Fragmentation and Duplication of Data for the Best Cloud Performance and Security (FDBPS)

Prof. Jaya Kumar B L[1], **Saranya C**[2]

[1]Assistant Professor, Department of Computer Science and Engineering, South East Asian College of Engineering and Technology

[2]M. Tech, Computer Science and Engineering, South East Asian College of Engineering and Technology

Abstract: ***The security mainly egresses from carrying over the data to a service provider which is of third party i.e. the cloud service. To optimally look into the cloud service, one would definitely pick out its service due to its huge quality of services offered. Contrary to its vantage point, it is proved to have malevolence in its overhaul. One such main liability in the cloud service is the compromise of its security which in turn derogates its overall attainment. This will betide due to two prudent reasons, i.e. either by the external attackers, or due to the compromise by the node within the cloud by itself. Therefore, high security metrics are required to be shield the data within the cloud. Along with the defending quality, the hired strategy of security must also deliberate on the optimization of the data retrieval time. Consequently, the proposed structure of Fragmentation and Duplication of data for the best cloud performance and security will collectively approach the security as well as the performance issues that were witnessed in the former cryptographic methods. In the FDBPS methodology, division of the files into fragments, and replicating the fragmented data over the cloud nodes are done. Only a lone fragment of a data file will be found in a single node. This ensures that there won't be a considerable amount of data casualties even when a node is being hacked or attacked.***

Keywords: security, service provider, third party, security metrics, optimization, Fragmentation and Duplication *of data*

1. Introduction

The cloud computing paradigm has reformed the usage and management of the information technology infrastructure. Cloud computing is characterized by on-demand self-services, ubiquitous network accesses, resource pooling, elasticity, and measured services. The aforementioned characteristics of cloud computing make it a striking candidate for businesses, organizations, and individual users for adoption. However, the benefits of low-cost, negligible management (from a users perspective), and greater flexibility come with increased security concerns. Security is one of the most crucial aspects among those prohibiting the wide-spread adoption of cloud computing. Cloud security issues may stem due to the core technology's implementation (virtual machine (VM) escape, session riding, etc.), cloud service offerings (structured query language injection, weak authentication schemes, etc.), and arising from cloud characteristics (data recovery vulnerability, Internet protocol vulnerability, etc.).

For a cloud to be secure, all of the participating entities must be secure. In any given system with multiple units, the highest level of the system's security is equal to the security level of the weakest entity. Therefore, in a cloud, the security of the assets does not solely depend on an individual's security measures. The fellow entities may stipulate a favorable circumstance to an attacker to tackle the user's defenses. The off-site data storage cloud utility requires users to move data in cloud's virtualized and shared environment that may result in various security concerns. Pooling and elasticity of a cloud, allows the physical resources to be shared among many users. Moreover, the shared resources may be reassigned to other users at some instance of time that may result in data compromise through data recovery methodologies.

Furthermore, a multi-tenant virtualized environment may result in a VM to escape the bounds of virtual machine monitor (VMM). The escaped VM can interfere with other VMs to have access to unauthorized data. Similarly, cross tenant virtualized network access may also compromise data privacy and integrity. Improper media sanitization can also leak customer's private data. A security framework is considered to be secured, if the underlying assumptions of the framework are weaker. The assumption parameter identifies the components that are assumed to be fully trusted, semi trusted, or distrusted to provide security features in an MCC environment. semi trusted means some functions are assumed to be done perfectly but some may be compromised, for example storage may be exposed but computation is properly conducted .

The data outsourced to a public cloud must be secured. Unauthorized data access by other users and processes (whether accidental or deliberate) must be prevented. As discussed above, any weak entity can put the whole cloud at risk. In such a scenario, the security mechanism must substantially increase an attacker's effort to retrieve a reasonable amount of data even after a successful intrusion in the cloud. Moreover, the probable amount of loss (as a result of data leakage) must also be minimized. A cloud must ensure throughput, reliability, and security. A key factor determining the throughput of a cloud that stores data is the data retrieval time.

In large-scale systems, the problems of data reliability, data availability, and response time are dealt with data replication strategies. Nevertheless, depositing replicas data on multiple of nodes elevates the attack surface for that discrete data. For instance, storing m replicas of a file in a cloud instead of one replica increases the probability of a node holding file to be chosen as attack victim, from 1 n to m n, where n is the total number of nodes. From the above discussion, we can deduce that both security and performance are critical for the next generation large-scale systems, such as clouds.

