# Parallel Web Content Mining in Multicore System using Apriori Algorithm

**Narendra Parmar [1], Dr. Vineet Richhariya [2], Jay Prakash Maurya[3]**

[1, 2, 3] *Computer Science & Engineering Department Lakshmi Narayan College of Technology, Bhopal, India*

***Abstract— The web content mining is a useful technique for discovering frequently and regularly occurring keywords in web data. Various algorithms have been proposed to speed up the web mining performance on the single and multicore systems. Unfortunately, when the volume of web dataset size is large, then both the computational cost and memory use can be extremely expensive. In this paper, we parallelized the Apriori algorithm using message-passing interface (MPI) on the multicore machines. We divided the large volume of database into the total number of CPU cores, and applied the integrated potential of all the CPU cores, to accomplish the maximum throughput. Our experimental result presents that the parallel execution of web content mining using Apriori algorithm achieves high speed up ratio as compared to serial execution. Finally, this paper is concluded with many future implementation and analysis.***

***Keywords—Web Content Mining, Data Mining, Apriori Algorithm, Parallel Processing.***

## 1. INTRODUCTION

One definition of web content mining is the deep understanding of variation in World Wide Web data sets. At its simplest, web content mining is concerned with finding and extracting the valuable information and the variations in complex World Wide Web [1]. At its main complexity, it is concerned with discovery of the most influential or most useful theoretical explanations for experimental variations.

For two reasons, web content mining is the essential application of data mining that parallel computing has been applying. First,analyzing deviation appears to be algorithmically difficult and hence it require levels of computing influence that only parallel computers can provide in a timely manner [2]. Second, the large web content data sets involved are huge and rapidly increasing larger and parallel computers are used to handle these large volumes efficiently and effectively, although some web content and data mining limitations are already demanding and challenging their limits [2][3].

This paper aims at presenting a large amount of broader view of the research of parallel web content mining, by discussing the parallelization using Apriori data mining algorithms [4]. Morespecifically, the parallelization of web content mining algorithms are discussed in the following two essential techniques first is Apriori algorithm and second one is MPI (Message passing Interface) [5].A typical web content mining application starts from the web content data set describing interactions between individual units and a central multifaceted unit[6]. The individuals unit are end users or customers and the central unit is a sales organization. The fundamental requirement is that the individual units revealvariationsand the central unit measures those variations. The web content miner requires to understand the deeper processes that generate those variation, for better market products in the primary case, to enhance production in the second case, and to discover new patterns about the web supported market in the third case [7]. The web content mining effectively enhances the marketing and business strategies to higher level. So there is a critical requirements to improve the existing web content mining techniques and methods [7][8].

This paper is organized as follows: Section 2 reviews the distinction between data parallelism and control parallelism. Section 3 discusses literature review for parallelizing web content mining. Section 4 discusses proposed method to overcome the limitations of existing work. Section 5 discusses the result of proposed work. Section 6 concludes with some future research directions.

## 2. DATA PARALLELISM VS CONTROL PARALLELISM

This Section reviews the distinction between data parallelism and control parallelism, which is crucial for an understanding of the mining process. In essence, data parallelism refers to the execution of the same operation, instruction on multiple large data subsets at the same time as illustrated in Figure 1 This is in contrast to control parallelism, or operation parallelism, which refers to the concurrent execution of multiple operations or instructions, as illustrated in Figure 2. Data parallelism has three main advantages over control parallelism:

(i). First, data parallelism gives itself to a type of automatic parallelization. The control flow of a data parallel program is fundamentally the same as the control flow of a sequential program, only the entrance to the data is parallelized. Hence, many previously written sequential

code can be re-used in a data-parallel fashion [9]. This simplifies programming and leads to a development time significantly smaller than the one associated with control-parallel programming.

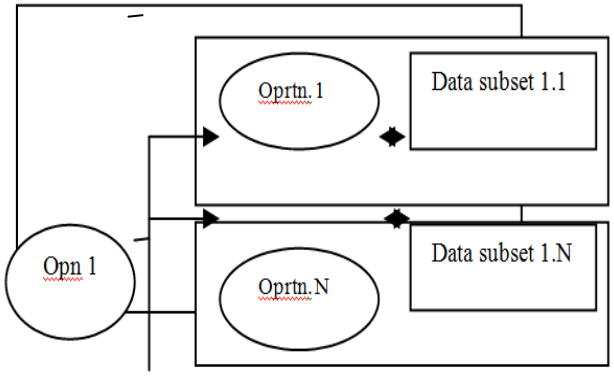


Figure 1: Data Parallelism (one operation executed on N processors, via data partitioning).

