JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

# Deadlock Prevention by Mutual Exclusion Process in Cloud Storage

[1]Deep Shikha, [2]Lalit Kumar
[1]Student, [2]Ph.D. Scholar
[1]CSED, [2]CSED
[1]Madan Mohan Malaviya University of Technology, Gorakhpur, India, [2]Motilal Nehru National Institute of Technology Allahabad, Prayagraj, India

***Abstract :*** Nowadays, users want to use cloud storage to remotely access and control resources without deadlock. For deadlock handing, developers used many deadlock handling strategies such as bully algorithm and other algorithms. In our work, we try to manage sessions to create a deadlock-free model by developing a modified version of the bully algorithm. Here in the bully algorithm, we try to manage various sessions by using the nested sleep function. These nested sleep function helps to create a deadlock-free system. This sleep function sleeps process for some milliseconds for completing the previous method. There are used sleep functions with enough time to manage these deadlock handling conditions without more delay.

*Index Terms* - **Cloud Storage, Mutual Exclusion, Deadlock, Bully Algorithm, Nested Sleep Function, Distributed System, Deadlock Prevention Techniques.**

## 1. INTRODUCTION

Cloud storage service has turned out to be fundamental pieces of the shopper's computing environment. Customers proceed to create and spend a large mass of data utilizing our computing device where Purchasers' information could be photographs, recordings, music, reports, well-being data, etc. Our computing device could be a Desktop, laptop, tablets, Personal Computers, cell phones, smart watch, and other wearable and additionally advance devices. As indicated by purchasers' computing design, a large amount of storage had been required. To download and transfer our information, they applied our own N.A.S. system. Since the establishment of N.A.S. is needed additional costs, the customers attach open cloud storage services [1][2][5].

However, the sum of free cloud storage available is insufficient to keep big volumes of buyer data. Many cloud-of-clouds services, which are a collection of open online storage, have been investigated to address this problem. A cloud-of-clouds is a cloud storage administration that uses many cloud storage providers. Cloud storage is the many monikers for a cloud-of-clouds that exists between these cloud storage options. A cloud-of-clouds gives a virus's cloud storage services to shoppers utilizing different-different open cloud storage services [4][6][8].

Regardless, examination of various customer's gadgets or possibly different purchasers to share their information is required. This shared prohibition method is fundamental when concurrent makes from multiple devices or potential purchasers occur on fused distributed storage benefits. Since public distributed storage changes over into distracted stockpiling, we can't use different APIs from open distributed storage. Although many billows of cloud administration frameworks are investigated, this can upset spreading them. Colossal assessments are proposed to give a common dismissal to a haze of mists administration framework. Regardless, their work relies upon the singular open-source cloud without consolidated capacity between distributed storage and correspondence between clients [7][9][10][11].

### 1.1 Mutual Exclusion

A mutex is a program object that averts concurrent entry to mutual exclusion. This idea is utilized in concurrent programming with the primary area (C. S.), a bit of code where processes or threads get to shared resources [15].

Various sorts of Mutual avoidance have side effects. For example, incredible semaphores concede halts, in which one procedure gets a semaphore; another approach gets a second semaphore. After that, both hold up till the other semaphore to be released. Different typical responses fuse starvation, in which a procedure never gets satisfactory assets to rushed to complete; need reversal, in which a higher need string sit tight for a lower-need series; and high idleness, in which response to hinders isn't expeditious[16][17].

Much research is a way for discarding the above effects, often with the target of guaranteeing non-blocking progress. No perfect arrangement is known. Blocking system calls are used to rest entire procedures. Until such calls form toward getting string safe, there was no suitable framework for relaxing a single series inside a process (see studying) [18].

### 1.1.1 Advantage of MutualExclusion

Favorable circumstances of Mutual exclusion are [20][21]:

- The facilitator allows only a solitary procedure at some random minute into each CS.
- It is sensible.
- Requests are yielded in the solicitation where they are gotten.
- No procedure holds up forever.
- It can be used for general resource distribution instead of administering overseeing common prohibition.

### 1.1.2 Disadvantage of Mutual Exclusion

- The organizer is a solitary purpose of failure, so in the event that it crashes, the whole system may go down [19].
- If process regularly hinders in the wake of making a solicitation, they can't recognize a dead organizer from "get to denied" since in the two cases facilitator doesn't answer.
- In a huge system, a solitary organizer needs to deal with all process.

