VOL. 10, NO. 8, MAY 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

IO WORKLOAD CHARACTERIZATION OF WINDOWS BASED ANALYSIS USING BIRCH ALGORITHM

N. Krishnamoorthi and G. K. Kamalam
Department of IT, Kongu Engineering College (Autonomous), Perundurai, India
E-Mail: krishnamoorthiinfo@gmail.com

ABSTRACT

The analyse of the windows based Input and Output workload is driven by various trace captured from running systems for various standard file system benchmarks. To find that many of the issues arise in SSD design appeared in the memory stack. To solving these difficult problems, there is considerable scope for design choice. The following issues are relevant to SSD performance are Data placement, Parallelism, Write Ordering, Workload Management. As SSDs increase in complexity and existing disk models will become incomplete for predicting performance. To specify the random write performance and disk lifetime will vary significantly due to the locality of disk write operations. To introduce a new model for characterizing this behaviour based on cleaning efficiency and suggest a new partition based algorithms for extending SSD lifetime. IO workload characterization has been a critical issue for operating system and storage community. One critical method is used for hot-data identification, in which a given logical block address (LBA) is verified to see if it contains frequently accessed data. Hot-data identification for flash-memory storage systems not only imposes great impacts on flash-memory garbage collection but also strongly affects the performance of flash-memory access and its life time. The advancement of NAND-based storage devices, which bear different physical characteristics from hard-disk-based storage devices, calls for an entirely new way of characterizing IO workloads. To revisit the issue of understanding and identifying the essential constituents of modern IO workloads from the viewpoint of the emerging NAND based storage device.

Keywords: birch algorithm, storage and operating systems, SSD, clustering, classification.

1. INTRODUCTION

"Data mining" is the process of extracting hidden patterns from large amounts of data. The goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use. It is generally used in a wide-ranging of summarizing practices such as marketing, Detection of fraud and surveillance. It is also popularly known as Knowledge Discovery in Databases (KDD), refers to the nontrivial, previously unknown, potentially and extraction of implicit useful information from data in databases. Several data mining techniques are classification, association and pattern recognition. A common approach for classifiers is to use decision trees to partition and segmentations. New segment records can be classified by traverse the tree from the root through nodes and branches, to a leaf represented classes. If the takes through a decision tree can then be represented as a rule. Clustering and classification are often used for purposes of segmenting the records, they have different achieve and objectives their segmentations through different ways. The proposed work will focus on challenges related to the integration of clustering and classification techniques. Classification has been identified as an important problem in the emerging field of data mining. Given our goal of classifying large data sets the focus is mainly on decision tree classifiers Decision tree classifiers are relatively fast as compared to other classification methods. A decision tree can be converted into simple and easy to understand classification rules.

2. CLUSTERING AND CLASSIFICATION

Two common data mining techniques for finding hidden patterns in data are clustering and classification analyses. It is mentioned in the same breath, they have different analytical approaches. There are a variety of algorithms used for clustering, but they all share the property of repeatedly assigning records to a cluster, calculating a measure (usually similarity, and/or distinctiveness) for identifying the throughput of the system. Cluster analysis has been the most popular statistical technique for automatically dividing the workload into workload classes. Block diagram of steps of evaluation and comparison. Apply classification Technique (Naïve Bayes classifier) using WEKA Tool. Figure-1. Shows the

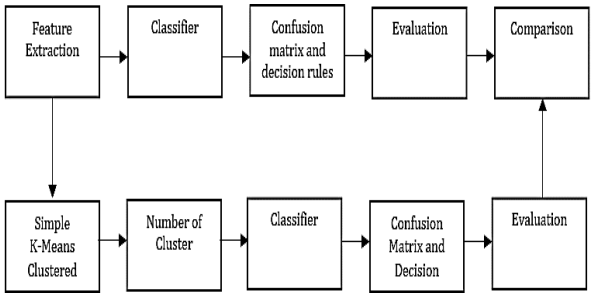

Figure-1. Block diagram.

Classification is two-step process, first, it build the classification model using training data. Every object of the dataset must be pre-classified i.e. its class label must be known, second the model generated in the preceding step is tested by assigning class labels to data objects in a

test dataset. The test data may be different from the training data. The accurate of this classification model is determined by comparing true class labels in the testing set with those assigned by the model. Apply the clustering method on the real data set using WEKA tool and now we are come up with a number of clusters. It also adds an attribute "cluster" to the data set. To apply the classification technique on the clustering assign data set. Then compare the results of classification and an integration of clustering and classification.