Security is one of the most crucial aspects among those prohibiting the wide-spread adoption of cloud computing. Cloud security issues may stem due to the core technology's implementation (virtual machine (VM) escape, session riding, etc.), cloud service offerings (structured query language injection, weak authentication schemes, etc.), and arising from cloud characteristics (data recovery vulnerability, Internet protocol vulnerability, etc.). For a cloud to be secure, all of the participating entities must be secure. In any given system with multiple units, the highest level of the system's security is equal to the security level of the weakest entity. Therefore, in a cloud, the security of the assets does not solely depend on an individual's security measures. The neighboring entities may provide an opportunity to an attacker to bypass the user's defenses.

The data outsourced to a public cloud must be secured. Unauthorized data access by other users and processes (whether accidental or deliberate) must be prevented. As discussed above, any weak entity can put the whole cloud at risk. In such a scenario, the security mechanism must substantially increase an attacker's effort to retrieve a reasonable amount of data even after a successful intrusion in the cloud. Moreover, the probable amount of loss (as a result of data leakage) must also be minimized.

To that end, this paper conjointly habituates an approach that meets the problem of security and performance with a secure data replication problem. this report constitutes the Fragmentation and Duplication of data for the Best cloud performance and security(FDBPS) that fairly fragments uploaded data into multiple parts and splits it to multiple nodes. In the FDBPS methodology, division of the files into fragments, and replicating the fragmented data over the cloud nodes are done. Only a lone fragment of a data file will be found in a single node. This ensures that there won't be a considerable amount of data casualties even when a node is being hacked or attacked.

2. Literature Survey

Bernd Grobauer, T Walloschek and E Stocker (2011), tells that, From a cloud customer perspective, the right hand side dealing with probable magnitude of future loss isn't changed at all by the cloud computing: the consequences and ultimate cost of, say, a confidentiality breach, is exactly the same regardless of whether the data breach occurred within a cloud or a conventional IT infrastructure. For a cloud service provider, things look somewhat different: because cloud computing systems were previously separated on the same infrastructure, a loss event could entail a considerably larger impact.[1].

Sudhi Seshachala (2015), conveys that Any discussion involving data must address security and privacy, especially when it comes to managing sensitive data, we mustn't forget "code space" and what happened to it after its AWS EC2 console was hacked and its data eventually deleted, forcing the company basically outsources everything it has. Of course, your cloud service provider is expected to manage and safeguard the underlying hardware infrastructure of a deployment, however remote access is your responsibility and in any case, no system is perfectly secure [2].

IEEE Published in 2012, Secure overlay cloud storage with access control and assured deletion. While we can now outsource data backup to third-party cloud storage services so as to reduce data management costs, security concerns arise in terms of ensuring the privacy and integrity of outsourced data. We design FADE; a practical, implementable, and readily deployable cloud storage system that focuses on protecting deleted data with policy-based file assured deletion. [3].

3. Problem Statement

In Existing system Outsourcing data to a third-party administrative control, as is done in cloud computing, gives rise to security concerns. The data compromise may occur due to attacks by other users and nodes within the cloud. Therefore, high security measures are required to protect data within the cloud. However, the employed security strategy must also take into account the optimization of the data retrieval time. For a cloud to be secure, all of the participating entities must be secure. In any given system with multiple units, the highest level of the system. Security is equal to the security level of the weakest entity.

Therefore, in a cloud, the security of the assets does not solely depend on an individual's security measures. The neighboring entities may provide an opportunity to an attacker to bypass the user's defenses.

4. System Architecture

We propose FDBPS that collectively approaches the security and performance issues. In the FDBPS methodology, we divide a file into fragments, and replicate the fragmented data over the cloud nodes. Each of the nodes stores only a single fragment of a particular data file that ensures that even in case of a successful attack, no meaningful information is revealed to the attacker.
Furthermore, the FDBPS methodology does not rely on the traditional cryptographic techniques for the data security; thereby relieving the system of computationally expensive methodologies.

We show that the probability to locate and compromise all of the nodes storing the fragments of a single file is extremely low. We also compare the performance of the FDBPS methodology with ten other schemes. The higher level of security with slight performance overhead was observed. The implementation is done in three steps as follows.

a) Fragmentation

The security of a large-scale system, such as cloud depends on the security of the system as a whole and the security of individual nodes. A successful intrusion into a single node may have severe consequences, not only for data and applications on the victim node, but also for the other nodes. The data on the victim node may be revealed fully because of the presence of the whole file.

