ISSN:0975-9646

Nested Software Pattern Testing on Cloud Storages

V.Nethaji.,M.C.A., M.Phil.
Karpagam University,
Pollachi Main Road, Eachanari Post,
Coimbatore, Tamilnadu, India.

C.Chandrasekar.M.C.A., M.Phil.,Ph.D.,
Computer Science Department,
Assistant Professor, Periyar University,
Salem, Tamilnadu, India.

Abstract— Cloud computing plays a vital role in the next phase of IT enterprise. The management of data in the cloud computing environment is processed with the centralized data centers which might not be trustworthy. So, it is difficult to provide a security over a data storage in cloud computing. Several software specifications are introduced to get back the erroneous data from the dataset stored in the large data centers. One of the root causes of the software products frequently come up with poor, incomplete, or yet without any recognized specifications. In an attempt to get better program understanding, extended a iterative software pattern testing [1] in which output patterns are repeat regularly within a program trace, or diagonally multiple traces. Frequent iterative patterns provide recurrent set of program activities that are processed similar to the software specifications but do not challenge with the software engineers using the software nested pattern testing in cloud environment. To overcome the issues of storage security in cloud computing environment, a novel nested software pattern testing leads to not only more compact yet also a complete result with better efficiency. Nested Software Pattern (NSP) - block algorithm for nesting software sequential pattern testing is mainly determined over candidate's generation and protection. NSP- BLOCK is an algorithm based on the concepts of memory blocks, works on exclusive occurrence of items in a sequential cloud database. A conception is a bigger software testing block can grasp the smaller of having same type in cloud zone to address the need of software engineers. Simulation experiments are conducted with various conditions in the cloud computing environment using Amazon EC2 dataset. Performance metric for evaluation of proposed NSP technique is measured in terms of Scalability, Pattern Prediction and run time (log scale).

Keywords — Cloud Environment, Software testing, nested pattern, NSP block algorithm, sequential cloud database.

I. INTRODUCTION

Many applications are involved in sequence software pattern testing in cloud environment. Distinctive examples comprise customer shopping sequences, Web click streams, biological sequences, sequences of events in science and engineering and nature and social development. Sequential software pattern testing is the testing of commonly occurring ordered events or subsequences as patterns. In industry the sequential software testing can be helpful for target marketing, customer retention and many other tasks in cloud environment. Other areas where sequential analysis used are web access software pattern analysis.

Cloud Computing is a representation of distributing the processing tasks to the resource pool which comprises of a huge number of computers, so that numerous application systems can receive the storage space, computing power, and a variety of software services provided with a demand over the data. The novelty of the Cloud Computing is that it almost takes out an restricted cheap storage space and processing power. This turn out to be a platform for storing and processing the huge amount of data.

Cloud computing united with data mining can present authoritative capabilities of storage and processing and an exceptional resource administration. Owing to the fiery data development and quantity of computation concerned in data mining, a competent and high-performance processing is very essential for a thriving data mining application. Data mining in the cloud computing environment can be measured as the prospect of data mining as of the rewards of cloud computing model.

Cloud computing provides larger capabilities in data mining and data analytics. The main apprehension about data mining is that the storage space needed by the processes and item sets is very huge. There are definite issues related with data mining in the cloud computing. The foremost problem of data mining with cloud computing is safety as the cloud provider containing inclusive control on the fundamental computing communications. Special attention has to be taken so as to guarantee the safety of data beneath cloud computing environment.

Similar to the closed item set mining algorithms, pattern mining tracks a safeguarding and test model, i.e., it desires to preserve the set of previously mined blocked series candidates which can be utilized to reduce search space and test out if a recently established frequent sequence is capable to be blocked. But, a closed pattern mining algorithm contain poor scalability in the number of regular closed patterns since a huge number of candidates will take up much memory and guide to outsized search space for the new pattern analysis, which is typically the case when the support threshold is small or the patterns turn into long. To provide a pattern mining to the data for security purpose, in this work, a solution is presented using NSP-BLOCK algorithm that mines the entire set of frequent closed sequences.