A. Clustering methods

The goal of clustering is to organize objects which are related to each other or have similar characteristics. Clustering group's similar objects (item) into same group.

B. Partitioning clustering

The partitioning method uses a set of M clusters and each object belongs to only one cluster. Every cluster can be represented by a cluster representative. The description of all the objects are contained in a cluster. This characterization will depend on the type of the object which is clustered. The real valued data are the arithmetic mean of the attribute vectors for all objects within a cluster provides an appropriate representative while alternative types of centroid may be required in other cases. The number of the clusters is larger than cluster representative, it can be further clustered which develop hierarchy within a dataset.

C. Hierarchal clustering

The algorithms require a pre-specified number of clusters as input and are non-deterministic. Hierarchical clustering outputs a hierarchical tree structure that is more informative than the unstructured set of clusters formed by flat clustering. This clustering is also does not requirement to specify the number of clusters in progressed. The clusters are created either by top-down or bottom up fashion by recursive partitioning. Hierarchical clustering are two types shown below Hierarchical Agglomerative methods and Hierarchical Divisive clustering methods.

D. Density based clustering

Density-based clustering algorithms try to find clusters based on density of data points in a realm. The basic idea for density-based clustering is that for each instance of a cluster the neighbourhood of a given radius (Eps) has to contain at least a minimum number of instances (MinPts). Density based clustering is based on probability distribution and points from one distribution are assumed to be part of one cluster. This approach is identifies the clusters and their parameters.

3. BACKGROUND

RELATED WORK

Bumjoon Seo *et al*. (2011) proposed the K-means clustering Algorithm. It is one of the simplest unsupervised learning algorithms that solve the well-known clustering problems. These concept is followed by simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori.

The main idea is to define k centroid, one for each clusters. The centroid need to place in a cunning way because of different location causes different result and the better choice is to place them as possible far away from each other. The result of this loop may notice that the k centroids change their location step by step until no more changes are done.

Mehrnoosh Sameki *et al*. (2008) proposed the IO Scheduling Algorithm to improve the efficiency of hard disk utilization, an Operating System (OS) reschedules IO requests to examine more read and write requests in one disk rotation. The pattern of the reordered IO requests is modified from its original sequential sequence to a random sequence. However, a random sequence of IO requests may impose large performance and endurance overhead to solid state devices.

LiPin Chang *et al*. (2007) described the Wear levelling algorithm. It is a process that is designed to extend the life of solid-state storage devices. Solid-state storage is performing up of microchips that store information in blocks. In each block can support a finite number of program or erase cycles before becomes unrelated. Wear levelling algorithm are organizes data, so that write or erase cycles are distributed evenly among all of the blocks in the device.

Sungjin Lee *et al.* (2007) proposed the hot-cold swapping Algorithm. The hot pages are likely to be kept in the hot partition; blocks that belong to the hot partition are intensively raised. On the other hand, blocks with cold data are rarely updated, thus erase counts of these blocks will be much smaller than those of other blocks. So we need to adopt a hot-cold swapping algorithm, which tries to balance erase cycles by periodically swapping the blocks containing hot data with blocks having cold data.

Jen-Wei Hsieh *et al*. (2005) proposed the Garbage Collection Algorithm to evaluate the performance of the multi-hash-function framework in terms of false hot-data identification. The performance of the pro-posed multi-hash-function framework might depend on the hash table size. A naive extension of the multi-hash-function framework was adopted for comparison, in which a hash table of a virtually unlimited size was adopted.

Jesung Kim *et al*. (2002) proposed the Single-Linkage Hierarchical Algorithm. The algorithm identifies the minimum spanning tree (MST) of a complete weighted graph with edge weights given by a distance function applied on the vertices, which is equivalent to solving the single-linkage hierarchical clustering problem.

The summary of the literature about IO Workload characterize the Simple K-Means algorithm yields the best silhouette value in the given vector matrix. To classifying the IO traces in the storage stack of the operating system. An SSD controller effectively exploits this classification model to characterize the incoming workloads.