A successful intrusion may be a result of some software or administrative vulnerability. The file owner specifies the fragmentation threshold of the data file is specified to be generated by. The file owner can specify the fragmentation threshold in terms of either percentage or the number and size of different fragments.

The percentage fragmentation threshold, for example, can dictate that each fragment will be of 5% size of the total size of the file. Alternatively, the owner can generate separate file containing information about the fragment number and size, for instance, fragment 1 of size 4,000 Bytes, fragment 2 of size 6,749 Bytes.

The owner of the file is the best candidate to generate fragmentation threshold as he is very well aware about the significant information from the file. The owner can best split the file such that each fragment does not contain significant amount of information. The default percentage fragmentation threshold can be made a part of the Service Level Agreement (SLA), if the user does not specify the fragmentation threshold while uploading the data file.

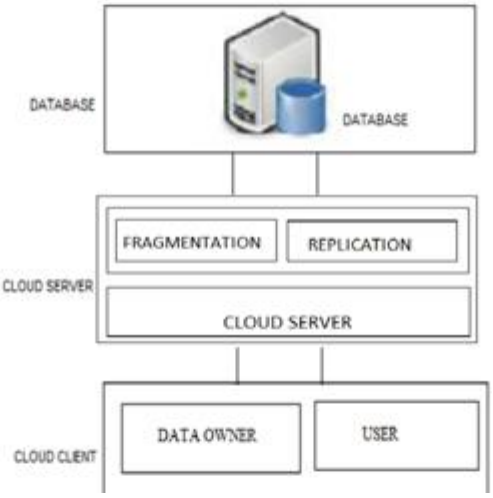


Figure 1: Software Architecture of FDBPS method

b) Fragment Placement

To provide the security while placing the fragments, the concept of fragmentation is used that was originally used for the channel assignment problem.

In this process, this loses some of the central nodes that may increase the retrieval time. But it achieves a higher security level.

If anyhow the intruder compromises a node and obtains a fragment, he cannot determine the location of the other fragments. The attacker can only keep on guessing the location of the other fragments. Because the nodes are separated by Fragmentation.

c) Replication

To increase the data availability, reliability, and improve data retrieval time, it also performs a controlled replication. It places the fragment on the node that provides the decreased access cost with an objective to improve retrieval time for accessing the fragments for reconstruction of original file.

User 1

1) First user makes registration by providing his details.
2) Users login by using username and password
3) After login he chooses the file and split that file in to number of fragments.
4) After splitting the files into fragments he view the fragments and details of the fragments
5) Next distribute the fragments by selecting add ,replicate, and delete operation to the servers
6) Receive the file from the server if u want to read the file

User 2

1) First user makes registration by providing his details.
2) Users login by using username and password
3) After login he chooses the file and split that file in to number of fragments.
4) After splitting the files into fragments he view the fragments and details of the fragments
5) Next distribute the fragments by by selecting add ,replicate, and delete operation to the servers
6) Receive the file from the server if u wants to read the file.

Server's module description

1) Receive the fragments (i.e...Wat user has distributed the splited fragments) from the user and store the fragments
2) Server receives the request details from user and give response for the particular request what user has requested
3) Division and Replication of Data in the Cloud can be done by the servers by Iterative Load Balancer

5. Proposed FDBPS method and Flow of Architecture

The FDBPS methodology does not reckon the traditional techniques of cryptographic for the prophylactic point of data; to that end obliterating the methodology which stands computationally expensive. The FDBPS makes an exposition that the relative likelihood to ascertain and imperil the entire fragments stored on the nodes of a single file is drastically low. Here the FDBPS methodology is being counterpart with ten other schemes. The elevated level of security with slight performance soar was observed. This developed scheme is for the carried over data that takes into consideration both the security and performance. The proposed scheme fragments and replicates the data file over nodes of the cloud. The proposed FDBPS scheme assures that even in the condition of a successful attack, no meaningful information is gained by the attacker. This scheme does not reckon the traditional techniques of cryptography for the ascertainment of the data.

The non-cryptographic innate vigor of the proposed scheme makes it fortified to perform the required operations (placement and retrieval) on the data. This in turn ensure a restrained (controlled) replication of the file fragments, where each of the fragments is replicated only once for the resolution of improved security.

To further improve the retrieval time, we judicially replicate fragments over the nodes that generate the highest read/write requests. The selection of the nodes is performed in two phases. In the first phase, the nodes are selected for the initial placement of the fragments based on the centrality measures. In the second phase, the nodes are selected for replication.

6. Implementation and Result

The FDBPS methodology does not rely on the traditional cryptographic techniques for the data security; thereby relieving the system of computationally expensive methodologies.

