Analysis of I/O Interrupt Request From The Trace Data For Measuring System Performance Using Clustering Algorithm

Bidisha Mahanta, Mirzanur Rahman

Department of Information Technology, Gauhati University, Guwahati, India

Abstract—This project is attempted to evaluate the effectiveness of elementary sequence mining techniques for characterizing I/O trace data. Elementary means the preliminary sequence mining. We have taken Interrupt vector table (IVT) as trace information for the application, and apply K-Means clustering algorithms to correlate particular trace sequences with phases of application execution. By using Interrupt vector table, a vector space model, text data can be treated as sparse numerical data vectors. It is a contemporary challenge to efficiency of system process and cluster very large document collections. In this paper we present a time and memory efficient technique for the entire clustering process, including the creation of the vector space model. We show that our approach can bring out correlations between I/O applications and phases of application execution, and also it can take hold assure for performance monitoring.

Keywords— IVT, interrupt request, clustering algorithm, q-grams.

I. INTRODUCTION

A designer always cannot give the exact analysis of the performance and the characteristics of the applications in which he is working. Since the same application may run for different resources and also different memory systems. So this become the responsibility of the designers to discover the all details of all the application and the runtime and also optimize the behaviours. For this work purpose we have to find out in which particular time an application has executed which are currently running on the system. Based on this information we can lead to on monitoring and improve the performance of

The applications which are running dynamically on the system. Now a days, a system contains so many application since the advent of grid computing, large systems have thousands of applications are running in a system at a time. Monitoring the behaviours and runtime of all applications is not so easy. So we collect some samples which have helped us to find performance and behaviours of the applications. In a particular time, for the accuracy of the results consider some amount of applications. This amount of data achieve the accuracy .This data are depends on the variable based on applications.

The performance of the applications are varies by the execution phases and some application are varies by the behaviour differently of the different machine on the respective of the machine. First however, we need to understand what the different execution phases of applications are, and we need to learn how to identify them based on observable information.

In this paper, our main concern is in obtaining the runtime of any respective application and ameliorates the performance by employing the clustering and Q-gram on the IVT table.

Our main motivation is that we want to cluster the applications that are running on the system taking a time constant. Thus our main emphasis is on high speed and scalability with modest main memory consumption. In clustering, it first reads the whole vector table from the system and analysis it. The IVT file shell have to be generated by an external programme and refreshed periodically.

We organized this paper as follows. In Section II we foreground the some prior works of this paper which we have gone through during the project work duration. Section III we depict the Characterization of data, where we describes how we tested our raw file and how it behaves. In Section 4 we describe how we have adapted existing algorithms for approximated results for data mining work with our data. Section 5 presents the results and discussions. Section 6 presents the conclusion and the possible future works of this paper.

II. LITERATURE OVERVIEW

There are several methods for this topic had been used in previous research.

Lu and Reed [4] have investigated low-overhead alternatives to event tracing called application signatures. They use curve fitting to characterize individual performance metrics, and use these to compare applications across platforms. This technique is comparable to our sequence-mining approach, but our research is focused more on leveraging powerful existing data-mining techniques on available data than on low overhead.

Hao Wang [3] has run tests on the same data, but his approach has did not consider the sequential nature of the data. Instead, he applied various data mining techniques to individual packets of data, without taking into account their order. Computer programs are inherently sequential, and their actions are difficult to characterize without examining their order. Streaming I/O references made by computer programs will also be strongly sequential, and mining for frequently occurring sequences of data will be more effective for characterization than analysis at the reference level.

Christopher LaRosa, LiXiong, Ken Mandelberg[6] has done on Frequent Pattern Mining for Kernel Trace Data. The introduction of low-impact kernel-level tracing tools allows for comprehensive and transparent reporting of process and operating system activity. An operating system trace log provides detailed, explicit information about which processes use which system resources at what time. This time series data contains underlying knowledge, such as common execution patterns. This information can assist in many systems-related tasks: application debugging, security enforcement, performance optimization, operating system debugging, and dynamic reconfiguration. However, while kernel trace collection tools have advanced and matured, there remains a lack of trace analysis tools for extracting useful knowledge from raw trace logs.

N. Nakka, A. Choudhary, W. K. Liao[7] - In this paper, they present a tool to extract I/O traces from very large applications running at full scale during their production runs. They analyze these traces to gain information about the application. They analyze the traces of three applications. The analysis showed that the I/O traces reveal much information about the application even without access to the source code. In particular, these I/O traces provide multiple indications towards the algorithmic nature of the application by observing the changes of data amount and I/O request distribution at the checkpoints.