The rest of this paper is organized as follows: Section 2 discuss the related work to pattern mining problem for storage space in cloud. Section 3 is focused on the NSPBLOCK algorithm. Section 4 present an experimental study with Amazon EC2 dataset and Section 5 describes the results comparison with the existing algorithms. Finally, the research is concluded with the conclusion in section 6.

II. LITERATURE REVIEW

Cloud computing allows vastly scalable services to be simply addicted over the Internet on an as-required basis. A major characteristic of the cloud forces is that users' data are typically practiced distantly in unidentified machines that users do not possess or work. While liking the handiness conveyed by the novel emerging expertise, users' worries of trailing control of their individual data (mainly, economic and health data) can turn out to be an important obstruction to the broad acceptance of cloud services. To deal with this problem, in [2], the author proposed a novel extremely decentralized information responsibility structure to keep track of the definite practice of the users' data in the cloud.

In [3], the author considered the trouble of conveying a set of clients with difficulties to a set of servers with capabilities and degree constraints. The objective is to discover a distribution such that the number of clients dispersed to a server is slighter than the server's amount and their general demand is lesser than the server's facility, while exploiting the general throughput.

Cloud computing, with its pledge of (almost) unrestrained calculation, storage space and bandwidth, is progressively suitable for the infrastructure of option for numerous organizations. As cloud contributions mature, service-based requests require to animatedly recompose themselves, to self-adapt to varying QoS requirements [4]. In [5], the author presented a decentralized method for such self-adaptation, by means of market-based heuristics. While this novel computing expertise needs users to commend their expensive data to cloud providers, there have been mounting security and seclusion provides on outsourced data. Numerous systems utilizing attribute-based encryption (ABE) have been planned for access control of outsourced data in cloud computing.

Since this new computing technology requires users to entrust their valuable data to cloud providers, there have been increasing security and privacy concerns on outsourced data. Several schemes employing attribute-based encryption (ABE) have been proposed for access control of outsourced data in cloud computing [6].

The privacy concerns caused by preserving intermediary datasets in cloud are significant but they are rewarded diminutive attention. Storage and computation services in cloud are counterpart from a reasonable viewpoint since they are stimulating in proportion to their procedure [6]. As a result, cloud users can amass expensive intermediary datasets selectively when giving out unusual datasets in data-intensive requests like medical diagnosis, so as to restrain the general expenses by avoiding recurrent re-computation to get hold of these datasets [12].

Officially, cloud computing is observed as an creative permutation of a sequence of tools, instituting a novel business representation by presenting IT services and employing economies of scale [7], [11]. Users in the commerce chain of cloud computing can help from this new model. Cloud customers can hoard gigantic resources venture of IT infrastructure, and focus on their own interior business [8]. As a result, numerous companies or organizations have been traveling or constructing their business into cloud. On the other hand, frequent potential customers are still cautious to take benefit of cloud due to security and privacy concerns [9], [10]. This consumes more time and expensive for storing the huge amount of data in the cloud computing environment.

To resolve the research gap, in this work, NSPBLOCK algorithm is presented to mine closed set of patterns in the cloud by managing the set of data in the cloud.

III. TESTING CLOSED SEQUENTIAL PATTERN MINING IN CLOUD USING NSPBLOCK ALGORITHM

NSP testing block presented in this work, generate nested sequential pattern by removing the unwanted memory blocks. NSP testing block efficiently mine the set of sequential patterns in the cloud storage environment to enhance the security over cloud. Amazon EC2 dataset is used where the software frequent unique events are identified from the list of software events in cloud zone. It generate nested software pattern testing to optimize the software testing process. The NSP testing block algorithm describes the concepts of memory blocks on exclusive occurrence of items in a sequential Amazon Elastic Compute Cloud (EC2) database. The block diagram of NSP testing is shown in the Fig 3.1.