## 1.2 Deadlock

Deadlock occurs when a massive number of techniques are blocked in light of how every method holds a benefit and sits tight for another advantage gotten by some different systems. Every single required thing is made underneath [22][23][25]:

- Deadlock is when a massive number of strategies are obstructed in light of how every method holds resources and sits tight for another benefit grabbed by a different methodology.
- A technique in the working system utilizes various resources and uses resources in a going with way.

1) Requests a benefit
2) Use the benefit
3) Releases the benefit

Consider a model when two trains are coming toward one another on a similar track, and there is just a lone track; none of the trains can move once they are before one another. The indistinguishable circumstance happens in the working structure when more methods hold two or three resources and keep them together for resources held by other(s). For instance, in the underneath figure, Process (1) is holding Resource (1) and sitting tight for resources (2) which are obtained by techniques (2), and systems (2) are keeping it together for resources (1).

Stop can create if the going with four conditions holds meanwhile (Necessary Conditions)

- **Mutual exclusion:** at least one than one asset is non-shareable (Only one procedure can use at once)
- **Hold and Wait:** A procedure is holding at any rate one asset and hanging tight for purchases.
- **No Preemption:** An asset can't purchase from a procedure except if the process discharges the purchase.
- **Circular Wait:** A lot of procedures are hanging tight for one another in a round structure.

### 1.2.1 Methods for taking care of stop

There are three unique approaches to manage gridlock [24].

- **Deadlock balancing activity or avoiding:** The reasoning is not to give the structure access to the gridlocked state. One can zoom into each class just; Prevention is finished by nullifying one of them as of late referenced special conditions for gridlock.
- **Avoidance is a kind of forefront in nature:** By utilizing "Avoidance," we need to make an assumption. We have to guarantee that all data about resources that systems WILL require is known to us going before executing the strategies. We utilize Banker's count (Which is such a blessing from Dijkstra) to maintain a strategic distance from gridlock.
- **Deadlock area and recovery:** Let stop occur, by then do apportionment to manage it once happened.
- **Ignore the issue entirely:** If gridlock is extraordinarily exceptional, by then, given it a chance to happen and reboot the system. This is the approach that the two Windows and UNIX take.3.2.3 Deadlock Handling Algorithm

For deadlock handling, we can utilize different algorithms. Be that as it may, we use a bully algorithm for deadlock handling on the cloud server in the proposed work. So the all insights concerning the bully algorithms, for example, definition, algorithms, and different things, are composed underneath.

## 1.3 Bully Algorithm

In appropriated registering, the domineering jerk calculations are a procedure for logically picking a facilitator or pioneer from a social event of conveyed P.C. forms. The techniques with the most effective procedure I.D. number from among the non-fizzled process are chosen as the Coordinator [12][15].

The calculations acknowledge that:

- ➢ The framework is synchronous;
- ➢ The Processes may lose at any minute, including in execution of the calculations;
- ➢ A process tumbles by forestalling and returns from disappointment by restarting;
- ➢ There is a disappointment discoverer which perceives fail out procedure;
- ➢ Message transport between approaches is trustworthy, and each process knows its very own procedure id and address and that of each unique process.

### 1.3.1 Algorithm:

The algorithms use the going with message types:

- ➢ Election Message: Sent to report the race.
- ➢ Answer (Alive) Message: Response to the Election message.
- ➢ Coordinator (Victory) Message: Sent by the champ of the race to report triumph

- When a procedure P recovery from the disappointment or the disappointment identifier demonstrates that the present facilitator has fizzled, P plays out the going with exercises:
- If P has the most significant procedure id, it sends a Victory message to each extraordinary procedure and transforms into the new Coordinator. Something different, P imparts an Election message to every remarkable strategy with higher procedure I.D.s than itself.
- If P finds no Solution consequent to sending an Election message, it imparts a Victory message to every unique technique and transforms into the Coordinator.
- If P finds a Solution from a procedure with a higher I.D., it sends no further messages for this decision and hanging tight for a Victory message. (If there is no Victory message after some time, it restarts the procedures close to the beginning.)
- If P gets an Election message from another procedure with a lower I.D., it returns an Answer message. It starts the choice strategy close to the begin, by sending an Election message to higher-numbered shapes.
- If P gets a message from the Coordinator, it sees the sender as the Coordinator.