S. Parthasarathy, M. J. Zakiy, M. Ogihara, S. Dwarkadas works on Incremental and Interactive Sequence Mining [8]. In this paper, we propose novel techniques for maintaining sequences in the presence of a) database updates, and b) user interaction (e.g. modifying mining parameters). This is a very challenging task, since such updates can invalidate existing sequences or introduce new ones. In both the above scenarios, we avoid re-executing the algorithm on the entire dataset, thereby reducing execution time. Experimental results confirm that our approach results in substantial performance gains.

Much work has been done on sequential pattern matching (string matching) and on cluster mining in the past. Sequence mining and approximate string matching have had important applications to biological data, and we apply these techniques to performance data. In particular, we use the K-Mean clustering algorithm [2] to measure of the similarity of sequences.

III. CHARACTERIZATION OF DATA

In this paper, we examine Interrupt Vector Table (IVT) trace file. Interrupt Vector File is a file contains all the interrupts of a system. Means it contains the record of each I/O event that occurred during the system is running. IVT file is benefited us so well because it has different I/O events records by which we can find the time it has occurred and also the resources that are using or relocate or kind of information's.

A. INTERRUPT VECTOR TABLE

This is of great gain and involvement to system programmers and administrators to adopt great apprehension of usage modes on large scale machines. The tracing [18] utility traces only I/O calls of the application interacting with the file system. File system interactions of the tracing utility itself are not traced. On detection of internal error conditions the tracing halts, and allows the application to continue without interruption if possible. Each process in the application job generates one file containing the I/O traces encoded in an efficient, platform independent, binary format. A dictionary of the underlying file system where the trace data is written. Describing the binary file format must be generated using a provided external utility on the host platform. The resulting traces can be decoded into readable format by using the dictionary and the provided binary decoder on any machine.

In pthis paper, we consider interrupt vector table, where represent I/O trace events, while others are actual I/O system calls made by the application. Each events represents different I/O events respectively such as read, write, open,

close etc. We consider only read, write operations in this paper.

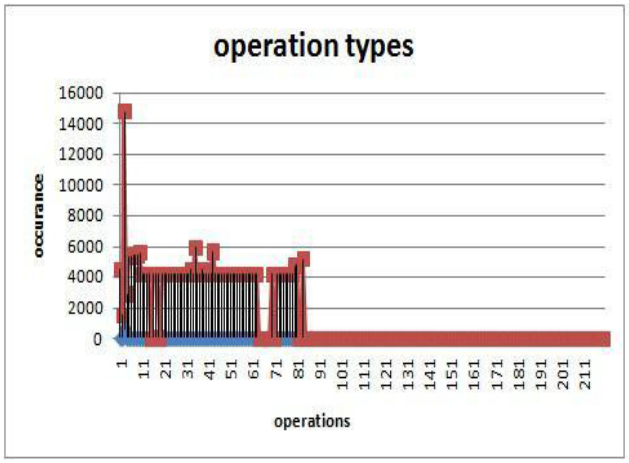

Figure3.1- I/O activity

From the IVT file we examined the durations of I/O operations which were most varied. The distributions of values of read and write are shown below.

