out of hundred percent data, ninety percent of the data is generated in the last two years. Millions of terabyte of log files are generated every day. As far as data organization is concerned the data are structured, unstructured and semi structured. The log file is under semi structured data. A log file is a copy of all the things or action which occurs in a particular server. Log files are maintained on the server to keep the information about the visitor from where they are coming, how they return, how they navigate through a site. Through these log files, a system administrator can determine what Web sites you've accessed, whom you are sending e-mails to and receiving e-mails from and what applications are being used. Log files can be found in the OS, in the Web browser in the form of a cache, in applications in the form of backups, E-mails etc. A log file also contains a list of events, which have been "logged" by a computer. Log files are often generated during software installation and are created by Web servers, but they can be used for many other purposes as well. The log files stores and keep track of all information about the visitor in each visit. In plain text format log files are saved.

*Author for correspondence

The website visitor data are recorded in web-server in the form of log files, where each visitor IP address, visit time and the page visited information are included. The log file also stores images, JavaScript or CSS files. Website statistics software is used for processing data, which can display the information in a user-friendly format, a user can view more detail information on a graph representation of each day on click and also it includes the visitor graph of the last month. By operating System, FTP programs and Software Utilities log files may also be generated. Mac-OS X stores a log of all program and system crashes; we can view it by using the built-in Console application. The event Viewer program uses to show Windows records applications, security, and system logs.

Most log files have a ".log" file extension, although numerous make use of the standard ".txt" extension or a different proprietary extension in its place [1]. Log file information is recorded following the order in which they occurred, and is located in the root directory, or rarely in a secondary folder, depending on how it is set up by the server [2].

Now a days size of computer networks increases, so it is quite difficult to monitor, control and secure them. These networks consist of several devices, sensors and a gateway which are spread over large geographical areas. Each of these devices produces

log file, which need to be analyzed to monitor and provide network security. Current information systems are replete with log file which are created in multiple places for multiple purpose. Log files may quickly grow to huge size and they must keep for a long period of time. Log files are an excellent source for determining the health status of a System and are used to capture the events happened within an organization's system and the network. Within an association, many logs enclose records associated with computer security. Common examples of these computer security, logs are audit logs that track user authentication attempts and security device logs that record potential attacks [3].

The log file grows quickly and store for a long period of time, so it needs a large amount of space also. The main aim of this paper is to transform a log file in a format to produce better compressible output than the original data. The transformation can be used in any kind of textual log file, despite of their source and size. The present paper uses a specialized log file transformation scheme Length index preserving transformation (LIPT) technique to transformed logs file. The rest of the paper is organized as follows: section 2 is the survey of related works, section 3 discusses the LIPT method, results and discussion is presented in section 4 while, the conclusion is given in section 5.

2.Related work

Rácz and Lukács [4] presented DSLC, a generalized scheme for web log compression. The DSLC is claimed to improve general purpose compression algorithm efficiency on the web log up to a factor of ten, it works well only on huge log file (over 1 GB) and it requires human assistance before the compression, it takes an average of two weeks for a specific log file.

Balakrishnan and Sahoo [5] proposed a scheme for compression of system logs of IBM Blue Gene/L supercomputer. The scheme consists of preprocessing stage and general purpose compression. The measure improvement of compression ratio was 28.3 % .

Skibinski and Swacha [6] proposed a couple of simple preprocessing variants intended to facilitate further compression of log files from various applications. They proposed five variants, where the simplest one merely encodes each line with reference to the previous line, storing the length of the longest match on a single byte (with the aid of symbols over 127 in ASCII), followed by the mismatching subsequence copied verbatim, until the nearest field end, where again the longest match in the previous line for the corresponding field is sought for. The next two variables are supplementary flexible in selecting the reference line which helps particularly for log types were not every line have matching structure.