## 2 Literature Review

Numerous methodologies have been investigated to give virtual distributed storage utilizing a few accessible open distributed storage. Several methods for providing virtual distributed storage using a few available distributed storage administrations have been studied. Friendbox, Cloud-RAID, Uni4Cloud, RAIN, and RAIC are some of the different types of explorations associated with haze of-mists. For cloud reinforcement stockpiling, NCCloud, an intermediary-based, multiple dispersed storage framework, is offered [2][3][5][6]. It eliminates non-critical failure adaptability and allows for a cost-effective remedy when a cloud fails. However, these works do not support a wide range of consumers or distinct gadget conditions. There isn't any attention given to a common avoidance approach. Methodologies for providing only assistance have been investigated. They are also worker-based methods.

Deepsky is a storage solution that uses commercial storage cloud services to support availability and secrecy. However, it is a server-side protocol. The rising popularity of cloud storage services has prompted businesses that handle sensitive data to consider using them for their storage needs.

Hybris [4] is a multi-cloud storage mechanism that the authors present in this paper. The data is spread across multiple public clouds through replication or eraser coding—this aids in the continuity of various authors. However, communication is required across clients and the cloud. We introduce Hybris Key-Value Store, the first reliable hybrid cloud storage system that addresses these issues by combining private and public cloud storage.

Hybris powerfully reproduces metadata on confided in private premises (private cloud), independently from information scattered (utilizing replication or deletion coding) across numerous untrusted public mists. Hybris keeps up with metadata put away on private premises in the request for not many bytes per key, keeping away from the versatility bottleneck at the private cloud. Thus, the crossover configuration permits Hybris to productively and powerfully endure cloud blackouts and expected perniciousness in mists without the overhead. In particular, to endure up to f malevolent mists, in the usual instance of the Hybris variation with information replication, composes repeat information across f + 1 mists, though peruses include a solitary cloud. In the most pessimistic scenario, simply up to f various mists are utilized.

This is impressively better than before multi-distributed storage frameworks requiring exorbitant 3f + 1 mists to veil f conceivably noxious mists. At last, Hybris uses solid metadata consistency to ensure solid information consistency to Hybris applications with no changes to the steady open mists in the long run. We executed Hybris in Java and assessed it is utilizing a progression of miniature and full-scale benchmarks. Our outcomes show that Hybris fundamentally beats practically identical multi-distributed storage frameworks and approaches exposed bone product public distributed storage execution.

S.L.A. [7] gives two trees and token-based circulated standard avoidance calculations for distributed storage. Be that as it may, it likewise needs between customer correspondence. — In Cloud Computing, Service Level Agreement (S.L.A.) is an agreement that defines a level and a sort of QoS between a cloud supplier and a customer. Since applications in the Cloud share assets, we propose two tree-based circulated standard avoidance calculations that help the S.L.A. idea. The first one is a modified adaptation of the need-based Kanrar-Chaki calculation [1]. Interestingly, the subsequent one is a clever calculation dependent on the Raymond calculation [2], where a cutoff time is related to each solicitation. In the two cases, we mean developing the Critical Section execution rate to diminish the quantity of S.L.A. infringement.

MetaSync [9] gives a record synchronization administration on top of distributed storage suppliers. Be that as it may, it offers affix log synchronization as it were—cloud-based file synchronization administrations, like Dropbox, region overall asset for some million so fusers. In any case, individual administrations regularly have tight asset limits, experience the ill effects of transitory out gesoreven closures, and at times quietly bad or hole client information. We configuration, carry out and assess MetaSync, a protected and dependable file synchronization administration that utilizes different cloud synchronization benefits as untrusted stockpiling suppliers.

Cloud Types [7][9] modify deliberations for distributed storage with synchronization supporting possible consistency. However, it additionally needs correspondence between customers. Cell phones ordinarily access shared information put away on a worker. To guarantee responsiveness, numerous applications keep up with nearby reproductions of the communicated information that remain immediately available regardless of whether the worker is slow or briefly inaccessible. Despite its clear effortlessness and shared trait, this situation can be shockingly difficult. Specifically, a right and solid execution of the correspondence convention and the conflict goal to accomplish possible consistency is overwhelming in any event for specialists.

