CHAPTER 6

DATA CONSISTENCY IN CLOUD STORAGE FOR CONCURRENT USERS

6.1. INTRODUCTION

Cloud applications generally use data that are dispersed across different data centres. In this multiuser cloud data store, multiple transactions simultaneously update the data at the same time. Since this data is accessed by concurrent users for their business purposes frequently, it must be maintained with the most recent information available so that it must be consistent. Data consistency means that all the instances of the application must have same data values at all time. Maintaining this data consistency in the cloud environment becomes a critical issue when concurrency and availability aspects arise.

Each and every web application and services are in need of cloud data. This data stored in cloud are used by individuals and organizations for performing various computations. Hence, it is necessary to maintain the cloud data with most recent information which must be consistent. It refers to the process of maintaining same set of values in all the instances of the application at all times. The transactional models are used to maintain consistency in which locks are used for preventing the modification of a data at the same time by various concurrent applications.

In the case of strong consistent system, concurrent requests for accessing a data at the same time is also blocked. The concept of pessimistic locking is also used. It locks the data while it is being updated by any of the application and the lock gets released when the updating is completed. In recent cloud applications, the data gets partitioned and are widespread across multiple data stores located at different sites. This process helps in load balancing across a large number of user requests thus improving the scalability, locating the data nearest to the users who access it thereby improving the response time in retrieving the data. The data get replicated and are distributed across multiple data stores thus improving the availability of data.
However, maintaining data consistency across various data stores is a challenging task across the cloud environment. The main problem is that the serialization and locking methodologies work efficiently if all the applications use the same data store and such applications must be designed in such a way that the locks are only short-lived. But, if the data get replicated across various storage servers, the locking and serializing data access for maintaining consistency of data creates a great overhead that affects the throughput, response time and scalability of the process.

6.2. DATA CONSISTENCY IN CLOUD ENVIRONMENT

With user cost minimization and resource usage maximization in highly demanding internet environment, the cloud storage plays a major role to accomplish the current demand of high valued applications usage. However, storage of user data at different cloud storage houses (servers) causes security threats on the confidential and private data sources of individual users. In addition, data consistency in cloud data storage service environment is another major area of concern to improvise concurrent data application users. The data consistency state needs to be handled effectively for improving the low cost communication and computation users of the cloud.

6.2.1. Dynamic Audit Services using Hash Table

The Dynamic Audit Services using Hash Table (DAS-HT) [55] presents dynamic audit services to enhance the performance of Third Party Auditor (TPA) and storage service providers for untrusted and outsourced storages. The performance of audit services was improved by applying probabilistic query and periodic verification. At the same time, the privacy of the data was also protected using fragment structure, random sampling and Index Hash Table (IHT).

6.2.2. Accountability for Data Sharing using Object Centered Approach

The Accountability for Data Sharing using Object Centered (ADS-OC) approach [25] presents a scheme for data sharing with user’s data and policies. A systematic approach that leverages the JAR programmable capabilities to create a
dynamic object is used to ensure user’s data trigger authentication and automated logging local to the JARs. The approach was also proven to be platform independent and highly decentralized, which in turn strengthens the user’s control in providing distributed auditing mechanisms. However, the consistency and coherence of the data storage mechanism in such an environment is not addressed in both of the above methods.

6.3. LINEAR ERASURE AND POISSON EXPONENTIAL DISTRIBUTION FRAMEWORK

In order to maintain data consistency in cloud data storage, an efficient integrated framework named Linear Erasure and Poisson Exponential Distribution (LE-PED) was designed. Initially, Linear Erasure Correction model is applied for several user transactions aiming at improving the cloud data storage capacity while reducing the transaction duration in cloud environment. Next, Poisson Exponential Distribution based Multi-threaded State Transition technique is designed aiming at improving the cyclic progression ratio of the cloud users therefore improving the data consistency state.

6.3.1. General Framework of Cloud Storage

![Cloud storage framework](image_url)

*Fig. 6.1. Cloud storage framework*
The building block of cloud storage framework is shown in Fig. 6.1. It includes two different entities, the cloud user and the cloud server. The cloud user, \( CU_i = CU_1, CU_2, ..., CU_n \) who store their data in the cloud heavily depend on the cloud environment for storing heterogeneous data. On the other hand, the cloud server \( CS \) is administered by Cloud Service Provider \( CSP \) with the sole purpose of providing services of data storage, managing computation resources and handling data storage in an efficient manner.