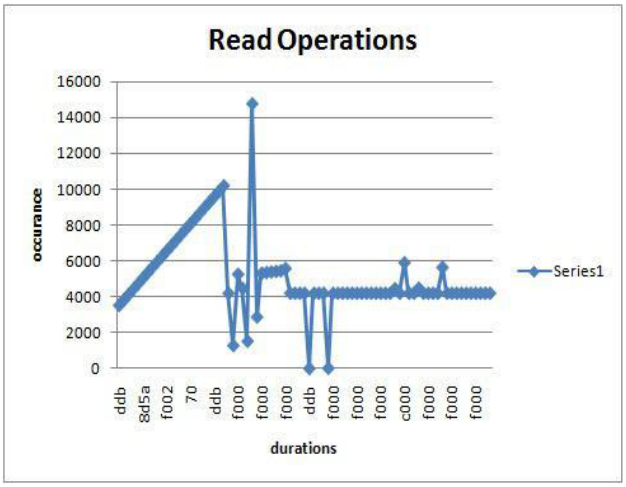

Figure3.2 - Durations of Read operation

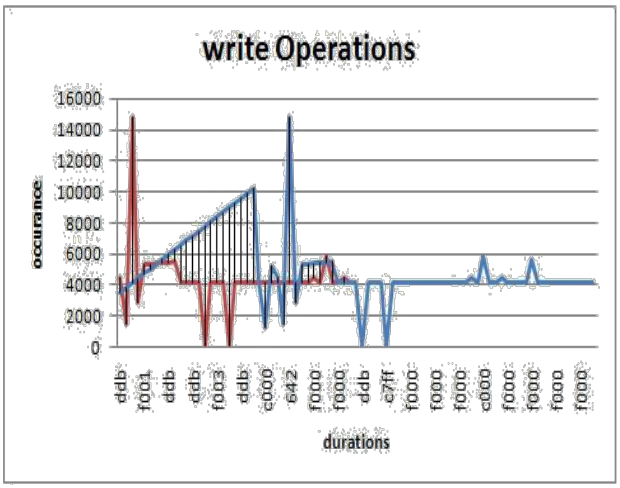

Figure3.3- Durations of Write operation

IV. ANALYSIS ON THE PERFORMED DATA

In this section, we depict how we have use algorithms for string matching and data mining on performed data up to a far extent.

A. Q-GRAM

String matching is delimit ate in terms of characters in sequence, or in the case of genetics, in terms of bases in DNA molecules.

Q-grams are [11], generally refers to taking a stream of inputs and breaking it into subsequence of a fixed length, q. In our case, we do just that: we split the data stream into q-grams of packets. As q-grams are sequential, comparing them instead of individual packets should give us more of a sense for the types of tasks the application is performing, rather than for the overall mix of operations it is using. To check the two q-grams are equal or not, there are some methods. We obviously [9, 3] cannot compare two packets for equality based on their timestamps, as this would result in none of them being equal. Instead, we define three different measures for packet equality based on the attributes available to us in the data, and we look for the q-grams that met these criteria.

We are followed two methods for considering two q-grams are equal [9] .They are as follows-

1) Operation-Type equality: If two packets have the same operation type, they are considered equal.

2) Exact duration equality: If two packets are of the same type, and they have the exact same duration, they are considered to be equal.

B. CLUSTERING

In data mining field, clustering is the process of finding similar group of objects from a large set of data. Clustering process involves in assessing the similar points. In this paper, we consider the points which are already passes through q-gram. A common measure [9] of dissimilarity of sequential strings is the edit distance. Edit distance is a measure of how many changes (e.g. adding, deleting and replacing of characters) would need to be made to make one string to another. Even two q-grams, we can again treat packets as characters, and using the equality measures.

Outlined above we can calculate the edit distance between two q-grams.

We apply K-Means clustering algorithm as a clustering algorithm. Because the only two operations we have defined on our q-grams are edit distance, we cannot use Density Clustering. K-Means requires the computation of a mean from the data points, but all we have is a measure of dissimilarity. Density clustering suffers from similar problems, and is only applicable to spatial data.

The k-means algorithm is an algorithm to cluster n objects based on attributes into k partitions, where k ¡ n.[5].It is similar to the expectation-maximization algorithm for mixtures of Gaussian s in that they both attempt to find the centres of natural clusters in the data. It assumes that the object attributes form a vector space. An algorithm for partitioning (or clustering) N data points into K disjoint subsets Sj containing data points so as to minimize the sum-of-squares criterion.

$$J = \sum_{j=1}^{K} \sum_{n \in S_j} |x_n - \mu_j|^2$$

Where, x_n is a vector representing the n^{th} data point and u_j is the geometric centroid of the data points in S_j. Simply speaking k-means clustering is an algorithm to classify or to group the objects based on attributes/features into K number of group. K is positive integer number. The grouping is done by minimizing the sum of squares of distances between data and the corresponding cluster centroid [5].

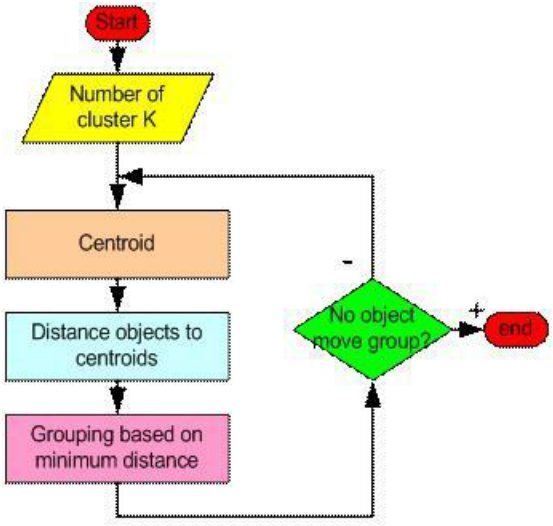

Figure 4.1: How the K-Mean Clustering algorithm works

It is worth [9] noting that the sequence clustering problem has been looked at in considerable detail by Wang, et al. They presented CLUSEQ, an algorithm for finding clusters in sequential data without a preset number of clusters to find, and without a preset length of sequences to cluster on (as our q-gram analysis inherently requires). Also, CLUSEQ measures similarity by statistical properties of sequences, rather than on a single distance metric. We believe that this algorithm would be very effective in analyzing I/O trace data, but it was not available at the time we performed our experiments. The algorithm is very sophisticated, and there was not time available to rewrite and then debug it. We believe that testing CLUSEQ holds promise for testing performance data in the future.

