Journal of Theoretical and Applied Information Technology
31st January 2017. Vol.95. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AN ADVANCED COST AND SEARCH SPACE REDUCTION AND OPTIMIZED PATTERN GENERATION USING MAPREDUCE

[1]HUSSAIN SYED, [2]DR.PRASANTH YALLA

[1]PhD Scholar, Department of computer science and engineering, K L University
[2]Professor, Department of computer science and engineering, K L University
E-mail: [1]syed.hussain4887@gmail.com, [2]prasanthyalla@kluniversity.in

ABSTRACT

Big-data takes us into new epoch of data, which commonly referred to be as a big data. It pose a challenges to the researchers with more velocity, more variety and large volumes and software engineers are working on variety of methods in avoiding cost of development process. Where the commonly used software's are not able to imprisonment, accomplish and process within the lapsed time. Furthermore, there is a need to discover new procedures for to process large volumes of data to optimization, datamining and knowledge discovery. This ambitions and motivations are drives the researchers to Big-data analytics and big-data mining. Over the earlier few centuries, different procedures have been proposed to use the MapReduce model-which decrease the space of search with distributed or parallel computing-for different big data mining and analytics tasks and these methods are used in providing a cost effective solution across various software development models using Bayesian Approaches. In this paper we propose an algorithm which reduces the search space based on user's perspective and MapReduce to mine valid frequent patterns, approximate patterns and rare patterns from high volumes of ambiguous data in a divide-and-conquer fashion and we evaluate the performance through Hadoop and this proposal proves that Bayesian approach will not only improves the strategies used in software development process and also cut down the cost of the project during the implementation. Our implementation results prove that proposed approach is more realistic across pervasive and distributed computing environments.

Keywords: Software development process, *Bayesian Approaches, Big-Data Analytics, MapReduce Model, Hadoop.*

1. INTRODUCTION

Over the past few years' technology becomes more advanced, it generates great volumes of valuable data from various real-life applications in modern organizations and society. Such as torrents of banking, financial, marketing, telecommunication, biological, medical, life science, and social data. Different mining and analytics are done past by the researchers which are web application processing, stock exchange analytics, marketing analytics and biological analytics. In web application processing requests from the customer are served by giving his required page with in elapsed time and also predict the customer navigation to give his required page which grouping similar people based on their interests or grouping similar constraints based on their properties (clustering). Stock exchange analytics are predict trading pattern to bought/ sold stocks which is comes under classification. Marketing analytics are used to identify the future directions of taking decisions.

Medical data also analyzed for future decisions. But all these techniques are up-to some limited volumes of data. Big-data takes us into new epoch of data, which commonly referred to be as a big data. It pose a challenges to the researchers with more velocity, more Variety and large volumes. Where the commonly used software's are not able to imprisonment, accomplish and process within the lapsed time. Furthermore, there is a need to discover new procedures for to process large volumes of data to optimization, datamining and knowledge discovery. This ambitions and motivations are drives the researchers to Big-data analytics and big-data mining. Over the earlier few centuries, different procedures have been proposed to use the MapReduce model-which decreases the space of search with distributed or parallel computing-for different big data mining and analytics tasks. Example tasks include clustering, outlier detection and structure mining.

Journal of Theoretical and Applied Information Technology
31st January 2017. Vol.95. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

