A Performance Analysis of Association Rule Mining Algorithms

Sallam Osman Fageeri[1], Rohiza Ahmad[2] and Hitham Alhussian[3]

[1] Faculty of Computer Science & Information Technology, Alzaiem Alazhari University, Sudan

[2,3]Computer & Information Sciences Department, Universiti Teknologi PETRONAS, Malaysia

[1]sallam_fageeri@hotmail.com, [2]rohiza@petronas.com.my, [3]halhussian@gmail.com

***Abstract*—In this paper, we evaluate the performance of association rule mining algorithms in-terms of execution times and memory usage using the CPU profiler of Java VisualVM. Mainly, and according to a previous work, we studied the performance of two main association rules algorithms which exhibits best results interms of execution time and memory usage: Binary-Based algorithm, and Eclat algorithm. The results of the CPU profiler of Java VisualVM showed that Binary-Based algorithm performs better than Eclat algorithm in-terms of both execution times and memory usage. In fact the results showed that Eclat algorithm sometimes even fails to produce any results and reports running out of memory exceptions.**

I. Introduction

The problem of mining association rules in large-scale databases is motivated by the fact that the volume of data doubles every year, however, techniques to extract the usefulness of these data on the other hand seems to be progressing at a much slower rate. Although traditional statistical techniques and data management tools already exist; but they are no longer adequate for analyzing the complexities of the immense data [1]. Similarly, typical tools of decision support systems are also not adequate for automated prospective analyses. Hence data mining techniques have arisen in the past recent years to tackle this problem. Although numerous number of frequent patterns and association rules mining approaches have recently been presented, but they still suffer the problem of low performance in-terms of execution times and main memory usage. This is due to the fact that the process of mining frequent patterns and association rules is known to be computationally expensive and input output intensive. In this paper, a performance analysis of association rules mining algorithms in-terms of execution time and memory usage is presented. According to a previous work [2], we studied the performance of two main algorithms which exhibits best results in-terms of execution time and memory usage: Binary-Based algorithm, and Eclat. The performance analysis has been conducted using Java VisualVM of oracle.

The rest of this paper is organized as follows: Section II reviews related works. Section III discusses the simulation methodology. Section IV presents and discusses the obtained results. The then concludes on Section V.

II. Literature Review

The problem of mining association rules can be decomposed into two sub problems:

1. Finding all sets of items (itemsets) having support which is greater than the user-specified minimum support value *Sup*. Itemsets with minimum support are called large frequent itemsets. All others items are said to be small, or infrequent items. The support *Sup* of an item (itemset) is calculated using equation 1.

$$Sup\ (X \Rightarrow Y, D) = \frac{Sup(XUY, D)}{|D|} \qquad (1)^1$$

2. Extract the association rules; the rules that have minimum confidence *Conf* greater than the user-specified minimum confidence, from the frequent itemsets. The confidence *Conf* between two items (itemsets) is calculated using equation 2.

$$Conf(X \Rightarrow Y, D) = \frac{Sup(XUY, D)}{Sup(X, D)} \qquad (2)$$

Frequent patterns and association rules mining algorithms can be categorized in two main categories [3]:

- Apriori and Apriori-like algorithms.
- Frequent Pattern (FP)-Growth and FP-Growth-like algorithms.

Apriori and Apriori-like algorithms suffer from two potentially very high computational costs [4]: (1) The cost of candidate itemsets generation and testing, and (2) the cost of repeatedly scanning the database transactions.

Hence, a new family of algorithms have been proposed without candidate itemsets generation referred to as FP-Growth and FP-Growth like algorithms. These algorithms builds a compact data structure known as FP-Tree [4] from the original database

[1] X and Y represents an itemsets in a database, where D represents database size.

 978-1-5090-2549-7/16/$31.00 ©2016 IEEE

transactions. However building the compact FP-tree takes a lot of time and consumes a lot of memory to store the database transactions (see Fig.1.2 which shows the failure of the well-known FP-Growth algorithm which ran out of memory in Pumsb datasets with 0.08 support).