C. IMPLEMENTATION

We have implemented our project that means Q-gram and clustering algorithm using JAVA. We use the Interrupt vector table as our back end since we are working on I/O applications. Firstly, reads the whole Interrupt Vector File continuously since it has dynamically updated as soon as new interrupts occurred. Second step is to splits the files into a fixed length. After spiting the file into q-gram finally, we run the K-means algorithm over there.

V. EXPERIMENTAL RESULTS AND DISCUSSION

In this article, we explain our experimental results obtained from mentioned algorithms.

Our approach has evaluated in “Net Beans” platform under windows operating system. During the work since IVT has

dynamically updated as new application has ran so after some milliseconds 256 interrupts we got.

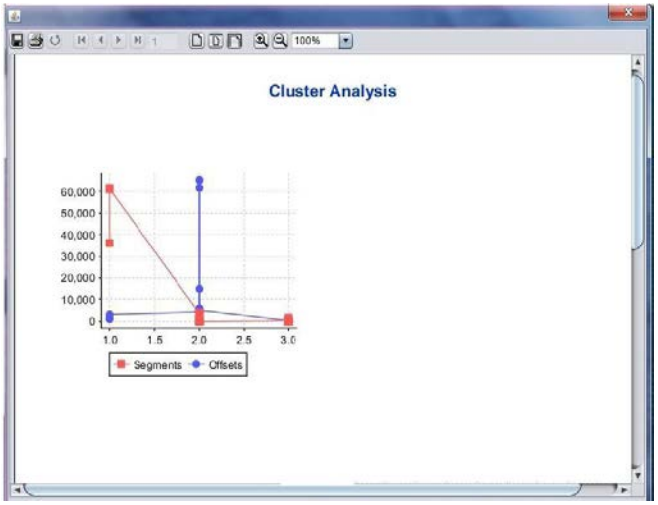

Figure 5.1: Result of clusters

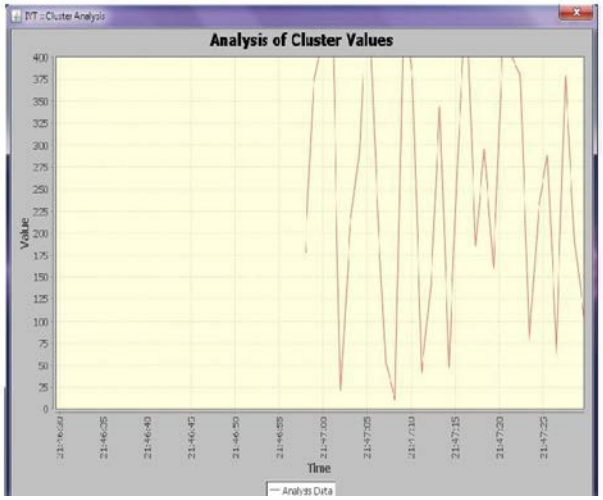

Figure 5.2: analysis of memory usages in 0-22 milliseconds

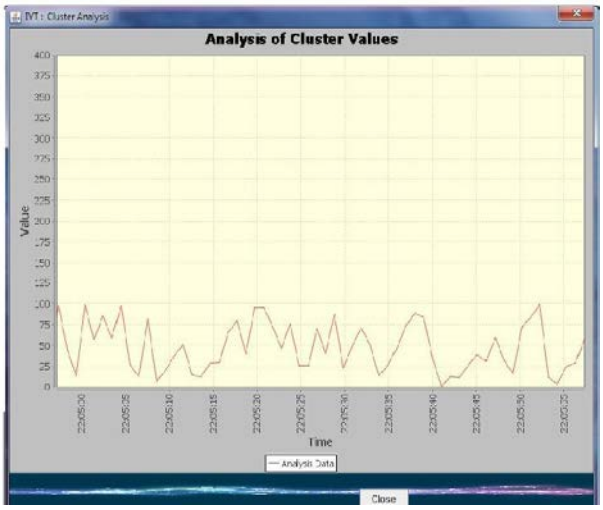

Figure5.3: analysis of memory usages in 22-50 milliseconds

Figure5.1 depicts the diagrammatic view of 3 clusters generated from the vector table. Figure5.2 and Figure5.3 display the analysis of memory usages in 0-22 milliseconds and analysis of memory usages in 22-50 milliseconds respectively.