Fourth, variant adds a dictionary substitution of words found in a pure pass (an idea used earlier, e.g. in [7], for plain text compression), and the fifth variant extends the previous one with compact encoding of the following patterns: numbers, dates, times and IP addresses. The transformed log files compressed with the default zip algorithm, in their experiments, i.e. deflate, where on average shorter by 37% than the non-preprocessed files submitted to zip. Major enhancement has also been pointed when stronger back-end compression algorithms (LZMA, PPMVC) were used [7]. They had proposed a specialized lossless Apache web log preprocessor and test it with the combination of several popular general purpose compressors. The result shows the proposed transform improves the compression efficiency of general purpose compressor on average by 65% in case of gzip and 52% in case of bzip2[7].

Feng et al. [8] suggested that the results of standard statistical tests performed on log-transformed data are often not relevant. Fegreei and Ahmad [9] experiment reveal a significant improvement in terms of the execution time of the log file's frequency mining calculation.Novel kinds of data, like XML, spatial, biomedical, or multimedia data should efficiency handle by Extraction, Transformation and Loading (ETL) applications [10].

3.Length index preserving transformation

Length Index Preserving Transform (LIPT) has been published by Awan and Mukherjee [11]. LIPT is a reversible lossless data transform which is an extension over LPT, RLPT and SCLPT, group of transforms which are in turn extensions of the Star-Encoding. It uses certain approaches proposed in LPT, and still yields better compression and time performance than any of that. It introduces frequent occurrence of common characters as well as it reduces the original text. The Reverse Length-Preserving Transform shortening the length of encoded words required to perform the one to one mapping with the original words. With LIPT, concept of sub-dictionaries and building dictionaries implement, here also, the dictionary should be pre-decided and it must be available to the compressor

and de-compressor. In LIPT, the codeword is made up of three components <*, LengthChar, OffSet> [12].

Example

The dictionary consisting of various sub-dictionaries considering the only words given in the following text data will be as shown in *Table 1*.

Table 1 Mapping of words

INDEX	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}
1				meta			measure	learning	education		Information
2				four			average	training	abilities		Improvement
3								Decoding			
4											
5											

Improvement, Education, learning, information, meta, abilities, visible, four, training, improvement, Decoding char length of meta, seed is 4 so assign in D4, character length of measure is 7 so assign in D7, likewise char length of average is 7 so assign 2nd column of D7 all other words are assigned. Char length of information and improvement is 11 so assign to D11 -1st column (information occurs first) and 2nd column (improvement occurs second). In the above *Table 1* number of rows can be increased so that word dictionary can be increased.

3.1Map the words in dictionary

Let Di[j] denote the word in dictionary i at index j. Let (*w) denote the codeword of word w. So, *(Di[j]) denotes the codeword for word at index j in dictionary i. The length of each codeword used for words in dictionary Di is i.

The 52 words are assigned codeword represented by a sequence of characters followed by a single alphabet letter from {a, b, …, z, A, B, ..., Z} as follows: at index j ($1 \leq i \leq 26$): 1st letter is i^{th} a lower case letter of the English alphabet, remaining next 26 codeword for words at index j ($27 \leq i \leq 52$): 1st letter is i^{th} upper case letter of the English alphabet. Next 52 words are assigned codeword as above; with a difference that 2nd letter is used from English alphabet for words at index 53 to 104. The concept is similar, except that the single alphabet letter (a, b,…, z, A, B,..., Z) is placed in the sequence in the second position. The process is repeated till codeword assigns to all words in all sub-dictionaries.

3.2Transformation process

Consider the notation Di[j] to denote the jth word in partition Di and EN(Di[j]) to denote the LIPT encoded word. The codeword EN(Di[j]) is made up of three components <*, Lengthchar, offset>.

At the start of a word denotes that it is an encoded word. If a word is not found in the dictionary it is left unaltered, and thus does not have a prefixed *.

