IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-2, Feb 2019 E-ISSN: 2347-2693

# Text Mining Using Frequent Pattern Analysis and Message Passing

**M. Deeba[1*], Mary Immaculate Sheela[2]**

[1]Manonmaniam Sundaranar University, Trirunelveli, India
[2] Department of Information Technologyy,Pentecost University College, Ghana

DOI: https://doi.org/10.26438/ijcse/v7i2.658667 | Available online at: www.ijcseonline.org

Accepted: 22/Feb/2019, Published: 28/Feb/2019

***Abstract*** Text mining is a Computer Science technique to analyze text data. Text mining is text analysis, is the process of deriving high quality information from text. Text mining is to convert text into data for suitable analysis. It allows us to investigate relationship among patterns which would otherwise be extremely difficult. Various techniques are used to mining the frequent patterns in the given text which are applicable to analyze the information in huge documents. The parallel construction of FP-Trees and parallel mining on multi cores is a popular tree projection based mining algorithm. Once each processor counts the frequency of each item using its local data partition, all worker processors send the local count to the master processor which combines them and generate global count. The parallel implementation of FP-tree may show good speedups but sending the local results to master on distributed environment and merging the patterns count on master core are overhead which consumes a considerable time. This study aims at to analyze various frequent pattern mining techniques used to extract information from texts especially on multi cores and going to adopt a new technique for finding frequent patterns, which used the Dictionary based compression algorithm(LZW). The new technique is implemented with single processor as so as with multi processor using message passing technique. The main objective of this research is enhancing the speed and reduce the memory consumption required to extract the frequent patterns form the given textual data. The parallel implementation of our proposed LZW based algorithm with three datasets Webdoc, Kosarak and Trump is compared with parallel implementation of FP-Growth on single and multi core. The results shows good performance in speedup, Latency and Efficiency in proposed LZW based algorithm.

***Keywords*** Parallel FP-Growth, Frequent Keywords Mining, Multi core Systems

## I. INTRODUCTION

Recent technological improvements have led to the availability of new type of information that was previously not available. The modern database includes both standard structure of data and unstructured data comprising words, images, and videos. New sources of text data such as text messages, social media activity, web searches and blogs. The tremendous availability of published texts, sophisticated technologies and interest in extracting information from text has led to replacing the human effort with automatic systems. Now a day's greater understanding of the text mining has led government authorities and private sectors to use this developing technology. The National Center for Text mining (NaCTem) is the first public funded text mining center in the world established by United Kingdom, operated by university of Manchester. Text analyzing text data is an one of the main element of Big data trends.

Text mining is to convert text into data for suitable analysis. There is a need for computational Artificial intelligence algorithms and statistical techniques to text documents. Various text mining techniques like Information Extraction, Information Retrieval, categorization and Clustering are used to extract useful information from Text data. The result of text mining could make text data as informative one. It allows us to investigate relationship among patterns which would otherwise be extremely difficult. With text mining, the text can be categorized and clustered which producing results such as word frequency count and predictions analysis.

An important subfield of data mining that is called pattern mining**.** Pattern mining consists of using data mining algorithms to discover interesting, unexpected and useful patterns in databases. Pattern mining algorithms can be applied on various types of data such as transaction databases, sequence databases, streams, strings, spatial data, and graphs. A wide variety of algorithms will be covered starting from Apriori. Many algorithms such as Eclat, TreeProjection, and FP-growth.

What is an interesting pattern**?** There are several definitions. For example, some researchers define an interesting pattern as a pattern that appears *frequently* in a database. Other researchers want to discover *rare patterns*, patterns with a high *confidence*, the top patterns, etc. The problem of frequent pattern mining has been widely studied in the literature because of its numerous applications to a variety of data mining problems such as clustering and classification. In addition, frequent pattern mining also has numerous applications in diverse domains such as spatiotemporal data, software bug detection, and biological data. The algorithmic aspects of frequent pattern mining have been explored widely.

Applying above pattern mining algorithms on multi cores using shared memory are now a day's ubiquitous. The parallel approaches to the frequent item sets using FP-Tree algorithm is a fast and popular tree projection based mining algorithm. Building several local FP- trees on multi cores parallel until all the frequent patterns are generated. The parallel algorithms with good workload balance provide higher speedups when are compared to sequential algorithm. The parallel implementation of FP-Trees and parallel mining on multi cores is a popular tree projection based mining algorithm. Once each processor counts the frequency of each item using its local data partition, all worker processors send the local count to the master processor which combines them and generate global count after the master processor removes the item with support count less than minimum support threshold. Once a complete frequent item set is constructed, it will broadcast to all the processors to the group. The parallel implementation of FP-tree may show good speedups but sending the local results to master on distributed environment and merging the patterns count on master core are overhead which consumes a considerable time.

One of the basic methods of parallel computing is the use of message passing. It is a transfer of data between instances of parallel program running on multiple processors in a parallel computing architecture. Parallel computers have two basic architectures Distributed memory and Shared memory.

We need algorithms that do not require multiple scans of the data base and leave small foot print in main memory at a given time. There are three factors initiate this research undertaken.

First one is the construction of FP-Tree on FP-Growth algorithm may not fit in memory also expensive to build. So, there is a need to minimize the consumption of memory. In FP-Growth algorithm, the construction of FP tree for each transaction will be in need of adding new nodes with existing ones of the tree. It is somewhat difficult to understand by the beginners. If transaction table has too many transaction ids then the tree will be large. So the second aim is to give an easy algorithm which is easily understood by the users. While we compare the speed of the algorithms Apriori, Eclat and FP-Growth, the FP-Growth gives high speed ups than the other two. But when Fp-Growth is applied with huge database speed is slow. So, the third aim is to improve the speed.

