



International Journal of Web Engineering and Technology

ISSN online: 1741-9212 - ISSN print: 1476-1289

<https://www.inderscience.com/ijwet>

Cloud storage framework for multivariate regression-based data mining: optimised LIFR and FIFR model

Saswati Sarkar, Anirban Kundu

DOI: [10.1504/IJWET.2025.10070076](https://doi.org/10.1504/IJWET.2025.10070076)

Article History:

Received:	14 December 2023
Last revised:	10 May 2024
Accepted:	17 November 2024
Published online:	02 April 2025

Cloud storage framework for multivariate regression-based data mining: optimised LIFR and FIFR model

Saswati Sarkar*

Computer Innovative Research Society,

West Bengal – 711103, India

Email: ssam.saswati@gmail.com

*Corresponding author

Anirban Kundu

Computer Innovative Research Society,

West Bengal – 711103, India

and

Netaji Subhash Engineering College,

Kolkata – 700152, India

Email: anik76in@gmail.com

Abstract: Authors propose a cloud-based storage framework to search data using optimised mapping. Multivariate regression-based data mining mechanism shows the utility of map tables, storage partitions, and data storage in cloud. The paper exhibits complexity of cloud-based searching algorithms in real-time scenario. The proposed cloud-based storage framework represents parallel disk searching technique to search data with less time consumption. Proposed technique exhibits linear time complexity. Drive partition concepts and map table concepts have been incorporated in this work for searching data in less time. It is observed that the overall execution time is inversely proportional to the number of drive partitions. Sequential and several parallel situations exhibit the comparisons using time graphs. The paper presents comparison graphs for execution time and time complexity between existing techniques and proposed approach with respect to storage partitions. Experimental observations and analysis using statistical data have been shown in this paper.

Keywords: optimised mapping; last in first read search; LIFR; first in first read search; FIFR; cloud storage; redundant array of independent drives; RAID5; map table; multivariate regression.

Reference to this paper should be made as follows: Sarkar, S. and Kundu, A. (2025) 'Cloud storage framework for multivariate regression-based data mining: optimised LIFR and FIFR model', *Int. J. Web Engineering and Technology*, Vol. 20, No. 1, pp.66–99.

Biographical notes: Saswati Sarkar is a PhD scholar in CSE in Computer Innovative Research Society (<http://www.cirs.co.in>), West Bengal, India. She has four international journal publication (published), two international book chapter publication (published), and five international conference publication (published). From September 2009–August 2018, she worked as an Assistant Professor of Information Technology Department at Adamas Institute of

Technology, West Bengal, India. From August 2008–September 2009, she was a Lecturer of Computer Science Department at Institute of Technology and Marine Engineering. From March 2008–August 2008, she worked as a Lecturer of Computer Science Department at Institute of Science and Technology. She has ten years' experience in teaching field.

Anirban Kundu is a Professor of Netaji Subhash Engineering College, Kolkata, India. He has been associated with Netaji Subhash Engineering College since April 2004. He worked as a Post-Doctoral Research Fellow at Kuang-Chi Institute of Advanced Technology, Shenzhen, China. He worked as a Foreign Expert in Information Science under Municipality Government of Shenzhen during 2011 to 2014. He was a Research Fellow in WIDiCoReL. He received his BE (Mech), BU (1999), PGD (FM), MSPI (2001), MTech (IT), IEST, Shibpur (formerly BESU) (2004), PhD (Engineering) (CSE), and JU (2009). His research interests include PR, ML, AI, SEO, WS, DC, CC, HPC, CA, IoT, multi-agent, fuzzy, storage, smart traffic, cryptography, and cybersecurity.

1 Introduction

1.1 Overview

Searching (Zhang, 2013) is one of the fundamental operations in computer science. Searching is a technique to fetch data from storage. There are different types of searching algorithms available in computer science, such as divide and conquer search, linear search, breadth first search, depth first search, binary search (Silberschatz et al., 2009; Mano, 2009). Different types of data structures are available in computer science, such as stack data structure, queue data structure, and so on. The main difference between stack and queue is stack uses last in first out (LIFO) method to store and access data, and queue uses first in first out (FIFO) technique to store and retrieve data. Stack and queue data structure are non-primitive data structures.

Cloud is a terminology used in network-based computing over the internet. Cloud computing is a model for on demand availability of resources for computer systems. Cloud computing services are divided into three categories, such as infrastructure-as-a-service (IaaS), platform-as-a-service (PaaS), and software-as-a-service (SaaS) (Curino et al., 2020; Nazir, 2012).

Cloud storage (https://electricalfundablog.com/cloud-storage-architecture-types/#Free_Cloud_Storage; Thomasian and Blaum, 2009) is a part of storage concept offering opportunities to fulfil demand of daily increasing data volumes. It allows organisations or companies to meet demand if budgets are decreasing or freezing. There are different kinds of cloud storage solutions available, such as object storage system, distributed data storage system, relational database storage system (Buyya, 2013).

Cloud storage architecture (Reddy et al., 2011; Chan et al., 2009) is scalable and multi-tenanted architecture. This architecture is scalable and high performance architecture (Gonzalez et al., 2012; Kusnetzky, 2011). The advantages of this framework are data protection capability and availability of resources. This is easy to manage and secured storage system. Since cloud storage architecture is related to distributed system architecture, facilities are designed as a distributed system (HPC, 2011).

1.2 *Aim*

Authors aim to develop a cloud storage multivariate regression-based framework for searching data from cloud disk storage in parallel and/or concurrent manner. Following are the major factors to be considered in this research:

- search data using map tables (MTs)
- last in first read (LIFR) searching technique
- first in first read (FIFR) searching technique.

1.3 *Scope*

The scope of this research is knowledge discovery in database within cloud, which consists of time complexity calculation, and execution time representation.

1.4 *Motivation*

To design a regression-based cloud storage system with optimisation for searching data in less time, and to have a storage analysis exhibiting time optimisation in a linear fashion with proper connectivity between drive partitions in reference of MTs.

1.5 *Challenges*

Main challenges of proposed multivariate regression-based cloud storage framework are to provide better searching technique compared to existing approaches to achieve less time complexity and usage MTs to detect drive's location of storage.

1.6 *Organisation*

Rest of the paper is organised as follows: storage framework preliminaries have been discussed in Section 2; design framework for cloud storage using regression is discussed in Section 3; multivariate data mining using MTs is discussed in Section 4; Section 5 shows analytical derivations on storage; Section 6 shows theoretical discussions for time optimisation; theoretical discussions on drive partitions and its mapping covered in Section 7; experimental discussions are depicted in Section 8; comparative analysis is shown in Section 9; Subsection 9.2 shows experimental analysis using statistical data; finally Section 10 concludes our research work in this paper.

2 **Related work**

2.1 *Literature review*

Storage architecture (Heger, 2011) of any system is a critical component of data transfer and accessing information. There are four keys for storage architectures (Wang et al., 2023) such as server attached storage (SAS), network attached storage (NAS) (Bender

et al., 2018; Liu et al., 2018), storage attached network (SAN), and tape libraries (Shuaiqing et al., 2020; Richa et al., 2019).

SAS (Wang et al., 2012; Rawat et al., 2014) is a most common type of storage. In this storage, all the disks and devices used to store data are installed in the server, rather than the separate devices (Shah et al., 2011; Paola et al., 2020). This technique is suitable for small servers that can access data from several computers. This type of storage is cost effective. NAS is a standard storage technology for centralised system for office or home. Redundant array of independent drives (RAID) (Huang and Wang, 2018) architecture uses NAS concept. SAN is an advanced storage network. This storage network (Xie et al., 2012; Bairavasundaram et al., 2007) can encompass one or more storage devices. These storage network (Carns et al., 2009; Shuaiqing et al., 2020) are powerful networks as to control their storage costs for allowing businesses using different hardware into single logical unit (Hou et al., 2016; Han et al., 2006). Tape libraries are efficient options to store backup files, data, and information.

RAID (Tian et al., 2011; Dholakia et al., 2008) or redundant arrays of independent disk is a technique for increasing the performance of storage using combination of multiple disks instead of using single disk. This technique increases data redundancy of storage. Data redundancy takes extra storage space and adds disk reliability. If one disk fails, then same data is backed up into another disk. It is possible to retrieve data and proceed with the specific operation. Key points of RAID system are availability, reliability, performance, and capacity of storage. This architecture is transparent as it appears as a single disk presenting as an array. Different RAID (Gibson, 2007; Jiang et al., 2008) levels are RAID0, RAID1, RAID4, RAID5, RAID10, etc.

RAID0 architecture uses stripping technique to store data. In this architecture, minimum two disks are used to store data. The efficiency of this technique is 100%. In RAID0 architecture, write performance is excellent. Sequential and random methods are used to read data from storage. Read performance is good for sequential and random methods. The cost of RAID0 architecture is low. The reliability of RAID0 architecture is 0 (Plank et al., 2013).

RAID1 architecture uses mirroring technique to store data. Minimum two disks are used to store in this architecture. The efficiency of this technique is 50%. The cost of RAID1 architecture is high. Write speed of this architecture is same as the write speed of single disk. Each write operation should be committed to all disks in this technique. Mirror protection is used in this technique to protect data (Shu, 2007).

RAID5 architecture uses minimum three disks to store data. The efficiency of this technique is 75%. The cost of RAID5 architecture is moderate to low. Writing performance of this architecture is fair. Random and sequential techniques are used to store data in this architecture. Read performance is good for sequential and random techniques. Parity protection is used in this technique to protect data (Thomasian, 2005; Zhang et al., 2018).

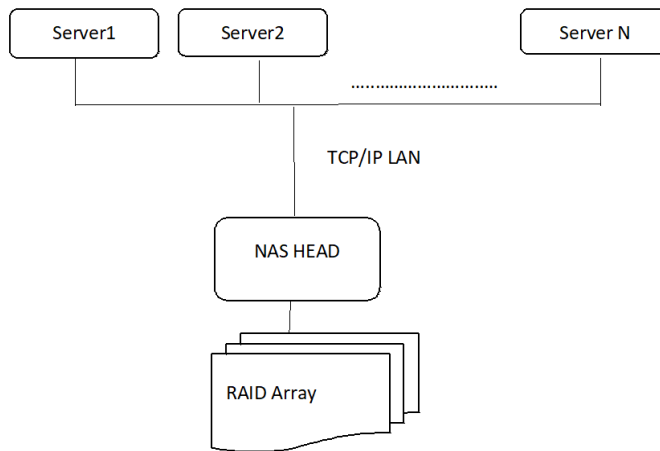
RAID10 architecture uses minimum four disks to store data. The efficiency of this technique is 50%. The cost of RAD10 is high. Writing performance of this architecture is fair. Read and write performances are good in this technique. Mirror protection is used in this technique to protect data. This architecture is the combination of RAID0 and RAID1 architecture.

The principle of multivariate analysis (MVA) is depending upon multivariate statistics (Tian et al., 2011). MVA is mainly applied for multiple measurements for each

experiment. Relationship between each measurement is important in MVA (Kumar and Sahoo, 2014) for analysis. It includes distribution theory, measurement of relationship, multivariate model, probability computations, etc. Multivariate regression (Villalpando et al., 2014; Timmons et al., 2016) is a technique where multiple data or variables are involved for analysis (Wang et al., 2023). It is an extension of multiple regressions for one dependent variable and one independent variable or more than one independent variable (Zhou et al., 2021; Zhao et al., 2022).

A device (storage) (Shuaiqing et al., 2020; Richa et al., 2019) attached directly to the network is called NAS. Network file system is one protocol in NAS architecture (Wang et al., 2012). NAS is used to store different types of data over the network. This architecture is capable to access different types of data from network such as Storage Network Industry Association (SNIA).