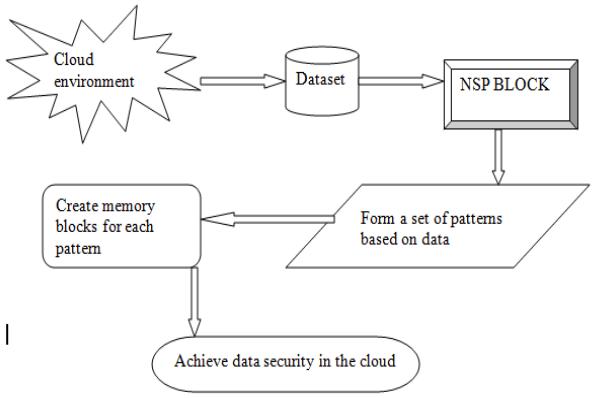

Fig 3.1 Architecture diagram of the proposed NSPBLOCK

Nested Software Pattern (NSP) block diagram in fig 3.1 describes the testing of software memory usage in cloud environment. Through the implementation of NSPBLOCK, a closed sequential pattern mining is achieved based on the creation of memory blocks. By keeping the sequential patterns regarding the data in the cloud in memory blocks, it will be easier for forming the set of closed patterns in the dataset.

3.1 System model

The network architecture for cloud data storage is illustrated in Fig. 3.2.

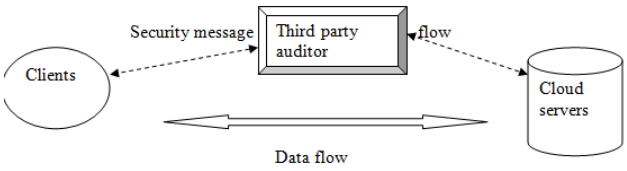

Fig 3.2 Cloud data storage architecture

Three diverse set of network units can be recognized as follows:

- Client: an individual consumers or an organization maintains huge data files to be processed in the cloud for data maintenance and computation.
- Cloud Storage Server (CSS): A computational resource for maintaining the clients' data;
- Third Party Auditor: A trusted entity, which has capabilities to render the possibility of cloud storage services on behalf of the clients.

In the cloud model, by placing the huge data files on the distant servers, the clients can be pleased of the load of storage and division. As clients do not process their locality servers for storing the data, it is of serious significance for the clients to make sure that their data are being appropriately stored and sustained. That is, clients should be prepared with definite security means so that they can occasionally authenticate the accuracy of the distant data even not including the subsistence of local copies. If the clients do not have time to check out the data, the trusted auditor will assist the clients based on their requests.

3.2 Mining closed sequential pattern in cloud

There are mainly three steps to be followed in NSPBLOCK to generate the set of closed sequential pattern they are:

- Identify the frequent unique items.
- Generate frequent patterns on projected database for each item.
- Determine closed patterns and remove them.

Before starts a mining process with the sample set of patterns in the database, the complexity of mining in the cloud is described in the subsequent steps. Let D = {d1, d2, d3, d4….. dn} be a set of distinct software events in cloud. A sequence 'P' is an ordered list of software events, denoted as {i1, i2, i3,…., im} where id is an events, i.e. id € D for 1 < d < m. From the NSP definition, it is known that software events occur multiple times in different events of a sequence as shown in Fig 3.3. The number of software events i.e., instances of software items in a nested sequence is called the length of the nested sequence and a nested sequence with a length l is also called an l-sequence, for example SSBBCA is a 6 sequence.

Let us consider a scenario with a sequence nsa= a1a2a3….an is contained in another nested sequence nsb= b1b2b3….bm if there exist integers 1 < d1 < d2< …. < dn < dm such that a1=b1d1, a2= b2d2 and so on. If nested sequence NS is contained in NS', NS is called nested subsequence of NS' and NS' is nested super-sequence of NS, denoted as NS [NS].

