ENSURING PERFORMANCE AND SECURITY THROUGH FAULT-TOLERANCE AND REPRODUCIBILITY IN CLOUD

Sathya S[1], Kamalesh S[2], Karthick J[3], Madhuvanth Kanichan S[4], Muruganantham D[5]

[1] *Assistant Professor, Computer Science and Engineering, Hindusthan College of Engineering and Technology, Tamil Nadu, India*
[2] *Student, Computer Science and Engineering, Hindusthan College of Engineering and Technology, Tamil Nadu, India*
[3] *Student, Computer Science and Engineering, Hindusthan College of Engineering and Technology, Tamil Nadu, India*
[4] *Student, Computer Science and Engineering, Hindusthan College of Engineering and Technology, Tamil Nadu, India*
[5] *Student, Computer Science and Engineering, Hindusthan College of Engineering and Technology, Tamil Nadu, India*

ABSTRACT

The increasing popularity of networks as an attractive alternative to classic information processing systems has increased the importance of its correct and continuous operation even in the presence of faulty components. In this paper, we introduce an innovative, system-level, modular perspective on creating and managing fault tolerance in networks. We propose a comprehensive high-level approach to shading the implementation details of the fault tolerance techniques to application developers and users by means of a dedicated service layer. In particular, the service layer allows the user to specify and apply the desired level of fault tolerance, and does not require knowledge about the fault tolerance techniques that are available in the envisioned network and their implementations.

Keyword: - *Fault Tolerance, Networks, Security, Resilience, Cloud Computing, Fragmentation*

1. INTRODUCTION

With the increasing scale of big data, data processing has put forward new requirements for high-performance computers. Interconnection network is a vital factor, which directly affects the performance of parallel computing system. It consists of a network of switching elements with a certain topology and control mode. It is used to realize the interconnection of multiple processors or multiple functional components within a computer system. Large-scale integrated circuit technology can be used to build complex Internet and predict the next generation of supercomputer systems. While adopting faster processors, it also achieves high speed and rapidity by increasing the number of processors. Therefore, how to design an excellent interconnection network to connect these processors is a technical difficulty in building supercomputer systems. Hypercube is one of the most commonly used interconnection structures. It has many good properties, such as regularity, recursive, low node degree and so on. Due to its powerful computing function and high efficiency, it is significant to run parallel algorithms on it. Almost all algorithms on linear arrays, cycles and trees can be effectively simulated on hyper cubes with only constant factor delay.

2. MODULES

- User Authentication
- Construction of Data Storage

- Replica Allocation & Data Encryption
- Fault Tolerance
- Data Retrieval

2.1 User Authentication

User Authentication is the process of identity verification you are trying to prove a user is who they say they are. For a user to prove their identity, a user needs to provide some sort of proof of identity that your system understands and trust. The authentication process starts with creating an instance of the Login Context. Various constructors are available; the example uses the Login Context variety. The first parameter is the name (which acts as the index to the login module stack configured in the configuration file), and the second parameter is a callback handler used for passing login information to the Log server. Callback Handler has a handle method which transfers the required information to the Log server. The example uses a very simple handler which saves the username and password in an instance variable, so that it can be passed on during the invocation of the handle method from the Log server. It's also possible to create callbacks that interact with the user to obtain user credentials, and transfer that information to the log server for authentication.

2.2 Construction of Data Storage

The data tier uses the storage service offered by the IP to store and retrieve its customer data, the application tier uses the IP's compute service to process its operations and respond to customer queries. This system architecture allows the banking service to meet its varying business demands with respect to scalability and elasticity of computing resources. However, using traditional methods, fault tolerance properties of the banking service remain constant throughout its life cycle. Therefore, in the client's perspective, it is easier to engage with the SP, specify its reliability and availability requirements based on business needs, and transparently obtains desired fault tolerance properties for its applications similarly, the working state of network links and VM instances must be maintained by the resource manager. We note that the database and the resource graph are essential for the service provider to ensure the correct behavior of fault tolerance mechanisms.

2.3 Replica Allocation & Data Encryption

This component supports the replication mechanisms by invoking replicas and managing their execution based on the client's requirements. We denote the set of VM instances that are controlled by a single implementation of a replication mechanism as a replica group. Each replica within a group can be uniquely identified, and a set of rules R that must be satisfied by a replica group are specified. The task of the replication manager is to make the client perceive a replica group as a single service, and to ensure that the fault free replicas exhibit correct behavior during execution time. To support a replication mechanism, the replica invoker first contemplates the desired replication parameters such as the style of replication (active, passive, cold passive, hot passive), number of replicas, and constraints on relative placement of individual replicas, and forms the replica group. In other words, the replica invoker takes the reference of a client's application as input from FTM Kernel, analyzes the expected fault tolerance properties, and interacts with the resource manager to obtain the location of each replica. Client uploaded data are encrypted for secure data storage in server. Encrypted data are stored in different virtual server with fragments.