Figure 1 Existing NAS architecture



NAS has fixed and predefined number of hard disks. RAID array (Rawat et al., 2014) is connected to the network which is magnetic tape drive. NAS has own IP address for RAID configuration (Shah et al., 2011; Paola et al., 2020). NAS head is a translator between data level NAS protocols (NFS, CIFS, etc.) and block level SAN protocols (fibre channel). NAS head is a NAS architecture which does not have any on-board storage. RAID arrays are independent hard disks which are redundantly aligned. If any hard disk fails in the NAS device, then RAID arrays are used as a backup hard disk.

2.2 Definitions

Definition 1: MT – MT is an index which represents the direction to search data from a particular storage partition.

Definition 2: RAID – redundant array independent disk (RAID) is a storage technology that combined multiple disks into one logical unit. The purpose of RAID architecture is to improve the performance of storage. There are different types of RAID architectures such as RAID0, RAID1, RAID10, RAID5, etc.

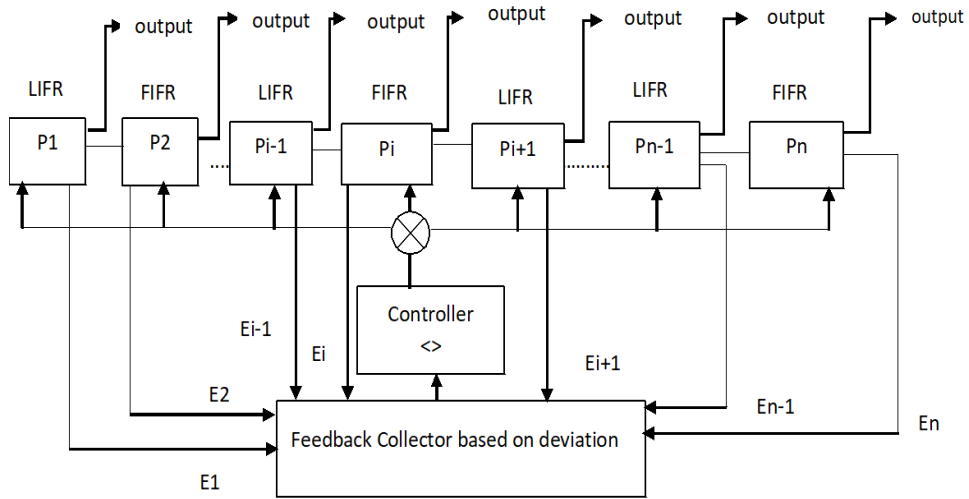
Definition 3: Data-based storage – data-based storage (Xie et al., 2012) is also called data storage in which data is stored and retrieved as a specific format. It stores data in a hierarchical manner. Data can be accessed using NFS protocol within the system framework.

Definition 4: Multivariate regression – multivariate regression is an extension of linear regression used to measure degree of variable based on the value of more than one variable (Reinaldo and Adelchi, 2021; Zhiyang, 2021).

3 Design framework for cloud storage using regression

In this section, proposed multivariate regression-based design architecture and related structural design are discussed as an initial impression. Proposed searching technique [Naive pattern searching (NPS)] is used maintaining distinct odd ($P_1, P_3, \dots, P_{k-1}$) and even partitions ($P_2, P_4, \dots, P_k$), and MTs as per the index.

Figure 2 Architecture of LIFR and FIFR

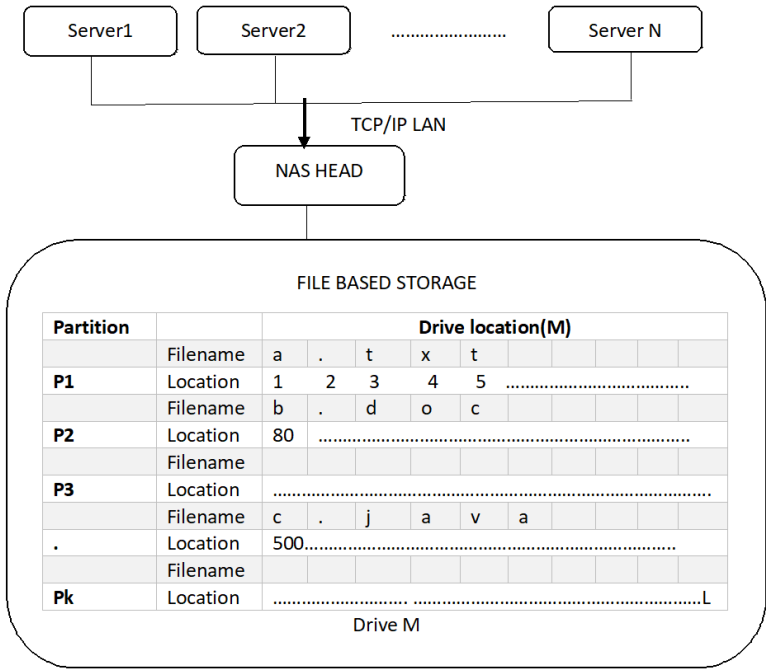


In Figure 2, multiple partitions ($P_1, P_2, \dots, P_n$) exist in proposed architecture. A controller is used to control searching in all partitions. Proposed parallel searching techniques (NPS(LIFR(last in first read))) and (NPS(FIFR(first in first read))) are applied to produce output. Feedback collector is used to collect individual deviation from each partition. The difference between expected search output and actual search output is termed as error ($E_1, E_2, \dots, E_n$), which is further considered as deviation in feedback collector.

In proposed framework, total drive is denoted by drive M (in Figure 3) where proposed searching technique has been applied. Total data (N_F) is stored in drive ' M ' using linear data structure method (refer Figure 1). Searching technique uses stack (NPS(LIFR)) and queue (NPS(FIFR)) algorithms combining with two-way pointers for searching particular data. Drive ' M ' is divided into ' k ' number of partitions and searching technique is applied in each partition using parallel fashion.

Searching technique
Partition $P1 = P1[0] \text{ to } P1[n] \rightarrow \text{NPS}(\text{LIFR}(P1[n] \text{ to } P1[0]))$
Searching technique
Partition $P2 = P2[0] \text{ to } P2[n] \rightarrow \text{NPS}(\text{FIFR}(P2[0] \text{ to } P2[n]))$
...
Searching technique
Partition $Pk - 1 = Pk - 1[0] \text{ to } Pk - 1[n] \rightarrow \text{NPS}(\text{LIFR}(Pk - 1[n] \text{ to } Pk - 1[0]))$
Searching technique
Partition $Pk = Pk[0] \text{ to } Pk[n] \rightarrow \text{NPS}(\text{FIFR}(Pk[0] \text{ to } Pk[n]))$

Figure 3 Storage structure used in proposed work



Structural design of proposed regression-based system has been depicted in Figure 4 indexing module creates index for MTs. Drive 'M' is divided into 'k' number of partitions. Partitions are divided into two parts (even and odd) depending on partition numbers. MTs contain partitions of drive 'M'. MTs determine the partition selected for finding a particular data (N_D) from a distinct drive location. Even partitions apply queue data structure along with two-way pointers.

$$[P2, P4, \dots, Pn] \Leftrightarrow (\text{NPS}(\text{FIFR}(N_D))); \tag{1}$$

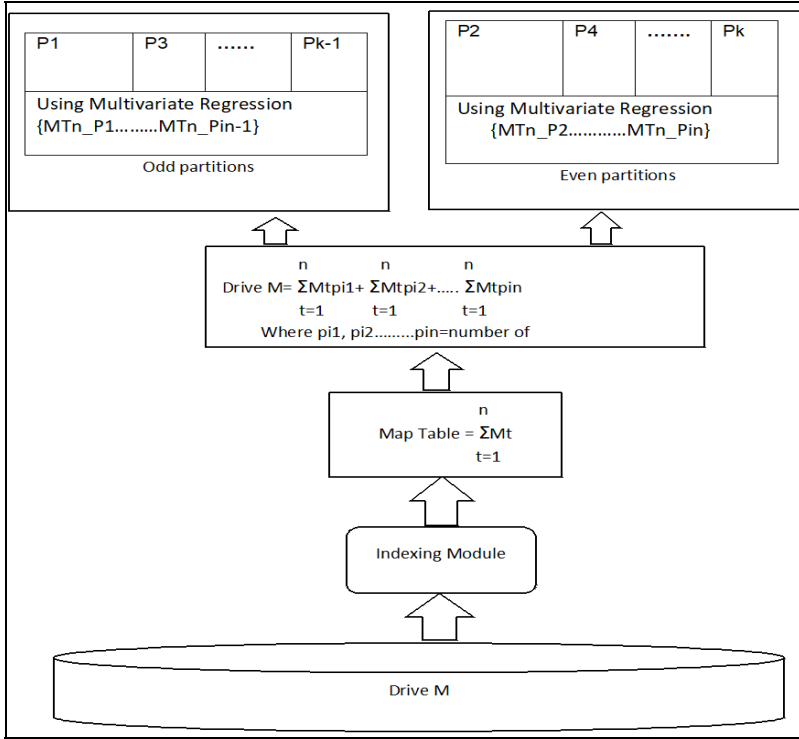
Equation (1) has been applied for searching data or file using queue data structure from even partitions using two way pointers. FIFR algorithm has been applied in case of even partitions $[P2, P4, \dots, Pn]$ for searching data using multivariate regression.

Odd partitions apply stack data structure along with two-way pointers

$$[P1, P3, \dots, Pn-1] \Leftrightarrow (NPS(LIFR(N_D))); \quad (2)$$

Equation (2) has been applied for searching data or file using stack data structure from odd partitions using two way pointers. LIFR algorithm has been applied in case of odd partitions $[P1, P3, \dots, Pn-1]$ for searching data using multivariate regression.

Figure 4 Data flow in proposed multi-tenant storage framework



4 Multivariate data mining using MTs

The multivariate data mining technique finds a particular data from total number of search data in parallel manner.

From equation (1) and (2),

$$Search_{Result(SR)} = Parallel_Search \left(NPS \left(LIFR | FIFR (P1, P2, \dots, Pk(N_D)) \right) \right) \quad (3)$$

Equation (3) represents search result using parallel search technique. Either LIFR or FIFR searching technique has been applied to achieve specific search results.

Total drive in cloud is divided into sub-drives (called partitions $(P1, P2, \dots, Pk)$) and required data is being searched in parallel manner for optimising time complexity.

