IJCAT International Journal of Computing and Technology, Volume 1, Issue 3, April 2014
ISSN : 2348 - 6090
www.IJCAT.org

Implementing Apriori Algorithm in Parallel

[1] Nilesh S. Korde, [2] Shailendra W. Shende

[1] Department of Information Technology, Nagpur University, YCCE
Nagpur, Maharashtra, India

[2] Department of Information Technology, Nagpur University, YCCE
Nagpur, Maharashtra, India

Abstract - A Huge amount of data gets collected from society with different sources. Hardly has it led to a useful knowledge. For finding useful knowledge an algorithm is required. Apriori is an algorithm for mining data from databases which shows items that are related to each other. The databases having a size in GB and TB need a fast processor. For fast processing multi-core processors are used. Parallelism is used to reduce time and increase performance, Multi-core processor is used for parallelizing. Serial mining can consume time and reduce performance for mining. To solve this issue we are proposing a work in which load balancing is done among processors. In this paper we have implemented Apriori algorithm in serial and parallel manner and comparison of both on the basis of varying support-count and time using parallel programming technique Open Multi-Processing (OpenMP).

Keywords **- Parallel processing, Apriori, OpenMP, Data mining, Multicore processing.**

1. Introduction

Technique for mining hidden knowledge from a large number of databases, knowledge which is never seen before Data mining is used [1]. For knowledge discovery data mining is used. Data mining has Association rule mining (ARM) as one of the functionality. To find the relation between two items Association rule mining is used, that frequently appearing in a large database with each other. Mining frequent itemsets from huge transactional databases Apriori algorithm is one that is widely used. Apriori algorithm scans database more than once for finding frequent itemsets [1].Increase in performance of algorithm is required for dealing with large volume of data during mining. Parallelizing Apriori algorithm can improve performance of Apriori algorithm drastically [2].Recently launched advanced computers provides multi-core architecture for parallelizing Apriori algorithm. Multi-core processors are available in advanced computers [3]. Multi-core processor shares the

load among cores with single thread or multiple threads, which reduce time for processing and increase performance [4].

2. Related Work

Chao Yang, Tzu Chang and Chih Chang presents the comparison of some tools that are specifically designed to extract the most of data parallelism on multi-core system using OpenMP [3].Ying Liu and Fuxiang Gao presents the cubic convolution interpolation algorithm for image processing and parallelized it by using OpenMP on multi-core processors [4]. Ketan shah and Sunita Mahajan present the performance of parallel Apriori algorithm on heterogeneous nodes with different datasets and n processors on a commodity cluster of machines [6]. Anuradha.T and Satya Prasad.R presents an evaluation of the performance of Apriori on a hyper threaded dual core processor compared to the performance on a non hyper threaded dual core processor using fread() and mmap() functions [7]. Rakesh Agrawal and Ramakrishna srikant presents two new algorithms Apriori and AprirориTid and combined into an Apriori hybrid algorithm and demonstrate in scale-up-properties [5]. Kyung Min Lee, Tae Houn Song evaluates a parallel programming model, parallel programs and OpenMP that can be benchmarked to multi-core processors of embedded boards using OpenMP and executed parallel programs on a dual-core embedded system, analyzing the performance of sequential programs and parallel programs by SERPOP analysis [9]. Anuradha.T, Dr.Satya Prasad.R, Dr.Tirumala Rao.S.N gives the performance of Apriori using Linux mmap() function compared to fread() function in both the serial and parallel environments [10]. Jaiwen Li, Zhang Zheng, Xuhoo Chen, Li Shen, and Zhiying Wang give a performance model for OpenMP parallelized loops to address the critical factors which influences the performance [8].

3. Theoretical Background

A. Apriori Algorithm

It is nothing but finding frequent itemsets using candidate generation. It uses Apriori property that all nonempty subsets of a frequent itemset must also be frequent. Two steps used in Apriori algorithm are

Step 1: The Prune Step: To find the count of each candidate in Ck the entire database is scanned. Candidate k-itemset is represented by Ck. To find whether that itemset can be placed in frequent k-itemset Lk to count each itemset in Ck is compared with a predefined minimum support count [1].