2.4 Fault Tolerance

The task of offering fault tolerance as a service requires the service provider to realize generic fault tolerance mechanisms such that the client's applications deployed in virtual machine instances can transparently obtain fault tolerance properties. To this aim, we define ft−unit as the fundamental module that applies a coherent fault tolerance mechanism to a recurrent system failure at the granularity of a VM instance. The notion of ft−unit is based on the observation that the impact of hardware failures on client's applications can be handled by applying fault tolerance mechanisms directly at the virtualization layer than the application itself For instance, fault tolerance of the banking service can be increased by replicating the entire VM instance in which its application tier is deployed on multiple physical nodes, and server crashes can be detected using well-known failure detection algorithms such as the heartbeat protocol. Where the primary and backup components are run in VM instances independent of the banking service's application tier. The design stage starts when a client requests the service provider to offer fault tolerance support to its applications. In this stage, the service provider must first analyze the client's requirements, match them with available ft−units, and form a complete fault tolerance solution using appropriate ft−units. We note that each

ft−unit offers a unique set of fault tolerance properties that can be characterized using its functional, operational, and structural attributes.

2.5 Data Retrieval

The data retrieval module describes the retrieve data from different virtual server only authenticate user. Data are encrypted in different virtual server with fragments. Data are retrieved from different virtual server and combine the data and convert to decrypted format. Decrypted data are bringing to original data for user extraction. The goal of this component is to achieve system-level resilience by minimizing the downtime of the system during failures. To this aim, this component supports ft−units that realize recovery mechanisms so that an error-prone node can be resumed back to a normal operational mode.