- LengthChar denotes the length of the actual word Di[j]. Note that the length is represented using characters <a-z> corresponding to length <1-26>.
- Offset represents the index of the actual word in partial dictionary Di. The index is represented using letters of the alphabet. Offset of the first word is "a", 26th word is "z", 27th word is "A", 52nd word "Z", the 53rd word "aa" and so on.
- Thus, the word information in the given example, would be encoded as **ka*, where ***** implies that the word is encoded; ***k*** means its length as 11; and **a** represents the index 1 in the dictionary D11.

3.3Reverse transformation process

The reverse transformation of the word EN (DI [j]) is fairly simple. If the word read from the encoded input file begins with *, it is to be decoded; otherwise it is sent to output as it is. Decoding is done as follows:

- Determine length i of word from 2nd character of codeword (encoded word). This determines sub-dictionary number.
- Determine index of word in sub-dictionary Di from offset.
- The resulting word is (Di[j]) for output.
- If there is a capitalization mask at the end of the encoded word, then it is applied to the decoded word [12].

For example: -***hc*** denotes that the encoded code word (denoted by first character *) and original word is of length 8 (corresponding to 'h') located at index 1 (offset 'c'). Hence the actual word is "***Decoding"*** located at index 3 in dictionary D8. *Figure 1* shows the pictorial view of transforming log file.

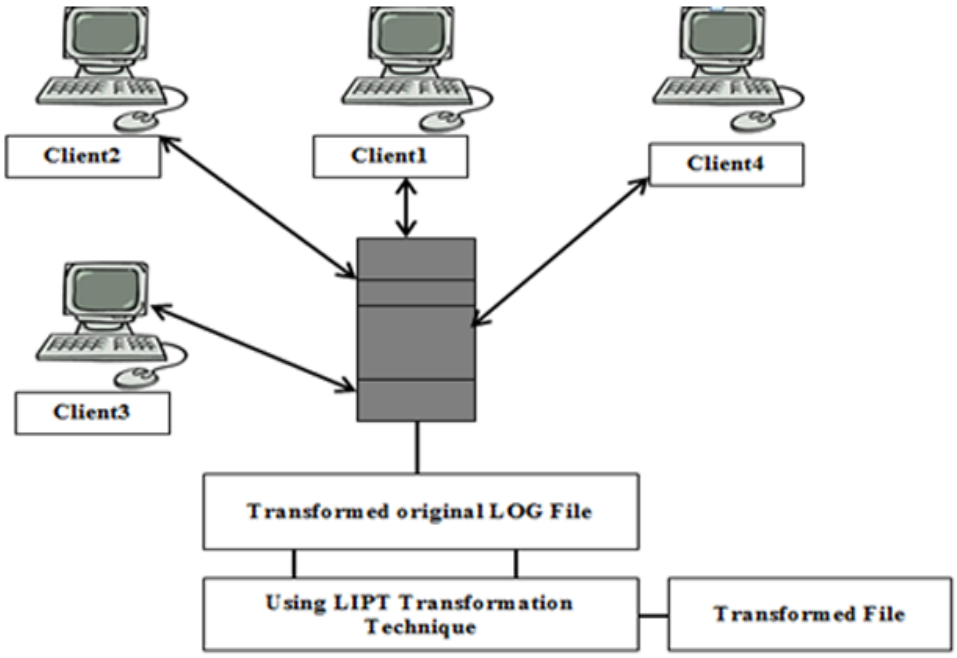

Figure 1 Pictorial view of transforming log files

3.4Log file transformation

The log files are plain text files in which every line corresponds to single logged event report. Lines are separated by end-of-file marks and each event description consists of at least tokens, separated by spaces. In many log file neighboring lines are very similar in both structure and content. Our proposed transformation technique transforms the tokens very short (in 3 characters) using LIPT, an algorithm is also proposed to do the work.

This transformation is an extension of core transformation (Variant 1) of Prezemyslaw Skibnski and Jakub Swacha [6].