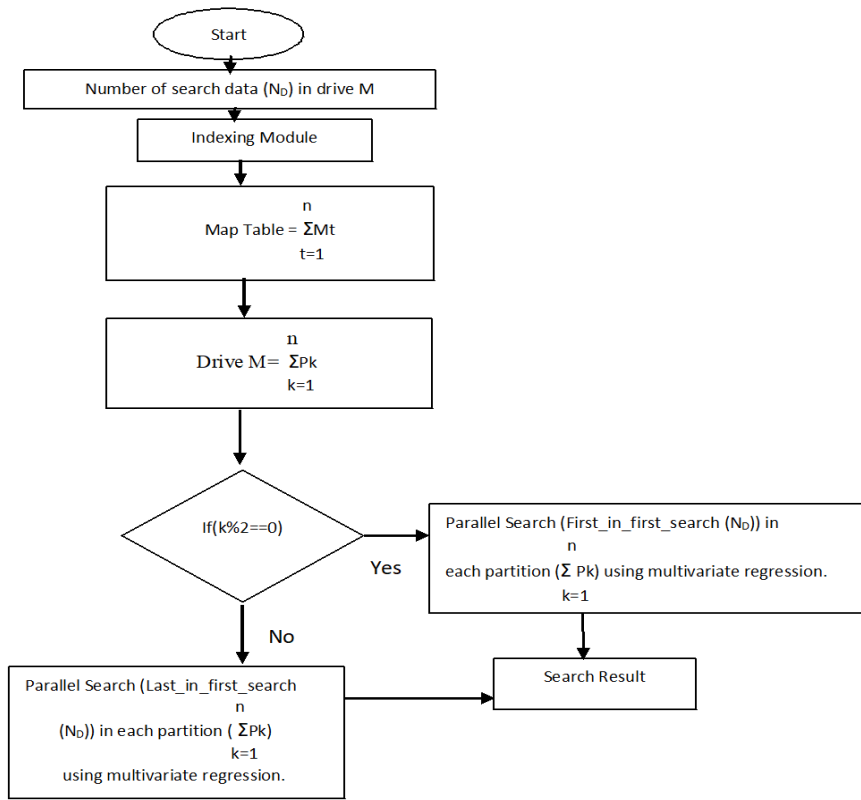
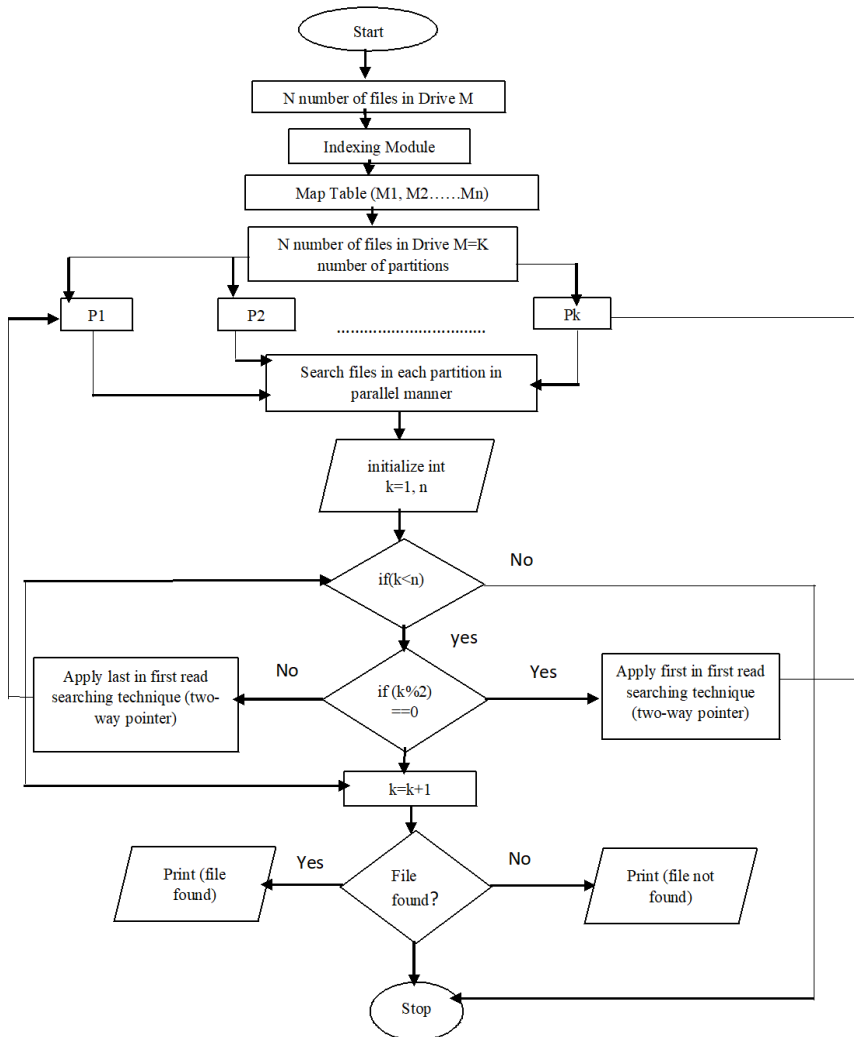
Figure 5 Framework for cloud disk search using parallel partitions

Figure 5 represents multivariate regression-based framework of parallel partitions for cloud disk search. Total number of elements (N_F) is divided into different partitions ($P_1, P_2, \dots, P_n$), and partitions are selected using MTs. MTs are used to select drive partitions where incoming elements (files) have to be stored. Cloud searching has been applied in parallel manner to find out an optimum solution regarding performance of the system framework. All search results [using equation (3)] are accumulated after searching each partition. In Figure 5, total number of data is represented by N_F and total number of partitions is represented by P_k . Elements are divided and further forwarded to different partitions for parallel disk searching.

$$\begin{aligned}
 P1_Elements &= N_F / P2_Elements + P3_Elements \dots Pk_Elements \\
 P2_Elements &= N_F / P1_Elements + P3_Elements \dots Pk_Elements \\
 &\dots \\
 Pk_Elements &= N_F / P1_Elements + P2_Elements \dots Pk-1_Elements \\
 \\
 P1_SR &= (NPS(LIFR(P1_Elements))) \\
 P2_SR &= (NPS(FIFR(P2_Elements))) \\
 &\dots \\
 Pk_SR &= (NPS(FIFR(Pk_Elements)))
 \end{aligned}$$

Figure 6 Detailed flowchart for proposed framework



If data exists, then search result would be produced.

$$SR = (P1_SR \| P2_SR \| \dots Pk_SR) \quad (4)$$

Equation (4) represents the search results of any partitions, if data or file is being found. SR is represented as search results. Proposed algorithm is applied on each partitions in parallel manner to find out searching file or data.

Figure 6 represents a workflow diagram of data mining technique in cloud. Total cloud-based drive contains 'N' number of elements. According to proposed approach, 'M' drive is divided into 'k' number of partitions. Cloud-based data mining procedure is applied in each partition in parallel manner. According to flowchart, if partition number is even [Peven] $\Leftrightarrow$ (NPS(FIFR(N_D))), then FIFR along with two-way pointers searching technique is applied for searching particular data [using equation (4)]. If partition number is odd [Podd] $\Leftrightarrow$ (NPS(LIFR(N_D))), then LIFR along with two-way pointers searching technique is applied for searching data [using equation (4)].

Cloud-based searching technique has been proposed using Algorithm 1.

Algorithm 1 Parallel_search

Input: 'n' number of data

Output: Find specific data

Step 1 Drive $M = \sum_{k=1}^n P_k$

Step 2 if $((k\%2) == 0)$

 First_in_first_parallel_search(data)

 else

 Last_in_first_parallel_search(data)

Step 3 Print 'Search Result'

Step 4 Stop

In Algorithm 1, 'n' number of data is considered as input dataset. In step 1, total drive (M) is considered as equal to 'k' number of partitions. If partitions are represented by P1, P2, P3, ..., Pn. Then $P_k = P_1 + P_2 + P_3 \dots P_n$ or $\sum_{k=1}^n P_k$. In step 2, check for odd or even partition number. If partition number is odd, then data has been searched using first in first search technique (refer Algorithm 2). If partition number is even, then data has been searched using first in first search technique (refer Algorithm 3). If data exists, then print the search results [using equation (3)].

Algorithm 2 Last_in_first_search

Input: Drive partitions (P_k)

Output: Search data (N_D)

Step 1 initialise top = First_index_of_partition [P1, P3, ..., Pk-1]

Step 2 initialise front = First_index_of_partition [P1, P3, ..., Pk-1]

Step 3 top $\leftarrow$ top + 1

Step 4 $N_D = \text{Parallel_Search}(\text{NPS}(\text{LIFR}(P1[\text{top}] \Leftrightarrow P1[\text{front}], P3[\text{top}] \Leftrightarrow P3[\text{front}], \dots, P_{k-1}[\text{top}] \Leftrightarrow P_{k-1}[\text{front}]))$

Step 5 Stop

In Algorithm 2, partitions [P1, P3, Pk-1] are considered as input. In step 1 and in step 2, variables 'top' and 'front' are initialised by first index number of partitions. In step 3, top is incremented by 1 for incoming new data. Finally parallel searching method [using equations (1) and (2)] with two-way direction has been initiated from partition [top] to partition [front]. LIFR with Native pattern searching has applied in each odd (Po) partitions in drive M to search data.

Algorithm 3 First_in_first_search

Input: Storage partitions (Pk)

Output: Search data (Nd)

Step 1 initialise top = First_index_of_partition[P2, P4, ..., Pk]

Step 2 initialise front = top

Step 3 top <- top + 1

Step 4 $N_D = \text{Parallel_Search}(\text{NPS}(\text{FIFR}(\text{P2}[\text{front}] \Leftrightarrow \text{P2}[\text{top}], \text{P4}[\text{front}] \Leftrightarrow \text{P3}[\text{top}], \dots, \text{Pk}[\text{front}] \Leftrightarrow \text{Pk}[\text{top}]))$

Step 5 Stop

In Algorithm 3, Searching technique has initiated in even partitions (PE) in drive M. partitions [P2, P4, ..., Pk] are considered as input. In step 1 and in step 2, variables 'top' and 'front' are initialised by first index number of even partitions. In step 3, top is incremented by 1 for incoming new data. Then searching method has been initiated from partition [front] to partition [top] in parallel manner with two-way pointers.

Figure 7 Drive partition formulation on data update

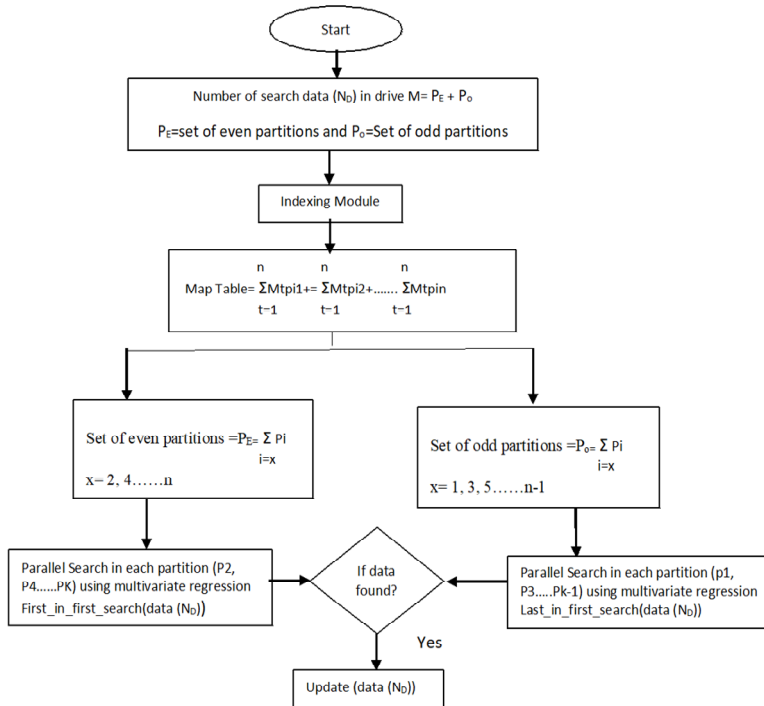


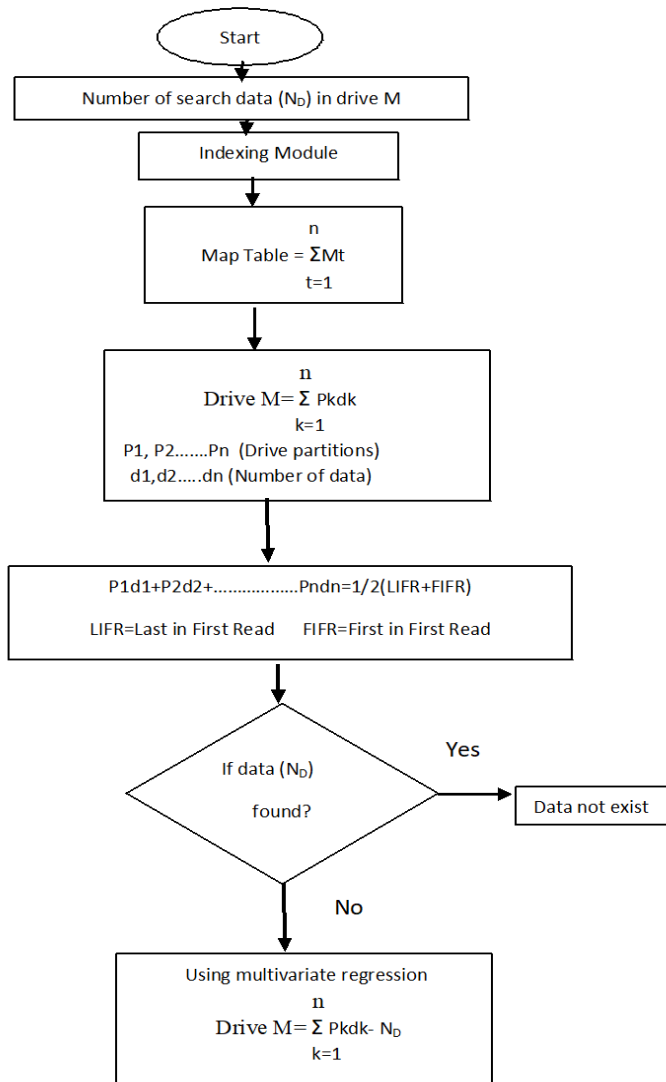
Figure 8 Formulation-based secure deletion

Figure 7 represents the workflow diagram for updating particular data. Drive M is divided into two partitions: even and odd. Indexing module maintains the index of MTs. If, it is required to edit, or to update data, then proposed multivariate regression-based parallel searching technique has been applied for searching particular data. NPS (FIFR) searching technique has applied in even partitions [P2, P4, Pk]. NPS (LIFR) searching technique has been applied in odd partitions [P1, P3, Pk-1].