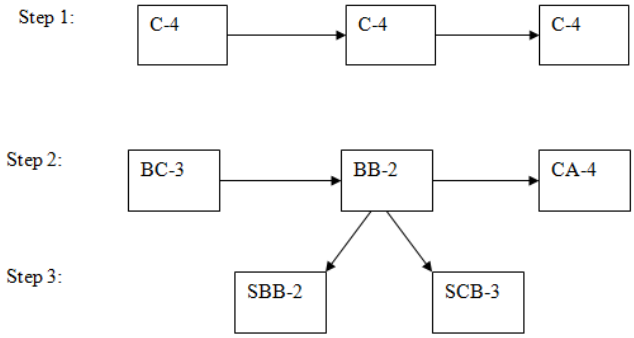

Fig 3.3 Output of Nested Software Pattern Testing

An input nested sequence database NSDB (Fig 3.3) is a set of tuples with a sequence identifier and input sequences. The number of software tuples in NSB is called base size of NSB, denoted as |NSB|. A software tuple (NSid and NS) is said to contain a sequence NS' if NS is a nested super-sequence of NS', for example in fig 3.3, software tuple (s, CSSBC) contain a software nested sequence CSS because CSSBC is a nested super sequence of CSS. The number of software tuples in NSB that contain NS' is called an absolute support of nested sequence NS in NSB.

Min_sup is a minimum software support threshold already assumed in cloud environment. A nested sequence NS is a frequent sequence on NSB if sup (NS) > min_sup. If nested sequence NS is frequent and there exist no proper nested super-sequence of NS with same support i.e. there exists no nested super-pattern ns' such that ns' ב ns, and ns' and ns have the same support it is called a nested software pattern.

TABLE 3.1 Sample Nested Software Sequence Testing

Nested Sequence identifier	Nested Sequence (NS)
1:1	CSSBC
1:2	SBAC
1:3	CSBC
1:4	SBBCA

NSP block are trying to implement the sequential software nested pattern mining by using separate memory blocks for separate software patterns like fig 3.3. Namely, 4 memory blocks for all software patterns starts with S and another block for all patterns starts with B, C and so on. On placing these sequential nested software patterns in memory blocks, finds nested pattern with unused memory easily while placing them to these blocks.

3.2.1 Identifying Software Frequent Pattern

Consider a amazon EC2 dataset, Hadoop, which provides a detailed information about the file processing systems. Now, the process of presenting the operation of read, write, update and delete block are taken as set of software events. From these set of software events, the frequent set of patterns are identified and processed.

Let us assume the frequent pattern be the sessions required to distribute the data in the cloud environment. The session software event should match the following constraint,

i) The session should appear frequently in numerous events

ii) The time taken to accomplish the session should lies minimal than the user pre-defined time (TL).

Initially all software events unique occurrences of sessions are estimated with their support are placing at top as root node or first level, recursively extend the memory boxes at level J in the flow by adding an events 'D' at top level.

- Subroutines Find unique events (D) find all unique items from database NS from fig 3.3 all unique occurrences are A B and C. The unique event occurrences of the sequential database are identified individually in cloud environment.

  That is, the unique sessions are identified from the dataset based on the unique processes of events like read, write, modify and delete block individually.
- Subroutine FindSoftwarePattern(UD, D) find all frequent software patterns in cloud where UD is the unique event generated in first step. NSP in return provide all possible frequent software patterns for one block; process be nested to find other possible frequent software patterns of different unique items generated earlier. After completion of this step, the outcome will be the set of software events with the own sequential frequent patterns.
- Since the large files of the dataset are processed in terms of blocks, each block contains a set of unique frequent sessions like allocating, read, write and delete. To form a nested software processing scheme, each block is analyzed from the set of blocks to identify the set of unique software event i.e., session based event based on TL.

The formation of frequent set of sessions from each block of the large file dataset is presented in the table below (table 3.2).