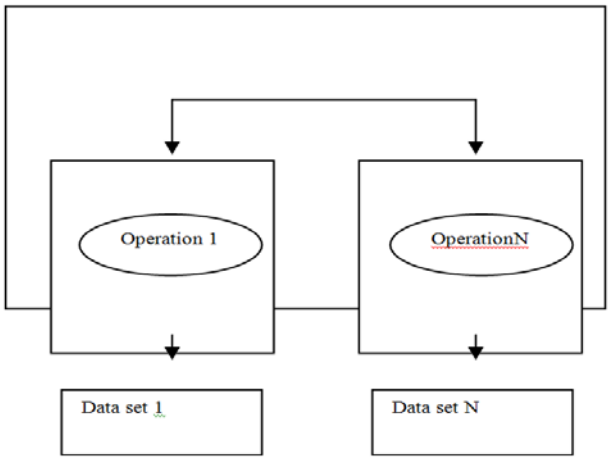


Figure 2: Control parallelism (N operations executed on N processors).

(ii). Second, data parallelism has a higher degree of machine architecture independence, as compared to control parallelism. However, the control flow of a data-parallel algorithm is still sequential only data handling is parallelized, there is no need to modify the control flow of the algorithm in the underlying parallel architecture [9]. This is in dissimilar with control parallelism, where this kind of modification is one of the major challenges of parallel programming [9]. Note that the problem of machine architecture. Dependence is not completely removed in data parallelism. This limitation is simply pushed down to a lower layer of software, hidden from the applications programmer, which leads to an enhancement in programmer productivity [9].

(iii). Third, intuitively data parallelism has better scalability for huge volume of databases than control parallelism [9]. In most database applications of data mining, the amount of data can increase arbitrarily faster, while the number of lines of code typically increases at a much slower rate. Simply, the more data is available, the more the opportunity to exploit data parallelism.

## 3. LITERATURE REVIEW

In this section, the existing tools, techniques and methods for parallelization of web content mining are analyzed and reviewed. In web content mining, web content is a form of data in which the attributes of data are defined in high dimensional attributes. The existing techniques which are basically applied for web content mining are follows:

### A. *The Count Distribution Algorithm*

Count Distribution, is a parallel Apriori-based algorithm which is used to mine for frequent patterns. Each processor computes its local candidate item set, along with their support count, by performing a single pass over its local data partition at that time. Information is maintained in a hash-table, which is identical for each processor. This process is accomplished by running a sequential Apriori on each processor. All local counts are then accumulated and summed together to form a global support count using a global reduction function [10]. Global reduction consists of two other operations. One of the operations is referred to as 'Reduce Scatter', which is responsible for obtaining local support count communication from a processor, and the other operation is called 'All Gather' operation, which is responsible for global support count communication.
Count Distribution seems to scale linearly to the number of records of the dataset as all computations to find support counts can be done locally at each processor having minor communication only at the end for accumulating the counts [10][11]. However, in case the hash-table structure cannot fit into the main memory of each processor, it must be partitioned and support counts are computed by performing multiple scans of the dataset, one scan for each partition of the hash-table. Thus, the algorithm is efficient only if candidate set generated leads to a hash-table with reasonable size relatively to the size of main memory. It is important to notice that the number of candidates becomes larger as the number of distinct items in the dataset increases or as the minimum threshold decreases [11]. In general, Count Distribution works better for small number of distinct items and high levels of minimum thresholds.

### B. *The Data Distribution Algorithm*

The Count Distribution algorithm is attractive in the sense that no data movement is performed. All the counts are

computed locally to each processor thus every processor can operate asynchronously on its own data. However, this limits the ability of taking advantage of non-local memory parallel machines provide. The Data Distribution algorithm solves this problem by allowing each processor to compute the support counts of its locally stored subset of the candidate item sets for all the transactions in the database [12].

In order for this to become feasible, the All-to-All broadcast is used, where each processor must scan its own partition of the data as well as other partitioned data located at remote processors. This results in every processor having to broadcast their data to all other participating processors as well as receive data from them [12]. Although this will solve the problem that Count Distribution carries with it, there are still negative effects as far as the burden placed in communication operations as there is a high communication overhead created due to data movement [12]. Furthermore, such a communication scheme as this one causes the processors to become idle while waiting for data to be broadcasted resulting in wasting time that could have been manipulated for useful processing.