4. PROPOSED METHOD

To find a set of representative patterns that can characterize IO workloads. By using Blktrace tool to collect IO trace in operating system over runtime. To use the computational analysis tool to find number of IO trace samples and patterns. To focus on challenges related to the integration of clustering and classification techniques. Classification has been identified as an important problem in the emerging field of data mining. Given our goal of classifying large IO Workload Trace data sets the focus is mainly on decision tree classifiers. Decision tree classifiers are relatively fast as compared to other classification methods. A decision tree can be converted into simple and easy to understand classification rules.

5. BUILDING A MODEL FOR IO WORKLOAD CHARACTERIZATION

To find a set of representative patterns that can characterize IO workloads. By using Blktrace tool to collect IO trace in operating system over runtime. To use the computational analysis tool to find number of IO trace samples and patterns. Tofocus on challenges related to the integration of clustering and classification techniques. Classification has been identified as an important problem in the emerging field of data mining. Given our goal of classifying large IO Workload Trace data sets the focus is mainly on decision tree classifiers. Decision tree classifiers are relatively fast as compared to other classification methods. A decision tree can be converted into simple and easy to understand classification rules.

A. IO trace data collection

The twenty features of IO trace events are listed in Table I Features F1 and F2 indicate the length of a trace in terms of pages and time. Feature F3 is the length of the longest segment. Features F4–F7 represent the total number of Input and Output requests, Break Points, Continued Points and non-random segments. The other features are the ratios (F8–F10) and basic statistics, such as arithmetic means (F11- F13), range (F14), and standard deviations (F16-F20) of other features. The features transform each IO trace sample into a vector of twenty dimensions for downstream analysis. It converts the input trace data into a matrix of 58,758 rows (recall that used 58, 758 traces in our experiments) and twenty columns and then the matrix normalized, so that every column has zero mean and unit variance.

The important feature can be determined by exploring a variety of (combinations of) features. However, due to the curse of dimensionality [1] (or lack of data separation in high-dimensional space), using too many features for data mining could make it difficult to separate IO traces in the original characteristic space. Need to select a set of characteristic for better clustering results.

Table-1. List of features for characterizing IO traces.

ID	ID future description	a	b
F1	Trace size in pages		
F2	Duration of trace (= sum of all intervals between IO requests)		○
F3	Length of the longest segment		
F4	Number of IO requests		
F5	Number of BPs (= segments)		
F6	Number of CPs		○
F7	Number of non-random segments	○	○
F8	Ratio of the number of pages in random segments to that of all pages		○
F9	Ratio of the number of CPs to that of BPs	○	○
F10	Ratio of the number of up-segments to that of all segments		
F11	Average segment length		
F12	Average interval between IO requests		
F13	Average access length of IO requests		
F14	Range of intervals between IO requests		
F15	STD dev. of access lengths		
F16	STD dev. of the starting LBAs of IO requests	○	
F17	STD dev. of the page indices of BPs		
F18	STD dev. of the page indices of CPs		
F19	STD dev. of the starting page indices of random segments		
F20	STD dev. of intervals between IO requests		

a - The features selected by backward elimination
b - The features used by tree-based model

To build a model for classifying IO traces first to see what type of IO traces appear in data storage systems. The particular number of all available patterns may be difficult to determine but expect that to find a small number of representative patterns. To summarize the patterns of IOtraces in an automated fashion and using Weka tool that patterns therefrom. The clustering is a perfect fit to using unsupervised learning enable us to find novel and previously unnoticed patterns besides known to expected ones. Classification is useful when the set of possible classes is already determined but as supervised learning techniques it cannot discover novel patterns unlike clustering.

The clustering analysis of IO traces are determined the set of features and the specific clustering algorithm used. Since the set of best features differs by which clustering algorithm is used. To determine each dimensions are the same time for the best result. Optimization all of these simultaneously and it will be difficult due to the huge size of search space. To determine

the set of features by a feature selection procedure with the clustering algorithm are fixed and respect to the determined set of features are to find the best clustering algorithm.

B. IO trace data description

By default, blkparse expects to run in a post-processing mode, where the trace events have been saved by a previous run of blktrace and blkparse. blktrace and blkparse is combining event streams and dumping formatted data. Blkparse are run in a live manner concurrently with combining it with the live option for blktrace and blktrace by specifying -i - to blkparse. An example is:

% blktrace -d /dev/sda -o - | blkparse -i - (1)

The number of blkparse batches event are set via the -b option, the default is to handle events in batches of 512. The event traces in blktrace is saved with different output names (via the -o option to blktrace), and the same input name to be specified via the -i option. The resultant data can be controlled via the -f or –F options.