A1-192.168.0.2/TCP_Miss/connect.client-channel.google.com

A2-192.168.0.2/TCP_Miss/Connect.client-loreld.fdfid.com

A1 and A2 are two log files A1 come first, then A2 both files are very similar in content and structure so we transform it using LIPT transformation.

Table 2 Mapping of A1 and A2

	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	W …	W. .	W_{26}
1			Com		Fdfid	client	connect	TCP_Miss						
2						google	channel							
3						loreld								
4														

From *Table 2* column W_1-W_{26} denotes the words of length, for small letter <1-26> and W_{27}-W_{52} for capital letter. Then a mapping is used to encode all token of a log file in each dictionary to transform the original log file. Char length of com is 3 so it is assigned to W_3, char length of connect is 7 so it is assigned to W_7. Char length of Client is 6 so it is assigned to W_6. The char length of google is 6 and char length loreld is also 6 but the google encounter 2^{nd} and loreld encounter 3^{rd} so they assigned in 2^{nd} and 3^{rd} row of W_6 of *Table2*, likewise char length of channel is 7 and encounter 2^{nd} so it is assigned to 2^{nd} row of W_7.

IP address placed as it is TCP_Miss assigned at first row of W_8 so it is encoded as *ha, where '*' denote the word is encoded 'h' denote 8 columns and 'a' denote first row. Connect is encoded as *ga, client is encoded as *fa, google *fb.

Algorithm
The algorithm of the transformation process is as follows:

Table 3 Transformation algorithm

Step1: Read original log file from left to right and break it into sub word in a sequence.
Step 2: Mapped the word in Table [**according to Map the word in Dictionary [section 3.1]**].
Conditions:
a) If the word is a string and it is appended with a numeric value, string with special symbol, then mapped the string into LIPT table, **according to LIPT Rule 3.1.**
b) If the sub-word is an IP address, date, any separate numeric value, and a symbol, then skip mapping, and continue reading.
Step 3: After mapping, transform the word using **Transformation process [Section 3.2]** and place it as according to key value stored in an array, where each transformed string is separated by delimiter.

After transforming A1 and A2 using an algorithm given in *Table 3* the result would
A1- 192.168.0.2;*ha;*ga;*fa;*-ga.;*fb.;*ca
A2- 192.168.0.2;*ha;*fa-;*fc;*ea.;*ca

Illustration of algorithm
The algorithm in *Table 3* is illustrated by taking an Apache Log Samples
L1:-lj1036.inktomisearch.com- - [07/Mar/2004:17:18:36] "GET /robots.txt HTTP/1.0"
L2:-64.242.88.10 - - [07/Mar/2004:17:21:44 -0800] "getwiki/bin/attach/tWiki/table*Plugin* http/1.1"
L3:-64.242.88.10 - - [07/Mar/2004:17:22:49 -0800] "GET /twiki/bin/view/TWiki/ManagingWebs?rev=1.22 HTTP/1.1" 200 9310
The mapping of L1, L2 and L3 is given in *Table 3*.

Table 4 Mapping of L1, L2 and L3

	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	$W_{…}$	$W_{..}$	W_{26}
1			Com	http	twiki	Ij1036	getwiki				tableplugin			
2			Mar			robots								
3			Get			attach								
4			Txt											
5			Bin											
6														

From *Table 4*, column W_1-W_{26} denotes the words of length, for small letter <1-26> and W_{27}-W_{52} for capital letter. Then a mapping is used to encode all token of a log file in each dictionary to transform the original log file. Char length of Mar is 3 so it is assigned to W_3, char length of getwiki is 7 so it is assigned to W_7. Char length of Ij103n6 is 6 so it is assigned to W_6. The char length of robots is 6 and char length, attach is also 6 but robots encounter 2nd and attach encounter 3rd so they assigned to 2nd and 3rd row of W_6 of *Table 4*, likewise other char of L1 and L2 are also assigned.

