A Novel AI Approach for Cloud Data Replication

Nilesh Suresh Jain
Principal Software Engineer at Oracle America Inc.

DOI: 10.29322/IJSRP.13.10.2023.p14206
https://dx.doi.org/10.29322/IJSRP.13.10.2023.p14206

Paper Received Date: 16th August 2023
Paper Acceptance Date: 24th September 2023
Paper Publication Date: 6th October 2023

Abstract- In cloud, replication enables automatic, asynchronous copying of objects across buckets. Replication serves two purposes, with the main one being to provide high availability in case nodes or shards fail. Enabling Object Storage replication is as simple as creating a replication policy on the source bucket that identifies the region and the destination bucket to replicate to. After the replication policy is created, the destination bucket is read-only and updated only by replication from the source bucket. Objects deleted from the source bucket after policy creation are automatically deleted from the destination bucket. When replication is enabled, accidental data change operation results in permanent data loss, deleting or corrupting objects from both source and destination bucket. This problem applies to most existing storage systems. In this research, an AI algorithm to detect and protect against accidental data loss is proposed, which takes the replication decisions in advance by monitoring and acting before the data loss occurs. In addition, algorithm is proposed to determine lifecycle policy for the objects created in target bucket. Experimental results show that proposed algorithm is the ideal candidate for replication of the cloud data across buckets, as it minimizes the accidental data loss, total number of SLA violations, average response time and total execution time to resolve each incident as compared to the existing setup.

***Index Terms*-** Artificial intelligence (AI), Backup/recovery, Cloud, Data Replication, Fault-tolerance, High availability, Machine Learning (ML), Reliability

I. Introduction

Enabling Object Storage replication is as simple as creating a replication policy on the source bucket that identifies the region and the bucket to replicate to. After the replication policy is created, the destination bucket is read-only and updated only by replication from the source bucket. Objects uploaded to a source bucket after policy creation are asynchronously replicated to the destination bucket. Objects deleted from the source bucket after policy creation are automatically deleted from the destination bucket. Objects uploaded to a source bucket before policy creation are not replicated.

Any object in the destination bucket with the same name as an object in the source bucket is overwritten by replication. The name, metadata, ETag, and MD5 value of a replicated object match those of the original object in the source bucket. These attributes are not replicated from the source because the archival state, modified timestamp, and creation timestamp can all vary.

A destination bucket is not automatically created when a replication policy is created. Prior to creating the replication policy on the source bucket, create the destination bucket. A bucket can be in the Standard (Object Storage) or Archive Storage tier as a source or destination. Replication in chains is not supported. A single replication policy can be applied on a given source bucket. Each bucket used as a destination for replication can only have one source. For every replication source bucket, there can only be one destination. A replication source can never also be a destination bucket.

Following the creation of the replication policy, the destination bucket becomes read-only and is only updated via replication from the source bucket. The destination bucket receives an automatic replication of any objects uploaded to the source bucket. Object from the destination bucket is automatically cleared when an object is deleted from the source bucket. A replication destination bucket cannot be deleted unless replication is stopped, and the bucket is made writable once more. Replication can be used in conjunction with Lifecycle Management rules that control object deletion and archiving. However, lifecycle policies must respect the replication destination bucket's read-only characteristics. It is impossible to implement a lifecycle policy that removes objects from the replication destination bucket. Any replication and lifecycle policy combination you implement should be carefully examined and tested.

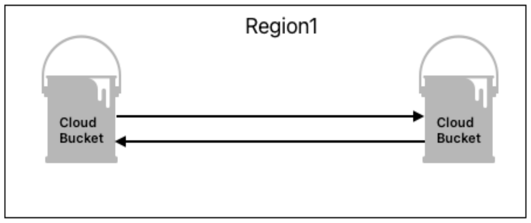


FIGURE 1. Object Storage Same-Region Replication

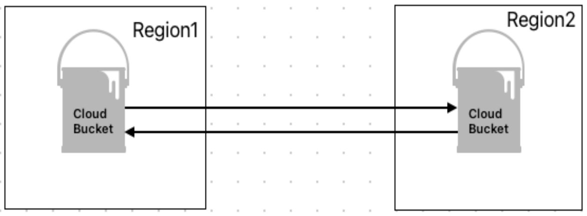


FIGURE 2. Object Storage Cross-Region Replication

For Disaster Recovery

Figure 1 represents the data replication setup in same region. With the configuration shown in Figure 2, you can activate the application in the primary region and store objects in a read-write bucket. The applications can be passive and synchronizing with the primary while the objects are replicated by Object Storage to the secondary region read-only bucket [6]. Data replicated to the destination bucket can be accessed for business continuity in the event of a disaster that prevents access to the source bucket.