One of the main innovative features of the LE-PED framework lies in its ability of generating cyclic progression for storage states and the threads are assigned in an on-demand transaction request of the user that combines the aspects of cyclic progression ratio and data storage capacity.

### 6.3.2. Methodology

The first step involved in the design of the LE-PED is the design of Linear Erasure Correction. With the objective of improving cloud data storage capacity, the LE-PED uses Linear Erasure Correction model that bears numerous failures in distributed storage environment. The Linear Erasure Correction model performs two important functions. The first function is that it obtains the original vector and obtains the linear vector. The second function is the generation of resultant matrix using identity matrix and parity vector. With these two functions in Linear Erasure Correction model, the cloud data storage capacity is ensured.

Next, it involves Poisson Exponential Distribution based Multi-threaded State Transition model. The cloud user relies on the data from the cloud server in order to improve cyclic progression ratio. In order to maintain data consistency, the LE-PED uses Poisson Exponential Distribution-based Multi-threaded State Transition.

The Poisson Exponential Distribution model stops the iterative process on the basis of the Multi-threaded State Transition with an arrival rate and an interval time. Finally, a cyclic progression for the storage states and the threads (i.e. the cloud user that has to be assigned with the cloud storage) are assigned in an on-demand transaction request of the user. The architecture diagram of the LE-PED framework is shown in Fig. 6.2.
Fig. 6.2. Architecture diagram of LE-PED framework

A Linear Erasure and Poisson Exponential Distribution (LE-PED) framework is designed to improve the data consistency rate. The data storage capacity is improved by applying the Linear Erasure Correction model where multiple threads are issued. Here, storage states are initialized with the help of using Linear Erasure and Poisson Exponential Distribution. The Poisson Exponential Distribution-based Multi-threaded State Transition demonstrates that LE-PED framework provides increased cyclic progression ratio. The data flow diagram of LE-PED framework is shown in Fig. 6.3.
Each cloud user is served by a server one at a time based on the threading model, according to the first-come to be served first. Upon successful completion of the service, the cloud user is reduced by one. As a result, the cyclic progression ratio increases with the increase in the number of cloud users to be served in the cloud environment.
6.3.3. **Construction of LE-PED Framework**

6.3.3.1. **Linear Erasure Correction**

Fig. 6.4. Block diagram of linear erasure correction model.

The first step in the design of the LE-PED framework is the Linear Erasure Correction model. With the objective of improving cloud data storage capacity, the Linear Erasure Correction model bears numerous failures in distributed storage environment. Fig. 6.4 shows the block diagram of Linear Erasure Correction model.

Let us consider a cloud environment comprising of cloud servers \(CS_i = CS_1, CS_2, ..., CS_n\) where cloud server task is to process the transaction
For each user transactions, multiple threads are issued and storage states are initialized with commit, ready, and sleep. This multiple threads result in reducing the transaction duration for several storage states. The cloud users ‘$CU_i = CU_1, CU_2, ..., CU_n$’ interact with the cloud environment through the cloud servers and therefore accomplishes their task in an efficient manner.

The data file $F$ is dispersed across ‘$d + v$’ distributed servers. The proposed framework uses the function ‘$\{d + v, v\}$’ with the objective of storing the data in an explicit manner. Here ‘$d$’ represents the data vectors which is used to create ‘$v$’ redundancy parity vectors such that the original ‘$d$’ data vectors are reconstructed from any ‘$d$’ out of any ‘$d + v$’ vectors. By keeping ‘$d + v$’ vectors on different servers, the original data ‘$d$’ is at the appropriate location without any loss of data. Let ‘$F_i = f_1, f_2, ..., f_n$’ denotes the column vector whereas ‘$n$’ denotes the size of data vector. Then, the Linear Erasure Correction model is computed for efficient cloud data storage which is mathematically formulated as given below.

\[
v = \begin{bmatrix}
    f_0 & f_1 \\
    f_0' & f_1' \\
    f_0'' & f_1'' \\
    \vdots & \vdots
\end{bmatrix}
\begin{bmatrix}
    p_0 & p_1 \\
    p_0' & p_1' \\
    p_0'' & p_1'' \\
    \vdots & \vdots
\end{bmatrix}
\]

Equation (6.1)