Therefore, search result (SR) is represented as follows:

$$SR = \text{Parallel_Search} \left[\left(\text{NPS} \left(\text{LIFR} \left(P_{\text{odd}}(N_F) \right) \right) \right) \parallel \left(\text{NPS} \left(\text{FIFR} \left(P_{\text{even}}(N_F) \right) \right) \right) \right] \quad (5)$$

Equation (5) represents search results using parallel searching technique using algorithm LIFR (for odd partitions) and FIFR (for even partitions).

If data is being found, then update data as per requirements: update (SR),

Algorithm 4 Update_Data

Input: 'n' number of data (N_F)

Output: specific data (N_D)

Step 1 Drive $M = \sum_{K=1}^n P_k$

Step 2 if $((k\%2) == 0)$

$N_D = \text{Parallel_Search}(\text{NPS}(\text{FIFR}(P_2, P_4, \dots, P_k)))$ [refer equation (5)]

else

$N_D = \text{Parallel_Search}(\text{NPS}(\text{LIFR}(P_1, P_3, \dots, P_{k-1})))$ [refer equation (5)]

Step 3 update (N_D)

Step 4 Stop

In Algorithm 4, 'n' number of data is considered as input dataset. In step 1, total drive is considered as equal to 'k' number of partitions. In step 2, check partition number is whether EVEN or ODD. If partition number is EVEN, then data has been searched using FIFR technique [refer Algorithm 3 and using equation (5)]. If partition number is ODD, then data has been searched using LIFR technique [refer Algorithm 2 and using equation (5)]. If data exists, then update the concerned data as required.

Figure 8 represents workflow diagram for deleting data. Proposed searching technique is applied to search particular data while requiring deleting data.

Drive M is divided into storage partitions. Proposed multivariate regression-based searching technique has applied in each partitions in drive M.

Drive $M = K$ number of partitions ($P_1 + P_2 \dots P_k$).

$SR = \text{Parallel_Search}(\text{NPS}(\text{LIFR} \parallel \text{FIFR}(\text{Drive } M)))$.

If data is found, then delete the data (drive $M-SR$).

Algorithm 5 Delete_Data

Input: 'n' number of data (N_F)

Output: Delete specific data (N_D)

Step 1 Drive $M = \sum_{K=1}^n P_k$

Step 2 if $((k\%2) == 0)$

First_in_first_search(N_D)

else

Last_in_first_search(N_D)

Step 3 Drive $M = \sum_{K=1}^n P_k - N_D$

Step 4 Stop

In Algorithm 5, ‘n’ number of data is considered as input dataset. In step 1, total drive is equal to ‘k’ number of partitions. In step 2, check partition number is odd or even. If partition number is odd, then data has been searched using LIFR technique (refer Algorithm 2). If partition number is even, then data has been searched using FIFR technique (refer Algorithm 3). If data exists, then delete the data from drive M. Data is represented as N_D which is deleted from drive M as required.

5 Analytical derivations on storage

Table 1 represents link between drive partitions and MTs. Drive partitions are divided into two parts: even partitions and odd partitions. MTs allocate the incoming data into particular partitions.

Table 1 Linkage between partitions and MTs in proposed storage

<i>Storage M</i>		<i>Partitions of storage M</i>		
		<i>P1, P2, P3, ..., Pk</i>		
		<i>Map table (MT)</i>		
MT1	MT1_P1	MT1_P2	...	MT1_Pi1
MT2	MT2_P1	MT2_P2	...	MT2_Pi2
...	...	...	...	...
...	...	...	...	...
MTn	MTn_P1	MTn_P2	...	MTn_Pin

As per Table 1:

Set of partitions (P) = $\{P1, P2, P3, \dots, PK\}$; where k = any even value

Set of even partitions (P_E) = $\{P2, P4, P6, \dots, PK\}$

Set of odd partitions (P_O) = $\{P1, P3, \dots, PK - 1\}$

Drive $M = P_E + P_O$ (6)

From equation (6), it is represented that, drive M is the summation of even partitions and odd partitions.

$MT1 = \{MT1_P1, MT1_P2, \dots, MT1_Pi1\}$

$MT2 = \{MT2_P1, MT2_P2, \dots, MT2_Pi2\}$

...

$MTn = \{MTn_P1, MTn_P2, \dots, MTn_Pin\}$

$MT1 + MT2 \dots MTn = \{MT1_P1, MT1_P2, \dots, MT1_Pi1\}$

$+ \{MT2_P1, MT2_P2, \dots, MT2_Pi2\}$

$+ \dots + \{MTn_P1, MTn_P2, \dots, MTn_Pin\}$

$$\begin{aligned}
 MT1 &= \{MT1_P1, MT1_P2, \dots, MT1_Pil\} \subseteq \text{Drive M} \\
 MT2 &= \{MT2_P1, MT2_P2, \dots, MT2_Pil\} \subseteq \text{Drive M} \\
 &\dots \\
 MTn &= \{MTn_P1, MTn_P2, \dots, MTn_Pin\} \subseteq \text{Drive M}
 \end{aligned}$$

Each MTs are subsections of drive M.

$$\therefore \text{Drive M} = P_E + P_O \text{ [refer equation (6)]}$$

$$P_E \subseteq \text{Drive M}$$

$$P_O \subseteq \text{Drive M}$$

$$\begin{aligned}
 MT1 &= \{MT1_P1, MT1_P2, \dots, MT1_Pil\} \\
 &= \{MT1_P1, MT1_P3, \dots, MT1_Pil-1\} \\
 &+ \{MT1P2, MT1P4, \dots, MT1Pil\}
 \end{aligned}$$

...

$$MTn = \{MTn_P1, \dots, MTn_Pin-1\} + \{MTn_P2, \dots, MTn_Pin\}$$

$$MT1 = P_E + P_O$$

$$MT2 = P_E + P_O$$

$$\therefore MTn = P_E + P_O$$

$$MT = P_E + P_O$$

$$P_E \subseteq MT \tag{7}$$

$$P_O \subseteq MT \tag{8}$$

Total MT is the summation of even and odd partitions. Therefore, even partitions and odd partitions are subsets of MT [refer equations (7) and (8)].

$$\begin{aligned}
 \text{For partition (P1)} &= \text{First in first read search (FIFR)} \\
 &+ \text{Last in first read search (LIFR)} \\
 &= 0 + \text{LIFR}
 \end{aligned}$$

$$\begin{aligned}
 \text{For partition (P2)} &= \text{First in first read search (FIFR)} \\
 &+ \text{Last in first read search (LIFR)} \\
 &= \text{FIFR} + 0
 \end{aligned}$$

$$\begin{aligned}
 \text{For partition (P3)} &= \text{First in first read search (FIFR)} \\
 &+ \text{Last in first read search (LIFR)} \\
 &= \text{LIFR} + 0
 \end{aligned}$$

$$\begin{aligned}
 \text{For partition (P4)} &= \text{First in first read search (FIFR)} \\
 &+ \text{Last in first read search (LIFR)} \\
 &= \text{FIFR} + 0
 \end{aligned}$$

$$\begin{aligned}
\text{For partition } (k-1)(P_{k-1}) &= \text{First in first read search (FIFR)} \\
&\quad + \text{Last in first read search (LIFR)} \\
&= 0 + \text{LIFR}
\end{aligned}$$

$$\begin{aligned}
\text{For partition } (k)(P_k) &= \text{First in first read search (FIFR)} \\
&\quad + \text{Last in first read search (LIFR)} \\
&= \text{FIFR} + 0
\end{aligned}$$

$$P_1 + P_2 + P_3 \dots P_K = \frac{1}{2(\text{LIFR} + \text{FIFR})} \quad (9)$$

LIFR algorithm is applied on odd partitions and FIFR algorithm is applied on even partitions. Overall, equation (9) is used to search files or data throughout the partition.

6 Discussion on time optimisation

Theoretical analysis has been established in this section with a focus on time optimisation using distinct factors, such as time complexity and execution time.

Motivation of Theorem 1: Time complexity is dependent on number of elements, number of drive partitions, and number of searching pointers. If number of drive partitions is increased, then time complexity is reduced linearly.

Theorem 1: Time complexity is linear in proposed approach.

Proof: In proposed approach, following conditions are being considered based on the number of partitions and number of pointers.

Number of partitions = k ; where $k = 1, 2, 3, 4, \dots, t$;

Number of pointers = 2

Number of dataset = n

Execution time $T(n) = 1$; when $n/2 * k = 0$

// when $n/2 * k = 0$ then execution time is 1

$$T\left(\frac{n}{2} * k - 1\right) + 1; \text{ when } \frac{n}{2} * k > 0$$

$$T(n) = T\left(\frac{n}{2} * k - 1\right) + 1$$

$$T(n-1) = T\left(\frac{n}{2} * k - 2\right) + 1$$

$$\text{Substitute } T(n) = T\left(\frac{n}{2} * k - 3\right) + 3$$

$$T(n) = \left[T\left(\frac{n}{2} * k - 2\right) + 1 \right] + 1$$

$$T(n) = T\left(\frac{n}{2} * k - 2\right) + 2$$

Substitute again:

$$T(n) = T\left(\frac{n}{2} * k - 3\right) + 3$$

Continue for m times:

$$T(n) = T\left(\frac{n}{2} * k - m\right) + m$$

$$\text{Assume } \frac{n}{2} * k = m$$

$$\text{Since } \frac{n}{2} * k = m$$

$$T(n) = T\left(\frac{n}{2} * k - \frac{n}{2} * k\right) + \frac{n}{2} * k // \text{ since } \frac{n}{2} * k = m \text{ then } (m - m) = 0$$

$$T(n) = T(0) + \frac{n}{2} * k$$

$$T(n) = 1 + \frac{n}{2} * k(1)$$

$$\text{Therefore, time complexity} = O\left(\frac{n}{2} * k\right)$$

Hence, it is proved that time complexity is linearly reduced in proposed approach (end of proof).

Motivation of Theorem 2: Execution time is dependent on number of drive partitions in case of searching a particular data from the number of data through parallel channels. If number of drive partitions is increased, then due to parallel searching execution time is decreased.

Theorem 2: Execution time is inversely proportional to the number of partitions (P).