We show that the probability to locate and compromise all of the nodes storing the fragments of a single file is extremely low. We also compare the performance of the FDBPS methodology with ten other schemes. The higher level of security with slight performance overhead was observed. The implementation is done in three steps as follows.

a) Fragmentation

The security of a large-scale system, such as cloud depends on the security of the system as a whole and the security of individual nodes. A successful intrusion into a single node may have severe consequences, not only for data and applications on the victim node, but also for the other nodes. The data on the victim node may be revealed fully because of the presence of the whole file.

A successful intrusion may be a result of some software or administrative vulnerability. The file owner specifies the fragmentation threshold of the data file is specified to be generated by. The file owner can specify the fragmentation threshold in terms of either percentage or the number and size of different fragments. The percentage fragmentation threshold, for example, can dictate that each fragment will be of 5% size of the total size of the file. Alternatively, the owner can generate separate file containing information about the fragment number and size, for instance, fragment 1 of size 4,000 Bytes, fragment 2 of size 6,749 Bytes.

The owner of the file is the best candidate to generate fragmentation threshold as he is very well aware about the significant information from the file. The owner can best split the file such that each fragment does not contain significant amount of information. The default percentage fragmentation threshold can be made a part of the Service Level Agreement (SLA), if the user does not specify the fragmentation threshold while uploading the data file.

b) Fragment Placement

To provide the security while placing the fragments, the concept of fragmentation is used that was originally used for the channel assignment problem. In this process, this loses some of the central nodes that may increase the retrieval time. But it achieves a higher security level. If anyhow the intruder compromises a node and obtains a fragment, he cannot determine the location of the other fragments. The attacker can only keep on guessing the location of the other fragments. Because the nodes are separated by Fragmentation.

c) Replication

To increase the data availability, reliability, and improve data retrieval time, it also performs a controlled replication. It places the fragment on the node that provides the decreased access cost with an objective to improve retrieval time for accessing the fragments for reconstruction of original file. Algorithms for dynamic model fragmentation here we give a brief explanation of the control flow of Algorithm 3 which partitions the workflow model into Fragments dynamically.

First let us discuss the situation that the source transition ts of F has only one output place p. If p has only one output transition, we can just cut off ts from F and receive a subsequent fragment Else if p has multiple output transitions, then each transition is the source transition of a new fragment multiple fragments are obtain although only one of them can really be enabled and executed since they are mutual exclusive. Another situation, which is more difficult to tackle, is the case that when ts has multiple output places. In this situation each output place forms at least one new fragment, and these fragments are to be executed in parallel. So we take some measure to avoid information redundancy when multiple fragments are generated, i.e. we introduce the idea of Transition restricted reachable sub-fragment, and when we do fragmentation, the set Tr is updated constantly to prohibit the unnecessary spanning of sub-fragments.

7. Algorithm for Dynamic model fragmentation

```
Input: Fragment ($t_s$, $F$)
Output: A list of fragments, denoted as F_LIST
If ($|t_s^\bullet| = 1$)
  Let $p = t_s^\bullet$
  If ($|p^\bullet| = 1$)// $p$ has only one output transition
          Let $t_{next} = p^\bullet$
          Add ($t_{next}$, RSF($t_{next}$, $F$)) to F_LIST
  Else//$p$ has multiple output transitions
    Let $\{t_1, t_2, \ldots, t_k\} = p$
    For each $t_i$ in $p$
      Add ($t_i$, RSF($t_i$, $F$)) to F_LIST
    End For
  End If
End If
Else//$t_s$ has multiple output places, i.e., $t_s$ is a AND-split
  Let $\{p_1, p_2, \ldots, p_k\} = t_s^\bullet$
  Let $t_{join}$ = JoinTrans ($t_s$)
  $T_r = \{t_{join}\}$
  For each $p_i$ in $t_s$
        If ($|p_i^\bullet| = 1$)


          Let $t_{i next} = p_i^\bullet$
          If ($i = 1$)
              Add $F_{i\ next}$ = ($t_{i\ next}$, RSF($t_{i\ next}$, $F$)) to F_LIST
              Update ($T_r$, $F_{i\ next}$, $F$)
          Else
              Add $F_{i\ next}$ = ($t_{i\ next}$, TRRSF ($p_i^\bullet$, $F$, $T_r$)) to F_LIST
              Update ($T_r$, $F_{i\ next}$, $F$)
          End If
    Else //$p$ has multiple output Transitions
        Let $\{t_{i1}, t_{i2}, \ldots, t_{iki}\} = p_i^\bullet$
        For each $t_{ij}$ in $p_i^\bullet$
          If ($i = 1$)
                Add $F_{ij\ next}$ = ($t_{ij}$, RSF ($t_{ij}$, $F$)) to F_LIST
          Else
          Add $F_{ij\ next}$ = ($t_{ij}$, TRRSF ($t_{ij}$, $F$, $T_r$)) to F_LIST
          End If
        End For
        Update ($T_r$, $F_{i1\ next}$, $F$)
    End If
  End For
End If
```