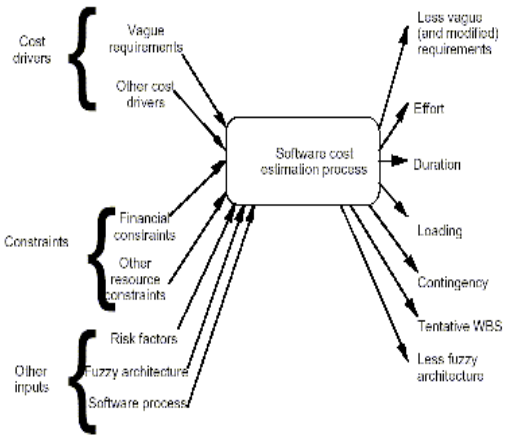

Figure 1: A Rough Architecture About Cost Management In Terms Of Module

These calculations finds fortifying data in the types of routinely happening information sets of produce things or occasions. Following the presentation of incessant example mining, various studies have been directed to mine successive examples from exact information (e.g., customary databases of market exchanges). With these conventional databases, clients certainly know whether a thing is available in (or is truant from) an exchange. Then again, information in some genuine applications are filled with vulnerability. It is in part because of inalienable estimation mistakes, testing and length of time blunders, system latencies, and purposeful obscuring of information to protect secrecy. Thusly, clients are normally dubious about the vicinity or nonappearance of things. As a solid sample, a meteorologist might suspect (yet can't promise) that extreme climate wonders will create amid an electrical storm. The vulnerability of such suspicions can be communicated as far as existential likelihood. The paper is composed as takes after area 1 talk about presentation, segment 2 examine related work, segment 3 talked about proposed work, segment 4 bargains results and assessments and segment 5 finishes up the paper.

1. Limitations and Assumptions:

This research is based on organizing a cost effective system across Hadoop Big data and Map Reduce. In providing an effective solution for better cost management system used in distributed system. In the previous research it is completely defined about cost control model across web based applications and Relational database management system. This is comparative study used in calculating the query processing time and cost over big data based systems.

2. RELATED WORK

Carson Kai-Sang Leung et al proposes "Lessening the Search Space for Big Data Mining for Interesting Patterns from Uncertain Data" As things in every exchange of these probabilistic databases of dubious information are generally connected with existential probabilities communicating the probability of these things to be available in the exchange, the quest space for mining from indeterminate information is much bigger than mining from exact information. This matter is exacerbated as we move into the period of Big information. Moreover, in some genuine applications, clients might be keen on just a small partition of this huge hunt space. To abstain from squandering heaps of time and space in registering every single successive example first and pruning uninteresting ones as a post-handling step, we proposed in this paper a tree-based calculation that (i) permits clients to express their enthusiasm for terms of compact hostile to monotone (SAM) limitations and (ii) utilizes MapReduce to mine unverifiable Big information for incessant examples that fulfill the client determined imperatives. Therefore, our calculation gives back all and just those examples that are intriguing to the clients. Additionally, in spite of the fact that we concentrated for the most part on taking care of SAM requirements, we likewise examined how our calculation handles limitations that are hostile to monotone (AM) yet not compact. As continuous work, we misuse how to handle limitations that are compact however not AM and also requirements that are neither concise nor AM. Trial results demonstrate the adequacy of our calculation in exploiting so as to lessen the pursuit space properties of requirements when mining obliged continuous examples from indeterminate Big information utilizing the MapReduce model.

As of late, Lin et al. proposed three Apriori-based calculations called SPC, FPC and DPC to mine continuous examples from exact information. Among them, SPC utilizes single-pass checking to discover incessant examples of cardinality k at the k-th pass (i.e., the k-th database examine) for $k \geq 1$. FPC utilizes settled passes joined tallying to discover all examples of cardinalities k, (k +1)... (k +m) in the same pass or database examine. From one viewpoint, this altered passes system settles the quantity of required goes from K (where K is the greatest cardinality of every single continuous example that can be mined from the exact information) to a client determined steady. Then again, because of consolidated checking, the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

quantity of produced competitors is higher than that of SPC. Interestingly, DPC utilizes dynamic-passes consolidated tallying, which considers the advantages of both SPC and FPC by taking the workloads of hubs when mining regular examples with MapReduce. Like these three calculations, our proposed calculation additionally utilizes MapReduce. Be that as it may, not at all like these three calculations (which mine continuous examples from exact information utilizing the Apriori-based methodology), our proposed calculation mines successive examples from questionable information utilizing a tree-based methodology. Note that the quest space for successive example digging for questionable information is much bigger than that for exact information because of the vicinity of the existential likelihood values.

Riondato et al proposed a parallel randomized calculation called PARMA for mining approximations to the top-k incessant examples and affiliation rules from exact information utilizing MapReduce. In spite of the fact that PARMA and our calculation both use MapReduce, one key distinction between the two calculations is that we plan to mine genuinely visit (rather than around continuous) designs. Another key contrast is that we mine from indeterminate information (rather than exact information). The third key contrast is that we center our calculation on finding those legitimate successive examples (i.e., those regular examples that fulfill the client indicated limitations) rather than all (unconstrained) continuous example

3. PROPOSED WORK

To reduce the search space in an uncertain collection of data that is commonly known as big data. Previously many algorithms are proposed by various researchers, but no one concentrated on user's perspective exactly. In our work we mainly focus on user's perspective and we divide the user's navigations into 3 major levels. Those are 1. Frequent patterns 2.Approximate patterns and 3.Rare patterns. Generally users wants search a particular thing to get but the data is very vast so searching the entire volume takes more time and requires more computational power. If the user daily checks or more number of users searches for the same pattern we make that as a frequent pattern. If the user does not know exactly what he needs in that case we use approximate pattern by taking that as an approximate pattern. And finally rare pattern it is rarely searched one so it is considered as a rare pattern.

