Multi-Keyword Searching and Top-K Encrypted Retrieval of Cloud Data

D. Pradeepa[1], Dr. P. Sumathi[2]

[1]Research Scholar, PG & Research Department of Computer Science, Government Arts College, Coimbatore INDIA.
[2]Assistant Professor, PG & Research Department of Computer Science, Government Arts College, Coimbatore INDIA

Abstract

Searchable encryption is a method that permits approved clients to look through encoded data without decrypting it. This is finished by indexing the data, so that it tends to be searched without compromising the security of the data. Guaranteeing the cloud data security is a main issue for corporate cloud supporters, and at times for the confidential cloud clients. The multi-keyword searching empowers more unambiguous questioning, and the top-k encrypted retrieval guarantees that the most significant data is uncovered. This approach can be utilized in different spaces, for example, healthcare, finance or legal, where data privacy is an urgent viewpoint. The proposed approach is helpful for Top-k Encrypted Retrieval of Cloud data and can be utilized by organizations to store and recover delicate data in the cloud securely. The proposed Cost based Iterative Deepening Depth-First Search (CIDDFS) is developed to reduce the searching time of the keyword in encrypted data. According to cosine similarity of the text, the cost of searching time is reduced. The proposed CIDDFS method is secure and safe, which is validated by various security analysis schemes. This approach ensures that user/owner does not need to perform time-consuming calculations during the process of searching the keywords or during the audit of the shared data.

Keywords: Cloud Storage, Ranking Search, Multi-Keyword, Depth-First Search, Encryption, Top-k retrieval.

1. INTRODUCTION

Cloud Computing permits new types of administrations through which computational and network resources are available by means of the Internet. Data rethinking is one of the most widely recognized cloud computing administrations. All the while, the security remotely put away data on an untrusted cloud server is a central issue. Delicated data, for example, individual wellbeing records, addresses, personal assessment and financial reports, etc., are generally rethought in encoded structure utilizing notable cryptographic methods to ease these worries. While encoded data capacity shields distant data from unapproved access, it confounds a few basic yet necessary data use administrations, for example, plaintext keyword search.

As data management and analysis are facing new challenges in the age of big data, the rationality and timeliness of the data processing methods are becoming the research hotspot of big data statistical analysis, and big data association analysis greatly increases the profits of enterprises.

As one of the significant examination directions of data mining, regular itemsets mining assumes a fundamental part in mining associations, correlations, causality and other significant data mining errands (Agrawal and Srikant et al.[2])which is a strong stimulus to the applications of association rules in business sectors selection, decision examination and business board (Liao et al. [3]);the current traditional successive itemsets mining algorithms are broadness first algorithm Apriori proposed(Agrawal et al. [2]) and depth-priority algorithm FP-Growth introduced by (Han et al. [4]) to work on the productivity of successive itemsets mining, numerous analysts have proposed a few strategies to optimize classical algorithms in the mean time, to address the bottleneck of mining execution and lessen memory consumption and computation cost of the machine in the single machine environment, equal and distributed algorithms are proposed (Agrawal and Shafer et al.[2], Tanbeer et al. [5]). For traditional techniques, the applications of corresponding algorithms for continuous itemsets mining in the huge scope datasets will effortlessly cause high CPU consumption, high memory cost, high I/O above, low computing execution and other different issues.

With the appearance of big data time, huge data are developing quickly. Be that as it may, in actuality, the majority of the enormous scope data are made out of monstrous little files. Little files for the most part allude to those file sizes, which are under 64 MB. As per a concentrate in 2007 at the National Energy Research Scientific Computing Center, 43% of the north of 13 million files on a common parallel file system are under 64 KB and close to 100% are under 64 MB (Petascale Data Storage Institute (2007)), and more logical applications consist of an enormous number of little files are portrayed in (Carns et al. [7], Essam et.al [8]). In any case, notwithstanding monstrous little file datasets, the constructed FP-tree in Parallel FP-Growth (PFP) algorithm can't squeeze into the memory, which frequently creates some issues, for example, memory overflow and enormous communication above. In the interim, the computing proficiency of the Hadoop stage generally relies upon the presentation of HDFS and MapReduce (White [6]), and Hadoop was from the get go, planned explicitly to deal with streaming enormous files, so while managing monstrous little files, there are huge limitations.

Huge little files will lessen the presentation of Hadoop, which is chiefly displayed in the accompanying two angles: (1) the entrance effectiveness of HDFS is diminished.
(2) The additional above of MapReduce is expanded.

Frequent Pattern-Growth

FP-development takes on two different ways those are isolating and vanquish way. Gridded FP-Growth, GFP-Growth is short and it is intended to run on a PC cluster. To stay away from memory flood, GFP-Growth takes on the projection technique to find all the contingent example bases straightforwardly without building a FP-tree unavoidable and lattice computing enormous scope calculation power, complex issues and furthermore exceptionally huge data stockpiling resources (Byreddi S et al.[9]). Matrix computing is loosely coupled minimal expense engineering in light of a web association.