TABLE 3.2 Frequent pattern mining outcome

S. No	Frequent Sessions	Session required (sec)	Event occurrences frequency (%)
1	Allocate block, read	11	20
2	Write block	9	24
3	Modify block	15	12
4	Delete	12	3
5	Analyze block	5	9

After identification of software events like read, write, update and delete blocks, the generation of patterns is done based on the session requirement. For reading the data in the block, the event takes approximately 11 seconds to read the data, 9 seconds to write the data in the block of file. The event occurrence frequency is determined based on the occurrence of nested software events with its session range value limit.

Based on the TL, the software events are classified with the corresponding session limit. The repeatability of software events is analyzed in each block of the file is identified and removed by following the closed sequential pattern mining. Now the closed set of software events are formed based on nesting procedures without any repeatability over the information and removed from the list with no data loss. The process of NSPBLOCK events are described in the figure below,

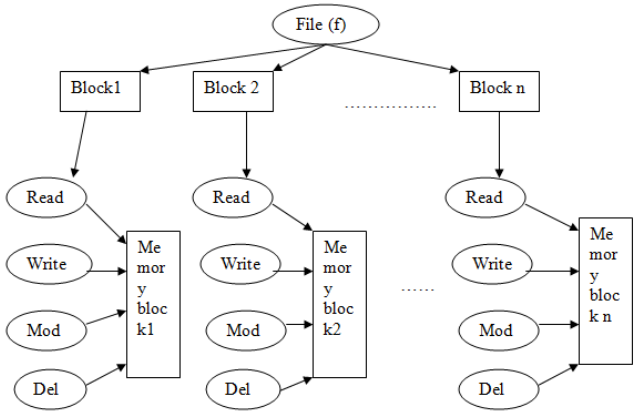

Fig 3.4 Process of NSPBLOCK

The processes of NSPBLOCK is described in fig 3.4 with the software events like read, write by maintaining the memory block simultaneously with the operations. The memory block maintains the information about the session limit, session consumed, software event, block name, file name. The memory of each of the block is specified in fig 3.5.

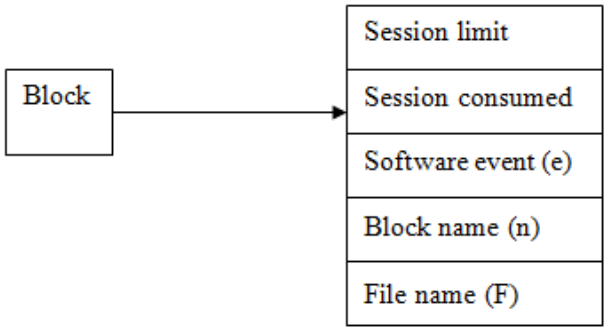

Fig 3.5 Information in memory block (stack)

With this information, a stack operation is presented. The NSPBLOCK followed the stack to provide the information about the frequent set of software events with its session limit. In each block, it maintains the information about the individual software event with its session limit. The repetition of information is identified through nested patterns which are formed based on the stack operation. Once the repetitiveness is identified, it is removed from the block. So, finally, the set of software events are maintained in the dataset with no repetitiveness in each block of the huge dataset.

To improve the performance, implements a checking for NSP in same level, software events like read, write, modify and delete operations are processed with underline. These software event operations are found nested based on the requirement of session for the processes of events so it is removed from the block to reduce memory and in step 2 will get all sequential nested software patterns of all blocks. Further steps are taken as sub process of step 2 in which checking is performed for finding the software frequent item sets which may occur in other blocks like block2, block 3 and block n are repeated in Block 2 and 4 so are closed and removed. Finally after reaching last step, will get all frequent software patterns. From Fig 3.3 found an output of nested software pattern testing for first extracted software event i.e. 'read (allocating data)'. The algorithm below describes the process of NSPBLOCK process with the set of software events.