Proof: By the concept of linear search (stack or queue), expression of execution time is as follows:

$$T(n) = n + c \text{ [from equation (1)]}$$

where $T(n)$ = execution time, n = number of dataset, and c = constant

$$T(n) = \frac{3n}{4} * \frac{4}{3} + c$$

$$T(n) = \frac{n(2^2 - 1)}{2^2} * \frac{4}{3} + c$$

If k number of partitions are considered,

$$\therefore T(n) = \frac{n(2^k - 1)}{2^k} * \frac{4}{3} + c; \text{ where } k = 1, 2, 3, 4, \dots, t$$

where $T(n)$ = execution time, n = number of dataset, c = constant, and k = number of partitions.

Now,

$$\therefore T(n) = \frac{n(2^k - 1)}{2^k} * \frac{4}{3} + c$$

$$\therefore T(n) = \left(\frac{n - n}{2^k} \right) * c1; \text{ where } c1 = \frac{4}{3} + c$$

$$\therefore T(n) = n \left(\frac{1 - 1}{2^k} \right) * c1$$

$$\therefore T(n) \propto 1/2^k$$

$$\therefore T(n) \propto 1/k$$

Hence, it is proved that execution time is inversely proportional to number of partitions (end of proof).

7 Discussion on drive partitions and its mapping

Motivation of Theorem 3: Drive partitions are divided into two sets: even partitions and odd partitions. Summation of size of even and odd partitions is equal to size of total drive. As a result, even partitions and odd partitions are subset of total drive.

Theorem 3: Set of even drive partitions and set of odd drive partitions are subset of drive M.

Proof: From equation (6), drive $M = P_E + P_O$

$$\therefore P_E \subseteq \text{Drive } M$$

$$\therefore P_O \subseteq \text{Drive } M$$

Hence, it is proved that drive partitions are subset of total drive (end of proof).

Motivation of Theorem 4: Total drive size is equal to addition of all drive partitions. Drive partitions are divided into two parts (even and odd) depending on drive numbers. Addition of even drive partitions and odd drive partitions are equal to total drive space.

Theorem 4: Total drive space is summation of even drive partitions and odd drive partitions.

Proof: From equation (6),

$$\text{Drive } M = M = P_E + P_O$$

Drive $M = P_E + (1 - P_E)$ where number of P_E and P_0 are equal.

$$\therefore \text{Drive } M = 1$$

Hence, it is proved total drive space is a combination of even and odd partitions (end of proof).

Motivation of Theorem 5: MT is the index of drive partitions. Drive is divided into drive partitions. MTs are subset of drive partitions and drive partitions are subset of MTs and total drive.

Theorem 5: Drive Partitions are subset of MTs and subset of drive M.

Proof:

$$\begin{aligned} MT1 + MT2 \dots MTn &= \{MT1_P1, MT1_P2, \dots, MT1_Pi\} \\ &+ \{MT2_P1, MT2_P2, \dots, MT2_Pi\} \\ &+ \dots + \{MTn_P1, MTn_P2, \dots, MTn_Pin\} \\ MT1 &= \{MT1_P1, MT1_P2, \dots, MT1_Pi\} \subseteq \text{Drive } M \\ MT2 &= \{MT2_P1, MT2_P2, \dots, MT2_Pi\} \subseteq \text{Drive } M \\ MTn &= \{MTn_P1, MTn_P2, \dots, MTn_Pin\} \subseteq \text{Drive } M \\ \therefore \{MT1_P1, MT1_P2, \dots, MT1_Pi\} &\subseteq MT1 \subseteq \text{Drive } M \\ \{MT2_P1, MT2_P2, \dots, MT2_Pi\} &\subseteq MT2 \subseteq \text{Drive } M \\ \dots \\ \{MTn_P1, MTn_P2, \dots, MTn_Pin\} &\subseteq MTk \subseteq \text{Drive } M \\ &[\text{refer equation (7) and (8)}] \end{aligned}$$

Hence, it is proved that drive partitions are subset of MTs and drive M (end of proof).

8 Experimental results

8.1 Overview

Experimental discussion has been initiated in this section to establish experimental setup, different types of observations, and several comparisons. Different types of analyses such as time complexity analysis have been discussed in observation sub-section. Comparisons between existing approach and proposed approach have been discussed in comparison sub-section.

8.2 Experimental Setup

Experimental setup is as follows:

- number of servers used in cloud – 200
- nature of configurations – homogeneous

- processor – 2.70 GHz
- RAM– 2.00 GB
- drive type – hard disk drive.
- capacity – 750 GB;
- system type – 64-bit Windows 11 operating system
- software used – VC++ compiler.

8.3 Experimental observations for initial disk mapping and disk partitions

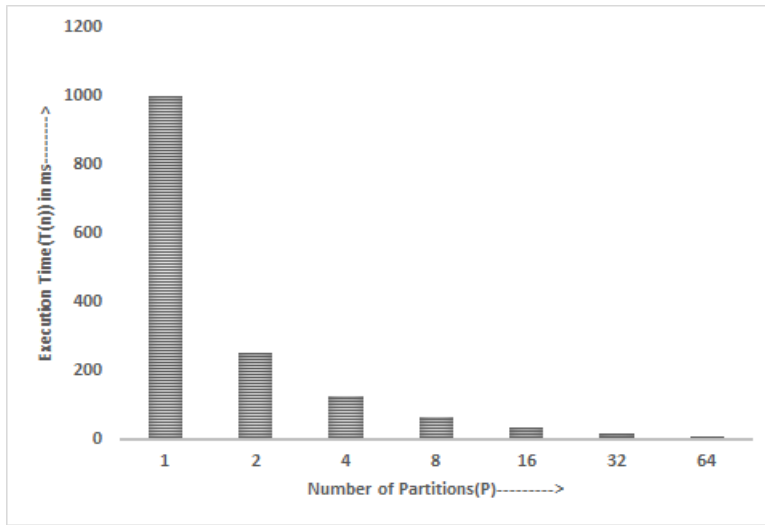
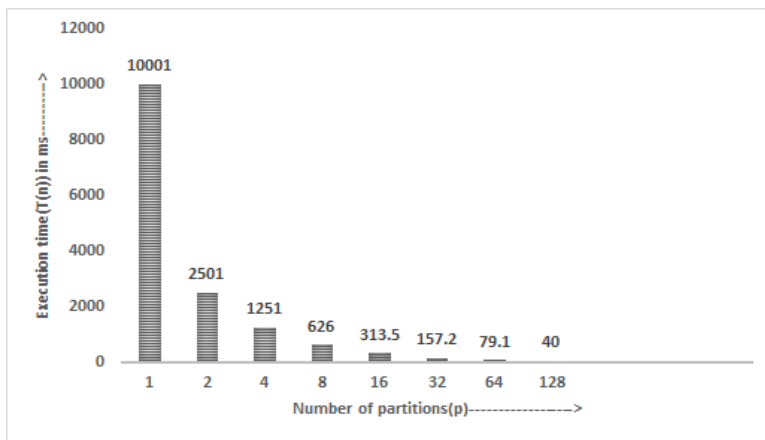
In Table 2, drive locations from 1 to 100,000 are allocated for storing different data. Table 2 represents initial disk space location as existing approach in which drive M is considered as a single partition. Linear searching technique (stack or queue) has been applied for searching particular data. Proposed multivariate regression-based searching technique is applied using two-way pointers for searching particular data.

Table 2 Initial disk space mapping

<i>Drive partitions (P)</i>	<i>Locations for drive M</i>						
	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>...</i>	<i>10,000</i>
<i>P1</i>	10,001	10,002	10,002	10,004	10,005	...	20,000
	20,001	20,002	20,002	20,004	20,005	...	30,000
	30,001	30,002	30,002	30,004	30,005	...	40,000
	40,001	40,002	40,002	40,004	40,005	...	50,000
	50,001	50,002	50,002	50,004	50,005	...	60,000
	60,001	60,002	60,002	60,004	60,005	...	70,000
	70,001	70,002	70,002	70,004	70,005	...	80,000
	80,001	80,002	80,002	80,004	80,005	...	90,000
	90,001	90,002	90,002	90,004	90,005	...	100,000

Table 3 Divide locations with two partitions

<i>Drive partitions (P)</i>	<i>Locations for drive M</i>	
<i>P1</i>	1	...
		...
		50,000
<i>P2</i>	50,001	...
		...
		...
		100,000

Figure 9 Search results w.r.t. execution time with 64 drive partitions on 1,000 data**Figure 10** Search results w.r.t. execution time with 128 drive partitions on 10,000 data

In Table 3, locations of drive M are divided into two partitions. From 1 to 50,000, locations are reserved for P1. From 50,001 to 100,000, locations are reserved for P2. Proposed searching technique is applied into partitions in parallel manner.

Drive locations can be divided into four partitions. From 1 to 25,000, locations are reserved for P1. From 25,001 to 50,000, locations are reserved for P2. From 50,001 to 75,000, locations are reserved for P3, and from 75,001 to 100,000, locations are reserved for P4. Proposed searching technique is applied into four drive partitions in parallel manner.

Drive locations can also be divided into eight partitions (P1, P2, P3, P4, P5, P6, P7, P8). From 1 to 12,500, locations are reserved for P1. From 12,501 to 25,000, memory locations are reserved for P2. From 25,001 to 37,500, locations are reserved for P3. From 37,501 to 50,000, locations are reserved for P4. P5 is reserved by locations from 50,001

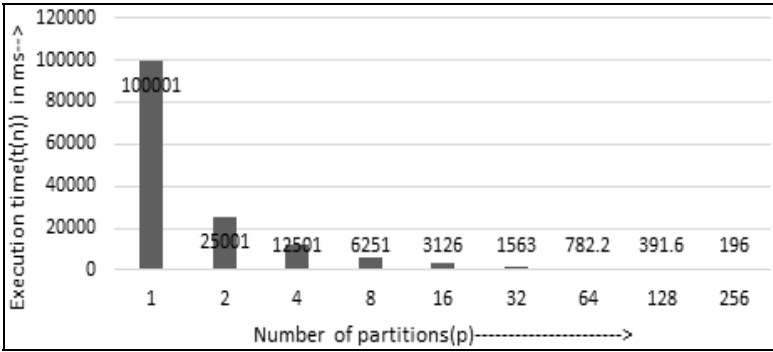
to 62,500. Address location 62,501 to 75,000 is reserved for P6. P7 is reserved by locations from 75,001 to 62,500. P8 is reserved by locations from 87,501 to 100,000.

Similarly, drive locations are also divided into 16 drive partitions (P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12, P13, P14, P15, P16) accordingly. Therefore, proposed searching technique is applied in each partition in parallel manner for searching particular data.

In Figure 9, ‘X’ axis is denoted by number of partitions (P) and ‘Y’ axis is denoted by execution time T(n). This graph is represented based on 1,000 data and 64 partitions where ‘1’ represents existing approach. It shows that execution time is inversely proportional with the number of partitions in storage. As per Figure 9, execution time of proposed approach (partitioned data) represents better result than existing approach (no partition data).

According to Figure 10, ‘X’ axis is denoted by number of partitions (P) and ‘Y’ axis is denoted by execution time T(n). In Figure 10, graph is depicted based on 10,000 data having 128 partitions. On ‘X’ axis, Number of partitions (P) = 1 represents existing approach with no drive partitions. It exhibits that, execution time to search elements is decreased while partitions of storage are being increased as it is going through less number of elements in each partitions. As per Figure 10, it has been graphically represented that execution time of proposed approach (partitioned data) exhibits better result than existing approach (no partition data).