It is heterogeneous as computing and stockpiling is a conventional cluster framework. Network computing is extremely simple and furthermore adds some extra computing resources

Make a tree-based file structure and propose an "Iterative Deepening - Depth-first Search (IDDFS)" calculation in view of it to achieve high search performance. In view of the remarkable structure of tree-based database, the proposed search plan will achieve sub-linear search times and manage record cancellation and addition. The proposed EHMRS plot utilizes the FPgrowth calculation to find continuous Itemsets in an exchange database without creating up-and-comers. The FPgrowth calculation addresses the database as a tree known as a regular example tree or FP tree. The relationship between the thing sets will be kept up with by this tree structure.

While encryption guarantees data protection, it makes routine search tasks troublesome. This is because of the way that plain-text inquiries can't be utilized expressly in light of the fact that they uncover the search terms to CSP. Moreover, because of the absence of a search protocol that works for both plain-text questions and scrambled reports, the encoded records alluding to these plain-text inquiries can't be recovered. Subsequently, searchable encryption plans arose, which produce scrambled search inquiries and give clients search capacities. It is necessary to research a successful SE plot to decrease the financial weight related with cloud use. Different plans have been proposed by the researchers to support effective recovery (Pramudiono I et.al [90]). These plans had the option to accelerate the search cycle, yet they had restricted search capability, as they just upheld conjunctive searching. Different Researchers, then again, attempted to give further developed search capacities, however coming up short on wanted level of search performance, bringing about a tradeoff between search effectiveness and search capability.

A hybrid multi-keyword search that profits matching data items in positioned request. The proposed safeguarded searchable encryption conspire takes into consideration multi-keyword questions over an encoded report corpus and recovers related records positioned by a similitude positioning. Not at all like every single past plan to search is sub-linear to the all out number of archives that contain the questioned set of keywords. Make a tree-based list structure and propose an "Iterative Deepening - Depth-first Search (IDDFS)" calculation in light of it to achieve high search performance (Chen YH et al. [10]). Due to the novel structure of tree-based database, the proposed search plan will achieve sub-linear search times and manage record erasure and addition. The proposed method utilizes the FPgrowth calculation to find continuous Itemset in an exchange database without producing up-and-comers. The FPgrowth calculation addresses the database as a tree known as a regular example tree or FP tree. The relationship between the thing sets will be kept up with by this tree structure.

2 EXISTING METHODOLOGIES

2.1 Agent Based Depth First Search Algorithm

Palanisamy Vet al. [17] proposed Agent Based Depth First Search Algorithm. Artificial intelligent procedures give various types of searching strategies in a tree structure and those techniques are carried out successive and equal way utilizing structured programming. Depth-first search is a tree search algorithm which empowers the crossing from the source hub to neighbor hubs, contiguous the source hub in a diagram and is executed sequentially in this design. The improvement of the consecutive Depth-first search is looked for in a few research studies. One of the solutions to the improvement of Depth-first search is the plan of a multi-agent framework for the searching issue. In this research study, the proposed multi-specialist structure for Depth-first search parcels the entire graph into a couple of groups using the bi-related region technique and consigns the specialists to each bunch to play out the Depth-first search consecutively. The proposed strategy uncovers a critical improvement as far as the performance proportion of the framework for successive Depth-first search. The performance proportion of the proposed strategy has been assessed as far as different perspectives, for example, all out calculation time, the quantity of communications.

2.2 Multi Period Degree Constrained Minimum Spanning Tree (MPDCMST)

Wamiliana Met al. [11] proposed Multi Period Degree Constrained Minimum Spanning Tree. The limitation of the connections on each vertex happens to keep the reliability of the organization. Additionally, the installation interaction is likewise separated into a periods because of asset constraints. This research will examine the mixture between the depth-first search technique and Kruskal's Algorithm applied to tackle the MPDCMST problem. From the outcomes above presume that when a vertex viable is introduced (associated) in the past period, and afterward the technique ought to proceed with the cycle without requesting more vertices and this will give an improved arrangement.

2.3 Improved Algorithm for Searching Maze Based on Depth First Search

Chen YHet al. [12] proposed an Improved Algorithm for Searching Maze Based on Depth First Search. They proposed an algorithm in light of a depth-first search to search for obscure mazes and build the guides. To get a total guide of the maze, we use stacks to store the directions and headings of the agents that poor person been searched, and we should leave the stack void to end the search. To accelerate the search, we further develop the depth-first search technique, utilizing two agents to search in the maze simultaneously, and exchange the searched maze data when the two agents met. In the wake of searching, we can track down the shortest way between any two focuses in the searched maze. Utilizing two agents to search for a maze is quicker than a solitary agent, in which various agents require some investment than a solitary agent. The outcome shows that the strategy is feasible.