The main objective of this paper is to improve the speed of mining work to extract frequent patterns from a given text mining by the parallel mode of work allotment on multi processors which are coordinating with each other by passing the messages among them to update the frequent pattern tables within them.

The paper is organized as follows: section1 is Introduction; section 2 is the literature survey where various related works of the authors are discussed in detail. section 3 describes about the Methodologies used. section4 is discussed about Results Analysis and section5 is the Conclusion of the discussions.

## II. LITERATURE SURVEY

A manual literature search was conducted to identify the literature available on various facets of the topic and to select relevant resources for the review. The search was performed using various Google Scholar search engine and various electronic databases also utilized. Keywords used included frequent pattern mining (FPM), Parallel pattern mining techniques, message passing multiprocessor systems etc. The search was limited to resource in the English language and was conducted over the period 1994 to2018. The researcher also referred list of relevant articles and particular resources were focused through on line.

This literature review begins with an overview of frequent pattern mining techniques which are already existed to assist the text mining process. The overview is followed by a discussion of the parallel FPM conceptual framework of this study. The factors influencing on parallel FPM are discussed within the theoretical framework and the various algorithms used for parallel FPM are also presented.

Frequent pattern mining on Text mining is an important tool to establish the incidence of patterns, which are used to extract frequent information from the text. Pattern mining is an important subfield of data mining. Various pattern mining algorithms can be applied on various types of data such as transaction databases, sequence databases, strings, spatial data, streams and graphs (CC Aggarwal, 2007). A range of widely used algorithms for finding frequent pattern on large transactional database (Sharmila Nasreen et.al. 2014). Apriori algorithm, Frequent Pattern Growth algorithm, Rapid Association Rule Mining (RARM), ECLAT algorithm and Associated Sensor Pattern Mining of Data Streams (ASPMS) Frequent pattern algorithms. The study focused on strength and weakness on each algorithm. The techniques used by Apriori are Breadth first search, RARM is Depth first search, FP-Growth used Divide and conquers. Regarding the data base scan Apriori Scans the database each time when the candidate key is generated. RARM scans few times to construct SOTriel tree. Apriori takes considerable execution time. The other algorithms execution time is less than Apriori. The major drawback of Apriori algorithm has to generate too many candidate itemset. ECLAT requires virtual memory to perform transformation. The construction of FP Tree in FP growth algorithm is expensive to build and it consumes more memory. The entire above frequent pattern algorithms can be applied for text mining.

Frequent Keyword Mining (FKM) is a useful tool for discovering frequently occurring keywords in data [1]. Many algorithms like Apriori, RARM and FP Growth have been developed to speed up mining performance on single core systems. When the data set size is huge it is good to parallelize FP-Growth algorithm on multi core machines. Partitioning the large database into number of cores and utilize the combined strength of all the cores to achieve maximum performance.

One of the major problems in frequent itemset mining is the explosion of the number of results which is directly effecting on the execution time of itemset mining algorithms[2]. To address this problem, closed itemsets have been proposed, which provides concise lossless representations of the original collection of frequent itemsets.

Associated Rule mining (ARM) is one of the fundamental tasks in data mining. It’s first application for the analysis of sales basket data which was introduced by Agarwal et al[3]. The important problems in data mining are discovering association rules from databases of transactions where each transaction consists of a set of items[4]. The most time consuming operation in this discovery process is the computation of the frequency of the occurrences of interesting subset of items (called candidates) in the database of transactions. Mining frequent patterns in transaction databases, time series databases, and many other kinds of databases has been studied popularly in data mining research [5]. Most of the previous studies adopt an Apriori-like candidate set generation-and-test approach. However, candidate set generation is still costly, especially when there exist long pattern.

Efficient utilization of shared memory MIMD parallelism is essential to improve the overall performance(L. Liu et al., 2007) as the large data movement and communication requirements of parallel association rule mining can be performed seamlessly exploiting the underlying shared memory [6].

FP-growth is usually a selection for large-scale mining applications due to its performance merits. In addition, the divide-and-conquer approach of FP-growth naturally lends itself to parallelism. Several parallel methods inspired by FP-growth have been proposed for shared memory multi-core systems. In the traditional FP-growth-based parallel approach, parallel processes cooperatively build a shared global FP-tree resulting in extensive use of costly synchronization locks to access each node of the tree[7]. A different approach called Tree Projection partitions the FP-tree into subsections with small portions shared among processes. Only access to the small shared sections would require locks for synchronization [8]. Although this approach reduces the synchronization cost considerably, it adds the overhead of extra partitioning of the workload and is harder to load balance.

Many of the methods for ARM have shown unstable performance for different database types and under utilize the benefits of multicore shared memory machines[9]. The proposed method, named ShaFEM, combines two mining strategies and applies the most appropriate one to each data subset of the database to efficiently adapt to the data characteristics and run fast on both sparse and dense databases. The ShaFEM[10] uses a new data structure named XFP-tree that is shared among processes to compact data in memory.

The characteristics of these data structures and the behaviors of their mining methods are quite different and will result D is scanned to specify all frequent items (or 1-itemsets) in D based on the minsup value. After this step, only data of frequent items are used to determine the frequent itemsets as well as to generate the association rules. This considerably reduces the memory usage and computation by avoiding a large amount of infrequent data from in different performance for a given database[11]. For example, algorithms like Apriori [12], FP-growth[13] and those making use of FP-array data structure[14] exploit horizontal format of data and perform efficiently on sparse databases (e.g. web document data or retail data) while Eclat [15] present data in a vertical format and run faster on the dense ones (e.g. biological sequence data). These mining methods perform unstably on differentdata types. Furthermore, the characteristics of data subsets D used to mine (k +1)-itemsets can change from very sparse to very dense as mining proceeds. Hence, applying a suitable mining strategy for each D is essential to improve the performance of ARM. It leads to the introduction of our parallel mining approach employing two mining strategies based on the characteristics of D. For large-scale transactional databases, apply parallel computing to speed up the mining.