To effectively manage data, one must replicate it. It gives you many advantages that can completely change your business by enabling you to create multiple copies of your data in various locations. Its support for real-time analytics is one of its key benefits. You can support your business intelligence and machine learning initiatives by synchronizing data from various sources in real-time, such as cloud-based reporting. Imagine being able to run predictive models on user behavior data to provide tailored recommendations in real-time or imagine being able to update your dashboards with the most recent data. Faster data access is a significant additional benefit. By storing data in multiple places, you can access it from the servers that are nearer to your users, lowering latency and enhancing data retrieval performance. As a result, users in various locations will be able to access data without annoying delays. With data replication, server performance can be greatly improved. By distributing data traffic across several servers, you can improve load balancing and resource utilization. For instance, offloading complex analytical queries to data warehouses or data lakes through replication can lighten the load on your operational databases, resulting in better system performance and scalability. Not only that, but data replication is a dependable disaster recovery plan. Any business may experience catastrophic data loss because of system malfunctions or disruptions, leading to monetary losses and operational disruptions. It mitigates these risks by creating redundant copies of data in multiple locations, allowing you to quickly switch to alternative data sources in case of disruptions or failures.

Data replication, a crucial process that ensures data backup, fault tolerance, and improved network accessibility, entails duplicating and regularly updating data in multiple locations. The choice of a particular type depends on the use for which the replicated data is intended and how it will be accessed. Here are the various forms of data replication.

Without actively checking for updates or changes, snapshot replication takes a "snapshot" of a database as it is at the beginning of the replication process. As a result, the replicated copy of the database is static and always displays the data as it did at the time. When significant changes occur quickly or when data in the database doesn't change frequently, snapshot replication is ideal. It allows for capturing a specific data state, which can be helpful for historical or reference purposes.

Using a powerful technique called transactional replication, a full copy of the database is made while new data is continuously being captured and copied in real-time as the database changes. As modifications are replicated in the order they are made, this guarantees that the replicated copy stays consistent with the original database. When you need to make sure that changes to data made by log-based incremental replication are replicated in real-time, transactional replication is especially effective when combined with key-based incremental replication. This method is appropriate for environments where there is a lot of data modification activity because it supports high volumes of read, write, and delete activity.

Data from various sources can be combined into a partial replication of a single database using the incredibly effective technique of merge replication. Merge replication makes sure that all alterations are made to the combined database by gathering and aggregating changes made by multiple users across various locations. Merge replication has the exceptional ability to quickly identify and resolve conflicting changes, which is a notable replication advantage. Conflicts may occur when merging changes made by multiple users at different locations into the replica.

Peer-to-peer replication depends on the continuous exchange of transactional data between nodes. A peer-to-peer setup ensures that data changes are propagated in real-time across all nodes by having every node in the same network constantly sync its databases with one another. Additionally, since all nodes are writable, data modifications are possible from anywhere in the world and are immediately reflected in all other nodes, guaranteeing real-time consistency irrespective of the point of origin of the change.

Restore and Backup Replication makes it possible to return replicated databases to the main database and server from which the backup was originally made. The replication settings, however, cannot be preserved if you need to retrieve a backup of a replicated database to a different database or server. In such circumstances, all publications and subscriptions would need to be created from scratch.

Replication of data in the cloud has several advantages, including distribution, accuracy, and accessibility. Accessibility is improved for both customers and employees when data is replicated across various hybrid cloud instances. To ensure that systems are always current and available, high-availability storage and active cluster failover can be supported by replicated data across multiple clusters.

Accuracy means making sure you always have access to data that is up to the minute accurate. Your organization can support identical copies of the same fundamental data that is accurate and current with the most recent customer interactions and database transactions with the right infrastructure and resources. Many organizations will use redundant cloud environments for a variety of geographical reasons to better serve their customers or distributed research teams. As a result, local users can perform better. These remote environments can sync successfully and make use of the most recent data thanks to a strategic cloud data replication plan. Due to the large number of redundant cloud servers that share the same data, emergency recovery is made possible, allowing for more effective and precise disaster recovery plans. Depending on your requirements, cloud data replication ensures that you can rely on their accurate data in hot or cold clusters for immediate recovery or long-term storage.