2.4 Distributed FP Growth Algorithm

Byreddi Set al.[9] proposed Distributed FP Growth Algorithm for Cloud Platform without exposing the individual transaction Data. Data mining is an idea of extricating the necessary patterns to take appropriate choices. One of the most significant challenges in data mining is to separate concealed patterns with security and protection from tremendous databases. Protection safeguarding is a technique used to extricate stowed away patterns with security. In this paper, Mining Association rules with a security saving system in the cloud stage are proposed (Zaki MJ et.al [13]). It is a strong technique to track down the secret pattern in the distributed database. For the present, numerous mechanisms have proposed, however, it has numerous downsides, not demonstrated and not explicit. In the cloud, the data is put away in the servers. The data is distributed to various servers on the cloud stage. Every server has one of the exchange data. The ongoing paper proposed the distributed FP growth algorithm for cloud platforms without uncovering the singular exchange data (Yang K et.al [14]). The outcomes demonstrated that the proposed algorithm is awesome to separate concealed patterns from the Cloud stage regarding proficiency. It utilizes the technique for AI notwithstanding the security safeguarding for the distributed servers. Likewise in the Hadoop stage, by applying the spurious data, for example, clamor to the conditional data in the public clouds, the outcome shows that the confidential data is gotten and simultaneously the expense of the execution isn't significant by mining the affiliation rules.

2.5 Big data Spark Framework with Fp-Growth Algorithm

Senthilkumar A.et al. [15] proposed,the data produced from various sources like mobile devices, sensors, and web cameras in everyday life is developing dramatically and is handled in huge data. These handled data have become highly significant for every one of the significant areas, like research, business and industry. One of the huge data processing platforms is Apache Spark which can deal with both group processing and ongoing streaming data. Cloud computing which can give required resources is utilized to meet the constant processing prerequisites of streaming applications (N. Bharill etal. [16]). In a virtualized cloud climate, where different large data applications are sent, the performance obstruction can likewise influence the performance of the streaming undertakings resulting in the performance corruption of the positions. Association Rule Mining is the algorithm used to track down the emphatically related patterns between itemsets. FPGrowth algorithms are the generally utilized Association Rule mining algorithm. In this paper, the parallel FP-Growth algorithm is executed in Spark Framework. Subsequently the more slanted kind of containers is dispensed with high CPU resources committed to these containers. Then, the asset distribution conspire likewise distinguishes how much info data and in light of the size of the data it chooses the quantity of containers to be apportioned, the optimal qualities are shown up, in view of the trial evaluation. In future work, dynamic expectation should be possible to anticipate the most optimal resources utilizing FP-Growth.

3. PROPOSED METHODOLOGY

The proposed scheme first conducts a conjunctive keyword search, in which documents containing the entire query words are easily retrieved and sorted based on their relevance to the query. If the number of obtained results exceeds the number of desired results, the search is efficient. The records containing the subset of question words are remembered for the aftereffects of disjunctive searching. The acquired outcomes are evaluated and imparted to the end clients. As a result, the proposed CIDDFS on the forward and inverted indexes strikes a balance between search efficiency and capability.
The proposed scheme should meet the following design objectives.

1. Multi-keyword search: It can handle several keywords in the search query, i.e., it should mimic real user search activity by allowing a different number of search terms.
2. Ranked retrieval: The answer to the search query should be ordered based on its relevance to the query, reducing post-processing computational overheads for end users.

   Multi-keyword searching represents end-user search activity, as a result, an encrypted search query containing all of the search terms entered by the end user is created in this process. The method used to generate the search query requires less computing resources and can be easily created by end users using any computer.

It consists of comparing the question to the outsourced indexes. In literature, various symmetric search schemes have been proposed. Methods, such as, document clustering and keyword binning based on different indexes are proposed in this phase to reduce the amount of time needed for document retrieval.

3.1 Depth-First Search

Depth-first search keeps away from this memory constraint. It works by continuously creating a relative of the most as of late extended node, until some depth cutoff is reached, and afterward backtracking to the following most as of late extended node and producing one of its relatives.

Since depth-first search just stores the flow way at some random point, it is bound to search all ways down to the end depth. To analyze its time complexity should characterize another boundary, called the edge spreading factor, which has the typical number of various administrators which are relevant to a given state. For trees, the edge and node fanning factors are equivalent, yet for graphs overall the edge stretching element might surpass the node spreading factor. For instance, the graph has an edge spreading variable of two, while its node stretching factor is only one. Note that a breadth-first search of this graph requires some investment while a depth-first search demands dramatic investment. Since the space utilized by depth-first search develops just as the log of the time required, the calculation is time-bound as opposed to space-bound by and by.

3.2 Depth-First Iterative-Deepening

A search algorithm which experiences neither the downsides of breadth-first nor depth-first search on trees is depth-first iterative-extending (DFID) (Palanisamy V et.al [17]). The algorithm fills in as follows: First, play out a depth-first search to depth one.
Second, disposing of the nodes produced in the first search, begin once again and do a depth-first search to even out two. Third, begin once more and do a depth-first search to depth three, and so forth, proceeding with this cycle until a goal state is reached.

Since DFID broadens all hubs at a given depth preceding, developing any hubs at a more noticeable depth, and finding a most limited length arrangement is guaranteed. Likewise, since at some random time it is playing out a depth-first search, and never searches further than depth, than the space is utilized.

The DFID is that it performs squandered calculation before arriving at the goal depth. At first look, it appears to be nonetheless, present an examination of the running time of DFID that shows that this squandered calculation influence the asymptotic development of the run time for remarkable tree searches. The natural explanation is that practically everything the work is finished at the most profound level of the search. Sadly, DFID experiences a similar downside as depth-first search on inconsistent graphs. Specifically, it should investigate all potential ways to a given depth.