C. Decision tree classification

Decision-tree-based classifier for classifying IO traces in runtime is constructed using Hierarchical classification method. This section explains how to train the classifier and validate it with real traces. Also, Compare the proposed tree-based classifier with other existing state of the art classifiers in terms of classification accuracy and run time. The purpose of the length of trace samples is also considered for classification performance.

D. Birch clustering algorithm

Phase-1: Scan all data and build an initial in-memory CF tree.

Phase-2: Condense into desirable length by building a smaller CF tree.

Phase-3: Global clustering.

Phase-4: Cluster refining – this is optional, and requires more passes over the data to refine the results.

Given a cluster of instances, we define:

$$\text{Centroid}: \quad \vec{x}_0 = \frac{\sum_{i=1}^{N} \vec{x}_i}{N}$$

$$\text{Radius}: \quad R = \left(\frac{\sum_{i=1}^{N} (\vec{x}_i - \vec{x}_0)^2}{N}\right)^{\frac{1}{2}}$$

$$\text{Diameter}: \quad D = \left(\frac{\sum_{i=1}^{N}\sum_{j=1}^{N} (\vec{x}_i - \vec{x}_j)^2}{N(N-1)}\right)^{\frac{1}{2}} \qquad (2)$$

E. Clustering features (CF)

The Birch algorithm builds a dendrogram called clustering feature tree (CF tree) while scanning the data set. Each entry in the CF tree represents a cluster of objects and is characterized by a 3-tuple: (N, LS, SS), where N is the number of objects in the cluster and LS, SS are defined in the following.

6. PERFORMANCE COMPARISON AND RESULT ANALYSIS

The performance of different classifiers approaches are compared in terms of their classification error and running Time. Figure-1 compares three classification methods Birch Clustering, Simple K-means, Logistic Regression in terms of their accuracies. The accuracy of the logistic regression-based classifier is the highest, and that of the K-Means Clustering is the lowest.

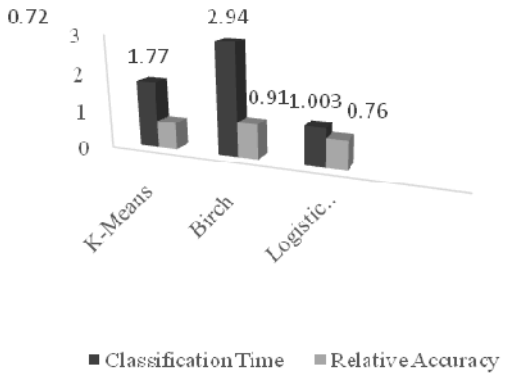

Figure-2. Performance comparisons of algorithms.

The performance of the Birch Algorithm, one of the most popular classifiers, is similar to that of the best one. The graph shows the running time required for classifying 58,758 IO traces by the methods used for comparison. As expected, the proposed method requires the least amount of time to complete the classification task. The advantage in running time obviously comes from the simplicity of the tree-based model.

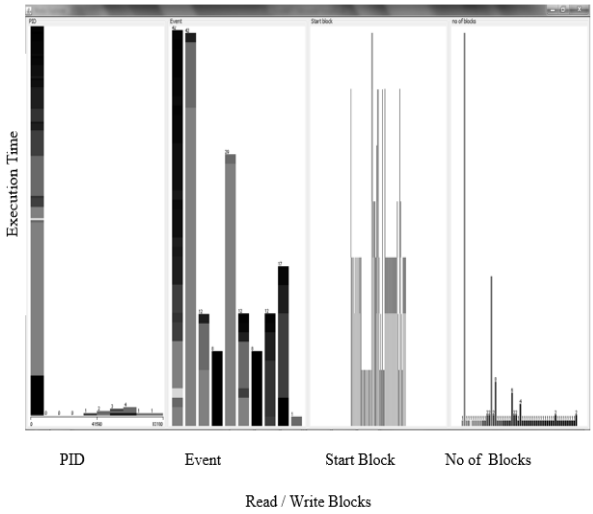

Figure-3. Classification of block IO workload.