7.1 Data Fragmentation

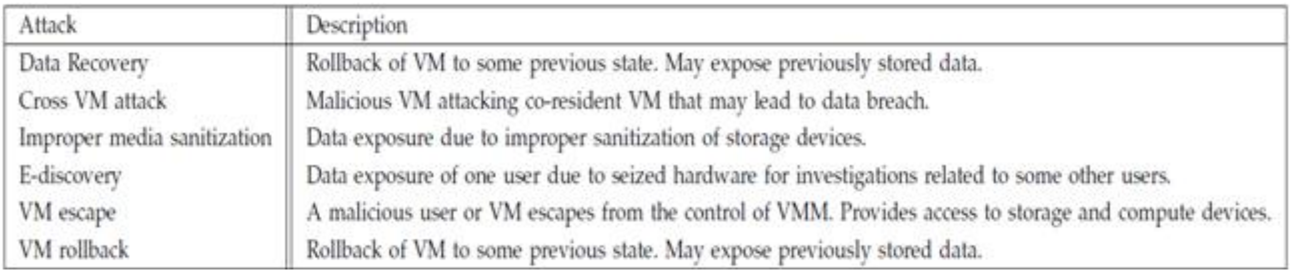
Cloud depends on the security of the system as a whole and the security of individual nodes. A successful intrusion into a single node may have severe consequences, not only for data and applications on the victim node, but also for the other nodes. The data on the victim node may be revealed fully because of the presence of the whole file. A successful intrusion may be a result of some software or administrative vulnerability. In case of homogenous systems, the same flaw can be utilized to target other nodes within the system. The success of an attack on the subsequent nodes will require less effort as compared to the effort on the first node. Comparatively, more effort is required for heterogeneous systems. However, compromising a single file will require the effort to penetrate only a single node.

Let s be the number of successful intrusions on distinct nodes, such that s>z. The probability that s number of victim nodes contain all of the z sites storing the file fragments (represented
By P(s, z)) is given as:

$$P(s,z) = \frac{\binom{s}{z}\binom{M-s}{s-z}}{\binom{M}{s}}.$$

7.2 Centrality

The centrality of a node in a graph provides the measure of the relative importance of a node in the network. The objective of improved retrieval time in replication makes the centrality measures more important. There are various centrality measures; for instance, closeness centrality, degree centrality, between’s centrality, eccentricity centrality, and eigenvector centrality. We only elaborate on the closeness, between’s, and eccentricity centralities because we are using the aforesaid three centralities in this work. For the remainder of the centralities, we encourage the readers to review. In the FDBPS methodology, we propose not to store the entire file at a single node. The FDBPS methodology fragments the file and makes use of the cloud for replication. The fragments are distributed such that no node in a cloud holds more than a single fragment, so that even a successful attack on the node leaks no significant information.



Attack	Description
Data Recovery	Rollback of VM to some previous state. May expose previously stored data.
Cross VM attack	Malicious VM attacking co-resident VM that may lead to data breach.
Improper media sanitization	Data exposure due to improper sanitization of storage devices.
E-discovery	Data exposure of one user due to seized hardware for investigations related to some other users.
VM escape	A malicious user or VM escapes from the control of VMM. Provides access to storage and compute devices.
VM rollback	Rollback of VM to some previous state. May expose previously stored data.

The communicational backbone of cloud computing is the Data Center Network (DCN). In this paper, we use three DCN architectures namely: **(a)** Three tier, **(b)** Fat tree, and **(c)** DCell. The Three tiers is the legacy DCN architecture. However, to meet the growing demands of the cloud computing, the Fat tree and Dcell architectures were proposed. Therefore, we use the aforementioned three architectures to evaluate the performance of our scheme on legacy as well as state of the art architectures. The Fat tree and three tier architectures are switch-centric networks. The nodes are connected with the access layer switches. Multiple access layer switches are connected using aggregate layer switches. Core layers switches interconnect the aggregate layer switches.. The Dcell is a server centric network architecture that uses servers in addition to switches to perform the communication process within the network.