II. Related Work

Users can store their files in the cloud so they can access them via the internet from any location [1]. Benefits of cloud storage include lower costs, easier IT management, better user experiences, and disaster recovery. Any Distributed File System (DFS) that requires replication must include it in its design. After a popular file is chosen for replication, Algorithm [2] decides how many additional replicas should be created for that file. Depending on the computed confidence values for the candidate files (popular files), three categories are created for the decision. Massive amounts of data can be stored and processed using the Hadoop Distributed File System (HDFS). One of the most crucial techniques in cloud storage systems is data replication [3]. By addressing some of the major issues in this category, such as availability, reliability, security, bandwidth, and data access reaction time, data replication's fundamental goal is to enhance performance for data-intensive applications. This study provides a thorough evaluation and classification of cutting-edge data replication techniques across number of currently available cloud computing platforms, using a traditional classification to characterize existing models and spot unresolved issues.

Data deduplication schemes, data auditing schemes, and data classification schemes are the three main categories in the classification. The author only covered the various replication classifications, not how to carry out the replication. The research focuses on ways to increase the cloud storage system's dependability, availability, and high performance with the arrival of the big data era [4]. Important factors that affect how cloud-based applications are managed include the number of cloud users and the amount of cloud resources [5]. Resource provisioning is one of the most challenging problems to solve for time-varying and diverse workloads in the resource management scope as the volume of traffic to cloud-based apps increases. Users of cloud computing can access storage as well as other services as needed.

The amount of data produced today necessitates a large amount of storage [10]. Users have the choice of off-site data storage and online data access. Of course, using the cloud gives the user the desired storage. The number of replicas that must be created and determined statically at the time of cloud system setup is known as static replication policies [8]. Data replication needs to be dynamic to account for shifting user demands and storage capacity trends. Data replication in cloud storage systems has benefits in terms of reliability and performance, including fault tolerance, data availability, data locality, and load balancing [9]. Data blocks stored on the failed data node must be restored each time it fails to maintain the replication level. For the system, which is already overworked from user application workloads, this could represent a sizable burden.

III. Problem Statement

Replication has two functions, the primary one of which is to ensure high availability if nodes or shards go down. Making a replication policy on the source bucket that specifies the region and destination bucket to replicate to is all that is necessary to enable Object Storage replication. Following the creation of the replication policy, the destination bucket becomes read-only and is only updated via replication from the source bucket.

After a policy is created, objects that are deleted from the source bucket are also automatically deleted from the destination bucket. Accidental data change operations that are enabled for replication cause permanent data loss by deleting objects from both the source and destination buckets. This can also be a result of corrupted data uploaded to source bucket by software program or due to manual error. Most of the storage systems in use today have this issue.

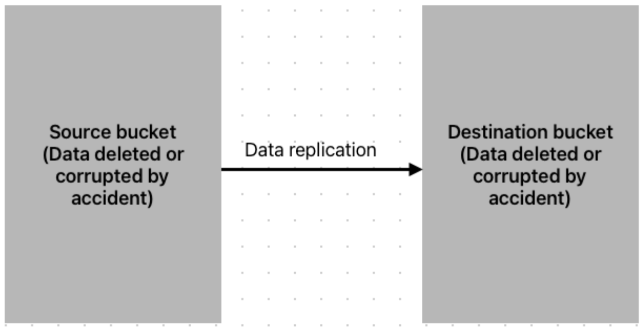

Source bucket (Data deleted or corrupted by accident)
Data replication
Destination bucket (Data deleted or corrupted by accident)

FIGURE 3. Object Storage Data Corruption Impacting Source And Target Bucket.

Figure 3 represents a scenario of possible data loss. In this case, corrupted data in source bucket results in corrupting the data in destination (backup) bucket.

Copying backups

You should back up your data in addition to replicating it. A backup will allow you to restore your data to the last known good state in the event of data corruption or accidental deletion. Additionally, not all data must be replicated instantly. For instance, scheduling backup replication may be more cost-effective if you are architecting for disaster recovery and your application has a longer RTO (Recovery Time Objective) and RPO (Recovery Point Objective).