Need of Work: This work is based on implementing an advanced cost model for Big data and Cloud based systems. The simulation result show that the proposed pattern based experiments will product efficient and accurate results and reduces cost of production and maintenance.

In order to find our required patterns in big data, we use the high level language to process the massive amount of data that is MapReduce. MapReduce mainly working with two key functionalities map and reduce.

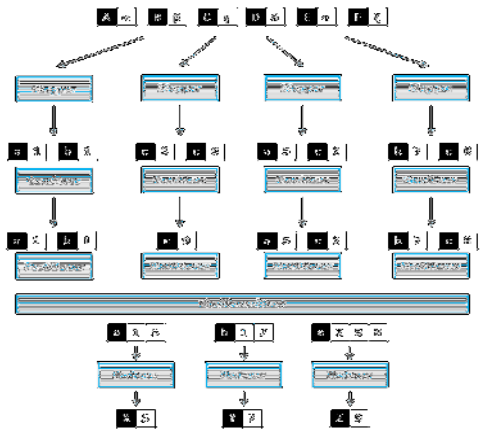

Figure 2: Framework For Mapreduce

The input data are scanned and portioned into several sub parts, and each is allocated to dissimilar processors. Each map function makes executed on one processor and produces pair of (key, value). The guide capacity takes this pair (key, esteem) and gives back a middle of the road aftereffect of (key, worth) sets. In this manner, these produced sets are rearranged and sorted. Exclusively the processor performs the lessen capacity on (i) a solitary key from this delegate result together with (ii) the rundown of all values that show up with this key in the moderate result. The decrease capacity "diminishes"— by consolidating, conglomerating, outlining, separating, or changing—the rundown of qualities connected with a given key (for all k keys) and returns (i) a rundown of k sets of keys and values, (ii) a rundown of k qualities, or just (iii) a solitary (accumulated or compressed) esteem. MapReduce applications can developed a reversed file and also the word checking of an article.

4. ALGORITHM

In this section, we propose our algorithm which uses MapReduce to mine valid frequent patterns,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

approximate patterns and rare patterns from high volumes of ambiguous data in a divide-and-conquer fashion (That is in tree-based pattern-growth fashion), The algorithm uses three sets of the "map" and "reduce" functions during the Big data mining process: (A) One set for mining frequent patterns (B) another set for mining approximate patterns (C) and another set for mining rare patterns.

This algorithm mainly focusing on dividing the ambiguous data into several patrons. And each pattern can be executed with one processor.

If the set of patterns that satisfies the user constraint that is taken based on the threshold value. If the constraint satisfies set of patterns taken then apply the map reduce to that patterns and taken them as a tree structure for easy accessing.

If the set of patterns satisfying the user constraint taken and then we divide the pattern type. Based on the threshold value that is number of patterns are more than 3, then we consider as a frequent pattern less than 3 considered as a rare pattern. If nothing pattern found but if we found nearest one then we consider this as an approximate pattern.

Example if some cities in India having temperature greater than 35 degrees. Then the constraint is k= max (temperature>=35) which express the user interest and here we get set of patterns and we classify the patterns here.

Here we apply the MapReduce algorithm to the frequent and rare patterns. But the approximate patterns are again consider with one more constraint after that only we can apply the MapReduce.