From Equation (6.1), the ‘$[f_0 \ f_1]$’ symbolizes the data vector where ‘$f_i \in d$’ whereas the ‘$[p_0 \ p_1]$’ symbolizes the parity vector that are distinct elements selected from the linear vector ‘$v$’. Upon successful completion of row transformations, the resultant matrix ‘$RM$’ is evaluated for reducing the errors (i.e. to avoid duplication of data) in cloud data storage using the LE-PED as given below.

\[
RM = [I * p] = \begin{bmatrix}
    1 & 0 & \cdots & 0 & p_{11} & p_{21} & \cdots & p_{1k} \\
    0 & 1 & \cdots & 0 & p_{11} & p_{22} & \cdots & p_{2k} \\
    \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \\
    0 & 0 & \cdots & 1 & p_{n1} & p_{2n} & \cdots & p_{nk}
\end{bmatrix}
\]

Equation (6.2)
From Equation (6.2), ‘I’ is the identity matrix whereas ‘p’ represents the parity matrix. The parity elements from the parity matrix represent a linear combination of the data symbols ‘d’ for secure data storage in cloud environment and is formulated as,

\[(p_1, p_2, ..., p_n) = (f_1, f_2, ..., f_n) \times GM\]  \hspace{1cm} \text{Equation (6.3)}

From Equation (6.3), the generator matrix ‘GM’ is obtained by multiplying ‘Fi’ by ‘RM’ and is mathematically formulated as given below,

\[GM = F_i \times RM\]  \hspace{1cm} \text{Equation (6.4)}

From Equation (6.4), the resultant generator matrix ‘GM’ contains the original data file vectors of \(F_i\), thereby ensuring cloud data storage capacity in an efficient manner. With the application of the generator matrix with multiple threads, the transaction duration is reduced in significant manner. The cloud data storage verification algorithm is given as,

\textbf{Algorithm 6.1. Cloud data storage verification algorithm}

\textbf{Input} : Cloud Servers ‘CSi = CS1, CS2, ..., CSn’, Transaction ‘Ti = T1, T2, ..., Tn’, Cloud Users ‘CUi = CU1, CU2, ..., CUn’, data vector \(F_i = f_1, f_2, ..., f_n\).

\textbf{Output} : Improved cloud data storage capacity

1. \textbf{Begin}
2. For each cloud users CUi
3. For each transaction Ti
4. Measure linear vector using (6.1)
5. Calculate resultant matrix using (6.2)
6. Evaluate parity matrix using (6.3)
7. Evaluate generator matrix using (6.4)
The algorithm, given above, performs two steps. The first step evaluates the linear vector for storing the cloud data and computes its resultant matrix for minimizing the errors in data storage (i.e. to avoid duplication of data). The second step measures the parity and generator matrix for secure data storage. The generator matrix reproduces the original data file vectors of $F_i$ which results in the improvement of cloud data storage capacity. The generator matrix with multiple threads helps in reducing the transaction duration in a significant manner.

6.3.3.2. Poisson Exponential Distribution based Multi-Threaded State Transition Technique

The cloud user relies on the data stored in cloud servers in order to improve the cyclic progression ratio. Fig. 6.5 shows the block diagram of Poisson Exponential Distribution-based Multi-threaded State Transition mechanism.
The cloud users ‘A’, ‘B’ and ‘C’ simultaneously access the cloud data storage ‘CDS’ in the cloud environment. In order to maintain data consistency, the LE-PED framework uses Poisson Exponential Distribution based Multi-threaded State Transition mechanism. The Poisson Exponential Distribution model is a continuous process whose state represents the set ‘\{1, 2, 3, ..., n\}’ that corresponds to the cloud users ‘CUi’. In this technique, ‘\(\lambda\)’ denotes the arrival rate of cloud users that occurs according to the Poisson Exponential Distribution model and it iterates the process.
from ‘$\hat{t}$’ to ‘$i + 1$’. Finally, a cyclic progression for the storage states and the threads are assigned based on on-demand transaction request of the user.