Step 2: The join step: Lk is natural joined with itself to get the next candidate k+1- itemset Ck+1.

The major step here is the prune step which requires scanning the entire database for finding the count of each itemset in every candidate k-itemset. If the database size is large, so to find all the frequent itemsets in the database, it requires more time [1]

B. OpenMP

For shared memory parallel programming OpenMP (OpenMulti-Processing) an Application programming interface (API) is used. This specification provides a model for parallel programming that is portable across shared memory architecture from different vendors [11]. The OpenMP API supports Compilers from numerous vendors. Compiler directives, Environment variables and Runtime library routines are the primary components of the OpenMP API [12]. OpenMP provides a task construct useful for the expression of unstructured parallelism and for defining dynamically generated unit of work [13]. The advantage of OpenMP is its comparative simplicity of use, because the detailed working for parallel program is up to the compiler [12]. OpenMP supports multi-platform shared memory multiprocessing programming in FORTRAN, C/C++ and on much architecture, including Microsoft Windows platforms and UNIX [13].

C. Multi-core Processor

Multi-core processors are those having two or more cores integrated on a single chip [14]. The core can be with one thread or two threads each [14]. For parallelism Multi-core processor plays very important role as the cores share work and helps to perform load balancing. The performance of multi-core processor depends on the software used. The work is equally shared among the processors in load balancing which minimizes the execution time and improve performance [14].

4. Experimental Work

We have implemented Apriori algorithm on a quad core prosessor in a serial and parallel manner. Our objectives are measuring the serial and parallel performance of Apriori agorithm on a quad core processor with respect to time and comparison between them. For our work we mainly used Ubuntu 12.10 Linux operating system, intel core 2 quad processor, and four standard datasets from Frequent Itemset Mining Repository. Different support counts from 30% to 70% for measuring performance is used. The Real time observations are as follows

Real time values for ACCIDENTS Dataset			
Minimum support count	Serial time in seconds	Parallel time in seconds	Item sets generated
30	115.980	112.21	149545
40	36.390	34.630	32528
50	6.040	5.240	8057
60	2.010	1.190	2074
70	1.450	0.650	529

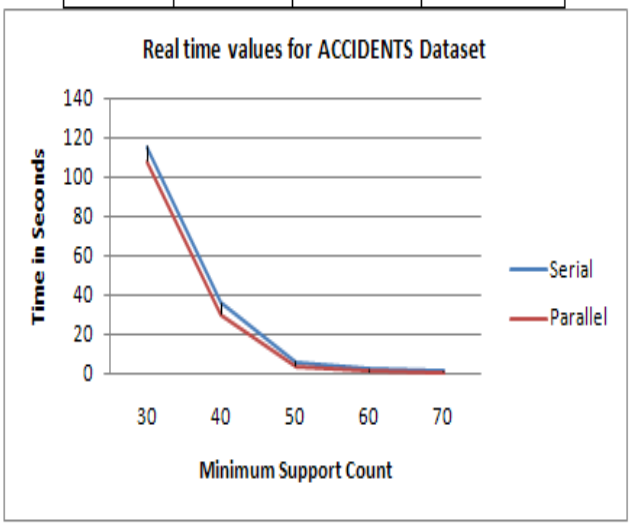

Fig1: Real time values for ACCIDENTS Dataset

Real time values for CHESS Dataset			
Minimum support count	Serial time in seconds	Parallel time in seconds	Item sets generated
30	285.520	283.89	37282962
40	82.030	80.020	64397
50	20.460	19.320	12729
60	3.510	2.910	2549
70	0.200	0.190	487