## III. METHODOLOGY

### FREQUENT ITEMSET GENERATION

Frequent items play an essential role in many Data Mining tasks that try to extract interesting patterns from databases. The association rule mining is one of the most popular problems of all these. The most basic tasks in Data Mining is identification of sets of items, products, symptoms and characteristics, which often occur together in the given database. The motivation for searching frequent sets came from the need to analyze the supermarket transaction data, that is, to analyze the customer behavior in terms of the purchased products. Frequent sets of items describe how often items are purchased together. Text mining is a technique that is used to extract useful information from large amount of data sets. Data mining rules like frequent pattern and association rule that is important for finding frequent patterns. The apriori based and tree structure-based algorithms are used in frequent pattern mining.. The tree structure-based algorithm is FP-Growth.

**Apriori algorithms Frequent itemset generation:**
Apriori is the first algorithm for finding frequent items. Apriori algorithm was invented by Rakesh Agarwal and Ramakrishnan Srikant[16]. It is a well known algorithm in data mining. It was originally applied to market basket transactions. Frequent itemsets generation of the Apriori algorithm is using a general transaction database. Each row in the table represents a transaction, which contains unique transaction identification number (TID) along with items. But it takes more time for finding the frequent item sets. It has to scan the database again and again which is consumes lot of time. Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time

**Frequent itemset generation with Eclat algorithm:**
Eclat (Equivalent Class Clustering and bottom up Lattice Traversal) algorithm is a data mining algorithm used to find frequent items from large amount of database. The limitations of Apriori algorithm is reduced by using vertical dataset which reduces the access time. Eclat algorithm finds the elements from bottom like depth first search. It is a simple algorithm not using horizontal database. It counts the support but not calculate the confidence. Both Apriori and FP-Growth use horizontal data format.

**Frequent itemset mining with FP-Growth algorithm:**
FP-Growth algorithm has an improvement over above two algorithms for finding frequency itemset in a database without candidate key generation. It was proposed by Han. It is fast and consumes less memory than Apriori. It constructs a special internal structure called FP-tree. It is a two step approach

Step1. Build a data structure called FP-Tree.

Step2: Extract the frequent items from FP-Tree.

FP-Tree construction has two passes. On pass1 It scan and find the support of each item then discard the infrequent items after that it arranged the frequent items in descending order based on the support.

On pass2 FP-Growth reads one transaction at a time maps it with path which can overlap when transactions share items. Pointers are used between nodes.

**Experimental datasets**
The performance in terms of execution time of three states of art methods such as Apriori , Eclat and FP-growth on experimental dataset is shown in the table3.10. From the results, it is found that while FP-growth runs fastest on the sparse and dense dataset compared to other two methods. Hence for multi core process, the FP-Growth method is adapted.

**Webdoc**
Webdoc dataset was donated by Claudio Lucchese, Salvatore Orlando, Raffaele Perego, and Fabrizio Silvestri [6] and was built from a spidered collection of web html documents. A huge real-life transactional dataset was made publicly available to the Data Mining community through the FIMI repository. The whole collection contains about 1.7 millions documents, mainly written in English, and its size is about 5GB. It has 1,523,346 number of transactions and 52, 676,657.

**Kosarak dataset**
The Kosarak dataset was provided to us by Ferenc Bodon[7] and contains click-stream data of a hungarian on-line news portal. Kosarak has 990,002 transactions and 41,271 numbers of items.

**Trump dataset**
Trump dataset [8] contains 56 major speeches by Donald Trump by June 2015 – November 2016.

**Performance analysis in terms of execution time of Apriori, Eclat and Fp-Growth.**
The performance in terms of execution time of three states of art methods such as Apriori , Eclat and FP-growth on experimental dataset is shown in the table3. From the results, it is found that while FP-growth runs fastest on the sparse and dense dataset compared to other two methods. Hence for multi core process, the FP-Growth method is adapted.

Table1 Performance analysis in terms of execution time of Apriori, Eclat and Fp-Growth

| Dataset | Apriori | Eclat | FPGrowth |
|---|---|---|---|
| WebDocs | 120056 | 88467 | 63294 |
| Kosarak | 8944 | 6467 | 4324 |