// NSPBLOCK Algorithm
Begin
Input: Software Sequence Events, NSB
//NSUnique (NSB, Vns)
1: Vns=θ; select distinct software item D in NSB;
2: Vns=D;
3: Return Vns;
//Software Frequent Pattern Block(NSb,min_sup,Vns,SFP)
4: For each unique software item D in Vns
5: Identify Callstack(D,nst1,nst2,NSB,Vns,NSto)
6: Store nsto elements in SFP
7: Return SFP
8: End For
// Callstack(D, nst1,nst2,NSB,Vns,nsto)
9: Create length (Vns)node for nst1 and nst2 and store D on all of them
10: For (no p from 1 to no of length (Vns)) do
11: Find all software event combination upto p+1 for all Item starts with D
12: Check if min_sup is same as previous stack
13: If (true)
14: Replace previous with new one
15: Else
16: create a new node and store software item in cloud
17: Return Cloud nst1
18: End If
End
Output: Complete Unique set of events

For huge software frequent patterns in cloud, the length of frequent nested software patterns is also large. The study shows it is true that nested software pattern has better expressive power and frequent software pattern testing does have a better efficiency with same analytical result. NSP algorithm test software frequent sequences in cloud with the help of stack as memory blocks and then software sequential patterns were generated in cloud to address the need of software engineers in cloud environment software testing.

IV. EXPERIMENTAL EVALUATION

An experimental evaluation is carried out with the Amazon EC2 dataset to estimate the performance of the proposed NSP BLOCK algorithm. Amazon Elastic Compute Cloud (Amazon EC2) presents resizable calculating capability in the Amazon Web Services (AWS) cloud. Using Amazon EC2 eradicates the requirement to spend in hardware up front, so that it is easy to enlarge and organize applications faster. Amazon EC2 allows scaling up or down to hold changes in necessities or points in recognition, dropping your need to estimate traffic.

Amazon EC2 provides a broad collection of instance types optimized on top form diverse use cases. Instance types encompass unreliable mixtures of memory, CPU, storage, and networking capacity and offer the litheness to decide the suitable mix of resources for the required applications. Every instance type comprises one or more instance ranges, permitting to level the resources to the necessities of the target workload.

Hadoop is an open source accomplishment of the construction for large-scale parallel data processing. Hadoop is fame in both systems research and data mining, so it is significant to appreciate its runtime activities, pattern formation and analyze its performance against the PADD systems. To compare our online approach directly against the PADD systems proposed in [1], set of logs are considered, which has over 20 million log messages with an uncompressed size of 2.3GB. The logs were produced from the set of nodes approximately 200 organizing Hadoop for 48 hours. The machine load differs from the work it precedes. The log consists of 575,319 event traces, with 575,319 dissimilar file blocks in Hadoop File. The performance of the proposed NSP BLCOK algorithm is measured in terms of pattern prediction, run time and scalability.

V. RESULTS AND DISCUSSION

In this section (section 5), the performance of the proposed NSPBLOCK is evaluated against Public Auditability and Data Dynamics system [PADD]. The below table and graph describes the evaluation results.

TABLE 5.1 No. of sequences vs. pattern prediction

No. of sequences	Pattern prediction (%)	
	Proposed NSPBLOCK	Existing PADD
5	23	10
10	29	13
15	35	19
20	40	23
25	46	27
30	57	35
35	65	40

The prediction of formation of closed set of patterns from the set of sequences formed from the dataset is described in the table 5.1. The value of the proposed NSPBLOCK is compared with the existing Public Auditability and Data Dynamics system [PADD].