As referenced above, since DFID produces all hubs at a given depth prior to extending any hubs at a more noteworthy depth, it generally tracks down a most brief way to the objective, or any other state for that matter. Hence, it is optimal in terms of solution length.

3.3 FP-Growth

FP-tree based frequent itemset mining technique, called FP-Growth, and achieves high capability, in contrast with Apriori-like methodology. The FP-Growth technique embraces the gap and vanquish framework, uses only two full I/O outputs of the database, and avoids iterative competitor age by (Pramudiono I et al. [18]). Frequent pattern mining comprises of two stages:

- Firstly,Building a minimized data structure, FP Tree (frequent pattern tree), which stores more data in less space.
- Secondly is working of a FP-tree based pattern development technique to recursively uncover each frequent pattern.

The essential output totals will help each and every thing and short time later itpicks things that satisfy minimum help. This methodology produces frequent 1-itemsets and after that stores them in recurrence descending order. The subsequent output constructs FP-tree. The FP-Tree is a compressed portrayal of the input.

Parallel FP-Growth

To make this independent, initially repeat the issue of FIM then, at that point, characterizes parameters utilized in FPGrowth, and portray the algorithm and present parallel FPGrowth algorithm, or PFP. FPGrowth works in a gap and overcomes in way which requires two sweeps on the database (Senthilkumar A et al. [15]). FPGrowth initially processes a rundown of frequent things arranged by frequency in descending order (F-List) during its most memorable database check. In its subsequent sweep, the database is compressed into a FP-tree. Then FP-Growth begins to dig the FP-tree for everything whose help is bigger than by recursively fabricating its conditional FP-tree. The algorithm performs mining recursively on FP-tree. The issue of finding frequent itemsets is switched over completely to look the developing trees recursively.

3.4 Iterative Deepening Depth First Search (IDDFS) using FPgrowth algorithm Search Process

The proposed scheme search protocol is a recursive procedure on the tree known as the "Iterative Deepening Depth-First Search" (ID-
DFS) algorithm. IDDFS consolidates the space-productivity of depth-first search with the speed of breadth-first search. IDDFS calls DFS for different depths beginning from a given worth. DFS is prohibited from going beyond the depth specified in each call. The basic principle of algorithm is to begin with a start node and then examine the node's first child (grandchild of the starting node), and so on, until a node has no more children. It then advances one step and examines the next boy. If there are no more children, it advances one step higher, and so on until it finds more children or reaches the start node. In this event, the objective node has not been reached subsequent to getting back from the last offspring of the beginning node, the objective node can't be found since all nodes have been navigated by that stage.

IDDFS combines the space-efficiency of depth-first search with the completeness of breadth-first search (when the branching factor is finite). At the point, when the way cost is a non-diminishing capability of the node depth, it is ideal Iterative extending might appear to be wasteful in light of the fact that it visits expresses on various times, however, it ends up being not so costly on the grounds that most nodes in a tree are in the base level, so it doesn't make any difference assuming the upper levels are visited on various times.

Implementation:

Step 1: -For each child of the current node
Step 2: - If it is the target node, return
Step 3: - If the current maximum depth is reached, return.
Step 4: - Set the current node to this node and go back to 1.
Step 5: - After having gone through all children, go to the next child of the parent (the next sibling)
Step 6: - After having gone through all children of the start node, increase the maximum depth and go back to 1.
Step 7: - If have reached all leaf (bottom) nodes, the goal node doesn’t exist.

Retrieval Process

The process of finding data (generally documents) in the form of text that matches the information required from a set of documents stored on a computer is known as information retrieval (IR). Incorrect user requests are a common problem on IRs; this is caused by users weaknesses in representing their needs in the query. Researchers have suggested numerous solutions to address these limitations. In this research, proposed an approach based on the FPGrowth algorithm for the quest for frequent itemsets. The key storing keywords is the cloud storage server, which is denoted by the letter K r(keyword0, keyword1, keyword2, and keyword3). Calculate the support for a single itemset by traversing each record and seeing if it includes an itemset.

Procedure: FPgrowth (DB, ξ)

```
Step 1:  Define and clear F-List: F [];
Step 2:  foreach T transaction Ti in DB do
Step 3: -       Foreach Item $a_j$ in Ti do
Step 4: -               F [$a_j$] ++;
Step 5: -       end
Step6: - end
Step 7: - Sort F[];
Step 8: - Define and clear the root of FP-tree: r;
Step 9: - foreach T transaction Ti in DB do
Step 10: -      Make Ti ordered according to F;
Step 11: -      Call Construct T tree (Ti, r);
Step 12: - end
Step 13: - foreach item $a_j$ in I do
Step 12: -      Call Growth(r,$a_j$, ξ);
Step 13: - end
```

This algorithm, begins by compressing the input database, resulting in a frequent pattern tree case. The compressed database is then divided into a few conditional databases, each representing a single specific frequent pattern. Finally, mining of each database is done separately. As a result, the search costs are greatly reduced, resulting in strong selectivity.