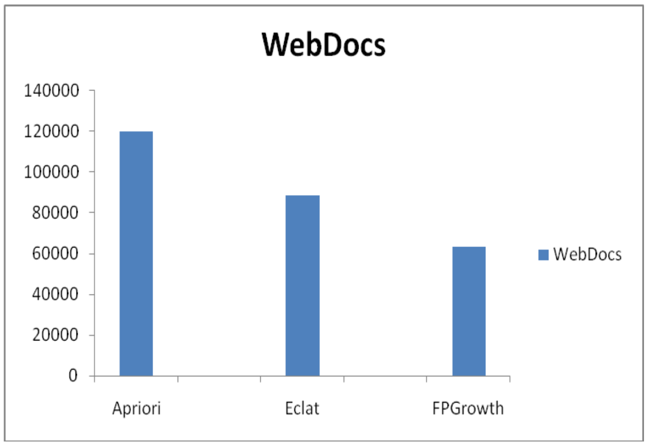


Figure1 Execution time for Webdoc with three algorithms

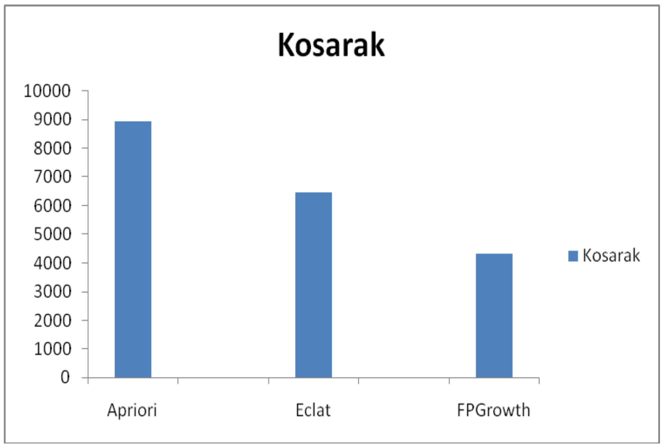


Figure2 Execution time for Kosarak with three algorithms

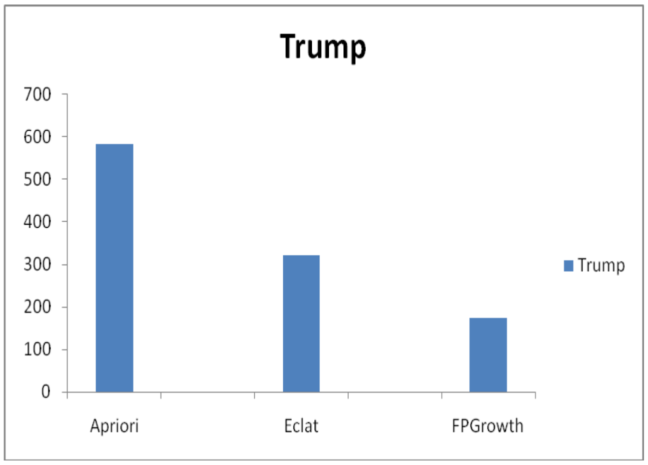


Figure3 Execution time for Kosarak with three algorithms

When we apply the three algorithms Apriori, Eclat and FP Growth on the datasets Webdoc, Kosarak and Trump, From the above three figures we come to know that FP Growth has high speed than other two.

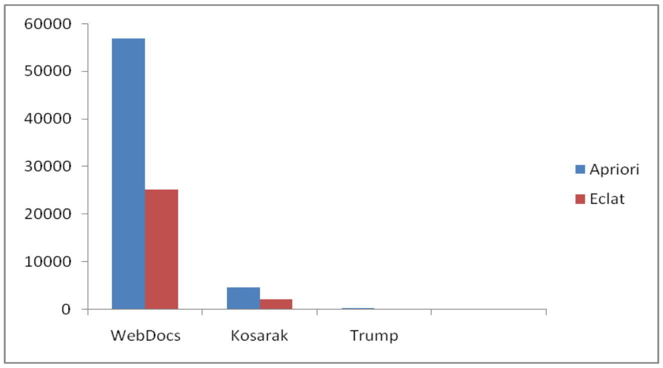


Figure4 Speedup of FP growth algorithm compared with Apriori and Eclat

From the above figure we come to know that, with WebDocs the execution speed of FP Growth algorithm is 56762 unit times faster than Apriori and 25173 unit times faster than Eclat algorithms. In case of Kosarak datasets, FP growth isssss 4620 unit times faster than Apriori and 2143 unit times faster than Eclat. In the third case with Trump dataset, 409 unit times faster than Apriori and 148 unit times faster than Eclat.

## PARALLEL TEXT MINING ON MULTICORES

The parallel data mining algorithms to be executed on multi core processors are used various architecture. The parallel method needs unified interfaces among the processes. For parallel execution additional functions are added to share data and models between the parallel threads. Besides such features have to obtain various parallel algorithm structures and implement various strategies of execution for different environment conditions. The described parallel method is illustrated through various algorithms Apriori, Eclat, ARM algorithms and FP-Growth algorithms.

The parallel implementation of FP-tree may show good speedups but sending the local results to master on distributed environment and merging the patterns count on master core are overhead which consumes a considerable time.A parallel formulation of the FP-Tree algorithm (E. H. Han et al., 2000)on a distributed memory environment consists of two main stages:

1. Parallel construction of FP-trees for each available processor, and 2.parallel formulation of the FP-Growth sequential mining method to mine each FP-tree.

2. Conditional FP-Trees (CFPT) are building recursively in parallel until all the frequent itemsets are generated. by proposing a master-worker based dynamic task scheduling technique to balance the workload at run time.

The number of trees to be built depends on several parameters of the database, such as average length of the transaction, number of transactions in the database, number of frequent items, and support threshold. We intend to improve the serial algorithm based on our findings and run our parallel algorithm for even higher number of processors with larger data sets, based on the resource availability on the machine environment.

**PFPMLZW(a parallel frequency pattern mining using LZW based algorithm)**

## THE OVERVIEW OF PFPMLZW

Various data mining algorithms like Apriori, ECLAT and FP-Growth are used to find interesting patterns from the data bases. Those algorithms can be applied on Transaction databases, streams, strings, spatial data and graphs. PFPMLZW implements a unified model for is proposed to discover frequent patterns using LZW based algorithm with message passing interface among n cores in parallel manner. Synchronous communication has been occurring among the processors for updating the table entries. PFPMLZW perform its mining task in the following major stages that are presented in detail in the following sections.

**Preprocessing work on given text:** To compact all data in memory the preprocessing work is done by stop words and stemming algorithms.