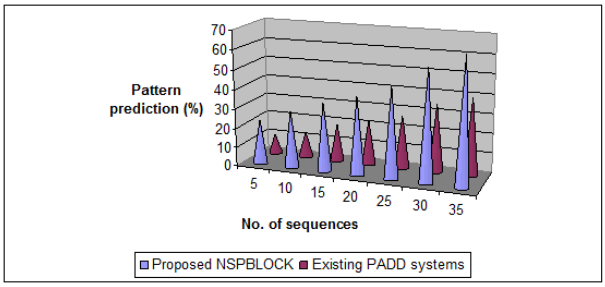

Fig 5.1 No. of sequences vs. pattern prediction

Fig 5.1 describes the prediction of formation of closed set of patterns from the set of sequences formed from the dataset. Compared to the existing PADD, the proposed NSPBLOCK provides high rate in predicting the formation of patterns. Because, the NSPBLOCK supports mining the closed set of sequences based on the formation of memory blocks i.e., stack to store in it. So, the formed set of patterns are available in all the time by forming the forward attribute selection. The variance in the proposed NSPBLOCK provides 15-20% high in predicting the set of patterns from the dataset.

TABLE 5.2 No. of records vs. running time

No. of records	Running time (sec)	
	Proposed NSPBLOCK	Existing PADD
100	75	100
200	100	120
300	130	160
400	150	180
500	200	230
600	250	270
700	320	350

The running time is measured for forming the closed set of patterns from the sequences based on the number of records is described in the table 5.2. The value of the proposed NSPBLOCK is compared with the existing Public Auditability and Data Dynamics system [PADD].

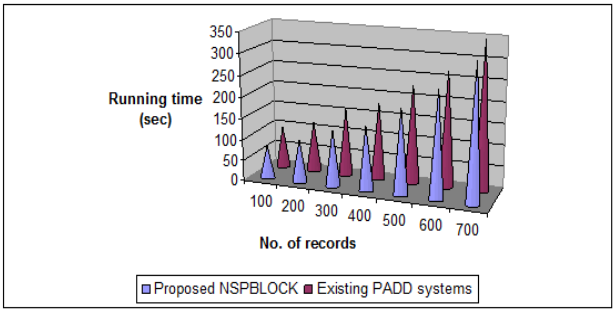

Fig 5.2 No. of records vs. running time

Fig 5.2 describes the running time is measured for forming the closed set of patterns from the sequences based on the number of records. Compared to the existing PADD, the proposed NSPBLOCK consumes less execution time in the formation of patterns. Because the NSPBLOCK forms the closed set of patterns from the given dataset, so the repeatability of patterns in the dataset is less . But in the existing PADD systems, the output patterns are repeated regularly and the process of identifying the sequential patterns for data storage. So, the consumption of processing the records in the existing PADD systems is high. The variance in the execution time is 10-13% less in the proposed NSPBLOCK.

TABLE 5.3 No. of patterns vs. scalability

No. of patterns	Scalability (%)	
	Proposed NSPBLOCK	Existing PADD
10	75	100
20	100	120
30	130	160
40	150	180
50	200	230

The scalability is measured based on the number of patterns formed from the set of sequences. The value of the proposed NSPBLOCK is compared with the existing Public Auditability and Data Dynamics system [PADD] is illustrated in table 5.3.

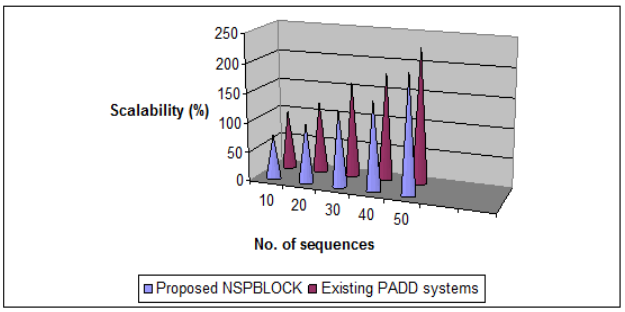

Fig 5.3 No. of patterns vs. scalability

Fig 5.3 describes the scalability which is measured based on the number of patterns formed from the set of sequences. Compared to the existing PADD, the proposed NSPBLOCK provides reliable scalability in mining the closed set of patterns. Since the proposed NSPBLOCK consumes less time and formed the set of closed patterns without any repeatability and data loss, the scalability is high in it.