SaveMe [12] is a clever online distributed storage framework that gives a secure cloud to put away private information by consolidating general society and individual distributed storage accessible to every client. There are many distributed storage specialist co-ops, and many of them give critical measures of free stockpiling. Clients ought to have the option to deal with the extra room accessible to them as a solitary brought together space. Something else, client records will be dissipated over various suppliers, invalidating the upside of utilizing distributed storage. One more critical worry of clients is that their information is at the removal of the specialist co-ops. Ultimately, as opposed to the promotion of exceptionally high accessibility, mishaps will, in general, occur. Indeed, even the most accessible cloud administrations are not available quite often or even each month.

Mutual Exclusion Method in Client-Side Aggregation of Cloud Storage [11] "Customers proceed to deliver and spend an immense measure of information utilizing shopper electronic gadgets. To store purchasers' information, an enormous volume of capacity is required. A haze of-mists virtual distributed storage administration using heterogeneous public distributed storage administrations can be an answer. In this paper, we propose a common prohibition technique for concurrent gets to in haze of-mists. The proposed plan incorporates a nuclear activity for haze of-mists, gridlock recovery, and race condition control.

DEPSKY is the expanding fame of distributed storage administrations that has lead organizations that handle basic information to contemplate utilizing these administrations for their capacity needs. Clinical record data sets, power framework verifiable data, and financial information are a few instances of fundamental information that could move to the cloud.

NCCloud[13][15]: A Network-Coding-Based Storage System in a Cloud-of-Clouds, "To give adaptation to non-critical failure to distributed storage, ongoing investigations propose to stripe information across numerous cloud merchants. Notwithstanding, if a cloud experiences a super durable disappointment and loses all of its knowledge, we need to fix the lost data with the assistance of the other enduring mists to safeguard information repetition. We present an intermediary-based capacity framework for issue open-minded numerous distributed storage called NCCloud, which accomplishes a practical fix for a long-lasting single-cloud disappointment.

## 3. Proposed Model

In this part of the paper, we discussed the methodology used to accomplish our goal of preventing deadlock conditions in cloud storage.

### 3.1 Introduction

To give a single archive structure in a cloud of clouds, tables of documents organization are keys. Archive tables should be secure in a distributed storage or diverse distributed storage. Using set away record table, clients can gain report system information of haze of mists. When a client exchanges a record, the client ought to invigorate a report table to deflect archive table brokenness among all clients. Report table of the dimness of clouds can be a record grain, an inventory grain, or a record system grain. In this paper, we offer record tables to give regular preclusion in a cloud of clouds.

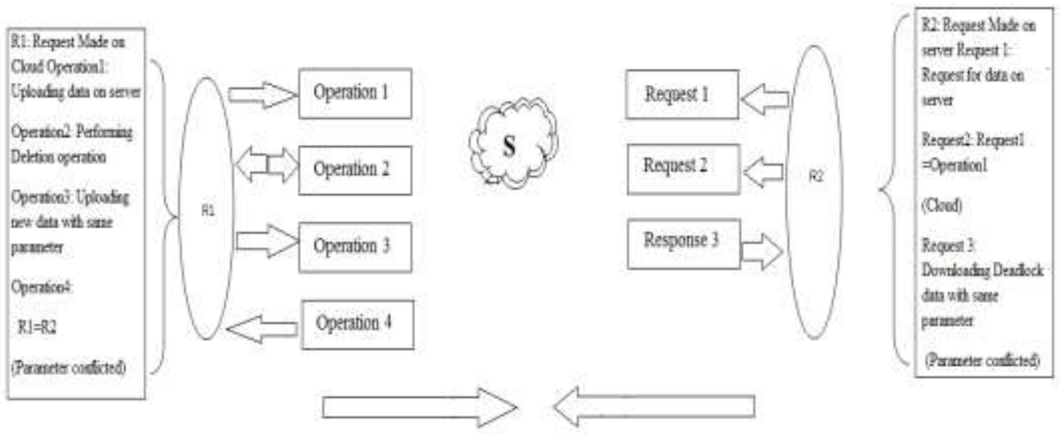
R1: Request Made on Cloud Operation1: Uploading data on server
Operation2: Performing Deletion operation
Operation3: Uploading new data with same parameter
Operation4:
R1=R2
(Parameter conflicted)
R1
Operation 1
Operation 2
Operation 3
Operation 4
S
Request 1
Request 2
Response 3
R2
R2: Request Made on server Request 1: Request for data on server
Request2: Request1 =Operation1
(Cloud)
Request 3: Downloading Deadlock data with same parameter
(Parameter conflicted)