Figure 11 Search results w.r.t. execution time with 256 drive partitions on 100,000 data



In Figure 11, ‘X’ axis is denoted by number of partitions (P) and ‘Y’ axis is denoted by execution time T(n). This graph is depicted based on 100,000 data having 256 partitions in which ‘1’ represent existing approach. Major advantage of proposed framework is the related higher performance due to inclusion of a number of partitions. Overall, the element of each partition is searched with less time complexity due to FIFO and LIFO concept applied to odd and even partitions. In Figure 11, higher result is achieved by increasing number of partitions and data size.

9 Comparative analysis

9.1 Comparison on execution time and time complexity of existing approach (no drive partitions and number of pointer is 1) and proposed approach

According to Table 4, in existing approach there is no partition in drive 'M'. In proposed approach, drive 'M' is divided into different number of partitions. It has been represented that proposed approach is better than existing approach after calculation of execution time.

Table 4 Execution time comparison between existing and proposed approaches

	<i>Number of drive partitions (P)</i>	<i>Number of pointers</i>	<i>Execution time (T(n))</i>
Existing approach	1	1	100,001
Proposed approach	1	2	50,001
	2	2	25,001
	4	2	12,501
	8	2	6,251
	16	2	3,126
	32	2	1,564
	64	2	782
	128	2	392
	256	2	196
	512	2	98
	1,024	2	49

Table 5 Time complexity comparison between existing and proposed approaches

	<i>Number of drive partitions (P)</i>	<i>Number of pointers</i>	<i>Time complexity</i>
Existing approach	1	1	$O(n)$
Proposed approach	1	2	$O(n/2)$
	2	2	$O(n/4)$
	4	2	$O(n/8)$
	8	2	$O(n/16)$
	16	2	$O(n/32)$
	32	2	$O(n/64)$
	64	2	$O(n/128)$
	128	2	$O(n/256)$
	256	2	$O(n/512)$
	512	2	$O(n/1,024)$
	1,024	2	$O(n/2,048)$

According to Table 5, in existing approach there is no partition in drive ‘M’. In proposed approach, drive ‘M’ is divided into different number of partitions, and it has been represented that proposed approach is better than existing approach with respect to time complexity.

Figure 12 Comparison for execution time between existing and proposed approaches with respect to storage partitions

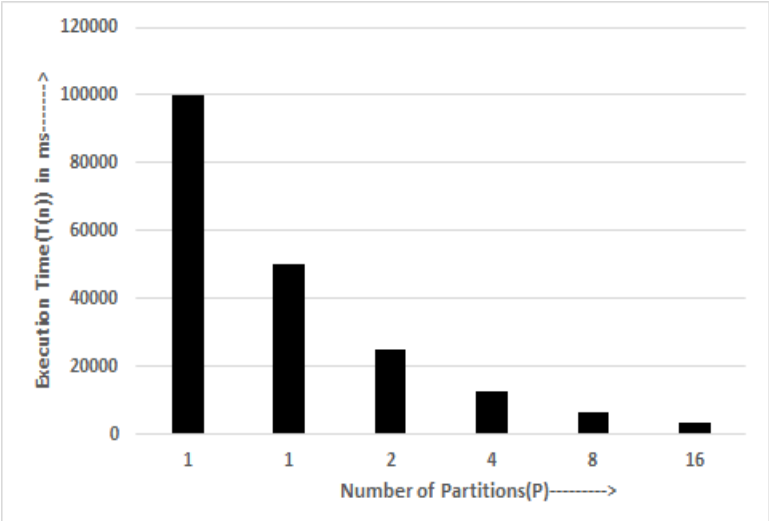


Figure 13 Comparison for execution time with 1,000 data between existing and proposed approach in storage partitions



In Figure 12, 'X' axis is denoted by number of partitions (P) and 'Y' axis is denoted by execution time $T(n)$. The graph exhibits execution time of proposed approach and existing approach (number of partitions = 1) having less time complexity for proposed approach.

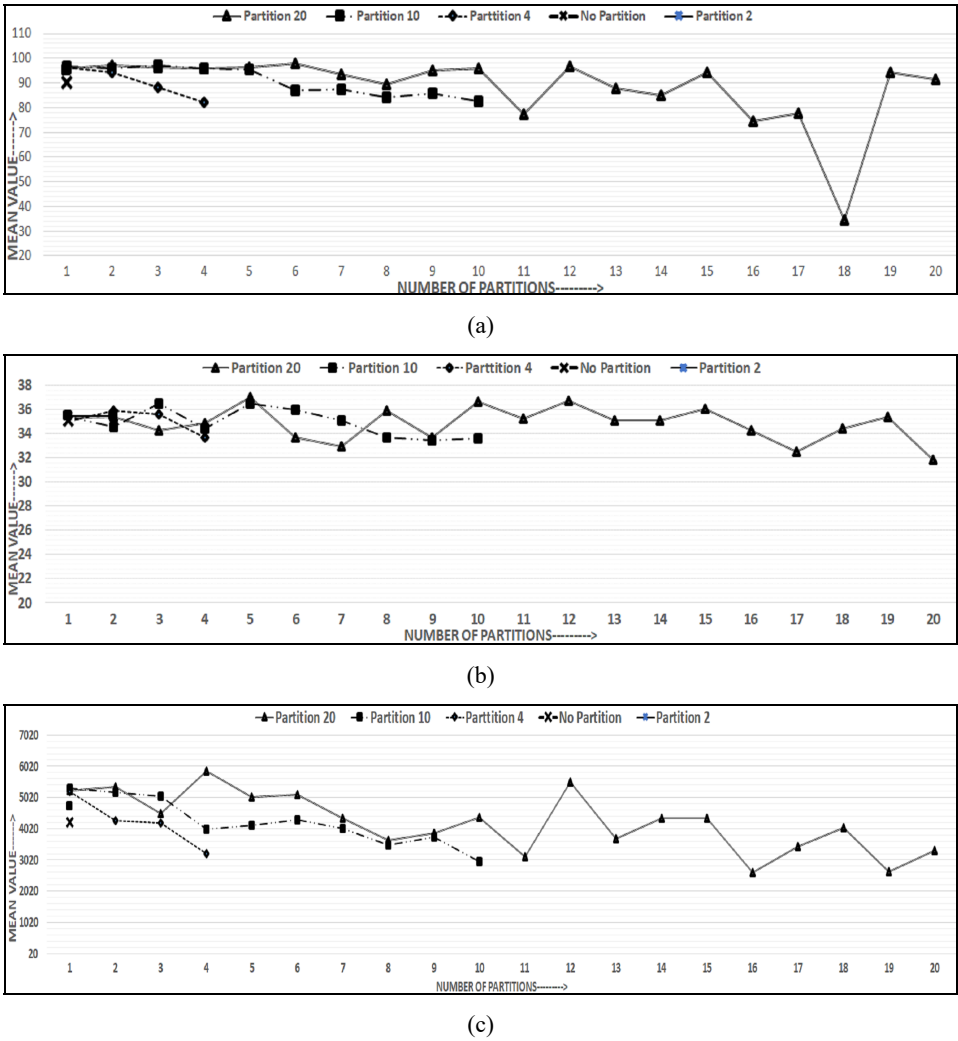
In Figure 13, 'X' axis is denoted by execution time $T(n)$ and 'Y' axis is denoted by number of elements. Figure represents that execution time is decreased when the partitions of drive are being increased. Therefore, it shows better results compared to existing approach (number of partitions = 1) while increasing number of partitions.

9.2 Comparison between existing approach and proposed approach using statistical data

In Figure 14(a), 'X' axis is denoted by number of partitions and 'Y' axis is denoted by mean value of quantity ordered dataset. According to Figure 14(a), the graph represents comparison of mean value of quantity ordered dataset between existing approach (no partition) and proposed approaches (partitions 2, 4, 10, and 20). Mean values of quantity ordered dataset for each partition are denoted by different graphs. In Figure 14(b), 'X' axis is denoted by number of partitions and 'Y' axis is denoted by mean value of price for each item dataset. According to Figure 14(b), the graph represents comparison of mean value of price for each item dataset between existing approach (no partition) and proposed approaches (partitions 2, 4, 10, and 20). In Figure 14(c), 'X' axis is denoted by number of partitions and 'Y' axis is denoted by mode value of sales dataset. According to Figure 14(c), the graph represents comparison of mean value of sales dataset between existing approach (no partition) and proposed approaches (partitions 2, 4, 10, and 20). Figure 14 shows mean values are higher in proposed approaches. It is known fact that low variance is associated with less risk tolerance. If mean values are increased in a system framework, then standard deviation and variance are decreased, and the system behaves better with respect to risk tolerance. According to Figure 14, majority of the mean values in case of proposed approaches are higher than existing approach. Therefore, proposed approaches represent better results.

In Figure 15(a), 'X' axis is denoted by number of partitions and 'Y' axis is denoted by median value of quantity ordered dataset. According to Figure 15(a), the graph shows comparison of median value of quantity ordered dataset between existing approach (no partition) and proposed approaches (partitions 2, 4, 10, and 20). In Figure 15(b), 'X' axis is denoted by number of partitions and 'Y' axis is denoted by median value of price for each item dataset. In Figure 15(c), 'X' axis is denoted by number of partitions and 'Y' axis is denoted by median value of sales dataset. According to Figure 15(c), the graph exhibits comparison of median value of sales dataset between existing approach (no partition) and proposed approaches (partitions 2, 4, 10, and 20). Median values of each dataset for each partition are denoted by different types of graphs. According to Figure 15, median values of proposed system framework shows higher values in majority of cases compared to existing approach. Therefore, proposed approach depicts better results with respect to existing approach.

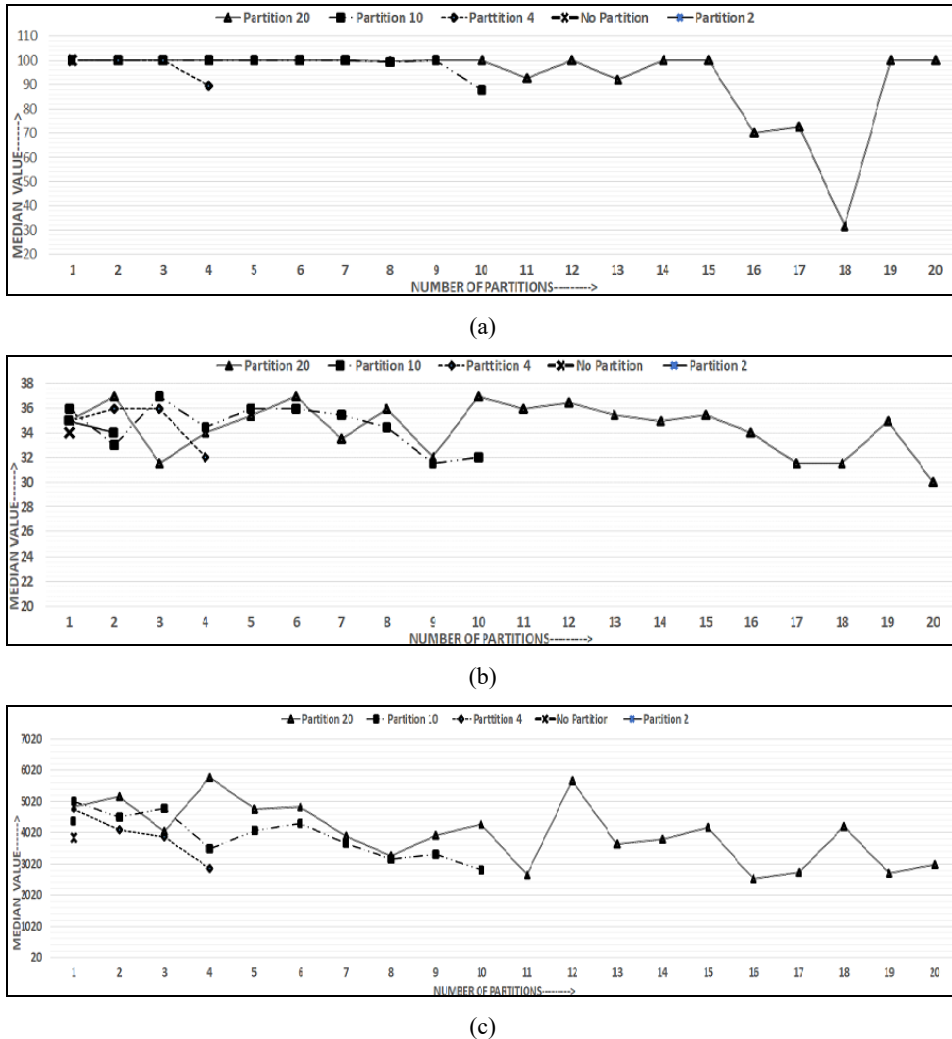
Figure 14 Comparison of mean values between existing and proposed approach using Kaggle dataset (see online version for colours)



Source: <https://www.kaggle.com/kyanyoga/sample-sales-data>

In Figure 16(a), ‘X’ axis is denoted by number of partitions and ‘Y’ axis is denoted by mode value of quantity ordered dataset. Mode values of quantity ordered dataset for each partition are denoted using distinct graphs. In Figure 16(b), ‘X’ axis is denoted by number of partitions and ‘Y’ axis is denoted by mode value of Price for each item dataset. According to Figure 16(b), the graph exhibits comparison of mode value of Price for each item dataset between existing approach (no partition) and proposed approaches (partitions 2, 4, 10, and 20). In Figure 16(c), ‘X’ axis is denoted by number of partitions and ‘Y’ axis is denoted by mode value of sales dataset. Figure 16 represents that majority mode values are decreased in proposed approaches. Therefore, proposed approaches show better results than existing approach.