7.3 Comparative techniques

We compared the results of the FDBPS methodology with fine-grained replication strategies, namely: **(a)** DRPA-star, **(b)** WA-star, **(c)** A_-star, **(d)** SA1, **(e)** SA2, **(f)** SA3, **(g)** Local Min-Min, **(h)** Global Min- Min, **(i)** Greedy algorithm, and **(j)** Genetic Replication Algorithm (GRA). The DRPA-star is a data replication algorithm based on the A-star best-first search algorithm. The DRPA-star starts from the null solution that is called a root node. The communication cost at each node n is computed as: cost(n) = g(n) + h(n), where g(n) is the path cost for reaching n and h(n) is called the heuristic cost and is the estimate of cost from n to the goal node. The DRPA-star searches all of the solutions of allocating a fragment to a node. The solution that minimizes the cost within the constraints is explored while others are discarded. The selected solution is inserted into a list called the OPEN list. The list is ordered in the ascending order so that the solution with the minimum cost is expanded first. The heuristic used by the DRPAstar is given as h(n) = max(0; (mmk(n)g(n))), where mmk(n) is the least cost replica allocation or the maxmin RC.

7.4 Workload

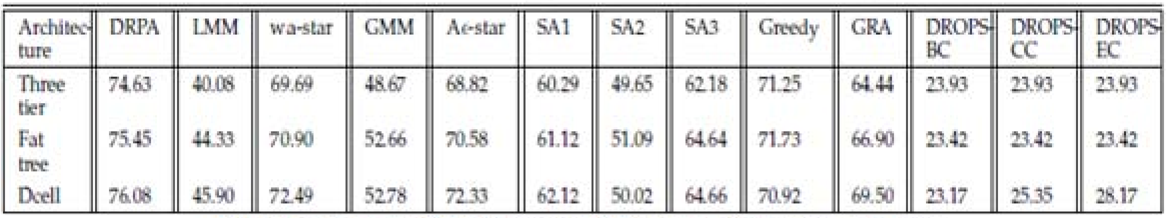
The size of files was generated using a uniform distribution between 10Kb and 60 Kb. The primary nodes were randomly selected for replication algorithms. For the FDBPS methodology, the Si.s selected during the first cycle of the nodes selection by Algorithm 1 were considered as the primary nodes. The capacity of a node was generated using a uniform distribution between (1 2CS)C and (3 2CS)C, where 0 B C C 1. For instance, for CS = 150 and C = 0:6 the capacities of the nodes were uniformly distributed between 45 and 135. Themean value of g in the OPEN and FOCAL lists was selected as the value of _, for WA-star and A_-star, respectively. The value for level R was set to _ d 2 _, where d is the depth of the search tree(number of fragments) The read/write (R/W) ratio for the simulations that used fixed value was selected to be 0:25 (The R/W ratio reflecting 25% reads and 75% writes within the cloud). The reason for choosing a high workload (lower percentage of reads and higher percentage of writes) was to evaluate the performance of the techniques under extreme cases. The simulations that studied the impact of change in the R/W ratio used various workloads in terms of R/W ratios. The R/W ratios selected were in the range of 0:10 to 0:90. The selected range covered the effect of high, medium, and low workloads with respect to the R/W ratio.

8. Results and Discussion

We compared the performance of the FDBPS methodology with the algorithms The behavior of the algorithms was studied by: **(a** increasing the number of nodes in the system, **(b)** increasing the number of objects keeping number of nodes constant, **(c)** changing the nodes storage capacity, and **(d)** varying the read/write ratio. The aforesaid parameters are significant as they affect the problem size and the performance of algorithms. Impact of increase in number of cloud nodes we studied the performance of the placement techniques and the FDBPS methodology by increasing the number of nodes. The performance was studied for the three discussed cloud architectures. The numbers of nodes selected for the simulations were 100, 500, 1,024, 2,400, and 30,000. The number of nodes in the Dcell architecture increases exponentially. For Dcell architecture, with two nodes in the Dcell0, the architecture consists of 2,400 nodes. However, increasing a single node in the Dcell0, the total nodes increases to 30, 000.