A different approach to deal with this problem is introduced by vertical data format algorithms such as Eclat [5, 6] in which each item is associated with a list containing all transaction identifiers (TID-list) in which the item appears. Such a representation is known as the vertical data layout format. The length of these lists of TIDs shrinks while moving towards itemsets with larger size. However, the main problem associated with this category is that the initial size of TID lists can be too large to fit into main memory, therefore such algorithms utilizes more complex techniques to compress the size of TID lists which might highly affect their performance.

In [2] a new binary-based approach for mining itemsets and association rules was proposed. The proposed approach uses a binary representation of the database transactions in order to simplify the process of identifying the frequent patterns as well as reduces the memory consumption. This has been done by representing the database transactions in binary way, and utilizing the usefulness of bitwise operations to process the data since they are executed faster by the processor compared to the other operations.

III. Methodology

Binary-Based approach as well as Eclat algorithm were implmented in Java language due to its supports of object orientation as well as its wealthy libraries of data structures available through its collections framework. The hardware used for the experiments is DELL PRECISION T1700 machine equipped with 4 cores running at 3.10 GHz speed and 8 GB of RAM memory. The operating system installed in the hardware is Windows 7 64 bit.

We have conducted the analysis of CPU and memory usage using a third party tool which is Java VisualVM of Oracle as being mentioned previously. Java VisualVM is a tool used to monitor both the CPU time as well as the memory usage for Java applications. Fig. 1 illustrates the main interface of Java VisualVM.

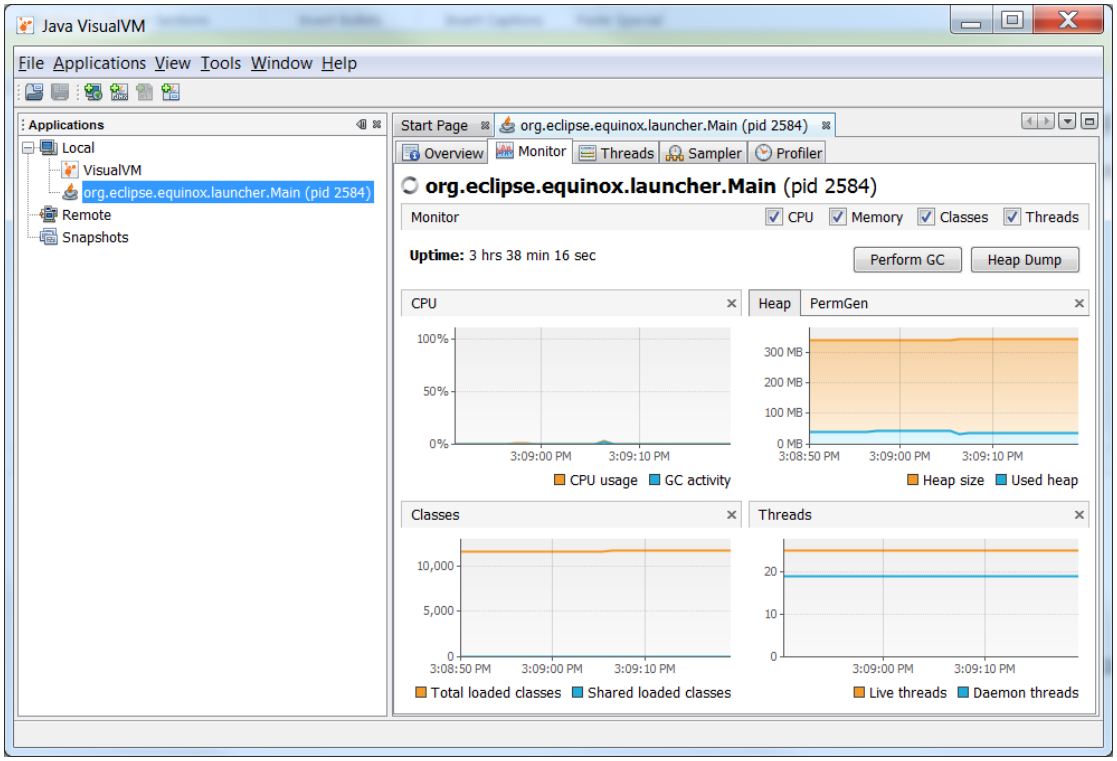


Fig. 1 Oracle Java™ VisualVM interface

Three different real and dense datasets have been used in this analysis. These datasets are Chess, Connect, and Pumsb datasets. Table 1 shows the characteristics of these datasets.