### C. *The Candidate Distribution Algorithm*

Both Count and Data distribution algorithms carry the limitation that there is some synchronization involved. Although in Count Distribution, each processor can compute its own candidates asynchronously, some synchronization is required when global counts are about to be summed. In case the workload is not perfectly balanced, some processors may have to remain idle until others are finished. Similarly, in Data Distribution, synchronization is needed when data is broadcasted around the processors [13]. Furthermore, since any database transaction could support any candidate item set, each transaction must be compared against the entire candidate set. For this reason, Count Distribution needs to duplicate the dataset in every processor and Data Distribution needs to broadcast all of the transactions.

Candidate Distribution [13] combines the ideas used in both previous algorithms in order to overcome the problems associated with idle time, communication, and synchronization issues. This is achieved by duplicating the data on every processor's local memory as well as partitioning the candidate set across processors. In this way every processor can proceed independently, using its part of candidates on its local data. There is no need to exchange data or counts using this algorithm. The only communication required is when pruning a local candidate set during the phase of pruning in candidate generation [13]. However, there is no need for synchronization at this stage thus no processor has to remain idle until pruning updates from other processors arrive.

### D. *Parallel Multipass with Inverted Hashing and Pruning (PMIHP) Algorithm*

The new Parallel Multipass with Inverted Hashing and Pruning (PMIHP) algorithm is a parallel version of the sequential Multipass with Inverted Hashing and Pruning (MIHP) algorithm. The MIHP algorithm is based on the Multipass approach [14] and the Inverted Hashing and Pruning (IHP) [14] that it is proposed. In PMIHP, the Multipass approach reduces the required memory space at each processor by partitioning the frequent items and processing each partition separately. Thus, the number of candidate itemsets to be processed is limited at each instance. The Inverted Hashing and Pruning is used to prune the local and global candidate itemsets at each processing node, and it also allows each processing node to determine the other peer processing nodes to poll in order to collect the local support counts of each global candidate itemset [14].

The Multipass algorithm partitions the frequent items, and thus partitions the candidate item-sets. The partition size is selected to be small enough to fit in the available memory of the processing node. Each partition is then processed separately. Each partition of items contains a fraction of the set of all items in the database, so that the memory space required for counting the occurrences of the sets of items within a partition will be much less than the case of counting the occurrences of the sets of all the items in the database.

## 4. PROPOSED METHOD

The previous existing algorithm "Multipass with Inverted Hashing and Pruning (MIHP)" and "Parallel Multipass with Inverted Hashing and Pruning (PMIHP)" works on data parallelization. These algorithms work as application programming on operating system. The solution is the requirement of task parallelization instead of data parallelization. Task parallelization is more efficient than data parallelization in context of time complexity. Also there is the requirement of system programming instead of application programming. The proposed method to overcome the limitations of previous method is combining Apriori Algorithm with MPI (Message Passing Interface).

*Apriori Algorithm:*

Apriori uses breadth-first search and a Hash tree structure to count candidate item sets efficiently. It generates candidate item sets of length k from item sets of length $k - 1$. Then it prunes the candidates which have an

infrequent sub pattern. According to the downward closure lemma, the candidate set contains all frequent k-length item sets. After that, it scans the transaction database to determine frequent item sets among the candidates. The pseudo code for the algorithm is given below for a transaction database T, and a support threshold of ϵ. Usual set theoretic notation is employed, though note that T is a multiset. $C_k$ is the candidate set for level k. At each step, the algorithm is assumed to generate the candidate sets from the large item sets of the preceding level, heeding the downward closure lemma. count[c] accesses a field of the data structure that represents candidate set c, which is initially assumed to be zero. Many details are omitted below, usually the most important part of the implementation is the data structure used for storing the candidate sets, and counting their frequencies. The Apriori algorithm is summarized as:

$$Apriori\ (T, \epsilon)$$
$$L_1 \leftarrow \{large\ 1-itemsets\}$$
$$k \leftarrow 2$$
$$while L_{k-1} \neq \emptyset$$
$$C_k \leftarrow \{a \cup \{b\} \mid a \in L_{k-1} \wedge b \notin a\} - \{c \mid \{s \mid s \subseteq c \wedge |s| = k-1\} \notin L_{k-1}\}$$
$$\textbf{for} transactions t \in T$$
$$C_t \leftarrow \{c \mid c \in C_k \wedge c \subseteq t\}$$
$$\textbf{for} candidates c \in C_t$$
$$count[c] \leftarrow count[c] + 1$$
$$L_k \leftarrow \{c \mid c \in C_k \wedge count[c] \geq \epsilon\}$$
$$k \leftarrow K + 1$$
$$\textbf{return} \bigcup_k L_k$$