VI. CONCLUSION AND FUTURE WORKS

In our project, we showed that we can bring out the correlations of I/O applications and different I/O data patterns by using data mining. We used Q-gram approach which gives us the small occurrence execution in the course of execution. Our clusters showed that on the application of high-level, qualitative comparison operation to these q-grams, that we can use clustering techniques to find larger groups of fairly similar q-grams. Cluster's are closely grouped in time basis, and could be used to build a monitoring adaptive performance. Q-grams monitored at runtime and identified by this classifier could be used to guide scheduling decisions and resource allocation for adaptive optimization. Since the whole project is not done yet so that its limitations cannot be figured out now. Till now documents are not gathered from different domains yet also. We have to overtake this and try to get the best result in the future.

This proposed solution will balance the load of the interrupts to a great extent, as it used the q-gram distance. We used the q-grams to compare for similarity, and this algorithm's running time of large values of q. We were unable to use this method for long sequences and large data sets. Furthermore, the K-Means clustering algorithm we used is not nearly as sophisticated as other sequence mining techniques in the literature. We believe that in the future, an algorithm like

”CLUSEQ” could be used which will give us far better results than this[9]. This algorithm uses a probabilistic approach to determine q-gram similarity, and does not require the number of clusters to be fixed from the outset. It might be able to eliminate the clusters we saw with very large standard deviations in time. Since we were successful to obtain the correlation between I/O application and time while using unsophisticated techniques upto a far extent, we believe that sequence mining is a promising approach for mining trace information and facilitating adaptation[9].

REFERENCES

[1] M.Ester,H.-P.Kriegel, J. Sander, and X.Xu. A Density-Based Algorithm for Discovering Clusters in large Spatial Databases with Noise. In Proceedings of the International Conference knowledge Discovery and Data Mining, 1996.

[2] L.Kauffman and P.J. Rousueeuw. Finding Groups in Data: an Introduction to Cluster Analysis. John Wiley and Sons, 1990.

[3] H.Wang. COMP 290-090 Final Project:Mining in Performance Data. 2004.

[4] C-D. Lu and D.A. Reed.Compact Application.

[5] Tutorial - Tutorial with introduction of Clustering Algorithms (k-means, fuzzy-c-means, hierarchical, mixture of gaussians) + some interactive demos (java applets).

[6] Frequent Pattern Mining for Kernel Trace Data.Christopher LaRosa, Li Xiong, Ken Mandelberg Department of Mathematics and Computer Science Emory University, Atlanta, GA 30322.

[7] Detailed Analysis of I/O traces for large scale applications. N. Nakka, A. Choudhary, W. K. Liao Electrical Engineering and Computer Science Northwestern University, Evanston, IL, USA.

[8] S. Parthasarathy, M. J. Zakiy, M. Ogihara, S. Dwarkadas works on Incremental and Interactive Sequence Mining. by Martin Fowler, Kent Beck (Contributor), John Brant (Contributor), William Opdyke, don Roberts .

[9] Todd Gomblin.”COMP 290 Data Mining Final Project Using sequence mining techniques for performance data”.

[10] R. Ng and J. Han. Efficient and Effective Clustering Methods for Spatial Data Mining. In Proceedings of VLDB, 1994.

[11] E. Ukkonen. Approximate String matching with q-grams and maximal matches. In Theoretical Computer Science, 1992.

[12] E. Ukkonen. Algorithms for approximate string matching. In Information and Control, 1985

[13] Inderjit S. Dhillon, James Fan and Yuqiang Guan.”Efficient Clustering Of Very Large Document Collections.

[14] D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W.Tukey. Scat-ter/gather: A cluster-based approach to browsing large document collec-tions. In ACM SIGIR, 1992.

[15] J. Heaps. Information Retrieval - Computational and Theoretical As-pects. Academic Press, 1978.

[16] Fang Chu.”Mining Techniques for Data Streams and Sequences ”.

[17] Wenzhi Zhou1, Hongyan Liu1, and Hong Cheng2.”Mining Closed Episodes from Event Sequences Efficiently”.

[18] N. Nakka, A. Choudhary, W. K. Liao,L. Ward, R. Klundt, M. I. Weston.”Detailed Analysis of I/O traces for large scale applications”