**Extract keywords and construct the initial keyword table:** Using RAKE ( Rapid Automatic Keyword Extraction ) extract the keywords and construct the keyword table by assigning code for each keyword.

**Partition the given text and distribute it to n cores:** By keeping load balance( is an even division of processing work between two or more computers and/or CPUs or other devices.

**Do the local computations:** For finding new patterns and updating the table entries using LZW algorithm with message passing mechanism.

**Extract the frequent patterns from the tables:** By setting minimum count value/ threshold value.

## THE DICTIONARY BASED /LZW ALGORITHM.

LZW is invented by Abraham Lempel, Jacob Ziv & Terry Welch. It is also called as Dictionary based Coding. It is a lossless compression algorithm used to compress the repeated patterns in the digital images. LZW compression start with simple dictionary called string table by assigning code to each string. The LZW compression algorithm takes each input sequence of bits of a given length (for example, 12 bits) and creates an entry in a table (sometimes called a "dictionary" or "codebook") for that particular bit pattern, consisting of the pattern itself and a shorter code. As input is read, any pattern that has been read before results in the substitution of the shorter code, effectively compressing the total amount of input to something smaller. The decoding program that uncompressed the file is able to build the table itself by using the algorithm as it processes the encoded input.

## FREQUENT PATTERN GENERATION ON SINGLE CORE USING LZW BASED ALGORITHM

On single core system, new pattern will be formed by combining a string with neighboring string. If a new pattern has been found and if the same pattern is already presented in the PC-table then increase the relevant frequent count by one otherwise put the new entry in the PC-table and assign a new code for it. The new code will be given by automatic increment of the event. The relevant sequential mining algorithm FPMLZWSC is given as follows.

**FPMLZWSC**(input-text,minimum-count-value)
// Input-text is the given text
Step1:Get the input text and do the preprocessing work.
Step2:Create the sting table and initial PC-Table in shared memory.
Step3: Find the new pattern from the given text.
    3.1:Check the pattern with table entries weather the pattern is already presented in the PC-Table or not. If not put the pattern in table entry and assign a code for that. If presented just increment its frequent count by 1.
    3.2: repeat the step 3 until the entire text has been scanned.
Step4:Arrange the frequency count column in ascending order.
Step5: Based on the minimum-count-value extract the patterns.
Step6:Exit.

## PROPOSED SOLUTION ON MULTI CORES

We partitioned the datasets after preprocessing work and give it to the cores by keeping the load balanced among the cores. The architecture diagram of proposed model with message passing is given as in the following figure--. .

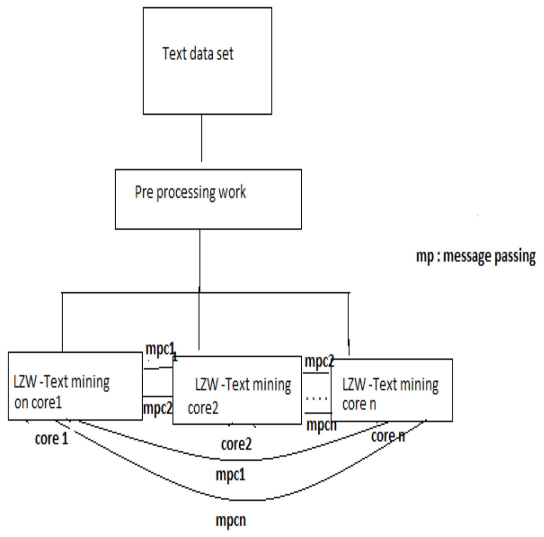


Figure6 Frequent pattern generation model with message passing.

If a processor is found a new pattern then it sends a message to other processors to check whether the pattern is presented in the Pattern Count table of other cores. If any one of the core's tables has that pattern, then it automatically increment it and that core send a send Boolean message as 1 to the sending processor. The other processors which are not having that pattern in their tables will send Boolean messages as 0. The sender cores will check all the Boolean values if all are 0 it comes to know that none of the core is having the pattern so it create a new entry in its table and put a new code for it and set the frequent count as 1. In this proposed scheme there is no much dependency on master core due to the intra message passing between worker cores. The main advantage of this approach is a pattern occurred only once in a core. There is no redundancy of a pattern in cores. Hence it automatically saves the memory. Due to this concept there is no requirement for merging process carried out by the main core as in existing system.

**PFPMLZW ALGORITHM**

The Parallel Frequent Pattern Mining using LZW algorithm is given below.

```
PFPMLZW (input-text, minimum-count-value/Threshold)
Step1: Load input -text data.
Step2: Do the pre-processing work by using Stop words and
        stemming algorithms.
Step3: Extract the Keywords using RAKE (Rapid Automatic
        Keyword Extraction)
Step4: Assign Code for Each String and construct initial
        keyword table.
Step5: Partitioning the given text and give it to cores, keeping
        the load balance.
Step6: Each core will perform the local computation.
6.1: Find the pattern from the given text.
6.2: Check the pattern with table entries whether the pattern
        is already Presented or not. If not send a message
        with new pattern to other cores. 6.2.1. On each
        processor
checks the pattern with PC-table entries that whether the
        pattern is already presented or not.
If presented increment its relevant frequent count by 1 and
        send the updated message as Boolean value 1
Else send message as 0 to the core which sent the
        message.
Step 6.3: Repeat the steps 6.1 and 6.2 until all patterns have
            been scanned.
Step7: By setting minimum count value extract the patterns
        from the tables of all the cores.
Step8: Exit.
```

## IV. RESULT ANALYSIS

The proposed multi core based frequent itemset mining algorithm is implemented with the help of Python 3.5 with Message Passing Interface. We have run the proposed parallel algorithm with 2, 4 and 6 processors and compared it with the FP-tree sequential algorithm. Both parallel and sequential algorithms were executed and the frequent itemsets were generated for a minimum support threshold of 0.1%.