3. FLOWCHART DIAGRAM

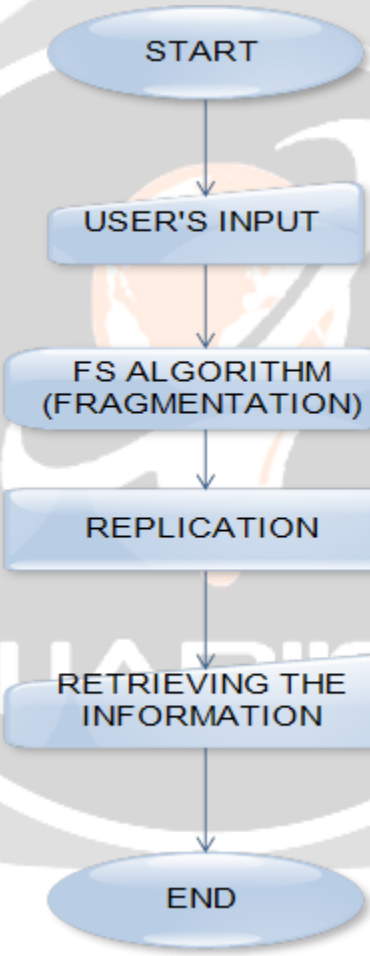

Fig. -1: Flowchart

4. SYSTEM ARCHITECTURE

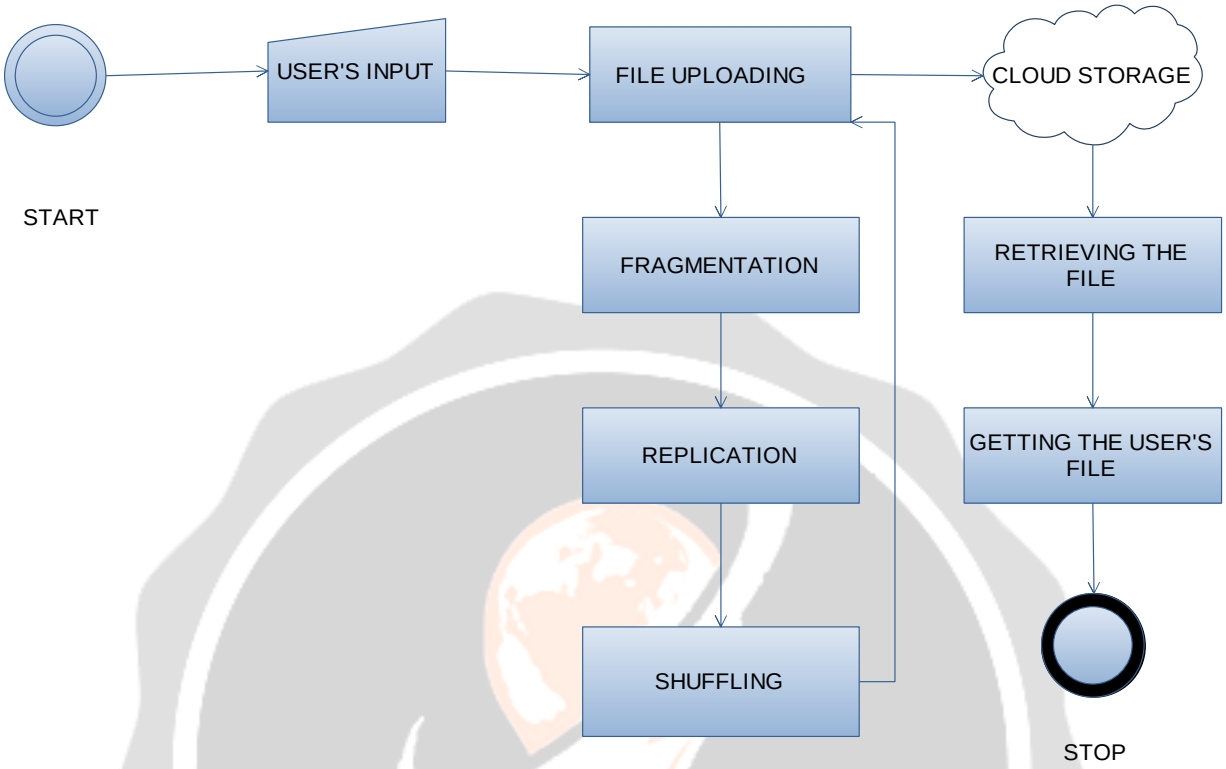

Fig. -2: Architecture Diagram

5. RESULTS

Fig. -3: New user registration

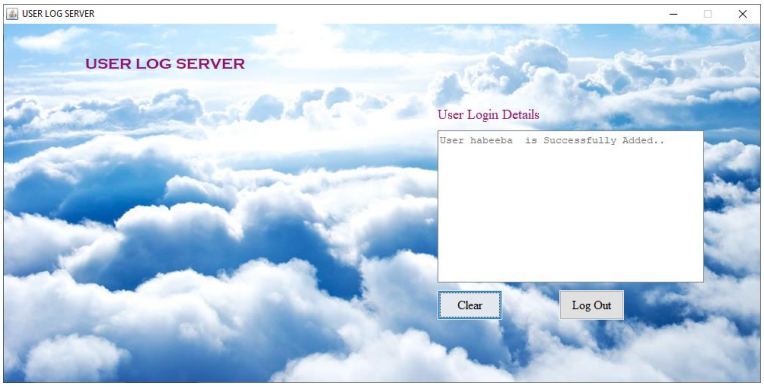

Fig. -4: Register details added in log server

Fig. -5: New user login

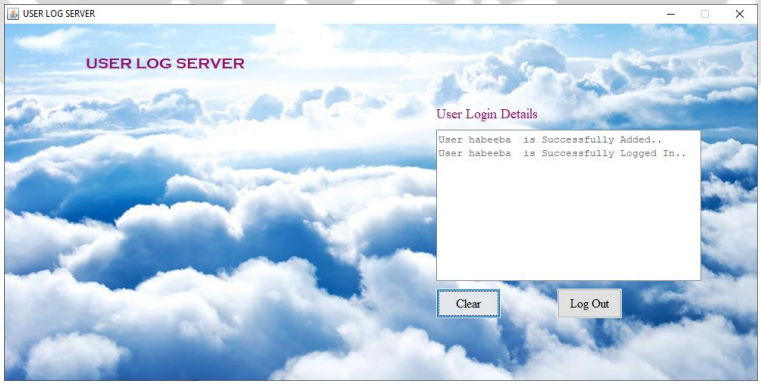

Fig. -6: Login details monitor in log server

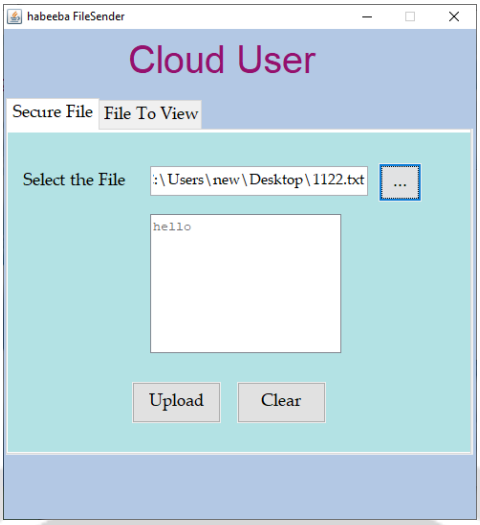

Fig. -7: File upload

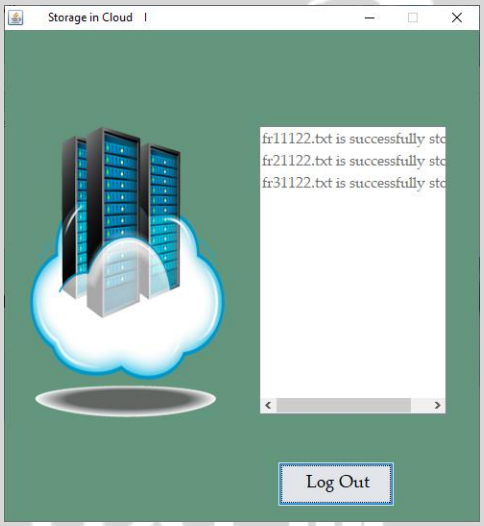

Fig. -8: Server Storage

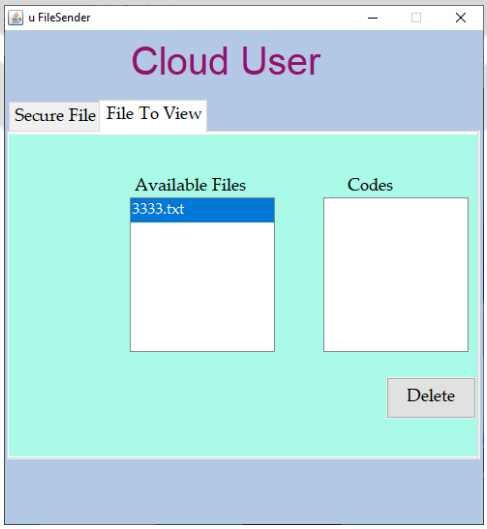

Fig. -9: File Download

6. CONCLUSIONS

A network 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 T-coloring. 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 fault tolerant 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.

7. REFERENCES

[1] I. Abraham, D. Delling, A. Fiat, A. V. Goldberg, and R. F. Werneck, "Highway dimension and provably efficient shortest path algorithms," J. ACM, vol. 63, no. 41, pp. 1–26, 2016.
[2] M. Adda and A. Peratikou, "Routing and fault tolerance in Z-fat tree," IEEE Trans. Parallel Distrib. Syst., vol. 28, no. 8, pp. 2373–2386, Aug. 2017.