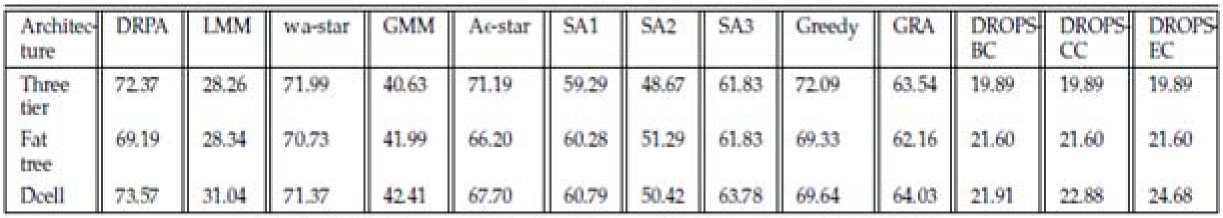
The number of file fragments was set to 50. For the first experiment we used C = 0:2. Fig. 2 (a), Fig. 2 (b), and Fig. 3 (a) show the results for the three tier, Fat tree, and Dcell architectures, respectively. The reduction in network transfer time for a file is termed as RC. In the figures, the BC stands for the between's centrality, the CC stands for closeness centrality, and the EC stands for eccentricity centrality. The interesting observation is that although all of the algorithms showed similar trend in performance within a specific architecture, the performance of the algorithms was better in the Dcell architecture as compared to three tier and fat tree architectures. This is because the Dcell architecture exhibits better inter node connectivity and robustness. The DRPA-star gave best solutions as compared to other techniques and registered consistent performance with the increase in the number of nodes.

TABLE 4: Average RC (%) savings for increase in number of fragments



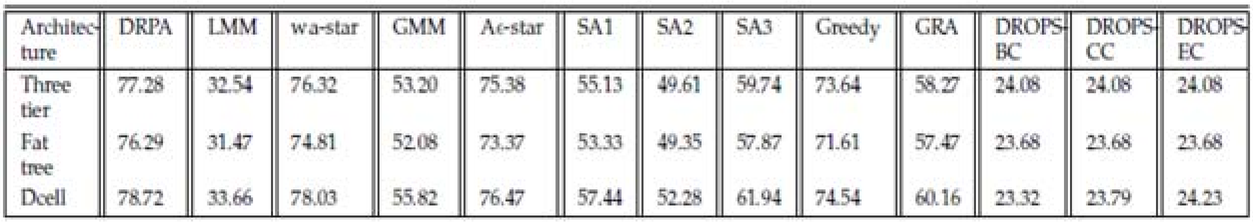
Architec-ture	DRPA	LMM	wa-star	GMM	Ae-star	SA1	SA2	SA3	Greedy	GRA	DROPS-BC	DROPS-CC	DROPS-EC
Three tier	74.63	40.08	69.69	48.67	68.82	60.29	49.65	62.18	71.25	64.44	23.93	23.93	23.93
Fat tree	75.45	44.33	70.90	52.66	70.58	61.12	51.09	64.64	71.73	66.90	23.42	23.42	23.42
Dcell	76.08	45.90	72.49	52.78	72.33	62.12	50.02	64.66	70.92	69.50	23.17	25.35	28.17

TABLE 5: Average RC (%) savings for increase in storage capacity



Architec-ture	DRPA	LMM	wa-star	GMM	Ae-star	SA1	SA2	SA3	Greedy	GRA	DROPS-BC	DROPS-CC	DROPS-EC
Three tier	72.37	28.26	71.99	40.63	71.19	59.29	48.67	61.83	72.09	63.54	19.89	19.89	19.89
Fat tree	69.19	28.34	70.73	41.99	66.20	60.28	51.29	61.83	69.33	62.16	21.60	21.60	21.60
Dcell	73.57	31.04	71.37	42.41	67.70	60.79	50.42	63.78	69.64	64.03	21.91	22.88	24.68

TABLE 6: Average RC (%) savings for increase in R/W ratio



Architec-ture	DRPA	LMM	wa-star	GMM	Ae-star	SA1	SA2	SA3	Greedy	GRA	DROPS-BC	DROPS-CC	DROPS-EC
Three tier	77.28	32.54	76.32	53.20	75.38	55.13	49.61	59.74	73.64	58.27	24.08	24.08	24.08
Fat tree	76.29	31.47	74.81	52.08	73.37	53.33	49.35	57.87	71.61	57.47	23.68	23.68	23.68
Dcell	78.72	33.66	78.03	55.82	76.47	57.44	52.28	61.94	74.54	60.16	23.32	23.79	24.23

The DRPA-star gave best solutions as compared to other techniques and registered consistent performance with the increase in the number of nodes. Similarly, WA-star, A_-star, GRA, greedy and SA3 showed almost consistent performance with various numbers of nodes. The performance of LMM and GMM gradually increased with the increase in number of nodes since the increase in the number of nodes increased the number of bins. The SA1 and SA2 also showed almost constant performance in all of the three architectures. However, it is important to note that SA2 ended up with a decrease in performance as compared to the initial performance. This may be due to the fact that SA2 only expands the node with minimum cost when it reaches at certain depth for the first time. Such a pruning for the first time might have purged nodes by providing better global access time. The FDBPS methodology did not employ full-scale replication. Every fragment is replicated only once in the system.