The parameters latency, Speedup and efficiency are measured with three datasets Webdoc, Kosarak and Trump by applying parallel and LZW based algorithm are given below for the reference

**LATENCY**

The Latency time comparison on three datasets between PMC and LZW based multicore algorithm is given below. By seeing the results, we come to know that, on 2 cores, the latency times of LZW based multicorealgorithm takes 114 seconds , nearly half of Parallel multicore system's latency 250 seconds. PMC algorithm takes 136 more seconds delay. In case of 3 cores, PMC algorithm is taking 68 more seconds delay than LZW based multicore algorithm. With 4 cores PMC takes 66 seconds more latency. For 5 cores PMV takes 82 seconds delay than proposed algorithm and with 6 cores PMC takes 45 seconds delay than LZW based multicore algorithm. More or less, the latency difference is not too much differ among the algorithm even though the number of cores increased.

**Trump**

Table2: Latency of Parallel multi core vs LZW based algorithm on Trump.

| Cores | PMC | LZW based Multicore |
|---|---|---|
| 2 | 250 | 114 |
| 3 | 164 | 96 |
| 4 | 124 | 58 |
| 5 | 129 | 47 |
| 6 | 82 | 37 |

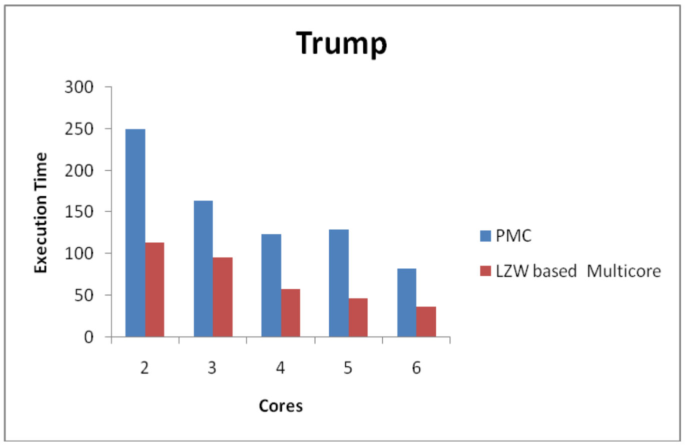


Figure7 Latency time comparison on Trump dataset between PMC and LZW based algorithm.

**Kosarak**

As in the case of Trump dataset Latency, the results show that, on 2 cores, the latency times of LZW based multicore algorithm takes 2938 seconds which is nearly half of Parallel multicore system's 5531 seconds. PMC algorithm takes 2593 more seconds delay. In case of 3 cores, PMC algorithm is taking1664 more seconds delay than LZW based multicore algorithm. With 4 cores PMC takes 1395 seconds more latency. For 5 cores PMV takes 1422 seconds delay than proposed algorithm and with 6 cores PMC takes 857 seconds delay than LZW based multicore algorithm.

Table3: Latency of Parallel multi core vs LZW based algorithm on Kosarak.

| Cores | Parallel Multicore | LZW based Multic |
|---|---|---|
| 2 | 5531 | 2938 |
| 3 | 3659 | 1995 |
| 4 | 2878 | 1483 |
| 5 | 2602 | 1180 |
| 6 | 1836 | 981 |

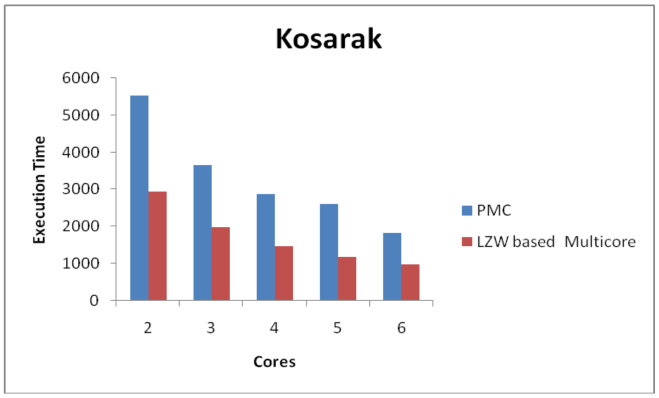


Figure8 Latency time comparison on Kosarak dataset between PMC and LZW based algorithm

**Webdocs**

On Webdocs with 2 cores, the latency times of LZW based multicore algorithm is 41037 seconds are nearly half of Parallel multicore system's latency 78808 seconds. PMC algorithm takes 37771 more seconds delay. In case of 3 cores, PMC algorithm is taking 21302 more seconds delay than LZW based multicore algorithm. With 4 cores PMC takes 16120 seconds more latency. For 5 cores PMV takes 12891 seconds delay than proposed algorithm and with 6 cores PMC takes 10753 seconds delay than LZW based multicore algorithm.