Finally, for vast frequent set of patterns where the time taken of forming the frequent closed patterns is also huge, the proposed NSPBLOCK shows that the closed pattern mining has enhanced communicative power with less time consumption as that of PADD systems i.e., frequent pattern mining.

VI. CONCLUSION

The issues of mining closed sequential patterns in the cloud environment are investigated in this paper and it is resolved by implementing the NSP BLOCK algorithm. NSP BLOCK algorithm is formulated efficiently to mine the frequent closed sequences of software events of the cloud storage management. In this paper, NSP BLOCK algorithm is presented for mining closed set of sequences which occurred frequently by assisting stack as memory blocks and then sequential patterns were produced to the closed patterns. NSP BLOCK outperforms Public Auditability and Data Dynamics system by mining the longer set of frequent sequences in the given dataset with low minimum support with no loss of information. Experimental evaluation is carried out with Amazon EC2 dataset to estimate the performance of the proposed NSP BLOCK algorithm against Public Auditability and Data Dynamics system. Performance results revealed that the proposed NSP BLOCK consumes less time in forming a set of software patterns with 16% accuracy in predicting the set of patterns compared to the existing scheme.

REFERENCES

[1] Qian Wang., Cong Wang., Kui Ren., Wenjing Lou., and Jin Li., "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing," IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 22, NO. 5, MAY 2011

[2] Smitha Sundareswaran., Anna C. Squicciarini., and Dan Lin., "Ensuring Distributed Accountability for Data Sharing in the Cloud," IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 9, NO. 4, JULY/AUGUST 2012

[3] Olivier Beaumont., Lionel Eyraud-Dubois., and Hejer Rejeb., "Heterogeneous Resource Allocation under Degree Constraints," IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS., 2012

[4] Vivek Nallur., Rami Bahsoon., "A Decentralized Self-Adaptation Mechanism For Service-Based Applications in The Cloud," IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, 2012

[5] Zhiguo Wan., Jun'e Liu., and Robert H. Deng., "HASBE: A Hierarchical Attribute-Based Solution for Flexible and Scalable Access Control in Cloud Computing," IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 7, NO. 2, APRIL 2012

[6] Xuyun Zhang., Chang Liu., Surya Nepal., Suraj Pandey., Jinjun Chen., "A Privacy Leakage Upper-bound Constraint based Approach for Cost-effective Privacy Preserving of Intermediate Datasets in Cloud," IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, 2012

[7] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica and M. Zaharia, "A View of Cloud Computing," Commun. ACM, vol. 53, no. 4, pp. 50-58, 2010.

[8] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg and I. Brandic, "Cloud Computing and Emerging It Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility," Fut. Gener. Comput. Syst., vol. 25, no. 6, pp. 599-616, 2009.

[9] L. Wang, J. Zhan, W. Shi and Y. Liang, "In Cloud, Can Scientific Communities Benefit from the Economies of Scale?," IEEE Trans. Parallel Distrib. Syst., vol. 23, no. 2, pp. 296-303, 2012.

[10] H. Takabi, J.B.D. Joshi and G. Ahn, "Security and Privacy Callenges in Cloud Computing Environments," IEEE Security & Privacy, vol. 8, no. 6, pp. 24-31, 2010.

[11] D. Zissis and D. Lekkas, "Addressing Cloud Computing Security [12] D. Yuan, Y. Yang, X. Liu and J. Chen, "On-Demand Minimum Cost Benchmarking for Intermediate Dataset Storage in Scientific Cloud Workflow Systems," J. Parallel Distrib. Comput., vol. 71, no. 2, pp. 316-332, 2011.Issues," Fut. Gener. Comput. Syst., vol. 28, no. 3, pp. 583-592, 2011.

[12] D. Yuan, Y. Yang, X. Liu and J. Chen, "On-Demand Minimum Cost Benchmarking for Intermediate Dataset Storage in Scientific Cloud Workflow Systems," J. Parallel Distrib. Comput., vol. 71, no. 2, pp. 316-332, 2011.