3.5 Cost based search as Iterative Deepening Depth-First Search (CIDDFS)

In the search tree, the nodes are visited by IDDFS every iteration like DFS as in the same order. The already-visited nodes are not considered by the IDDFS implementation and do not provide better performance for undirected graphs. However, the major drawback of this algorithm is high cost for searching time because it visited the iterative deepening states multiple times. The upper level of the algorithms is visited multiple times, because most of the nodes in a tree are presented only in the bottom level. To overcome these issues, the proposed method implements the cost based search as Iterative Deepening Depth-First Search (CIDDFS).

The cost of the search can be identified by using the following equation:

$$\boldsymbol{cost} \propto \frac{\mathbf{1}}{\boldsymbol{cosine\ similarity\ of\ text}}$$

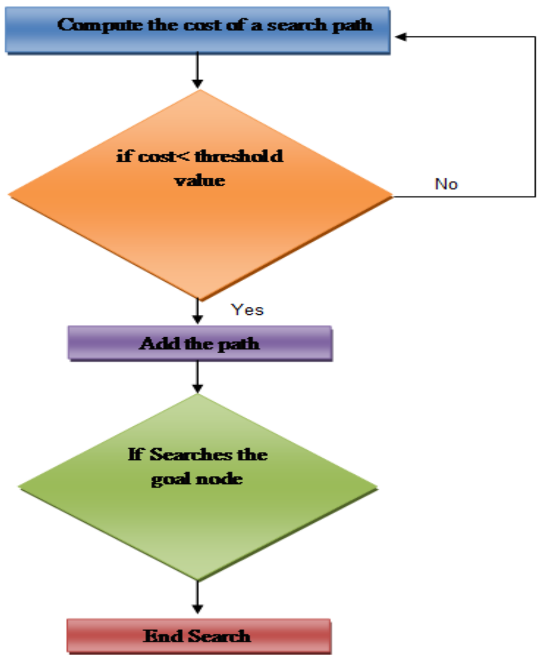

Figure 1.1 Cost Identification

The process of finding data (generally documents) in the form of text that matches the information required from a set of documents stored on a computer is known as information retrieval (IR). Incorrect user requests are a common problem on IRs; this is caused by user weaknesses in representing their needs in the query. Researchers have suggested numerous solutions to address these limitations; proposed an approach based on the FP-Growth algorithm for the quest for frequent itemsets. The proposed CIDDFS scheme uses the FPgrowth algorithm to find frequent Itemset in a transaction database without generating candidates.The FPgrowth algorithm represents the database as a tree known as a frequent pattern tree or FP tree. The relationship between the item sets will be maintained by this tree structure.

Enhanced Contributions

Step 8: Tdb = Total Transactional Database
Step 9: S Tdb= Sub transactional Database = Tdb / ntr
Step 10: Check the number of clusters in Hadoop, if cluster c= Cn
Step 11: Then Every Cn is allotted with the STdb
Step 12: Compare and map cn(i)=STdb (i)
Step 13: Repeat the process and continuously check
Step 14: Finally transfer the transaction to machine.
Step 15: Identify the difference and frequent itemset pattern (FIP)
Step 16: Recursively computes the independent sub-transaction databases generated
Step 17: Aggregate the local frequent itemsets generated from each node in the cluster
Step 18: Finally get the global frequent itemsets GFI

A small file processing program or the Sequence File is used to merge all massive small files, which are composed of a large amount of transaction datasets stored in HDFS, into a large transaction data file (transaction database).

Equally divide the transaction database into several sub-transaction databases and then assign them to different nodes in Hadoop cluster.

Then it is automatically operated by IDFS, when necessary we can use the balance command enabling its file system to achieve load balancing. The Map function compares the item of each transaction in the sub-transaction database with the item. Then, distribute the corresponding transaction to the machine. The Reduce function recursively computes the independent sub-transaction databases generated. FP Tree isgenerated, identify the difference and frequent itemset pattern. Aggregate the local frequent itemsets generated from each node in the cluster by MapReduce, and finally get the global frequent itemsets.

4. EXPERIMENTS AND RESULTS

Precision

$$\boldsymbol{Precision} = \boldsymbol{true\ positive} \,/\, (\boldsymbol{true\ positive} + \boldsymbol{false\ positive})$$

While comparing the Existing algorithm and proposed CIDDFS with FP-growth algorithm, provides the better results. The existing algorithm values start from 66 to 86, 65 to 74 and proposed CIDDFS with FP-growth algorithm values starts from 87 to 97. The proposed method provides the great results