The Poisson Exponential Distribution model stops the iterative process on the basis of the Multi-threaded State Transition with an arrival rate ‘$\lambda$’ and an interval time ‘$\alpha$’. The updates of the cloud user (i.e. the cloud user to be served next in the thread) are then mathematically formulated as given below.

$$\beta (\text{updates}) = \left[ \frac{\lambda}{\alpha} \right]$$  \hspace{1cm} \text{Equation (6.5)}

From Equation (6.5), updates ‘$\beta$’ of cloud user is performed in an effective manner with the objective of improving the data consistency in cloud storage. Each cloud user is served by a server one at a time based on the threading model, according to the first come to be served first. Upon successful completion of the service, the cloud user is reduced by one. Therefore, Poisson Exponential Distribution based Multi-threaded State Transition is then formulated as,

$$\begin{bmatrix}
\lambda & \cdots & \cdots \\
\alpha & (\lambda + \alpha) & \cdots \\
\cdots & \alpha & (\lambda + \alpha) \\
\cdots & \cdots & \cdots \\
\end{bmatrix}$$  \hspace{1cm} \text{Equation (6.6)}

From Equation (6.6), the Poisson Exponential Distribution model is considered stable only if ‘$\lambda < \alpha$’. On the other hand, if the arrival of cloud users is faster than service completions, the thread will grow indefinitely long and the Poisson Exponential Distribution model will not have a stable distribution. Therefore, the probability that the stable process is in state ‘$\hat{t}$’ is given as,

$$\text{Prob } (\text{state}_i) = (1 - \beta) \times \beta^i$$  \hspace{1cm} \text{Equation (6.7)}

From Equation (6.7), the number of cloud users is geometrically distributed by ‘$1 - \beta$’ with the average number of cloud users in the cloud environment being
Finally, a cyclic progression for the storage states \( \text{state}_i \) and the threads (i.e. the cloud user that has to be assigned with the cloud storage) are assigned depending on the on-demand transaction request of the user. As a result, the cyclic progression ratio increases with the increase in the number of cloud users to be served in the cloud environment. The algorithmic description of PED model is given as:

**Algorithm 6.2. Poisson exponential distribution algorithm**

**Input:** Cloud Servers \( \{CS_1, CS_2, \ldots, CS_n\} \), Transactions \( \{T_1, T_2, \ldots, T_n\} \), Cloud Users \( \{CU_1, CU_2, \ldots, CU_n\} \), arrival rate \( \lambda \), interval time \( \alpha \).

**Output:** Increased cyclic progression ratio

1. **Begin**
2. For each Cloud Users \( CU_i \)
3. For each Transactions \( T_i \)
4. Measure the updates using Equation (5)
5. Evaluate multi-threaded state transition using Equation (6)
6. Evaluate the probability measure using Equation (7)
7. End for
8. End for
9. **End**

The Poisson Exponential Distribution algorithm is described in three steps. It determines the arrival of cloud user on the basis of Poisson model with the access of cloud storage being performed in an Exponential Distribution model. Initially, this algorithm measures the arrival of cloud user (i.e. updates user arrivals in cloud) for improving data consistency in cloud storage. Then, the multi-threaded state transition is evaluated for providing the services to users in cloud. Finally, it measures the probability for identifying whether the stable process is in state‘\( t \)’. In this way, the cyclic progression for storage states are generated with the threads being assigned on the basis of on-demand transaction request of the cloud user resulting in improved cyclic progression ratio.
6.4. EXPERIMENTAL EVALUATION

The Amazon Access Samples dataset information is used for the transaction processing between cloud users and cloud servers. The information included in the Amazon Access Samples dataset comprises of dense dataset where less than 5% of the attributes were used for evaluating the data consistency state on the transactions using the proposed Linear Erasure and Poisson Exponential Distribution framework. The Amazon Access Samples dataset includes four categories of attributes including Person_Attribute, Resource_ID, Group_ID and System_Support_ID.

Experiments are conducted on several cloud users to identify the performance level of data consistency against the existing methods. The performance of Linear Erasure and Poisson Exponential Distribution (LE-PED) framework is measured based on the following parameters such as,

i) Cloud data storage capacity
ii) Transaction duration
iii) Cyclic progression ratio
iv) Data consistency state

Cloud data storage capacity refers to the data storage performed in cloud environment based on the number of transaction threads, data to be stored and the time taken for data storage. It is measured in terms of kilobits per second (Kbps) and is mathematically formulated as,

\[ DSC = T \times d \times time \]  

Equation (6.8)

From Equation (6.8), ‘DSC’ refers to the cloud data storage capacity, ‘T’ represents the number of transaction threads, ‘d’ states the data to be stored with respect to time ‘time’.

Transaction duration is the product of transaction data size ‘TDS’ and time taken for each data storage ‘Time (DS)’. The mathematical formulation of transaction duration is,
In Equation (6.9), the transaction duration for data storage is measured in terms of milliseconds (ms). Lower the transaction duration, more efficient the method is said to be.