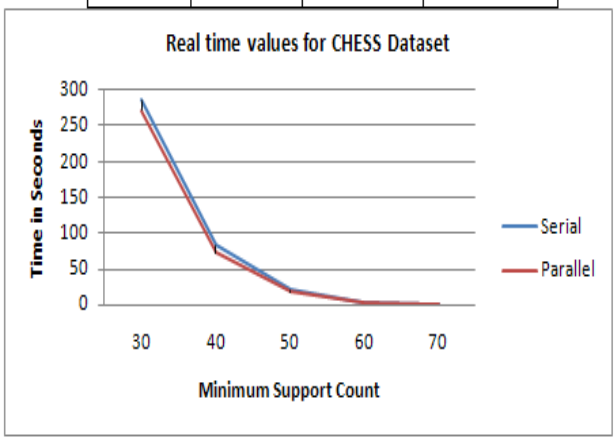

Fig2: Real time values for CHESS Dataset

Real time values for RETAIL Dataset			
Minimum support count	Serial time in seconds	Parallel time in seconds	Item sets generated
30	0.184	0.063	7
40	0.180	0.060	5
50	0.170	0.040	4
60	0.120	0.022	2
70	0.100	0.011	1

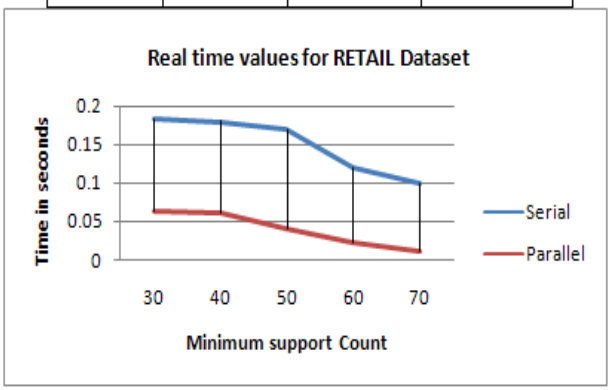

Fig3: Real time values for RETAIL Dataset

Real time values for MUSHROOM Dataset			
Minimum support count	Serial time in seconds	Parallel time in seconds	Item sets generated
30	0.026	0.010	2735
40	0.020	0.005	565
50	0.010	0.003	153
60	0.010	0.003	51
70	0.010	0.001	31

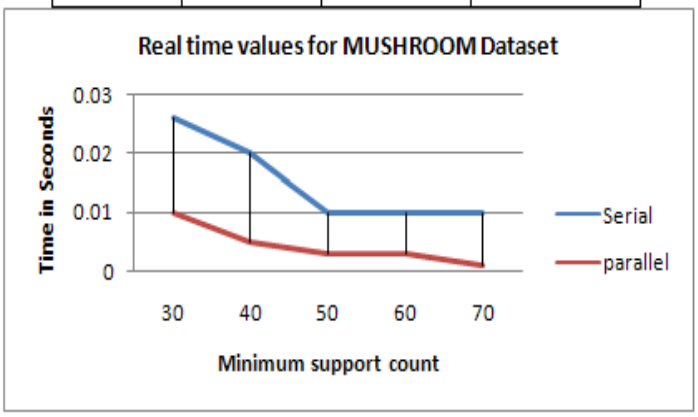

Fig4: Real time values for MUSHROOM Dataset

real time observations has shown that as support count increases the time in seconds decreases. we need to run Apriori algorithm in parallel to reduce the time and to increase the performance further. OpenMP is going to use for parallel implementation of Apriori algorithm on a quad core processor.OpenMP uses Fork and Join concept for parallelism [14].

In an OpenMP Fork and Join model the program begines with master thread, some part of program can be make work in parallel by creating chlid threads. Master thread work in a serial manner until parallel construct is encounterd[15]. Master thread create a team of child thread which work in parallel in a parallel region.The work is divided among the child threads equally by parallel construct. After the execution the master thread continues but the child threads get synchronized and enumerated[16].