**Figure 1:** Deadlock Condition in Cloud Server

Deadlock Condition in Cloud Server: Figure 4.3 shows an existing scenario of deadlock condition while serving different clients request in cloud server. This is because if both processes access the resource at same time, it leads to error. In fig 1.1 there is two recourses R1 and R2 sharing same cloud at responsive time R1 performing operation2 and R2 performing Request2. As a result there will be error due to R1.R1 performing operation2 for same file which is requested by R2 but there is an open source cloud detected on S .Hence it is difficult for server and may cause a deadlock and excepted error will be generated by cloud S.

### 3.2 Algorithm

The algorithm for proposed work is:

```
Step 1.    Begin
Step 2.    turn=0, flag [0] =flag [1] =false;
Step 3.    P1 executes 8, 6, and 4 (Since flag [0] was false, the next  instruction P1 will execute is 2.)
Step 4.    P1 executes and now P0 and P1 are both in their critical sections.
Step 5.    Flag[1]==true
Step 6.    flag [0]==true
Step 7.    turn==1
```

### 3.3 Working Process of Proposed Model

```
Step 1: Begin
Step 2: Declare variable Sleep.
Step 3: Initialize variables
If (Sleep 8)
{
            If (Sleep 6)
            {
                    If (sleep 4)
                    {
                            If (sleep 2)
                            {
```

```
Step4
                                  }}}
```

Step 4: Read request from user.
Step 5: If sleep=0
Load Data
Step 6: Stop

### 3.4 Flow Diagram for Proposed Model

The flow diagram for uploading any file in the cloud server is carried out as shown in the flow diagram in figure 1.

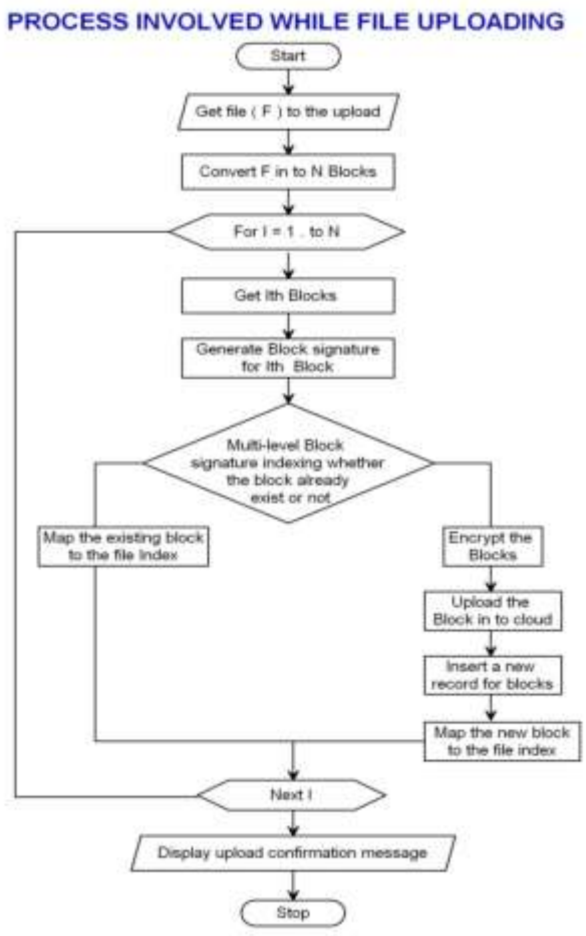
PROCESS INVOLVED WHILE FILE UPLOADING
Start
Get file ( F ) to the upload
Convert F in to N Blocks
For I = 1 . to N
Get Ith Blocks
Generate Block signature for Ith Block
Multi-level Block signature indexing whether the block already exist or not
Map the existing block to the file Index
Encrypt the Blocks
Upload the Block in to cloud
Insert a new record for blocks
Map the new block to the file index
Next I
Display upload confirmation message
Stop

**Figure 2:** Flow Diagram for Proposed Model

### 3.5 Use Case Diagram

Use Case Diagram for the Proposed work is designed below:

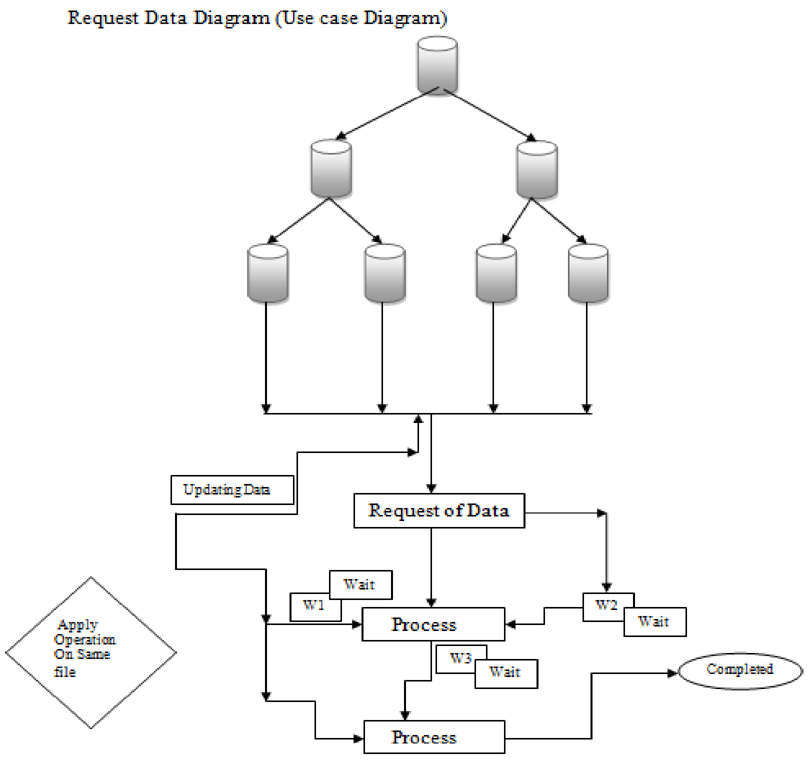

**Figure 3:** Use case diagram of Proposed Work

### 3.5 Mutual Exclusion Process

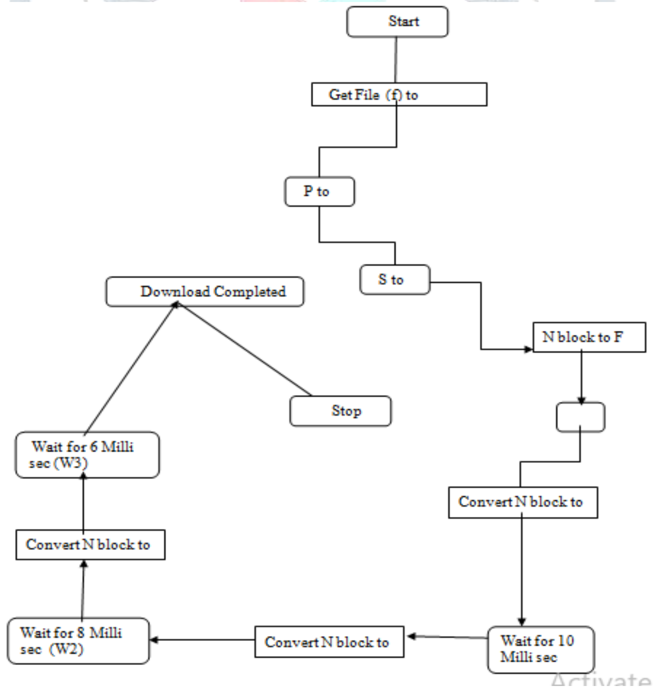

**Figure 4:** Mutual Exclusion Process for Proposed System

### 4. Result & Analysis of Proposed Work

The algorithm discussed in this work is implemented using various Visual Studio frameworks. A cloud space was created on the system itself and dummy cloud users and admin accounts were created. The various results obtained while testing the developed algorithm for mutual exclusion operation has been shown here.

The results obtained are shown figure 4.1. Green line used for data provider red line for cloud server and blue line for user .W1,w2,w3,w4 are represent waiting for server response left side scaling shows the sleeping scale during execution of request on server."As it is clear from graph that data provider will work without invoking any wait operation " and the decreasing sloping of user and cloud line shows that there will be less treading possible when sleeping time become less for both the server and user.