When executed in an embedded CPU, the proposed classifier would further widen the performance gap among the ratio of the number of continued points and the number of break points, and the standard deviation of the starting LBAs of IO requests. For IO classes, nine essential workload classes and three random, three sequential and three mixture classes are considered. Finally a classification model of IO traces and is developed a pattern classifier is constructed for classifying IO traces in runtime. Figure-3 shows the median silhouette values each of the clustering methods used reports. The features of above graph for PID (Process Id), Event, Start Block and No of Blocks are the selecting the four features determined in the previous backward elimination step. For each clustering method, we try multiple combination of parameters at stated above, although Figure-3 shows only some of the results of the obtained (including best one) due to limited space. According to Figure-3, the partition-based algorithms (i.e., for PID, Event, Start Block and No of Blocks) give the best result.

The results of various data mining algorithms are shown in Figure-2 shows that the Simple K-mean compare Birch clustering algorithm provides better results than hierarchical and DBSCAN algorithm. Simple k-mean is a faster and safer algorithm than the other data mining algorithms are used.

SUMMARY AND CONCLUSIONS

The novel IO workload characterization and classification are done based on the data mining approach. In this method for IO workload clustering and Birch Clustering yields the best silhouette value despite its simplicity. The work needs to be done in the general field of I/O cache performance. Each file type has a distinct size distribution and reuse ratio. File type and size information can direct I/O requests to different parts of an attribute cache. The attribute cache is allocating a wide amount of space to capturing these requests.

The portion of the cache should have small blocks to effectively capture the temporal nature. Small executable and data files tend to have temporal access patterns, whereas large executable and data files tend to have sequential access patterns. A cache should direct large files to a sequential sub cache, and smaller files to a temporal sub cache. The sequential sub cache has large blocks to capture sequential locality while the temporal sub cache has small blocks to minimize unused space. I/O caches should make use of file types to improve their efficiency. The future work is to develop FTL techniques that can effectively utilize the classification model proposed in this work.

ACKNOWLEDGMENT

First of all I would like to extend my sincere gratitude to my supervisor Dr. G.K. Kamalam for providing me the opportunities of taking the part in Master of Computer and Communication Program and her ideas and suggestions, which have been very helpful in the project. I am so deeply grateful for her help professionalism and valuable guidance throughout this project.

REFERENCES

[1] Bumjoon Seo, Jaehyuk Cha, Jongmoo Choi, Sooyong Kang, Youjip Won, Sungroh Yoon. 2013. ”IO Workload Characterization Revisited: A Data-Mining Approach”, in IEEE Transactions on Computers, accepted for Publication, Vol. 6, No. 1, pp. 1541-1551.

[2] Jen-Wei Hsieh, Li-Pin Chang, Tei-Wei Kuo. 2006. ”Efficient On-line Identification of Hot Data for Flash-Memory Management”, IEEE Transactions on Computers, Vol. 6, No.1, pp. 581-590.

[3] Hunki Kwon, Eunsam Kim, Jongmoo Choi, Donghee Lee, Sam H.Noh. 2010. “Janus-FTL: Finding the Optimal Point on the Spectrum Between Page and Block Mapping Schemes “Proc. International Conference on Embedded software, Vol. 4, pp. 169-178.

[4] Hsieh J.W, Chang L.P., and Kuo T.W. 2005. “Efficient on-line identification of hot data for flash-memory management”, International Symposium in Applied computing, Vol. 1, pp. 838–842.

[5] Jesung Kim, Jong Min Kim, Sam H. Noh, Sang Lyul Min and Yookun Cho. 2002. “A Space-Efficient Flash Translation Layer for Compact Flash Systems”, IEEE Transactions on Consumer Electronics, Vol. 48, No. 2, pp. 366-375.

[6] Sungjin Lee, Shin, Kim Y.-J and Kim J. 2008. “LAST: Locality-Aware Sector Translation for NAND Flash Memory-Based Storage Systems”, International symposium in Operating Systems Design and Implementation, 2008, Vol. 42, pp. 36–42.

[7] LiPin Chang. 2007. “On Efficient Wear Leveling for Large Scale Flash Memory Storage Systems”, Proc. International Conference in Applied Computing, Vol. 6, pp. 1126-1130.

[8] Mehrnoosh Sameki, Amirali Shambayati. 2008. “An IO Scheduling Algorithm to Improve Performance of Flash-Based Solid State Disks”, International Conference in Hossein Asadi Sharif University of Technology, Vol. 4, pp. 833-762.

[9] Priya Kakkar, Anshu Parashar. 2014. “Comparison of Different Clustering Algorithms using WEKA Tool”, International Journal of Engineering and Science. Vol. 1, No. 2, pp. 633-662.