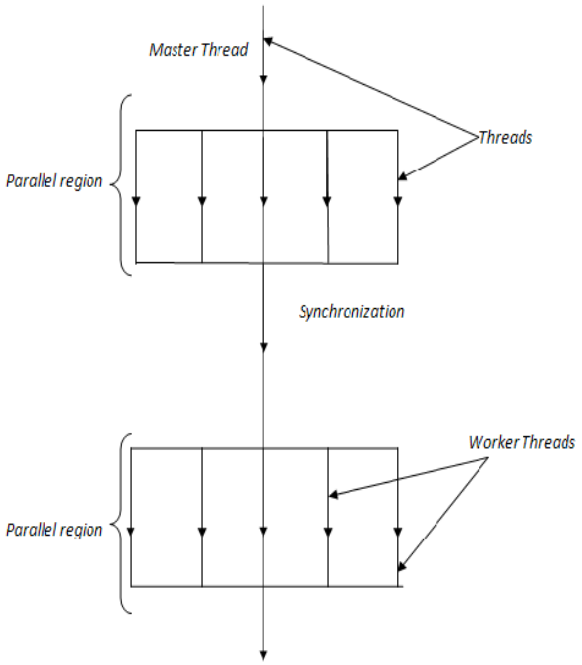

Fig5: Fork and Join Model

5. Conclusion

In this paper, we have implemented Apriori algorithm on a quad core processor in a serial and parallel manner with four standard datasets using different support counts varing from 30% to 70%. From our work we can say that parallel results are better than serial once.Our aim of measuring the serial and parallel performance of Apriori agorithm on a quad core processor with respect to time and comparison between them get satisfied.

References

[1] Han and Micheline Kamber, *Data Mining concepts and Techniques* 2nd edition Morgan Kaufmann Publishers, San Francisco 2006.

[2] Anuradha.T, Satya Prasad R and S N Tirumalarao. *Parallelizing Apriori on Dual Core using OpenMP. International Journal of Computer Applications* 43(24):33-39, April 2012. Published by Foundation of Computer Science, New York, USA.

[3] Chao-Tung Yang, Tzu-Chieh Chang, Hsien-Yi Wang, William C.C. Chu, Chih-Hung Chang. Performance Campion with OpenMP Parallelization for Multi-core Systems. Ninth IEEE International Symposium on Parallel and Distributed Processing with Applications, 2011 IEEE, pp 232-237.

[4] Ying Liu, Fuxiang Gao. Parallel Implementations of Image Processing Algorithms on Multi-Core. 2010 Fourth International Conference on Genetic and Evolutionary Computing, 2010 IEEE, pp 71-74.

[5] Agrawal R, Srikant R "Fast algorithms for mining association rules" In: Proceedings of the 1994 international conference on very large data bases (VLDB‟ 94), 1994 Santiago, Chile, and pp 487–499.

[6] Ketan D. Shah, Dr. (Mrs.) Sunita Mahajan. Performance Analysis of Parallel Apriori on Heterogeneous Nodes. 2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies, 2009 IEEE, pp 42-44.

[7] Anuradha.T, Satya Prasad. *Parallelizing Apriori on Hyper-Threaded Multi-Core Processor.* International Journal of Advanced Research in Computer Science and Software Engineer. *Volume 3*, *Issue 6*, June 2013.

[8] Zhong Zheng, Xuhao Chen, Zhiying Wang, Li Shen, Jiawen Li. Performance Model for OpenMP Parallelized Loops. 2011 International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE) December 16-18, Changchun, China, 2011 IEEE, pp 383-387.

[9] Kyung Min Lee, Tae Houn Song, Seung Hyun Yoon, Key Ho Kwon, Jae Wook Jeon. OpenMP Parallel Programming Using Dual-Core Embedded System. 2011 11th International Conference on Control, Automation and Systems Oct. 26-29, 2011 in KINTEX, Gyeonggi-do, Korea, pp762-766.

[10] Anuradha.T, Dr.Satya Prasad.R, Dr.Tirumala Rao.S.N *Performance evaluation of apriori with memory mapped files*. International Journal of Computer Science Issues *Vol.10*, *Issue 1*, no1, January 2003.

[11] "OpenMP Application Program Interface" Version 3.0 May 2008.

[12] Tim Mattson and Larry Meadows. "A 'Hands-on' Introduction to OpenMP" Intel Corporation.

[13] Kent Milfeld. "Introduction to Programming with OpenMP" February 6th 2012, TEXAS ADVANCED COMPUTING CENTER (TACC).

[14] Multi-core Processor Wikipedia [Available] en.wikipedia.org/wiki/Multi-core processor.

[15] *Blaise Barney, Lawrence Livermore.* Introduction to ParallelComputing.https://computing.llnl.gov/tutorials/parallel_comp/.

[16] Frank Willmore. "Introduction to Parallel Computing", February 6, 2012.