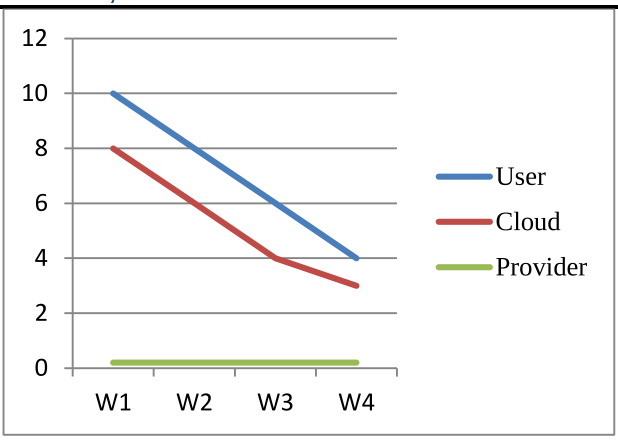

**Figure 5:** Comparison graph for File mutual Exclusion

Here Pie chart (Figure 5) represent waiting time for file downloading after using the sleep method at various level.

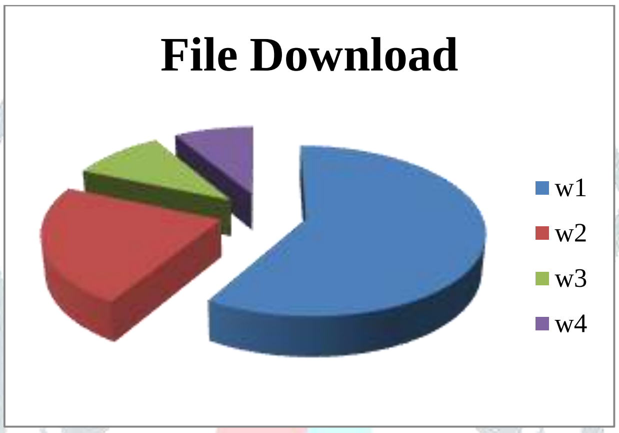

**Figure 6:** File downloads Wait Time

File Data Download = Download Data(Sequential Time Interval 1stwait on server)*
+ Download Data (Sequential Time Interval 2nd wait on server)
+ Download Data (Sequential Time Interval 3rd wait on serve
+ Download Data (Sequential Time Interval 4th wait on server).

DATA will downloaded from cloud side to client side and after a mili sec client site encryption back on cloud and check for changes made by provider and again do the same work untill file is being uploaded on cloud and downloaded on server.

## 5. Conclusions

Here this proposed works helps the cloud server provider to manage more clients at the channel. Due to this we used a "modified version of the bully algorithm". In "the modified version of the bully algorithm", we used some sleep functions 3x times. Using the sleep function we try to manage the file's access on the cloud server without any fault and error. This sleep method is holding the process before execution in the critical section. This holding process is done here using sleep functions at various levels. This whole work helps to remove the deadlock condition during the service of the cloud. This work is also helping to increase the accessibility of data over the cloud.

## REFERENCES

[1] G. Song, S. Kim, and D. Seo, "SaveMe: Client-Side Aggregation ofCloud Storage," IEEE Trans. Consumer Electronics, vol. 61, no. 3, pp.302-310, Aug. 2016.
[2] B. Mao, S. Wu, and H. Jiang, "Exploiting workload characteristics and service diversity to improve the availability of cloud storage systems," IEEE Trans. on Parallel and Distributed Systems, vol.27, no.7, 2016, pp.2010-2021.
[3] S. Han, H. Shen, T. Kim, A. Krishnamurthy, T. Anderson, and D. Wetherall, "MetaSync: File Synchronization Across Multiple Untrusted Storage Services," In Proc. of USENIX Annual Technical Conf., 2015, pp.83-95.
[4] E. Bocchi, I. Drago, and M. Mellia, "Personal Cloud Storage: Usage, Performance and Impact of Terminals," In Proc. of IEEE CloudNet, 2015, pp.106-111.
[5] A. Bessani, M. Correia, B. Quaresma, F. Andre, and P. Sousa, "DepSky: dependable and secure storage in a cloud of clouds", In Proc. of the 6th conf. on Computer systems, 2011, pp.31-46.
[6] M. A. AlZain, E. Pardede, B. Soh, and J. A. Thom, "Cloud Computing Security: From Single to Multi-Clouds", In Proc. of the 45th Hawaii Intl. Conf. on System Science, 2012, pp.5490-5499.
[7] A. Sampaio and N. Mendonca, "Uni4Cloud: an approach based on open standards for deployment and management of multi-cloud applications", In Proc. of the 2nd Intl. Workshop on Software Engineering for Cloud Computing, 2011, pp.15-21.
[8] R. Gracia-Tinedo, M. Sanchez-Artigas, A. Moreno-Martınez and P. Garcıa-Lopez, "FriendBox: A Hybrid F2F Personal Storage Application", In Proc. of IEEE 5th Intl. Conf. on Cloud Computing, 2012, pp.131-138.