Algorithm to reduce search space (user constraint, frequent pattern, rare pattern and approximate pattern)

User constraint= U_c; Frequent pattern= F_p; Rare pattern= R_p; Approximate pattern= A_p;

Step1: If (U_c € F_p)
{
Identify the valid patterns from set of frequent patterns that is those are satisfies the user constraint and also satisfies the threshold value constraint, those are taken as valid patterns V_p.
Apply map function to the list of valid patterns;
Apply reduce function after applying the map function; }

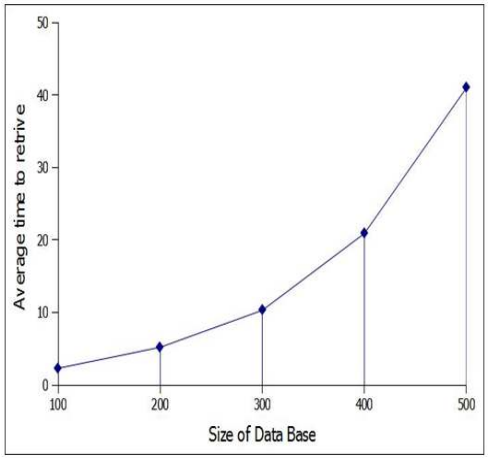

Figure 3:Average Time To Retrieve Any Pattern From DB

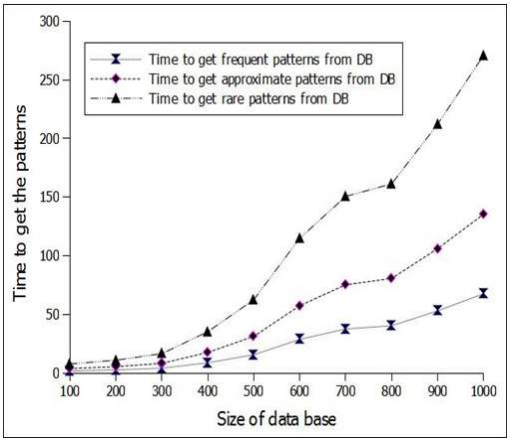

Figure 4: Comparison Of Patterns Retrieving Times From Different Sized Dbs

Step2: Else If (U_c€ R_p)

{
Identify the valid patterns from set of rare patterns that is those are satisfies the user constraint and also satisfies the threshold value constraint, those are taken as valid patterns V_p.

Apply map function to the list of valid patterns;

Apply reduce function after applying the map function;
}

Step3: Else

{
Take one more user speciation constraint to get exactly about the patterns;

If(U_c € F_p)

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Goto step2;

Else goto step3;
}

Map function:

The map function takes the input as transaction id T_{id}, and content of the transaction C_t. And produces the (key, value) pair for the respected T_{id} and C_t.

class Mapper

method Map(transaction id(T_{id}), content of the transaction (C_t))

H = new AssociativeArray

for all transactions T_{id} and content of the transaction C_t.

H{t} = H{t} + 1

for all transctionT_{id} in H do

Emit(transactions T_{id}, count H{t})

Reduce function:

class Combiner

method Combine(transaction T_{id}, [c1, c2,...])

sum = 0

for all count c in [c1, c2,...] do

sum = sum + C_t

Emit(transaction T_{id}, count sum)

class Reducer

method Reduce(transaction T_{id}, counts [c1, c2,...])

sum = 0

for all count C_t in [c1, c2,...] do

sum = sum + C_t

Emit(transaction T_{id}, count sum)

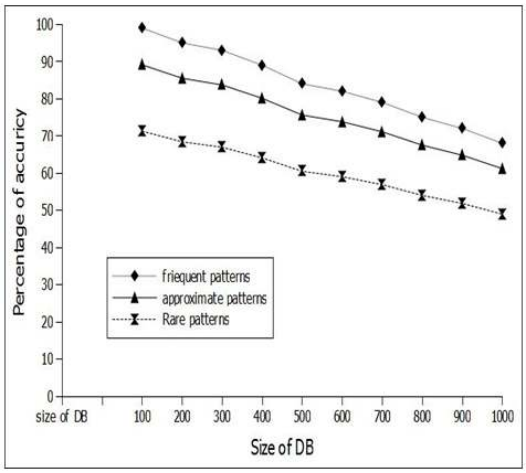

Figure 5: Accuracy Percentage Of Different Retrieval Patterns

5. EXPERIMENTAL EVALUATION

Experimental setup is made with Hadoop. Here we are evaluating the retrieval of data from huge data base based on user's perspective. Our algorithm shows the different types of user's patterns. And also the results show the patterns retrieval. Figure-3 shows the average time taken to retrieve any type of pattern based on the size of the database. That will results the time increases when the size of the database increase. Figure-4 shows the Comparison of Patterns retrieving times from different sized Databases. It results we can easily retrieve the frequent patterns from any huge database because these are most used ones and next we retrieve approximate patterns and then rare patterns takes enormous time compared to other patternsFigure-5 shows the percentage of accuracy for different patterns retrieval. Results shows that frequent patterns having more accurate results compared other patterns.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