Table4 Latency of Parallel multi core vs LZW based algorithm on Webdocs.

| Cores | Parallel Multicore | LZW base Multicore |
|---|---|---|
| 2 | 73308 | 41037 |
| 3 | 48874 | 27482 |
| 4 | 36657 | 20537 |
| 5 | 29311 | 16420 |
| 6 | 24432 | 13679 |

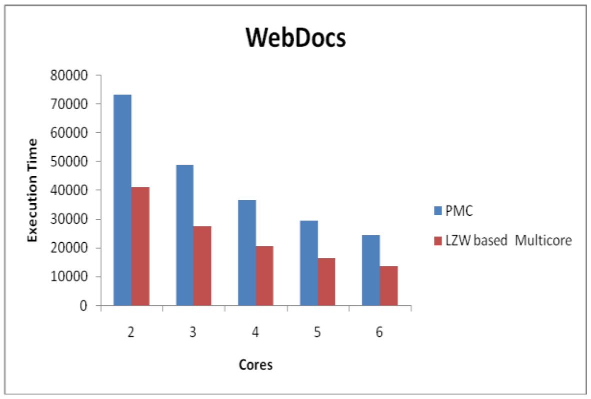


Figure9 Latency time comparison on Webdoc dataset between PMC and LZW based algorithm

**SpeedUp:**

The following table shows the speedup of PMC than single core and the speedup of LZW based algorithm with PMC. From the resultant dataset, the speedup of PMC vs single cores on all three datasets are coming around 2.3 seconds and the speedup of LZW based multicore vs single core for all three types of datasets are nearly around 4.5 seconds.

Table5 Speedup time of Parallel multi core vs Single core on three datasets.

| Datasets | PMC VS SC | LZW based Multicore VS SC |
|---|---|---|
| Trump | 2.590619 | 4.627093 |
| Kosarak | 2.352557 | 4.407747 |
| Webdocs | 2.121951 | 4.702703 |

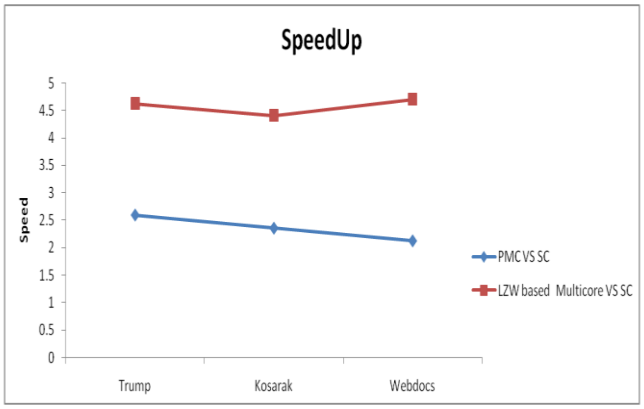


Figure10 The Speedup time comparison on three datasets between PMC and LZW based algorithm

**Efficiency:**
The efficiency of algorithms PMC vs single core and LZW based multicore vs single cores is given in the following table.

Table6 Efficiency of Parallel multi core vs Single core on three datasets.

| Datasets | PMC VS SingleCore | LZW based Multicore VS Singlecore |
|---|---|---|
| Trump | 0.43177 | 0.771182 |
| Kosarak | 0.392093 | 0.734625 |
| Webdocs | 0.353659 | 0.783784 |

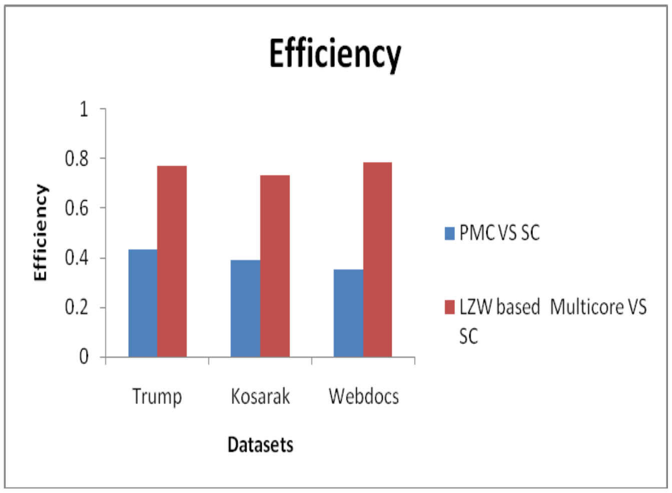
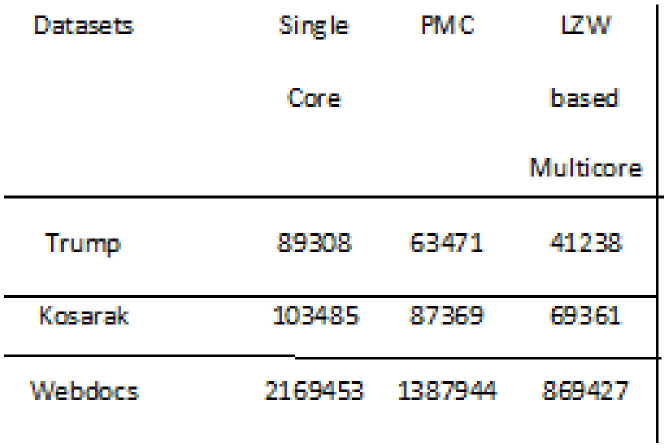


Figure11 The Efficiency comparison on three datasets between PMC and LZW based algorithm

**Memory Usage:**
The memory usage for execution of the algorithms single core, PMC and LZW based multicore on three datasets Trump, Kosarak and Webdoc are given below. In case of Trump dataset, the single core takes 89308 KB, the PMC algorithm takes 63471 KB memory and LZW based multicore takes 41238 KB.

For Kosarak dataset Single core takes 103485 KB, the PMC algorithm takes 87369 KB memory and LZW based multicore takes 69361 KB. With Webdocs dataset the Single core takes 2169453 KB, the PMC algorithm takes 1387944 KB memory and LZW based multicore takes 869427 KB.