[9] G. Zhaoz, M. G. Jaatun, A. Vasilakos, A. A. Nyre, S. Alapnesy, Q. Yue,and Y. Tangz, "Deliverance from Trust through a Redundant Array of Independent Net-storages in Cloud Computing", In Proc. of IEEE Infocom 2011 Workshop on Cloud Computing, 2011, pp.631-636.
[10] M. G. Jaatun, G. Zhao, A. Vasilakos, A. A. Nyre, S. Alapnes, and Y. Tang. (2012, July). The design of a redundant array of independent net-storages for improved confidentiality in cloud computing. Journal of Cloud Computing [Online]. 1(13).
[11] D. Decasper, A. Samuels, and J. Stone, "Redundant array of independent clouds", US 20120047339 A1, Aug. 20, 2012.
[12] D. Dobre, P. Viotti, and M. Vukolic, "Hybris: Robust Hybrid Cloud Storage," In Proc. of ACM SoCC, 2014, pp.1-14.
[13] H. Graupner, K. Torjura, P. Berger, C. Meinel, and M. Schnjakin, "Secure Access Control for Multi-Cloud Resources," In Proc. of IEEE LCN, 2015, pp.722-729.
[14] H. C. H. Chen, Y. Hu, P. P. C. Lee, and Y. Tank, "NCCloud: A Network-Coding-Based Storage System in a Cloud of clouds," IEEE Trans. Computers, vol.63, no.1, pp.31-44, Jan. 2014.
[15] A. Moreno-Martinez, R. Gracia-Tinedo, M. Sanchez-Artigas, and P. Garcia-Lopez, "Friendbox: A cloudified f2f storage application," in Proc. IEEE International Conference on Peer-to-Peer, 2012, pp. 75-76.
[16] M. Schnjakin, and C. Meinel, "Evaluation of Cloud-RAID: A Secure and Reliable Storage Above the Clouds," In Proc. IEEE Intl. Conf. on Computer Comm. and Net., 2013, pp.1-9.
[17] N. E. Beckman, "A Survey of Methods for Preventing Race Conditions", 2006.
[18] M. Benchaiba, A. Bouabdallah, N. Badache, and M. Ahmed-Nacer, "Distributed mutual exclusion algorithms in mobile ad hoc networks: anoverview", ACM SIGOPS Operating Systems Review, vol. 38, no. 1,pp.74-89, Jan. 2004.
[19] K. W. Preslan, S. R. Soltis, C. J. Sabol, M. T. O'Keefe, G. Houlder and J. Coomes, "Device Locks: Mutual Exclusion for Storage Area Networks", In Proc. of 16th IEEE Symp. on Mass Storage Systems, 1999, pp.262-274.
[20] S. Lang, R. Latham, D. Kimpe, and R. Ross, "Interfaces for Coordinated Access in the File System", In Proc. of Cluster Computing and Workshops in IEEE Intl. Conf. on CLUSTER, 2009, pp.1-9.
[21] M. T. O'Keefe, "Shared File Systems and Fibre Channel", In Proc. of the 6th Goddard Conf. on Mass Storage Systems and Technologies in Cooperation with the 15th IEEE Symp. on Mass Storage Systems, 1998, pp.1-16.
[22] J. Lejeune, L. Arantes, J. Sopena, and P. Sens, "Service Level Agreement for Distributed Mutual Exclusion in Cloud Computing", In Proc. Of IEEE/ACM Intl. Symp. on CCGrid, 2012, pp.180-187.
[23] S. Burckhardt, M. Fahndrich, and D. Leijen, "Cloud Types for Eventual Consistency," LNCS, vol. 7313, 2012, pp.283-307.