Table 1: Characteristics of Chess, Connect and Pumsb datasets

Dataset	#Items	#Transactions	Average of transaction	Max |T|	Type
Chess	76	3196	37	37	Dense
Connect	130	67557	43	43	Dense
Pumsb	7117	49046	74	74	Dense

IV. Results and Discussion

Fig. 2 as well as Fig. 3 respectively show the obtained results of CPU and memory usage recorded while monitoring the execution of Binary-Based approach and Eclat algorithm using Java VisualVM at support value = 0.9 on Chess dataset.

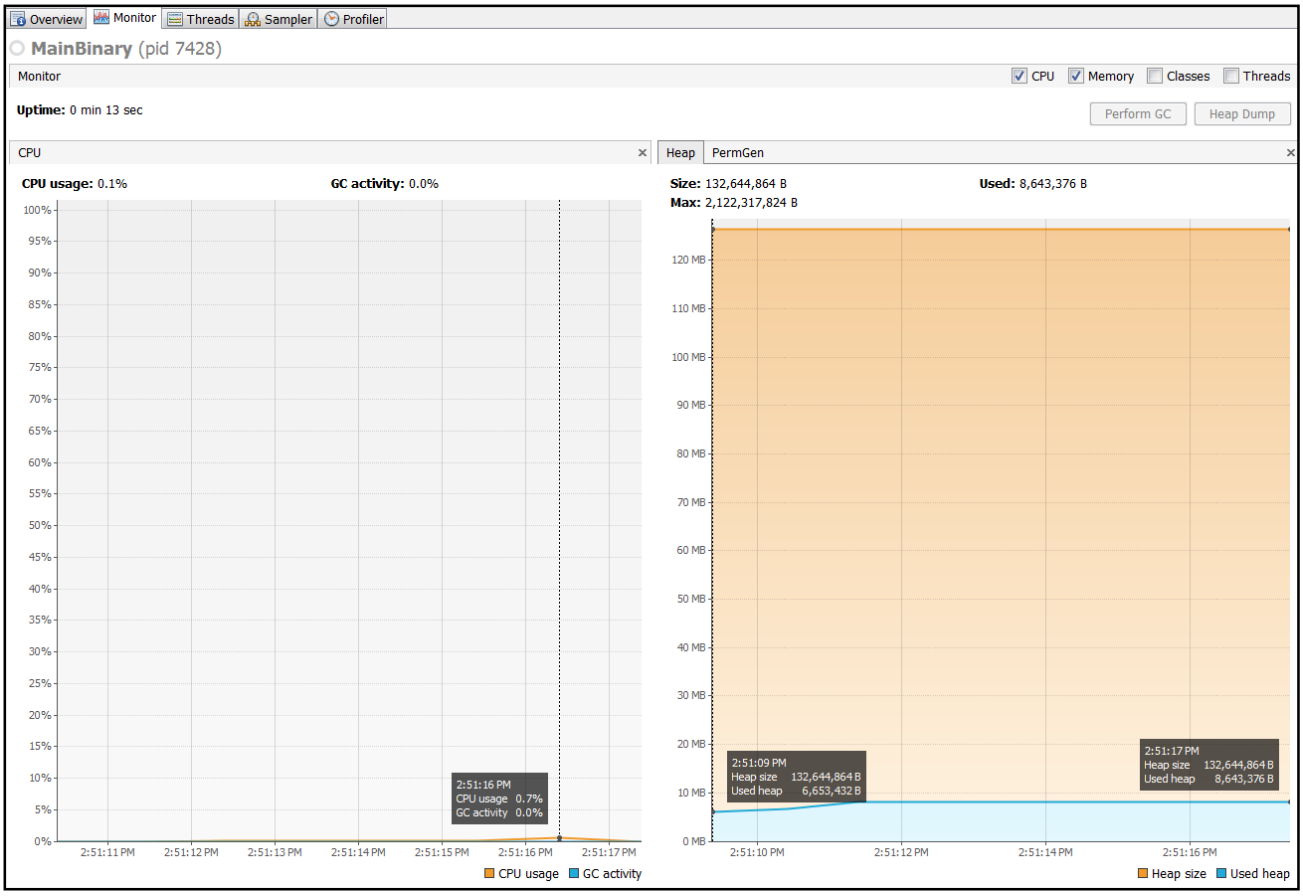


Fig. 2 CPU and Memory Usage for Binary-Based approach (Chess, Sup=0.9)