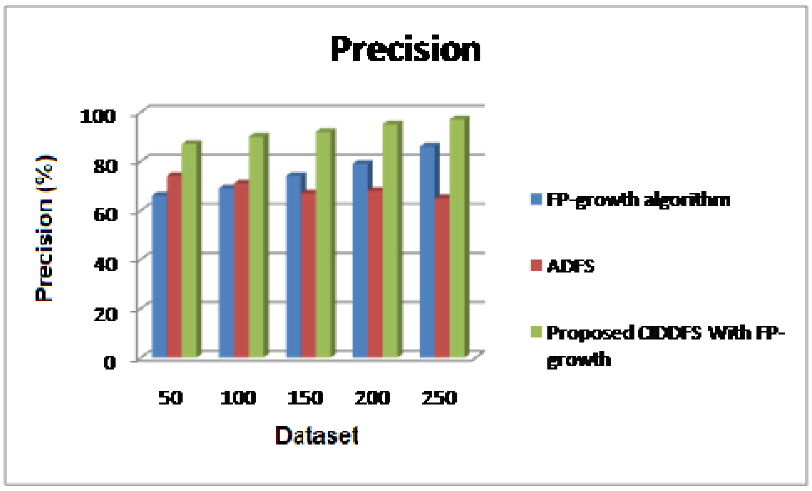

Figure 1.2 Comparison chart of Precision

The figure 1.2 shows the comparison chart of Precision demonstrates the existing FP-growth algorithm, ADFS Algorithm and proposed CIDDFS with FP-growth algorithm. X axis denote the Dataset and y axis denotes the Precision ratio. The proposed CIDDFS with FP-growth algorithm values are better than the existing algorithm.

Recall

$$\boldsymbol{Recall} = \boldsymbol{True\ Positives} - (\boldsymbol{True\ Positives} + \boldsymbol{False\ Negatives})$$

FP-growth algorithm, provides the better results. The existing algorithm values start from 0.62 to 0.75, 0.59 to 0.72 and proposed CIDDFS with FP-growth algorithm values starts from 0.83 to 0.96. The proposed method provides the great results.

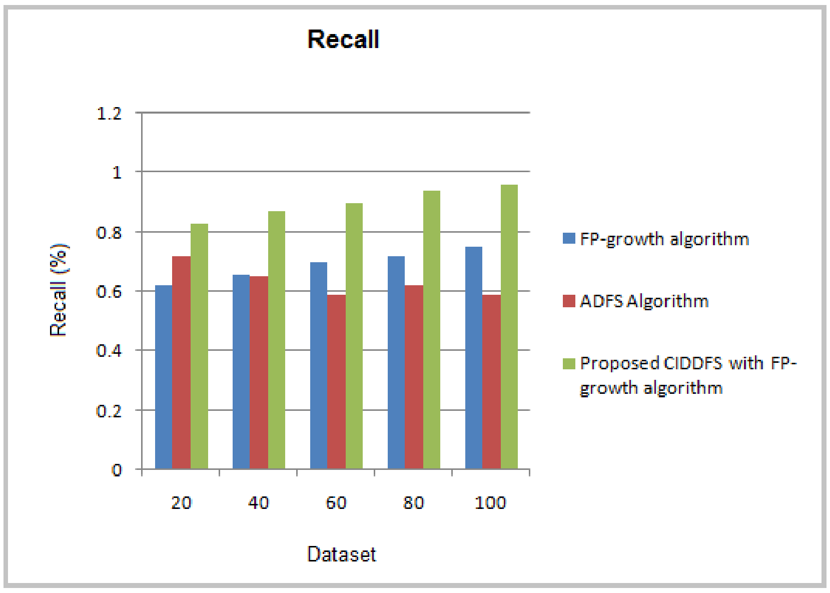

Figure 1.3 Comparison chart of Recall

The Figure 1.3 shows the comparison chart of recall demonstrates the existing FP-growth algorithm, ADFS Algorithm and proposed CIDDFS with FP growth algorithm. X axis denote the Dataset and y axis denotes the Recall ratio. The proposed CIDDFS with FP-growth algorithm values are better than the existing algorithm.

F -Measure

$$\mathbf{F-Measure = (2 * Precision * Recall) / (Precision + Recall)}$$

While comparing the Existing algorithm and proposed CIDDFS with FPgrowth algorithm, provides the better results. The existing algorithm values start from 0.82 to 0.89, 0.61 to 0.72 and proposed CIDDFS with FP-growth algorithm values starts from 0.92to 0.98. The proposed method provides the great results.