Transformation process for L1:
L1:-lj1036.inktomisearch.com - - [07/Mar/2004:17:18:36] "GET /robots.txt HTTP/1.0"
L1:-*fa;*ma;*ca--[07/mar/2004:17:18:36] *cc;*fa.;*cd

Here lj1036 is mapped at number six column, which is denoted by ‘f’ and first row of W_6 which is denoted by ‘a’. So lj1036 is encoded as *fa. Char length of Inktomisearch 13 so it will assign to W_{13} at the first row and encoded as *ma, likewise other word is mapped and encode. The log File L2 has transformed is given below.
L2:-64.242.88.10—
[FD:17:21:44];*ga;*ce;*fc;*ea;*ka

4. Result & discussion

We have taken three different log files 2005 Access, Apache log and NetAccess as given in *Table 5*. The result shows in *Table 5* is the original log file in KB. Second is the transformed Log File by using LIPT. The third one is the percentage reduces after transformed. The fourth one is differences of the original and transformed in KB.
For the experiment access log file, apache server, net access log file is taken in Kilobytes.

Table 5 Transforming results of three log files

Log Files	2005access	Apache Log	Netaccess	Average
original log file in KB	14.899	17.575	31.051	21.175
Size After Using LIPT transformation	8.19445	10.1935	17.07805	11.822
% Reduce between raw & transform	45	42	45	44
Difference	6.70455	7.3815	13.97295	9.353

Here the *Table 5* shows the original log file size in KB, which reduces by 42% in size after transforming.