Figure 15 Comparison of median values between existing and proposed approach using Kaggle dataset (see online version for colours)

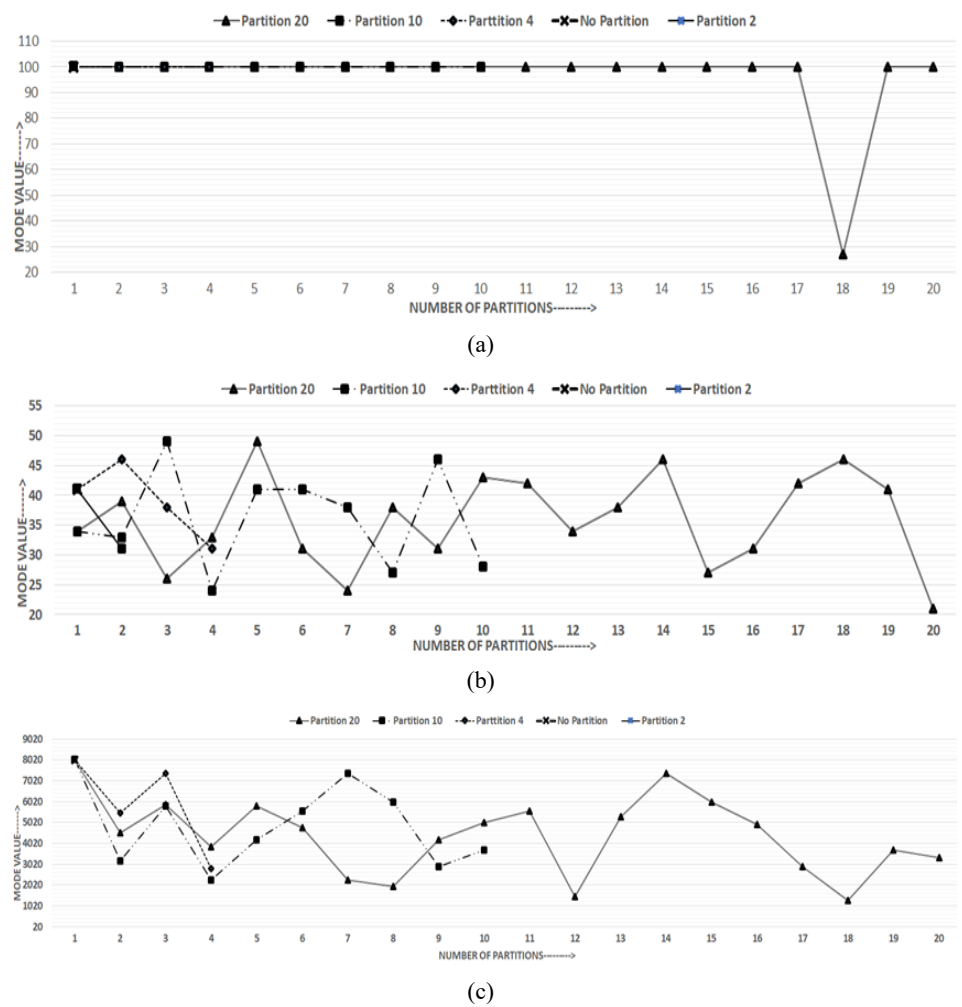


Source: <https://www.kaggle.com/kyanyoga/sample-sales-data>

In Figure 17(a), 'X' axis is denoted by number of partitions and 'Y' axis is denoted by standard deviation value of quantity ordered dataset. According to Figure 17(a), the graph represents comparison of standard deviation value of quantity ordered dataset between existing approach (no partition) and proposed approaches (partitions 2, 4, 10, and 20). In Figure 17(b), 'X' axis is denoted by number of partitions and 'Y' axis is denoted by standard deviation value of Price for each item dataset. According to Figure 17(b), the graph represents comparison of standard deviation value of price for each item dataset between existing approach (no partition) and proposed approaches (partitions 2, 4, 10, and 20). In Figure 17(c), 'X' axis is denoted by number of partitions and 'Y' axis is denoted by standard deviation value of Sales dataset. Figure 17 shows low standard deviation in case of majority of proposed approaches compared to existing approach. If

standard deviation is decreased, then variance is also decreased, which is better to manage risk tolerance. In Figure 17, proposed approaches exhibit better results compared to existing approach.

Figure 16 Comparison of mode values between existing and proposed approach using Kaggle dataset (see online version for colours)

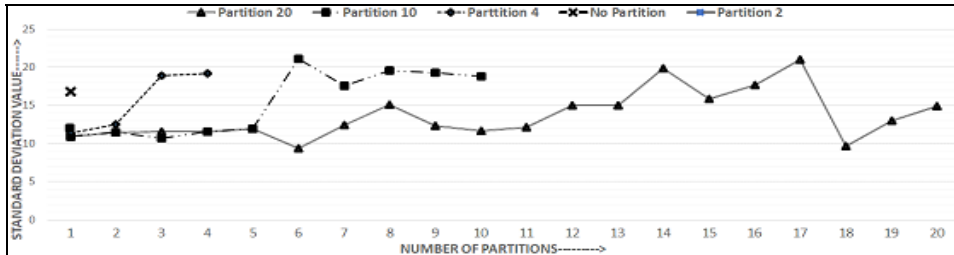


Source: <https://www.kaggle.com/kyanyoga/sample-sales-data>

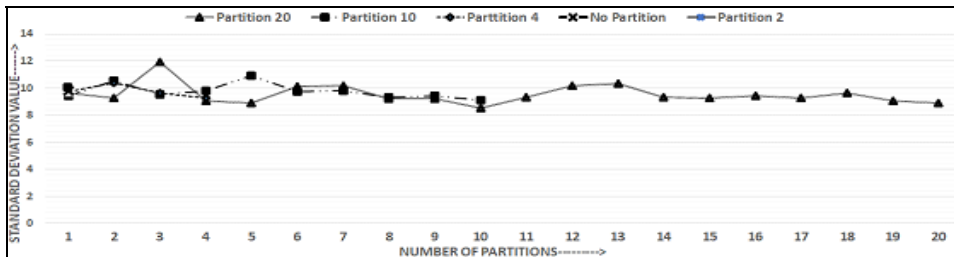
In Figure 18(a), ‘X’ axis is denoted by number of partitions and ‘Y’ axis is denoted by variance value of quantity ordered dataset. In Figure 18(b), ‘X’ axis is denoted by number of partitions and ‘Y’ axis is denoted by variance value of price for each item dataset. In Figure 18(c), ‘X’ axis is denoted by number of partitions and ‘Y’ axis is denoted by variance value of sales dataset. Plots are represented by the comparison of variance value

of each dataset between existing approach and proposed approaches. According to Figure 18, variance values are decreased in proposed approaches in majority of the cases compared to existing approach. Low variance is associated with less risk tolerance. It has been proved that variance of proposed approaches exhibit better results compared to existing approach.

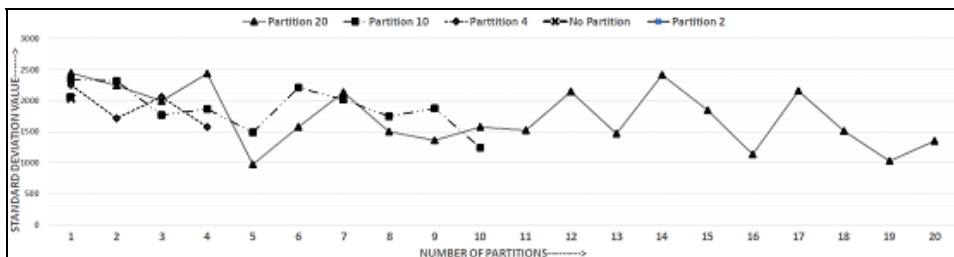
Figure 17 Comparison of standard deviation values between existing and proposed approach using Kaggle dataset (see online version for colours)



(a)



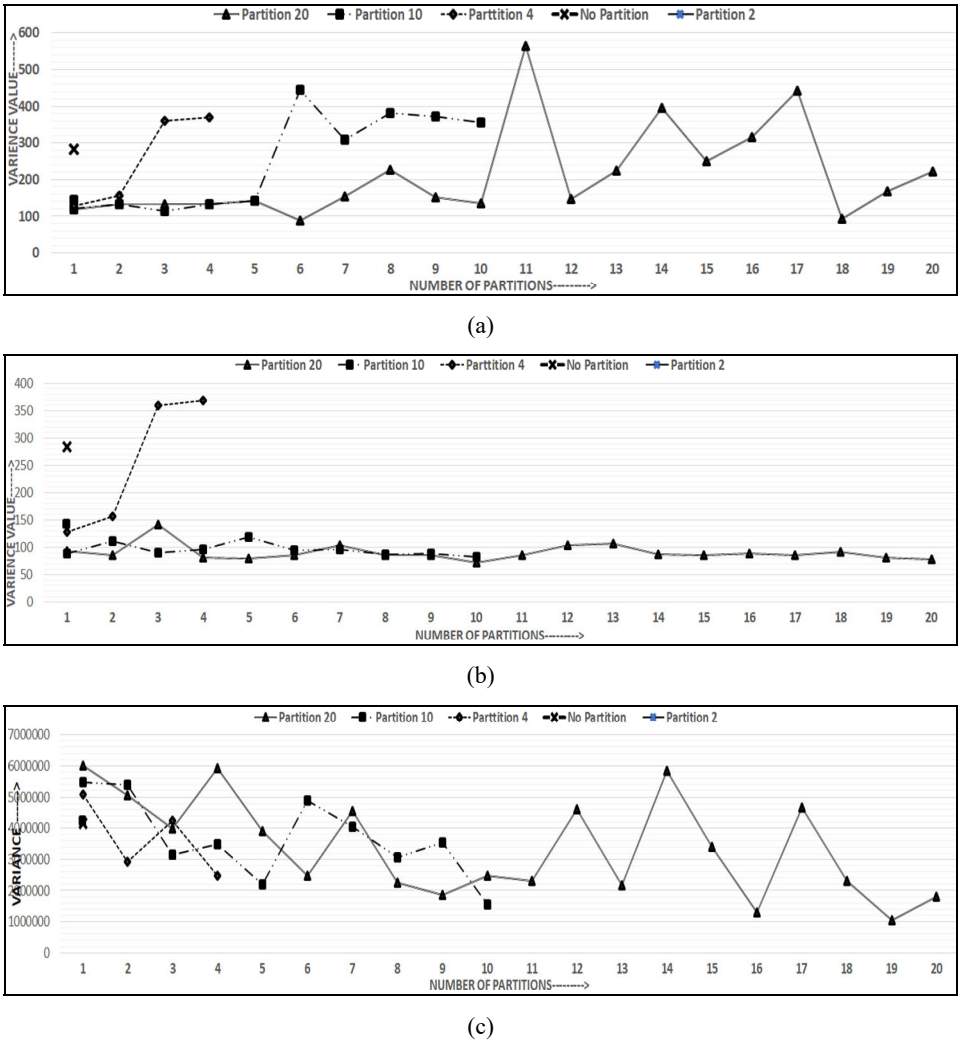
(b)



(c)

Source: <https://www.kaggle.com/kyanyoga/sample-sales-data>

Figure 18 Comparison of variance values between existing and proposed approach using Kaggle dataset (see online version for colours)



Source: <https://www.kaggle.com/kyanyoga/sample-sales-data>

10 Conclusions

The proposed multivariate regression-based disk searching technique using MTs for facilitating data searching is achieved in less time consumption. LIFR and FIFR techniques are effectively used by proposed design framework. Storage analysis has shown new perspectives with theory development for time optimisation, drive partitions and mapping. In experimental section it is graphically exhibited that execution time is decreased with the increasing number of elements and storage partitions. Proposed storage-based framework has shown efficient results compared to existing architecture

with respect to time complexity and execution time, mean, median, mode, standard deviation and variance.