*Message Passing Interface (MPI):*

Message Passing Interface (MPI) is a message passing library standard based designed by a group of researchers from academia. It is mainly used for task parallelization into CPU cores. It open source, standard, core of library routines useful in parallel computing for C, C++ & Fortran program. It is introduced as a header file # include ”mpi.h” in C programming language. Programming process have multiple threads sharing a single address space. Message Passing Interface **(**MPI) is for communication among processes, which have separate address spaces. MPI uses MIMD parallelism.

## 5. RESULT

*Serial execution Vs parallel execution***:**

*Consideration of database (size=5577)*

here we have taken a database of size 5577 transaction and perform mining of item serial as well as parallel. The corresponding result is shown in table. We analyze it through graph.

Table between execution of time and threshold in serial and parallel algorithm

Table 1: Execution time in serial and parallel algorithm

| Database-(5577) | Serial Execution (seconds) | Parallel Execution (seconds) |
|---|---|---|
| Threshold--.10 | 0.104 | 0.048 |
| Threshold--.15 | 0.06 | 0.04 |
| Threshold--.20 | 0.044 | 0.038 |

*Scalability Evaluation:*

Serial run-time is indicated using $T_s$

Parallel run-time using $T_p$.

- *Speed Up:*

Speed-up, S, is defined as “the ratio of the serial run-time of the best sequential algorithm for solving a problem to the time taken by the parallel algorithm to solve the same problem on p processors:

$$S = \frac{T_s}{T_p}$$

Graph Corresponding to Table 1:

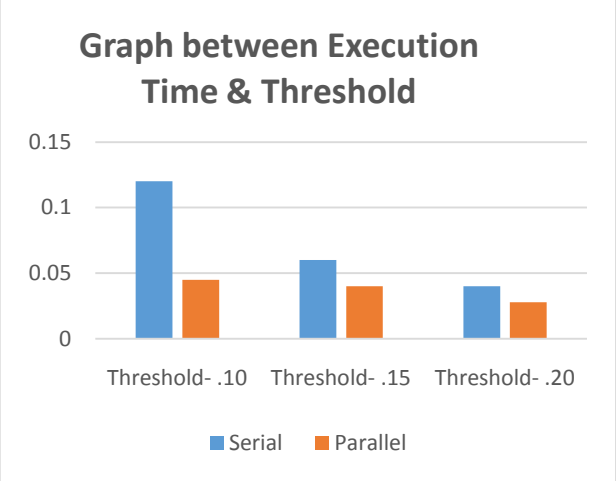


Figure 3: Graph between execution time and threshold

- *Efficiency:*

Efficiency E, is "a measure of the fraction of time for which a processor usefully employed"; it is defined as "the ratio of speed-up to the number of processors".

$$E = \frac{S}{P}$$

Result corresponding to Speed up and Efficiency

Table 2 : Table for speed up and efficiency

| Database=5577 | Threshold=.10 | Threshold=.15 | Threshold =.20 |
|---|---|---|---|
| Speed up | 2.16 | 1.5 | 1.15 |
| Efficiency | 0.72 | 0.5 | 0.38 |

Graph corresponing to table 2:

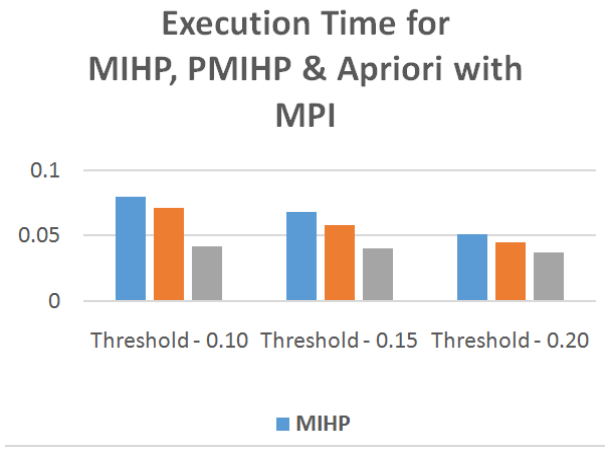


Figure 4: Graph of Execution Time for MIHP, PMIHP & Apriori with MPI

For lower threshold values, the proposed method presents high efficiency, however, for higher threshold values, it presents lower efficiency.