The cyclic progression ratio (CPR) measures the ratio of cloud users assigned with data storage by cloud server in cloud environment to the total number of cloud users. The cyclic progression ratio is measured in terms of percentage (%).

\[
CPR = \frac{\text{No. of cloud users assigned with storage space}}{\text{Total cloud users}} \times 100
\]

Equation (6.10)

From Equation (6.10), the cyclic progression ratio ‘\text{CPR}’ is measured with respect to the cloud users ‘\text{CU}’. Higher the cyclic progression ratio, more efficient the method is said to be.

Data consistency level is defined as the ratio of the amount of data maintained correctly as per the last update by users to the total amount of data.

\[
\text{Data consistency level} = \frac{\text{Amount of data maintained as per last update}}{\text{Total amount of data (KB)}} \times 100
\]

Equation (6.11)

From Equation (6.11), the data consistency level is measured in terms of percentage (%).

6.5. RESULTS AND DISCUSSION

The LE-PED framework is compared with the Dynamic Audit Services using Hash Table (DAS-HT) [55] and Accountability for Data Sharing using Object
Centered (ADS-OC) [25] schemes. The tabulation for different parameters that are considered in proposed framework is explained below.

6.5.1. Cloud Data Storage Capacity

Table 6.1. Tabulation for cloud data storage capacity

<table>
<thead>
<tr>
<th>Number of transaction threads (T)</th>
<th>Cloud data storage capacity (Kbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Existing schemes</td>
</tr>
<tr>
<td></td>
<td>DAS-HT</td>
</tr>
<tr>
<td>2</td>
<td>76.4</td>
</tr>
<tr>
<td>4</td>
<td>85.1</td>
</tr>
<tr>
<td>6</td>
<td>95.3</td>
</tr>
<tr>
<td>8</td>
<td>99.2</td>
</tr>
<tr>
<td>10</td>
<td>105.2</td>
</tr>
<tr>
<td>12</td>
<td>115.6</td>
</tr>
<tr>
<td>14</td>
<td>128.1</td>
</tr>
</tbody>
</table>

Table 6.1 represents the cloud data storage capacity with different number of transaction threads. The number of transaction threads sent between the cloud user and cloud server is varied between 2 and 14 at different time intervals using Amazon Access Samples dataset.
From the Fig 6.6, it is illustrated that the cloud data storage capacity has increased using the proposed LE-PED framework when compared to the two other existing methods. This is because of the application of the Linear Erasure Correction that bears numerous failures in distributed storage environment and the evaluation of generation matrix using parity and identity matrix with the objective of improving the linearity in an efficient manner. This results in the improvement of cloud data storage capacity by 10 - 17 % when compared to the existing DAS-HT and ADS-OC schemes.

6.5.2. Transaction Duration

The comparison of transaction duration at the cloud server is presented in Table 6.2 with respect to the transaction data size ranging from 100 KB to 700 KB taken up for experimental purpose.
### Table 6.2. Tabulation for transaction duration

<table>
<thead>
<tr>
<th>Transaction data size (KB)</th>
<th>Transaction duration (ms)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Existing schemes</td>
<td>Proposed</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DAS-HT</td>
<td>ADS-OC</td>
<td>LE-PED</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>86.3</td>
<td>76.4</td>
<td>70.8</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>95.3</td>
<td>85.1</td>
<td>78.6</td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>110.5</td>
<td>95.3</td>
<td>85.4</td>
<td></td>
</tr>
<tr>
<td>400</td>
<td>115.2</td>
<td>99.2</td>
<td>91.6</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>119.8</td>
<td>105.2</td>
<td>97.7</td>
<td></td>
</tr>
<tr>
<td>600</td>
<td>122.2</td>
<td>115.6</td>
<td>106.8</td>
<td></td>
</tr>
<tr>
<td>700</td>
<td>138.4</td>
<td>128.1</td>
<td>121.9</td>
<td></td>
</tr>
</tbody>
</table>

From Fig. 6.7, it is illustrative that the transaction duration using LE-PED framework is reduced because the framework uses a cloud data storage verification algorithm where the parity elements from the parity matrix is a linear combination of the data symbols. With this linear combination, the generator matrix with multiple threads uses a multiplicative function. Therefore, the transaction duration with respect to data storage is reduced by 31 – 38 % when compared to existing DAS-HT and ADS-OC schemes.
6.5.3. Cyclic Progression Ratio