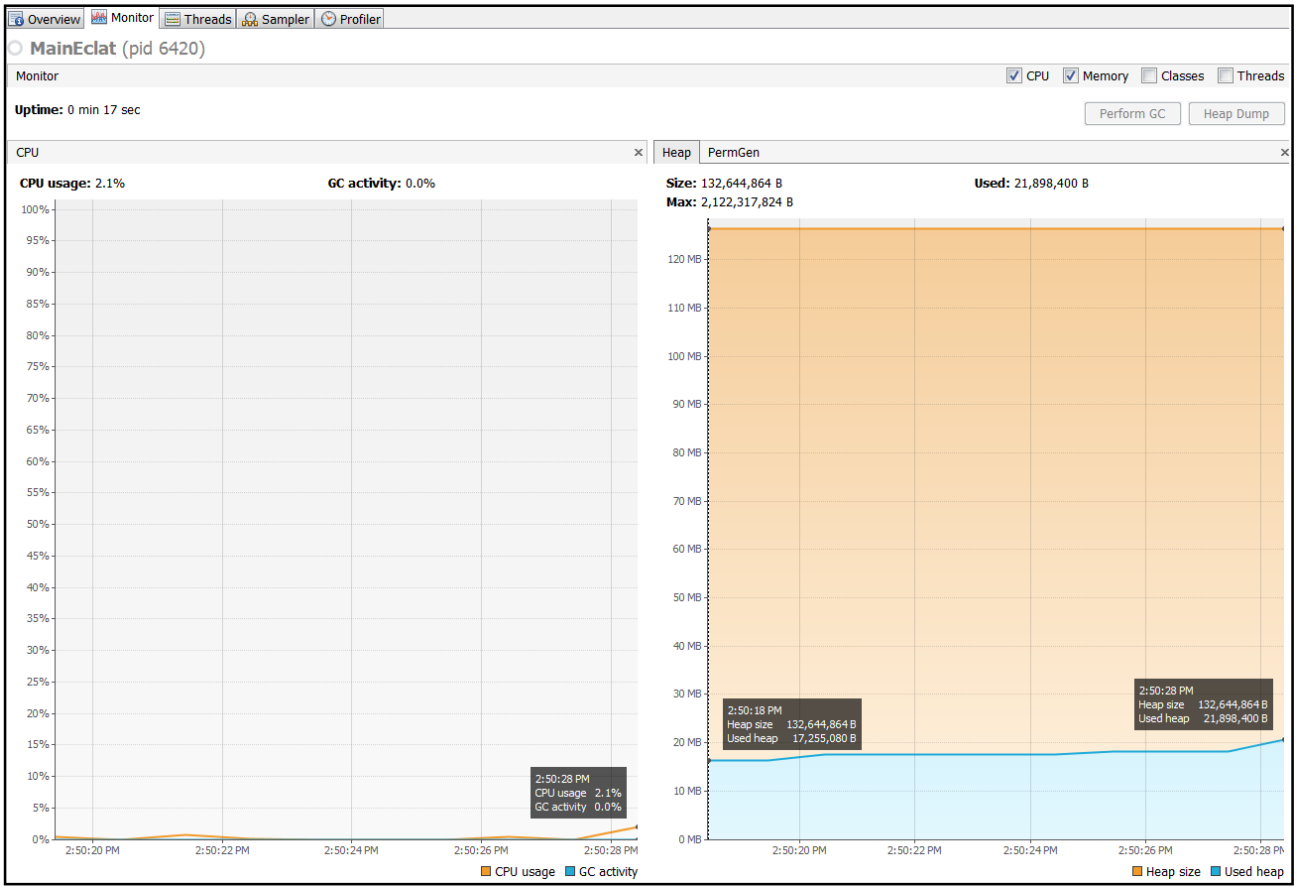


Fig. 3 CPU and Memory Usage for Eclat algorithm (Chess, Sup=0.9)

It can be clearly seen from both Fig. 2 and 3 Binary-Based approach reveals best performance than Eclat algorithm in both CPU and memory usage. For example, the maximum CPU usage recorded for Binary-Based approach was 0.7%, while Eclat recorded 2.1% maximum CPU usage. With regard to memory usage results, Binary-Based started with initial used heap size 6.7 MB, while Eclat started with 17.3. The final used heap size recorded by Binary-Based was 8.7 MB, while in Elcat the final recorded size was 22.0 MB. Table 2 summarizes both CPU and memory usage results.

Table 2: Summary of CPU and Memory usage (Chess, Sup=0.9)

Algorithm	CPU Usage %	Memory Usage (MB)			
		Heap Size		Used Heap Size	
		Initial	Final	Initial	Final
Binary-Based	0.7	132.6	132.6	6.7	8.7
Eclat	2.1	132.6	132.6	17.3	22.0

Fig. 4 as well as Fig. 5 respectively show the obtained results of CPU and memory usage recorded while monitoring the execution of Binary-Based approach and Eclat algorithm using Java VisualVM at support value = 0.5.