[3] D. Aguirre-Guerrero, M. Camelo, L. F_abrega, and P. Vil_a, "WMGR: A generic and compact routing scheme for data center networks," IEEE/ACM Trans. Netw., vol. 26, no. 1, pp. 356–369, Feb. 2018.
[4] A. Bossard and K. Kaneko, "A set-to-set disjoint paths routing algorithm in a torus-connected cycles network," in Proc. 31st Int. Conf. Comput. Appl., 2016, pp. 81–88.
[5] E. Cheng, K. Qiu, and Z. Shen, "Diagnosability problems of the exchanged hypercube and its generalization," Int. J. Comput. Math.: Comput. Syst. Theory, vol. 2, no. 2, pp. 39–52, 2017.
[6] E. Cheng, K. Qiu, and Z. Shen, "A strong connectivity property of the generalized exchanged hypercube," Discrete Appl. Math., vol. 216, pp. 529–536, 2017.
[7] J.-M. Chang et al., "Locally exchanged twisted cubes: Connectivity and super connectivity," Inf. Process. Lett., vol. 116, no. 7, pp. 460–466, 2016.
[8] K. Day, N. Alzeidi, and A. Touzene, "Multipath routing in a 3D torus network on chip," in Proc. 2nd Int. Conf. ACM Future Netw. Distrib. Syst., 2018, pp. 1–8.
[9] W. J. Dally and H. Aoki, "Deadlock-free adaptive routing in multicomputer networks using virtual channels," IEEE Trans. Parallel Distrib. Syst., vol. 4, no. 4, pp. 466–475, Apr. 1993.
[10] S. P. Dwivedi and R. S. Singh, "Error-tolerant graph matching using node contraction," Pattern Recognit. Lett., vol. 116, pp. 58–64, 2018.