## 6. Conclusion

In this paper, the web contents mining frequent item through MPI in transactional database was performed for making difference between sequential and parallel algorithm, the time of making APRIORI Tree was noted for both algorithms on different threshold. After compare the result of different metric with idle parallel system metric, conclusion is that proposed method and technique works good in parallel environment. MPI is used here for message communication among processes and multiple threads to achieve the true task parallelization. This method and technique helps MPI to communicate in parallel environment.

In the paper, processing times with parallel computing and that with single processor computing have been compared. The future extensions to this paper include some updates such as using optimization to make the database efficient. The database here comprised of numeric values. The MPI can be integrated with fp-growth algorithm in future, also can be applied with Big Data.

## REFERENCES

[1]. B.Santhosh Kumar and K.V.Rukmani, "Implementation of Web Usage Mining Using Apriori and FP Growth Algorithms, International Journal of Advanced Networking and Applications", Ketti, The Nilgiris, Vol 01, pp.400- 404, 2010.

[2]. J. Pei et al. H-Mine. "Hyper-structure mining of frequent patterns in large databases" In proceeding of the IEEE Conference on Data Mining, November 2001.

[3]. Clemens Költringer, Astrid Dickinger, "Analyzing destination branding and image from online sources: A web content mining approach", ournal of Business Research, Elsevier, Vol. 68, Issue 9, September 2015, pp. 1836-1843.

[4]. José Antonio Iglesias, Alexandra Tiemblo, Agapito Ledezma, Araceli Sanchis, "Web news mining in an evolving framework", nformation Fusion, Elsevier, Vol. 28, March 2016, pp. 90-98.

[5]. Petar Ristoski, Heiko Paulheim, "Semantic Web in data mining and knowledge discovery: A comprehensive survey", Web Semantics: Science, Services and Agents on the World Wide Web, Vol. 36, January 2016, pp. 1-22.

[6]. Amit Dutta; Sudipta Paria; Tanmoy Golui; Dipak K. Kole, "Structural analysis and regular expressions based noise elimination from web pages for web content mining", International Conference on Advances in Computing, Communications and Informatics (ICACCI, 2014, pp. 1445 – 1451.

[7]. Unil Yun, Gangin Lee, "Incremental mining of weighted maximal frequent itemsets from dynamic databases", Expert Systems with Applications, Elsevier, Vol. 54, 15 July 2016, pp. 304-327.

[8]. Mingxing Wu, Liya Wang, Ming Li, Huijun Long, "An approach of product usability evaluation based on Web mining in feature fatigue analysis", Computers & Industrial Engineering, Elsevier, Vol. 75, September 2014, pp. 230-238.

[9]. Cheng Wang; Ying Liu; Liheng Jian; Peng Zhang, "A Utility-Based Web Content Sensitivity Mining Approach", IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT '08. Vol. 3, pp. 428 – 431.

[10]. Tarique Anwar; Muhammad Abulaish; Khaled Alghathbar, "Web content mining for alias identification: A first step towards suspect tracking", IEEE International Conference on Intelligence and Security Informatics (ISI), 2011, pp. 195 – 197.

[11]. R. Etemadi; N. Moghaddam, “An approach in web content mining for clustering web pages”, Fifth IEEE International Conference on Digital Information Management (ICDIM), 2010, pp. 279 – 284.

[12]. YoonKyung Cha, Craig A. Stow, “Mining web-based data to assess public response to environmental events”, Environmental Pollution, Elsevier, Vol. 198, March 2015, pp. 97-99.

[13]. Petar Ristoski, Christian Bizer, Heiko Paulheim, “Mining the Web of Linked Data with RapidMiner”, Web Semantics: Science, Services and Agents on the World Wide Web, Elsevier, Vol. 35, Part 3, December 2015, pp. 142-151.

[14]. D.A. Adeniyi, Z. Wei, Y. Yongquan, “Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method”, Applied Computing and Informatics, Elsevier, Vol. 12, Issue 1, January 2016, Pages 90-108.Jo R. Etemadi; N. Moghaddam