Table7 Memory usage of Single core, Parallel multi core and LZW based Multicore on three datasets.



| Datasets | Single Core | PMC | LZW based Multicore |
|---|---|---|---|
| Trump | 89308 | 63471 | 41238 |
| Kosarak | 103485 | 87369 | 69361 |
| Webdocs | 2169453 | 1387944 | 869427 |

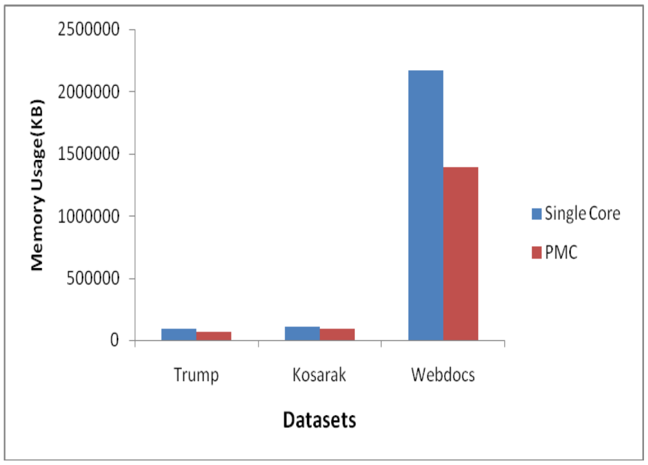


Figure12 Memory usage of Single core, Parallel multi core and LZW based Multicore on three datasets

The above results with 2 to 6 cores on three datasets show that when we move from single core to multicore parallel implementation, the performance is increased on various measures like speedup, latency, execution time, efficiency and memory usage. But when we applied the proposed algorithm the performance is higher than other cases. So we are making proof that the new algorithm is efficient one and it is simple for implementation on multicores with message passing mechanism.

## V. CONCLUSION

Without the novel computational technique, it would be hardly possible to analyze the extremely diverse combination patterns. The parallel implementation of frequent itemset mining algorithms are applied on databases in single core and multicore environments. We are concentrating on three popular pattern mining algorithms Apriori, Eclat and FP-Growth and do analysis work as in step by step manner as follows (i) Apply the algorithms on three types of datasets Webdoc, Kosarak and Trump separately on single core and compare the speedups among them. ( ii) A parallel Implementations of the algorithms with the same dataset is done and analyze the speedups of them. The next step is applying our proposed algorithm PFPMLZW on multicores, that is a parallel implementation LZW algorithm on multi core message passing environment is used to find the frequent items is explained and tested with three experimental datasets Webdoc, Kosarak and Trump on 6 processors environment. Our experiments showed speedups for almost all the cases.

When we intend to improve the multi cores to distribute in future, we can apply the proposed algorithm to distributed environment with large datasets. The processing is depends on the resource availability on the machine environment.

## REFERENCES

[1] Krishna Gadia & Kiran Bhowmick, ‘Parallel text mining in multi core systems using FP-Tree algorithm’, ScienceDirect Procedia Computer Science 45(2015)111-117, 2015

[2] S.K. Tanbeer, C.F. Ahmed, B.S. Jeong, ’Parallel and distributed frequent pattern mining in large databases’, in: Proceeding of the 11th IEEE International Conference on High Performance Computing and Communications, pp. 407–414, 2009

[3] R. Agrawal, R. Srikant, ’ Fast algorithms for mining association rules’, in: Proceedings of the 20th International Conference on Very Large Databases, , pp. 487–499, 1994.

[4] E. H. Han, G. Karypis, & V. Kumar.’ Scalable parallel data mining for association rules’,IEEE Transactions on Knowledge and Data Engineering, Vol. 12, No. 3,2000

[5] J. Han, J. Pei, and Y.Yin. Mining Frequent Patterns without Candidate Generation. In ACM SIGMOD, 2000.

[6] R. Rabenseifner, G. Hager & G. Jost,2009,’ Hybrid MPI/OpenMP parallel programming on clusters of multi-core SMP nodes’, in: Proceeding of the 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing (Feb. 2009), pp. 427–436.

[7] R. Garg & P.K. Mishra,2009,’Some observations of sequential, parallel and distributed association rule mining algorithms’, In: IEEE Proceeding of the International Conference on Computer and Automation Engineering (March 2009), pp. 336–342.

[8] D. Chen, C. Lai, W. Hu, W. Chen, Y. Zhang & W. Zheng, 2006,’ Tree partition based parallel frequent pattern mining on shared memory systems’, in: Proceeding of the 20th International Conference on Parallel and Distributed Processing, pp. 313–320.27.

[9] Lan Vu & Gita Alaghband, 2014, ‘Novel parallel method for association rule mining on multi-core shared memory systems’, ELSEVIER, Parallel computing 40(2014)768-785.

[10] Vu, G. Alaghband, 2012.’ Mining frequent patterns based on data characteristics’, in: Proceedings of the International Conference on Information and Knowledge Engineering, pp. 369–375.20.

[11] CC Aggarwal, 2007, ‘Data streams, models and algorithms’, Springer Science + Business media, books.google.com

[12] Krishna Gadia & Kiran Bhowmick, 2015, ‘Parallel text mining in multi core systems using FP-Tree algorithm’, ScienceDirect Procedia Computer Science 45(2015)111-117.

[13] J.S.Park, M.S.Chen & P.Yu,1995,’ An effective Hash based algorithm for mining association rules’, in Proc: ACM SIGMOD international conference on management of Data, Vol24, pp. 175-186.

[14] H.Mannila, H.Tojvonen & A.I. Verkamo, 1997,’Discovery of frequent episodes in event sequences’, Data Min. Knowl. Discovery 1(3)259-289.