References

- Bairavasundaram, L.N., Goodson, G.R., Pasupathy, S. and Schindler, J. (2007) 'An analysis of latent sector errors in disk drives', *Proc. Int. Conf. Meas. Modeling Computer Syst.*, San Diego, California, USA, pp.289–300.
- Bender, G., Kindermans, P., Zoph, B., Vasudevan, V. and Le, Q. (2018) 'Understanding and simplifying one-shot architecture search', *Proceedings of Machine Learning Research (PMLR)*, Vol. 80, pp.550–559.
- Buyya, R. (2013) 'Introduction to the IEEE Transactions on Cloud Computing', *IEEE Transactions on Cloud Computing*, Vol. 1, No. 1, pp.3–21.
- Carns, P.H., Lang, S., Ross, R.B., Vilayannur, M., Kunkel, J.M. and Ludwig, T. (2009) 'Small-file access in parallel file systems', *Proc. IEEE International Symposium on Parallel Distributed Processing*, Rome, Italy, pp.1–11.
- Chan, W.K., Mei, L. and Zhang, Z. (2009) 'Modelling and testing of cloud applications', *IEEE Asia-Pacific Services Computing Conference*, Singapore, 7–11 December, pp.111–118.
- Curino, J., Popa, M., Wu, B. and Zeldovich (2020) 'Relational cloud, a database-as-a-service for the cloud' [online] <https://www.scribd.com/document/365889717/CIDR11Paper331-pdf>.
- Dholakia, A., Eleftheriou, E., Hu, X-Y., Iliadis, I., Menon, J. and Rao, K. (2008) 'A new intra-disk redundancy scheme for high-reliability raid storage systems in the presence of unrecoverable errors', *Trans. Storage*, Vol. 4, No. 1, pp.1–42.
- Gibson, G.A. (2007) 'Reflections on failure in post-terascale parallel computing', *Proc. Int'l. Conf. Parallel Process (ICPP '07)*, Xi'an, China.
- Gonzalez, N., Miers, C., Redigolo, F., Simplicio, M., Carvalho, T., Naslund, M. and Pourzand, M. (2012) 'A quantitative analysis of current security concerns and solutions for cloud computing', *Journal of Cloud Computing, Springer Open Journal*, Vol. 1, No. 11, pp.1–18
- Han, D.Z., Yu, S.Z. and Xie, C.S. (2006) 'An IP-based high-available mass network storage system', *Chinese Journal of Electronics*, Vol. 34, No. 11, pp.2012–2017.
- Heger, D.A. (2011) *Devices – An Introduction to the Android Operating Environment Design, Architecture, and Performance Implications*, DHTechnologies [online] <http://dhtusa.com> (accessed 23 February 2023).
- Hou, H.X., Shum, K.W., Chen, M.H. and Li, H. (2016) 'BASIC codes, low-complexity regenerating codes for distributed storage systems', *Proc. IEEE Transactions on Information Theory*, Vol. 62, No. 6, p.1.
- HPC (2011) *A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard* [online] <https://www.hpc.msstate.edu/computing/MPI.pdf> (accessed 23 January 2023).
- Huang, Z. and Wang, N. (2018) 'Data-driven sparse structure selection for deep neural networks', *Proceedings of the European Conference on Computer Vision (ECCV)*, Munich, Germany, pp.317–334.
- Jiang, W., Hu, C., Zhou, Y. and Kanevsky, A. (2008) 'Are disks the dominant contributor for storage failures? A comprehensive study of storage subsystem failure characteristics', *Proc. 6th USENIX Conf. File Storage Techno*, San Jose, California, pp.1–15.
- Kumar, R. and Sahoo, G. (2014) 'Cloud computing simulation using CloudSim', *International Journal of Engineering Trends and Technology*, Vol. 8, No. 2, pp.82–86.
- Kusnetzky, D. (2011) *Application Performance Requires Better Storage Performance*, Document No. 20110110, The Kusnetzky Group.

- Liu, C., Zoph, B., Shlens, J., Hua, W., Li, L., Fei-Fei, L., Yuille, A., Huang, J. and Murphy, K. (2018) 'Progressive neural architecture search', *Proceedings of the European Conference of Computer Vision (ECCV)*, pp.19–34.
- Mano, M.M. (2009) *Computer System Architecture*, Pearson Edition, ISBN: 978-81-317-0070-9.
- Nazir, M. (2012) 'Cloud computing: overview & current research challenges', *IOSR Journal of Computer Engineering (IOSR-JCE)*, Vol. 8, No. 1, pp.14–22, ISSN: 2278-0661, ISBN: 2278-8727.
- Paola, P., Roberto, C., Alberto, B. and Lorenzo, P. (2020) 'Amazon, Google and Microsoft solutions for IoT, architectures and a performance comparison', *IEEE Access*, Vol. 8, pp.5455–5470.
- Plank, J.S., Blaum, M. and Hafner, J.L. (2013) 'SD codes, erasure codes designed for how storage systems really fail', *Proc. 11th Usenix Conf. File Storage Technologies (FAST '13)*, San Jose, CA.
- Rawat, A.S., Koyluoglu, O.O., Silberstein, N. and Vishwanath, S. (2014) 'Optimal locally repairable and secure codes for distributed storage systems', *IEEE Transactions on Information Theory*, Vol. 60, No. 1, pp.212–236.
- Reddy, V.K., Thirumal Rao, B., Reddy, L.S.S. and Sai Kiran, P. (2011) 'Research issues in cloud computing', *Global Journal of Computer Science and Technology*, Vol. 11, No. 11, pp.59–64.
- Reinaldo, B.A. and Adelchi, A. (2021) 'A formulation for continuous mixtures of multivariate normal distribution', *Journal of Multivariate Analysis*, Vol. 185, No. 104780, DOI: 10.1016.
- Richa, S., Isaac, W., Glaucio, C. and Alagan, A. (2019) 'Mobile cloud storage over 5G: a mechanism design approach', *IEEE Systems Journal*, Vol. 13, No. 4, pp.4060–4071.
- Shah, N.B., Rashmi, K.V. and Kumar, P.V. (2011) 'Information-theoretically secure regenerating codes for distributed storage', *Proc. IEEE Global Telecommunications Conference (GLOBECOM 2011)*, Houston, Texas, USA, pp.5–9.
- Shu, F. (2007) *Dataset Management Commands Proposal for ATA8-ACS2*, T13/e07154r6, Technical Committee, USA.
- Shuaiqing, Q., Qisheng, Z., Qimao, Z., Feng, G. and Wenhao, L. (2020) 'Hybrid seismic-electrical data acquisition station based on cloud technology and green IoT', *IEEE Access*, Vol. 8, No. 99, p.1.
- Silberschatz, A., Galvi, P.B. and Gagne, G. (2009) *Operating System Concepts*, Willy – India Edition, ISBN: 978-81-265-2051-0.
- Thomasian, A. (2005) 'Reconstruct versus read-modify writes in RAID', *Information Processing Letters*, Vol. 93, No. 4, pp.163–168.
- Thomasian, A. and Blaum, M. (2009) 'Higher reliability redundant disk arrays: organization, operation, and coding', *ACM Transactions on Storage*, Vol. 5, No. 3, p.7.
- Tian, L., Cao, Q., Jiang, H., Feng, D., Xie, C. and Xin, Q. (2011) 'Online availability upgrades for parity-based raids through supplementary parity augmentations', *ACM Transactions on Storage*, Vol. 6, No. 3, pp.1–23.
- Timmons, A.L., Fomin, P. and Wasek, J. (2016) 'Modeling cloud storage, a proposed solution to optimize planning for and managing storage as a service', *Journal of Information and Computing Science*, Vol. 11, No. 1, pp.70–80.
- Villalpando, L.E.B., April, A. and Abran, A. (2014) 'Performance analysis model for big data applications in cloud computing', *Journal of Cloud Computing*, Vol. 3, No. 19, pp.1–20.
- Wang, C., Wang, Q., Ren, K., Cao, N. and Luo, W.J. (2012) 'Toward secure and dependable storage services in cloud computing', *Proc. IEEE Transactions on Services Computing*, Vol. 5, No. 2, pp.220–232.
- Wang, Q., Yu, D., Zhou, J. and Jin, C. (2023) 'Data storage optimized model based on improved simulated annealing algorithm', *Sustainability*, Vol. 15, No. 9, p.7388.

- Xie, B., Chase, J., Dillow, D., Drokin, O., Klasky, S. and Oral, S. (2012) 'Characterizing output bottlenecks in a supercomputer, ROC', *Int. Conf. High Perform. Comput. Netw. Storage Anal.*, Salt Lake City, UT, USA, pp.1–11
- Zhang, G., Huang, Z., Ma, X., Yang, S., Wang, Z. and Zheng, W. (2018) 'RAID+: deterministic and balanced data distribution for large disk enclosures', *Proceedings of USENIX International Conference on File and Storage Technologies (FAST)*, Oakland, CA, USA, p.279.
- Zhang, Y. (2013) 'DifferCloudStor: differentiated quality of service for cloud storage', *IEEE Transactions on Magnetics*, Vol. 49, No. 6, pp.2451–2458.
- Zhao, X., Li, L., Tao, Y., Lai, S., Zhou, X. and Qiu, J. (2022) 'Aggregated operation of heterogeneous small-capacity distributed energy resources in peer-to-peer energy trading', *Int. J. Electr. Power Energy Syst.*, October, Vol. 141, No. 108162.
- Zhiyang, Z. (2021) 'Fast implementation of partial least squares for function-on-function regression', *Journal of Multivariate Analysis*, 21 December, Vol. 185, No. 104769.
- Zhou, S., Xing, L., Zheng, X., Du, N., Wang, L. and Zhang, Q. (2021) 'A self-adaptive differential evolution algorithm for scheduling a single batch-processing machine with arbitrary job sizes and release times', *IEEE Trans. Cybern.*, Vol. 51, No. 3, pp.1430–1442.

Websites

https://electricalfundablog.com/cloud-storage-architecture-types/#Free_Cloud_Storage.
<https://www.kaggle.com/kyanyoga/sample-sales-data>.