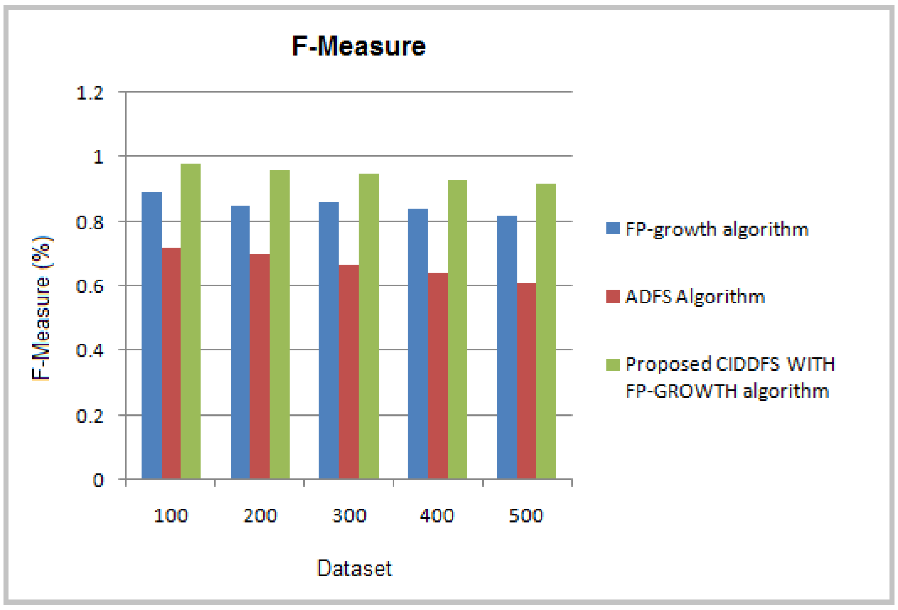

Figure 1.4 Comparison chart of F -Measure

The figure 6.4 shows the comparison chart of F -Measure demonstrates the existing FP-growth algorithm, ADFS Algorithm and proposed CIDDFS with FP-growth algorithm. X axis denote the Dataset and y axis denotes the F -Measure ratio. The proposed CIDDFS with FP-growth algorithm values are better than the existing algorithm.

Sensitivity

$$\boldsymbol{Sensitivity = True\ Positive/(True\ Positive + False\ Negative\ )}$$

While comparing the Existing algorithm and proposed CIDDFS with FP-growth algorithm, provides the better results. The existing algorithm values start from 83.48 to 93.66, 81.22 to 90.65 and proposed CIDDFS with FP-growth algorithm values starts from 86.87 to 96.91. The proposed method provides the great results.

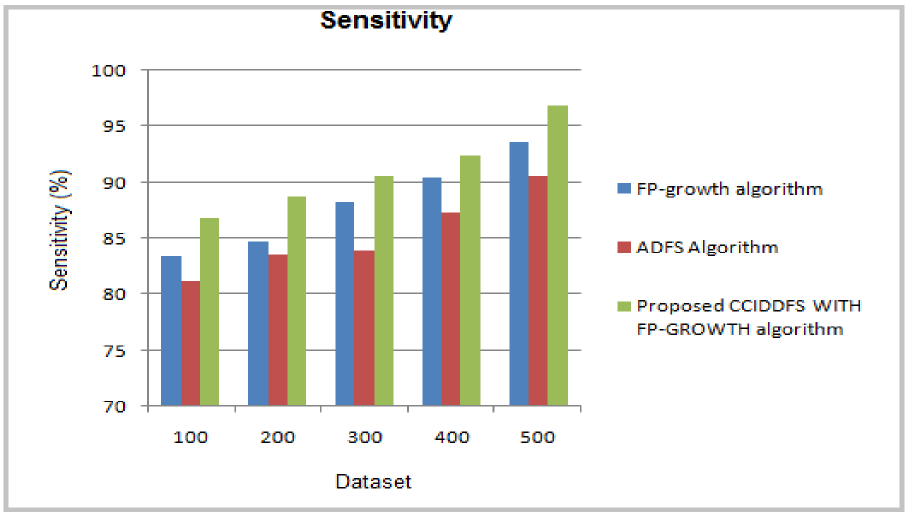

Figure 1.5 Comparison chart of Sensitivity

The figure 1.5 shows the comparison chart of sensitivity demonstrates the existing FP-growth algorithm, ADFS Algorithm and proposed CIDDFS with FP-growth algorithm. X axis denote the Dataset and y axis denotes the Sensitivity. The proposed CIDDFS with FP-growth algorithm values are better than the existing algorithm.

Accuracy

$$\boldsymbol{Accuracy\ =\ (true\ value\ -\ measured\ value)\ /\ true\ value\ *\ 100\%}$$

While comparing the Existing algorithm and proposed CIDDFS with FP-growth algorithm, provides the better results. The existing algorithm values start from 73 to 72, 65 to 69 and proposed CIDDFS with FP-growth algorithm values starts from 79 to 80. The proposed method provides the great results.

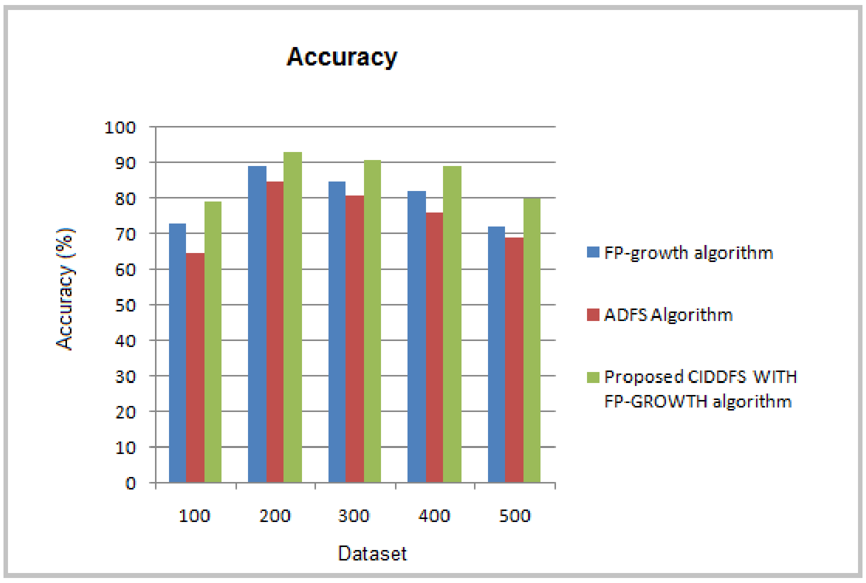

Figure 1.6 Comparison chart of Accuracy

The figure 1.6 shows the comparison chart of F-Measure demonstrates the existing FP-growth algorithm, ADFS Algorithm and proposed CIDDFS with FP-growth algorithm. X axis denote the Dataset and y axis denotes the Accuracy ratio. The proposed CIDDFS with FP-growth algorithm values are better than the existing algorithm.