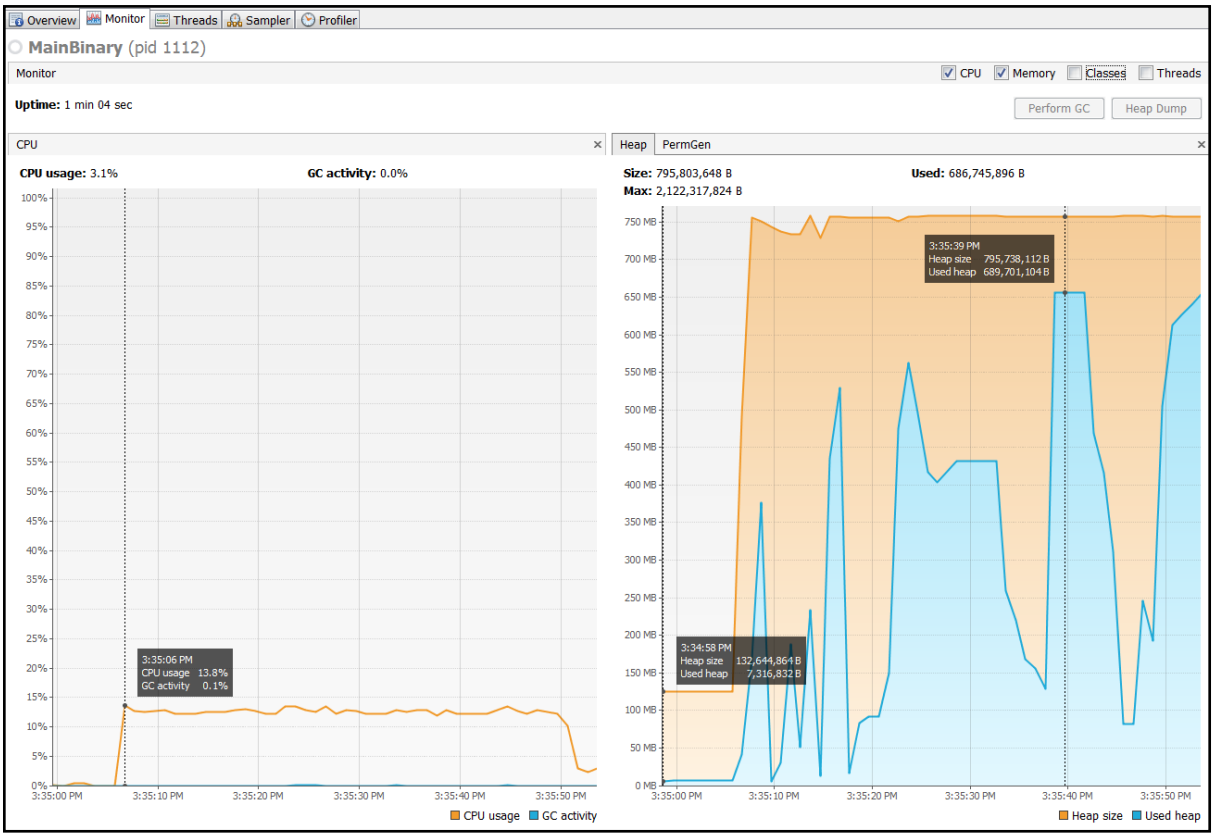