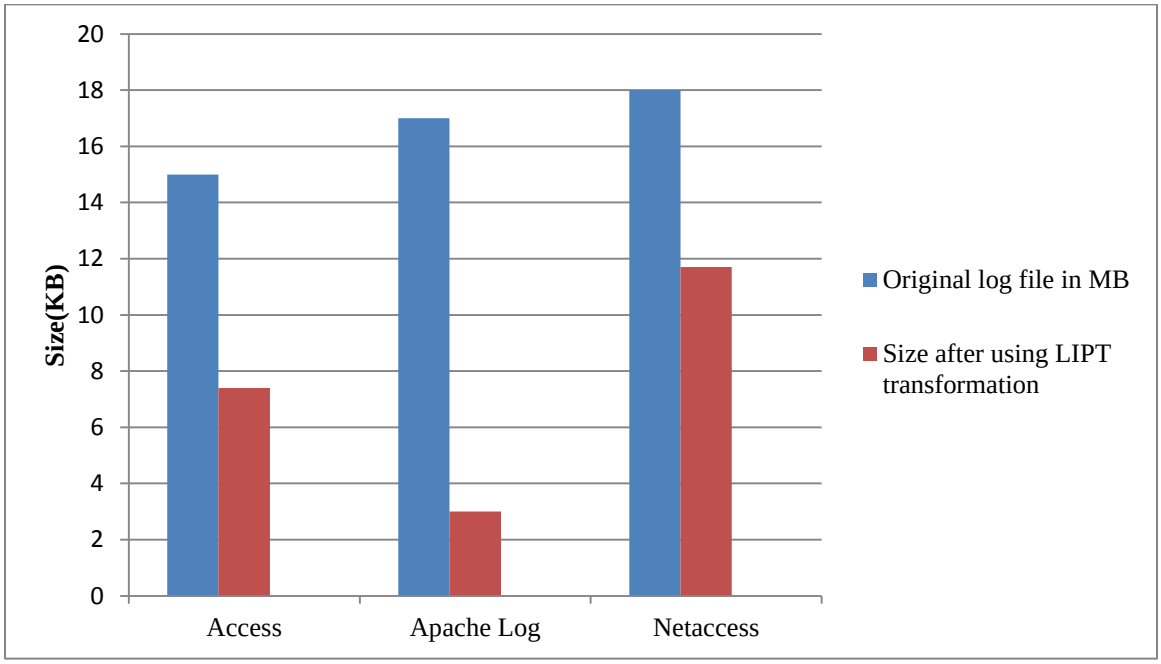

Figure 2 Reduction of log files

It has been observed from *Table 4* and *Figure 2* that the LIPT method is better to reduce log file size, before compression. The *Figure 2* show three log files in MB in three different server Access 2005, Apache and Netaccess, the size of Access 2005 is 15 MB, size of Apache is 17 MB and the size of Netaccess is 18 MB. After using transformation method the size of Access2005 is reduced from 15 MB to 7.4 MB, Apache logs reduces from 17MB to 10 MB and Netaccess reduces 18Mb to 11MB.

5. Conclusion and future scope

In this paper an attempt is to transform the log files Using LIPT method. It has been observed that LIPT is an effective way to transform log files for size reduction. From *Table 3* it has been observed that the file size gets reduced to the average of 44% before the compression. During the process of transformation the redundant character gets reduced. It is suggested that before compression of the log file if the file is transformed using LIPT technique significant amount of storage space get saved. We are currently in the process of transforming the log file using LIPT technique by our own software tool and analyze the impact of storage using HDFS architecture in Hadoop framework. In future we analyze the impact of storage of log file using HDFS architecture in Hadoop framework which will more reduce the size and provide optimal storage.

Acknowledgment

None.

Conflicts of interest

The authors have no conflicts of interest to declare.

References

[1] http://techterms.com/definition/logfile. Accessed 01 October 2015.

[2] Sree PK, Babu IR. FELFCNCA: fast & efficient log file compression using nonlinear cellular automata classifier. International Journal on Communications. 2012; 1(1): 7-11.

[3] Singh NK, Tomar DS, Roy BN. An approach to understand the end user behavior through log analysis. International Journal of Computer Applications. 2010; 5(11):9-13.

[4] Rácz B, Lukács A. High density compression of log files. In proceedings of data compression conference 2004 (p. 557). IEEE.

[5] Balakrishnan R, Sahoo RK. Lossless compression for large scale cluster logs. In parallel and distributed processing symposium (IPDPS) 2006 (pp. 1-7). IEEE.

[6] Skibiński P, Swacha J. Fast and efficient log file compression. In proceedings of 11th east-European conference on advances in databases and information systems (ADBIS) 2007 (pp. 56-69).

[7] Grabowski S, Deorowicz S. Web log compression. Automatyka/Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie. 2007; 11(3):417-24.

[8] Feng C et al. Log-transformation and its implications for data analysis. Shanghai archives of psychiatry. 2014; 26(2):105-9.

[9] Fageeri SO, Ahmad R. An efficient log file analysis algorithm using binary-based data structure. Procedia-Social and Behavioral Sciences. 2014; 129:518-26.

[10] Anitha J, Babu M. ETL work flow for extract transform loading. International Journal of Computer Science and Mobile Computing. 2014;3(6):610-7.

[11] Awan FS, Mukherjee A. LIPT: a lossless text transform to improve compression. In international conference on information technology: coding and computing 2001 (pp. 452-60). IEEE.

[12] http://shodhganga.inflibnet.ac.in/bitstream/10603/28063/7/07_chapter1.pdf. Accessed 30 October 2015.

Amit Kumar Sinha has received his Master of Computer Application degree from KIIT, University, Bhubaneswar in the year 2008, Presently he is working as an Assistant Professor in UMESL, Kolkata, India Since 2009.

Email: amitkr2k3@gmail.com

Vinay Singh has received his Master of Computer Application degree from IGNOU, New Delhi, India in the year 2003 and Master of Technology in Computer science and Engineering from BITs Mesra, Ranchi, India in 2009. He has submitted Ph.D from BIT's Mesra, India. Presently he is working as an Associate Dean of Information Technology in UMESL, Kolkata, India since 2008. He is also empanelled with Wipro Technologies as a corporate trainer. He has published twelve papers in the International Journal and Conference. His Research area is Software Metrics and Quality.