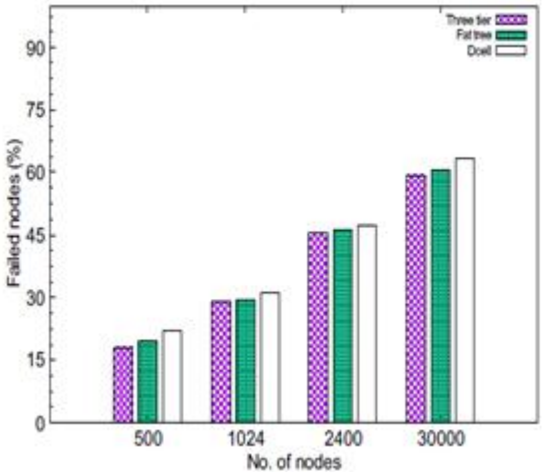


Figure 2: Fault tolerance level of FDBPS

9. Conclusion

We proposed the FDBPS methodology, a cloud storage security scheme that collectively deals with the security and performance in terms of retrieval time. The data file was fragmented and the fragments are dispersed over multiple nodes. The nodes were separated by means of Dynamic fragmentation. The fragmentation and dispersal ensured that no significant information was obtainable by an adversary in case of a successful attack. No node in the cloud, stored more than a single fragment of the same file. The performance of the FDBPS methodology was compared with full-scale replication techniques. The results of the simulations revealed that the simultaneous focus on the security and performance resulted in increased security level of data accompanied by a slight performance drop.

References

[1] B. Grobauer, T.Walloschek, and E. Stocker, “Understanding cloud computing vulnerabilities,” IEEE Security and Privacy, Vol. 9, No. 2, 2011, pp. 1-57.

[2] Sudhi Seshachala, “Disadvantages of cloud computing”, March 17, 2015.

[3] K. Bilal, S. U. Khan, L. Zhang, H. Li, K. Hayat, S. A. Madani, N. Min-Allah, L. Wang, D. Chen, M. Iqbal, C. Z. Xu, and A. Y. Zomaya, “Quantitative comparisons of the state of the art data center architectures,” Concurrency and Computation: Practice and Experience, Vol. 25, No. 12, 2013, pp. 1771-1783.

[4] K. Bilal, M. Manzano, S. U. Khan, E. Calle, K. Li, and A. Zomaya, “On the characterization of the structural robustness of data center networks,” IEEE Transactions on Cloud Computing, Vol. 1, No. 1, 2013, pp. 64-77.

[5] D. Boru, D. Kliazovich, F. Granelli, P. Bouvry, and A. Y. Zomaya, “Energy-efficient data replication in cloud computing datacenters,” In IEEE Globecom Workshops, 2013, pp. 446-451. .

[6] Y. Deswarte, L. Blain, and J-C. Fabre, “Intrusion tolerance in distributed computing systems,” In Proceedings of IEEE Computer Society Symposium on Research in Security and Privacy, Oakland CA, pp. 110-121, 1991.

[7] B. Grobauer, T.Walloschek, and E. Stocker, “Understanding cloud computing vulnerabilities,” IEEE Security and Privacy, Vol. 9, No. 2, 2011, pp. 50-57.

[8] W. K. Hale, “Frequency assignment: Theory and applications,” Proceedings of the IEEE, Vol. 68, No. 12, 1980, pp. 1497-1514.

[9] K. Hashizume, D. G. Rosado, E. Fernndez-Medina, and E. B. Fernandez, “An analysis of security issues for cloud computing,” Journal of Internet Services and Applications, Vol. 4, No. 1, 2013, pp. 1-13.

[10] M. Hogan, F. Liu, A.Sokol, and J. Tong, “NIST cloud computing standards roadmap,” NIST Special Publication, July 2011.

[11] W. A. Jansen, “Cloud hooks: Security and privacy issues in cloud computing,” In 44th Hawaii IEEE International Conference on System Sciences (HICSS), 2011, pp. 1-10.