5 Conclusions

In this phase, owing to the small files processing strategy, proposed the Cost based Iterative Deepening Depth-First Search (CIDDFS) algorithm can reduce memory cost greatly and improve the efficiency of data access, thus avoids memory overflow and reduces I/O overhead. In the mean time, the CIDDFS algorithm is moved to the MapReduce environment, which can finish successive itemsets mining productively and accordingly enhance the general exhibition of FP-Growth algorithm. The experimental results demonstrate the way that IPFP algorithm can make an advancement where CIDDFS algorithm has its deformities in dealing with enormous little files datasets, and has a decent speedup and a higher mining effectiveness.

6. REFERENCES

1. Leung K, Ma K, Wong W.K, Leong P.H.W,“FPGA implementation of a microcoded elliptic curve cryptographic processor” In Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (Cat. No. PR00871), pp. 68–76, 2000.
2. Agrawal R, Shafer JC, “Parallel mining of association rules”, IEEE Transactions on knowledge and Data Engineering, in IEEE Access, vol. 8, no.6,pp. 962-969, 1996.
3. Liao L, Lin TP, Zhang Y, “Corporate board and corporate social responsibility assurance: Evidence from China” Journal of Business Ethics,vol. 150, no. 1 pp.211-25, 2018.
4. Han J, Pei J, Yin Y.,“Mining frequent patterns without candidate generation”, ACM sigmod record, vol. 29, no. 2, pp. 1-2, 2000.
5. Tanbeer SK, Ahmed CF, Jeong BS, “Parallel and distributed algorithms for frequent pattern mining in large databases”, IETE technical review, vol. 26, no.1, pp.55-66, 2009.

6. White T,“Hadoop: The definitive guide", O'Reilly Media, Inc.".
7. Carns P, Lang S, Ross R, Vilayannur M, Kunkel J, Ludwig T., “Small-file access in parallel file systems”,IEEE International Symposium on Parallel & Distributed Processing, pp. 1-11, 2009
8. Essam, M. A. Abdel-Fattah and L. Abdelhamid, "Towards Enhancing the Performance of Parallel FP-Growth on Spark”,IEEE Access, vol. 10, pp. 286-296, 2022.
9. Byreddi S, Reddy AR.,“Distributed FP Growth Algorithm for Cloud Platform without Exposing the Individual Transaction Data”, International Journal of Recent Technology and Engineering (IJRTE), vol.8, issue-3, pp. 4983 – 4989, 2010.
10. Chen YH, Wu CM,“An improved algorithm for searching maze based on depth-first search”, IEEE International Conference on Consumer Electronics-Taiwan (ICCE-Taiwan), pp. 1-2, 2020.
11. Wamiliana W, Usman M, Elfaki F, Azram M,“Some greedy based algorithms for multi periods degree constrained minimum spanning tree problem”, ARPN Journal of Engineering and Applied Sciences,vol.10, no.21, pp.10147-52, 2015.
12. Chen YH, Wu CM,“An improved algorithm for searching maze based on depth-first search”, IEEE International Conference on Consumer Electronics-Taiwan (ICCE-Taiwan), pp. 1-2, 2020.
13. Zaki MJ, Parthasarathy S, Ogihara M, Li W, “ Parallel algorithms for discovery of association rules” Data mining and knowledge discovery, vol.1, no.4, pp.343-73,1997.
14. Yang K, Zhang K, Jia X, Hasan M. A & Shen X, “Privacy-preserving attribute-keyword based data publish-subscribe service on cloud platforms”, Information Sciences, vol.387 no.C, https://doi.org/10.1016/j.ins.2016.09.020, 2016.
15. Senthilkumar A, Hariprasad D, “A Spark Based Frequent Itemset Mining Using Resource Management for Implementation of Fp-Growth Algorithm in Cloud Environment”, Annals of the Romanian Society for Cell Biology, vol15, pp.6050-59, 2021
16. N. Bharill, A. Tiwari and A. Malviya, "Fuzzy Based Scalable Clustering Algorithms for Handling Big Data Using Apache Spark," in IEEE Transactions on Big Data, vol. 2, no. 4, pp. 339-352, 2016.
17. Palanisamy V, Vijayanathan S,“A novel agent based depth first search Algorithm”, IEEE 5th International Conference on Computing Communication and Automation (ICCCA), pp. 443-448, 2020.
18. Pramudiono I, Kitsuregawa M, “Parallel FP-growth on PC cluster”, Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 467-473, 2003.