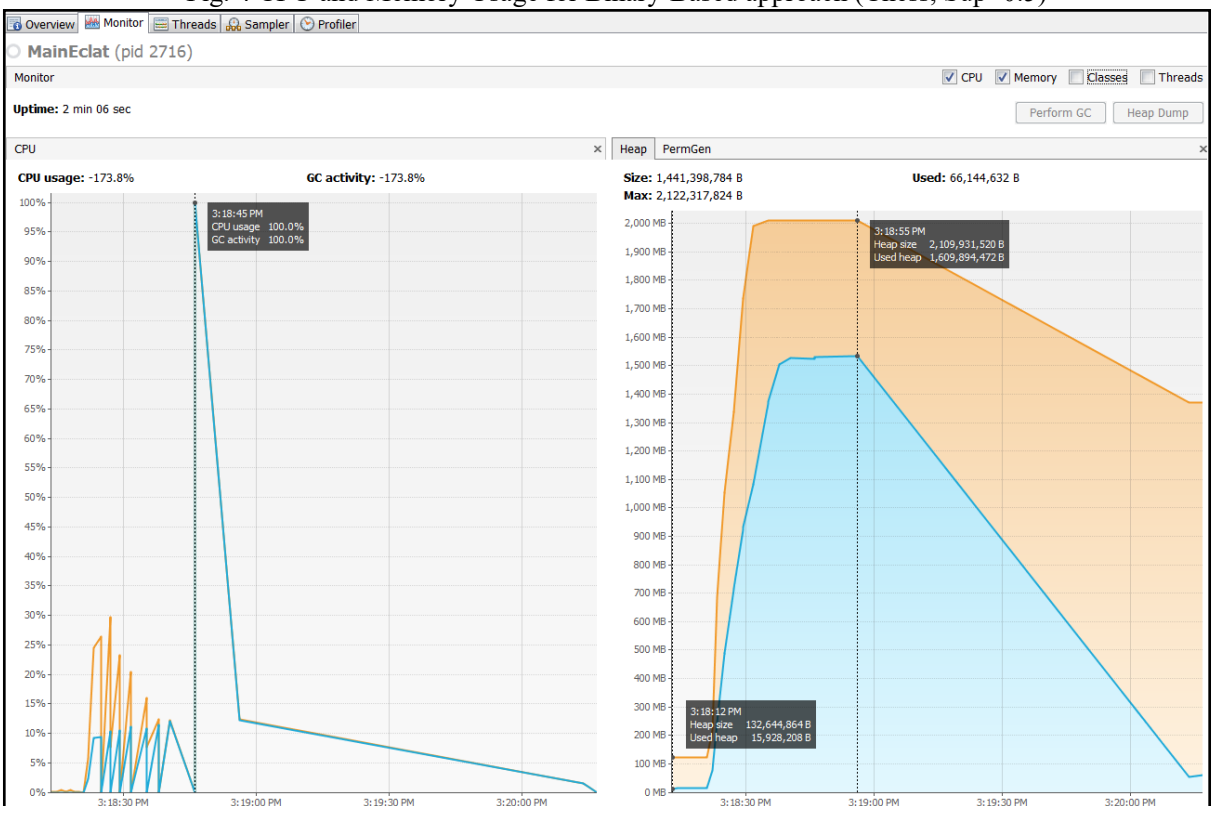