The cyclic progression ratio of LE-PED scheme is measured and is shown in Table 6.3. The measurement was considered with cloud users in the range of 5 to 35 using CloudSim simulator. With the increase in the number of cloud users, the cyclic progression ratio is also increased. Thus, it maintains the efficiency using the LE-PED framework.
Table 6.3. Tabulation for cyclic progression ratio

<table>
<thead>
<tr>
<th>No. of cloud users</th>
<th>Cyclic progression ratio (%)</th>
<th>Existing schemes</th>
<th>Proposed LE-PED scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>DAS-HT</td>
<td>ADS-OC</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>54.25</td>
<td>42.45</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>60.14</td>
<td>46.11</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>69.28</td>
<td>55.25</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>71.2</td>
<td>45.17</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td>73.3</td>
<td>49.27</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>71.11</td>
<td>51.08</td>
</tr>
<tr>
<td>35</td>
<td></td>
<td>74.23</td>
<td>55.2</td>
</tr>
</tbody>
</table>

As illustrated in Fig 6.8, the cyclic progression ratio is observed to high using LE-PED framework. This is because of the application of Poisson Exponential Distribution model, where the entry of cloud users is continuous. The on-demand transaction request of the user is made in an efficient manner. This results in the improvement of cyclic progression ratio by 18 – 38 % when compared to DAS-HT and ADS-OC schemes.
Fig. 6.8. No. of cloud users vs. cyclic progression ratio

6.5.4. Data Consistency Level

The data consistency state of LE-PED framework is measured and shown in Table 6.4. Different cloud users with varying transaction data size in the range of 50 to 500MB are considered using CloudSim simulator. Based on the tabulated values, the LE-PED framework has provided a better performance than the existing DAS-HT and ADS-OC schemes.
Table 6.4. Tabulation for data consistency level

<table>
<thead>
<tr>
<th>Transaction data size (MB)</th>
<th>Data consistency level (%)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Existing schemes</td>
<td>Proposed LE-PED scheme</td>
</tr>
<tr>
<td></td>
<td>DAS-HT</td>
<td>ADS-OC</td>
</tr>
<tr>
<td>50</td>
<td>67.36</td>
<td>70.36</td>
</tr>
<tr>
<td>100</td>
<td>69.10</td>
<td>72.10</td>
</tr>
<tr>
<td>150</td>
<td>72.23</td>
<td>74.20</td>
</tr>
<tr>
<td>200</td>
<td>74.35</td>
<td>76.52</td>
</tr>
<tr>
<td>250</td>
<td>75.68</td>
<td>79.10</td>
</tr>
<tr>
<td>300</td>
<td>77.10</td>
<td>80.20</td>
</tr>
<tr>
<td>350</td>
<td>78.36</td>
<td>82.20</td>
</tr>
<tr>
<td>400</td>
<td>82.35</td>
<td>84.63</td>
</tr>
<tr>
<td>450</td>
<td>83.65</td>
<td>86.35</td>
</tr>
<tr>
<td>500</td>
<td>85.68</td>
<td>88.32</td>
</tr>
</tbody>
</table>

From Fig 6.9, it is illustrative that the data consistency state is improved using the LE-PED framework. This is because of applying Poisson Exponential Distribution algorithm in LE-PED framework. With the application of Poisson Exponential Distribution algorithm, the on-demand transaction request of the cloud user is not only simplified for user transaction, but also the number of cloud users is geometrically distributed. This in turn maximizes the data consistency state by 8 - 12% using LE-PED framework when compared to DAS-HT and ADS-OC schemes.
6.6. SUMMARY

A Linear Erasure and Poisson Exponential Distribution (LE-PED) framework is designed to improve the data consistency level. The data storage capacity is improved by applying the Linear Erasure Correction model where multiple threads are issued. The storage states are initialized using Linear Erasure and Poisson Exponential Distribution. The LE-PED framework offers less transaction duration using cloud data storage verification algorithm. Finally, the application of Poisson Exponential Distribution-based Multi-threaded State Transition demonstrates that LE-PED framework provides increased cyclic progression ratio and therefore improves the data consistency level. The performance of LE-PED framework was found to be efficient when compared to other cloud data sharing methods namely DAS-HT and ADS-OC respectively.