	Average	Median	Min	Max	Stdev
Productivity (SLOC/Hour)	1.71	1.53	0.08	6.75	1.12
Size (SLOC)	103743.31	26972.5	120	2486000	272504.46
Effort (Hours)	60672.18	16594.5	673	1320667	147677.38
Duration (Months)	25.49	22	0	92.05	16.11
Peak Staff (Heads)	10.34	1.44	0	131	19.67
SW Effort Distribution					
Requirements Analysis (%)	4.41	0	0	53.45	8.71
Architecture and Detailed Design (%)	6.45	0	0	44.61	11.79
Coding and Unit Testing (%)	20.78	0	0	91.94	25.7
SW INT and System/Software INT(%)	2.08	0	0	26.32	5.49
Qualification Testing (%)	1.52	0	0	34.8	4.84
Developmental Test and Evaluation(%)	0.47	0	0	13.15	1.69
Program Management, SQA, SCM (%)	14.28	0	0	100	21.04
Staff Experience Distrubtion					
High Personel Experience (%)	17	0	0	100	29.4
Normal Personel Experience (%)	9.81	0	0	70	18.27
Entry Personel Experience (%)	3.93	0	0	42	10.14
Volatility (Rating 1-5)	0.76	0	0	5	1.5

6. RESULTS

Calculated Average, Median, Min, Max values across productivity and SLOC [Size across Lines of Code]

Effort Management number of hours in terms of Effort, Time Duration, Staff required to organize this application data.

Table show about how the cost minimization works across various phases of SDLC methodologies and proves these practices can be used for minimizing cost in program management, Software Quality Assurance and Software Change Management.

7. CONCLUSION

This paper mainly aims on user's perspective to divides the pattern into three categories which reduces the search space based on user's perspective and MapReduce to mine valid frequent patterns, approximate patterns and rare patterns from high volumes of ambiguous data in a divide-and-conquer fashion. This implementation proves that this is best cost effective model across various software developments models can use this techniques application rely on big data and hadoop databases for best performance and practices. Simulation Results Proved that implemented algorithms provide more cost effective solution over hadoop based large data set can be easily implemented through optimized cost model techniques. Our results proved that cost model implemented in Map Reduce along with implemented methodologies provides solution for many unsolved problems.

REFERENCES

[1] Carson Kai-Sang Leung et al "*Reducing the Search Space for Big Data Miningfor Interesting Patterns from Uncertain Data*" IEEE International Congress on Big Data, 2014, pp. 315-322

[2] P. Agarwal, G. Shroff, & P. Malhotra, *"Approximate incremental big data harmonization,"* in IEEE Big Data Congress 2013, pp. 118–125.

[3] R. Agrawal& R. Srikant, *"Fast algorithms for mining association rules,"*in VLDB 1994, pp. 487–499.

[4] A. Azzini& P. Ceravolo, "*Consistent process mining over Big datatriple stores,*" in IEEE Big Data Congress 2013, pp. 54–61.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[5] T. Condie, P. Mineiro, N. Polyzotis,& M. Weimer, *"Machine learning for Big data,"* in ACM SIGMOD 2013, pp. 939–942.

[6] R.L.F. Cordeiro, C. Traina Jr., A.J.M. Traina, J. L´opez, U. Kang, &C. Faloutsos, *"Clustering very large multi-dimensional datasets with MapReduce,"* in ACM KDD 2011, pp. 690–698.

[7] J. Dean & S. Ghemawat, *"MapReduce: simplified data processing onlarge clusters,"* CACM 51(1): 107–113, Jan. 2008.

[8] A. Koufakou, J. Secretan, J. Reeder, K. Cardona, & M. Georgiopoulos, *"Fast parallel outlier detection for categorical datasets using MapReduce,"* in IEEE IJCNN 2008, pp. 3298–3304.

[9] A. Kumar, F. Niu, & C. R´e, *"Hazy: making it easier to build andmaintain Big-data analytics,"* CACM 56(3): 40–49, Mar. 2013.

[10] L.V.S. Lakshmanan, C.K.-S. Leung, & R.T. Ng, *"Efficient dynamic mining of constrained frequent sets,"* ACM TODS 28(4): 337–389, Dec.2003.