Fig. 4 CPU and Memory Usage for Binary-Based approach (Chess, Sup=0.5)



Again, as it is clearly seen from both Fig. 4 and Fig.5, Binary-Based approach continued to reveal best performance than Eclat algorithm in both CPU and memory usage. For example, the maximum CPU usage recorded for Binary-Based approach was 13.8%, while Eclat recorded 100.00% maximum CPU usage and lastly stopped working reporting an out of memory error. With regard to memory usage results, Binary-Based started with initial used heap size 7.3 MB, while Eclat started with 15.9 MB. The final used heap size recorded by Binary-Based was 689.7 MB, while in Elcat the final recorded size was 1610.0 MB. Table 3 summarizes both CPU and memory usage results.

Table 3: Summary of CPU and Memory usage (Chess, Sup=0.5)

Algorithm	CPU Usage %	Memory Usage			
		Heap Size		Used Heap Size	
		Initial	Final	Initial	Final
Binary-Based	13.8	132.6	795.7	7.3	689.7
Eclat	100.0 %	132.6	2110.0	15.9	1610.0

Due to space limitations, we show in Table 4 and Table 5 respectively the CPU profiler results for Connect and Pumsb datasets using *Sup* value equal to 0.9.

Table 4: Summary of CPU and Memory usage (Connect, Sup=0.9)

Algorithm	CPU Usage %	Memory Usage			
		Heap Size		Used Heap Size	
		Initial	Final	Initial	Final
Binary-Based	14.0	132.6	795.6	7.3	672.1
Eclat	47.7	132.6	2122.3	12.0	1641.3

Table 5: Summary of CPU and Memory usage (Pumsb, Sup=0.9)

Algorithm	CPU Usage	Memory Usage			
		Heap Size		Used Heap Size	
		Initial	Final	Initial	Final
Binary-Based	16.7	132.6	261.0	6.7	90.5
Eclat	36.0	416.2	2122.3	159.7	1650.2

The reason behind using *Sup* value of 0.9 is that Eclat failed to execute at this value and throws an out of memory exceptions. This means that for less *Sup* values Eclat would definitely ran out of memory since less values of *Sup* produces more output in-terms of itemsets and association rules.

V. Conclusion

In this paper, we have conducted a performance analysis of two main association rules mining algorithms: Binary-Based algorithm against Eclat algorithm. The performance has been conducted using Java VisualVM of Oracle to monitor the execution time and memory usage of each algorithm. The Chess dataset was used with different support values. The results showed that Binary-Based algorithm outperforms Eclat in-terms of both execution time and memory usage.

VI. References

[1] S. J. Abdulkadir and S.-P. Yong, "Scaled UKF–NARX hybrid model for multi-step-ahead forecasting of chaotic time series data," *Soft Computing,* vol. 19, pp. 3479-3496, 2015.

[2] S. O. Fageeri, R. Ahmad, and B. B. Baharudin, "BBT: An Efficient Association Rules Mining Algorithm using Binary-Based Technique," *International Journal of Advancements in Computing Technology,* vol. 6, p. 14, 2014.

[3] Z. A. Zailani Abdullah, T. H. Tutut Herawan, A. N. A. Noraziah, and M. M. D. Mustafa Mat Deris, "Fast Determination of Items Support Technique from Enhanced Tree Data Structure," *International Journal of Software Engineering and Its Applications,* vol. 8, pp. 21-32, 2014.

[4] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *ACM SIGMOD Record*, 2000, pp. 1-12.

[5] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, "New Algorithms for Fast Discovery of Association Rules," in *KDD*, 1997, pp. 283-286.

[6] M. J. Zaki, "Scalable algorithms for association mining," *Knowledge and Data Engineering, IEEE Transactions on,* vol. 12, pp. 372-390, 2000.