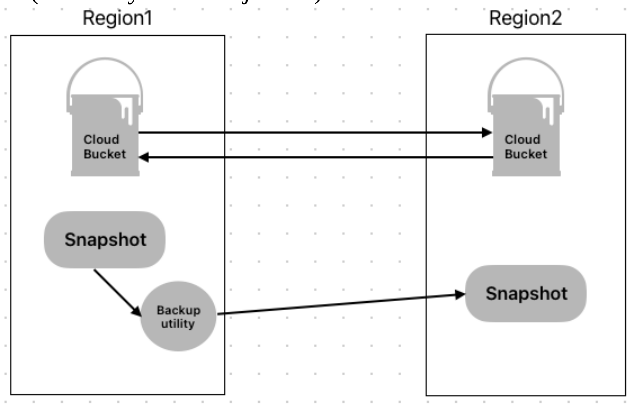


FIGURE 4. Replication With Data Backup For Snapshots.

For these use cases, the Backup utility shown in Figure 4 can automate data backup to meet business needs, and these backups can be set up to copy backups automatically to one or more Regions or accounts [7]. The length of a copy depends on the size of your backup and the distance it must travel. Run tests first to see if your defined RTO and RPO will be impacted by this cross-Region copy time. More services are being supported, which is particularly beneficial for those that don't offer real-time replication to another Region. For more information, consult your cloud provider's manual. A full copy of each dataset in the bucket is made when copying backups or taking snapshots, which increases the processing and storage costs.

Storage Lifecycle

Object Lifecycle Management operates by automatically acting in accordance with your defined rules. These rules tell Object Storage what to do on your behalf within a specific bucket, including deleting supported resources, moving objects to a different storage tier, and deleting uncommitted multipart uploads. An object lifecycle policy refers to all the lifecycle rules for a bucket. Object Lifecycle Management supports a variety of resources, including objects, object versions, and failed or uncommitted multipart uploads.

Actions come in two varieties: Transition actions and Expiration actions. When objects move from one storage class to another, transition actions determine this. When an object expires depends on its expiration actions. Costs associated with object lifecycle expiration vary depending on the time duration you choose. See Expiring objects for more details. Using the cloud console or command line interface, you can also configure the lifecycle. Storage lifecycle policies combined with replication and backup software only partially address the issue. Lifecycle policies can only be used at the bucket level, and processing and maintaining backup copies of all datasets is very expensive.

IV. ALGORITHM

A. *Algorithm idea*

Proposed AI algorithm in Table I detects and protects against accidental data loss. Data deletions can now be performed in either hard or soft mode where AI program determines the mode of data operation and lifecycle policy for the backup object. Hard mode performs direct delete while soft mode creates a copy before performing delete on the target object.

TABLE I

```
Algorithm 1:    Mode of data operation
1.      foreach Oi in ListOfObjects {
2.      find if Oi exists in Backup bucket
3.          if (Oi exists) {
4.            Use logic to determine the mode of operation.
              (Input: object usage, versions, etc.)
5.            if (mode == SOFT) {
6.                Set mode of operation to SOFT mode
7.                Create a copy of existing object
8.                Assign lifecycle policy
9.                Replace existing object
              }   // endif
10.         } else if (Oi NOT exists) {
11.           Set mode of operation to HARD mode
12.           Add new object to backup bucket
13.           } // endelse
14.     }   // endforeach
```

B. *Algorithm description*

Proposed AI algorithm detects and protects against accidental data loss. Data deletions can now be performed in either hard or soft mode where AI program determines the mode of data operation and lifecycle policy for the backup object. Hard mode performs direct delete while soft mode creates a copy before performing delete on the target object. Usage of existing objects along with its size, number of existing backups determines the outcome of algorithm. Assigning lifecycle policy and marking artifacts as a critical can also be achieved by suppling necessary configs to the program.

V. PERFORMANCE EVALUATION

In this section, first, the simulation parameters for running the simulation are defined. The details of the experimental setup are then discussed. Finally, the simulation results are presented.

A. *Simulation Parameters*

The simulation parameter values are taken from Table II. Requests generated randomly as a combination of same and different type of datasets performing add, update, delete operation with and without configuration parameters. Datasets used were combination of valid and corrupted datasets.

TABLE II. THE FOLLOWING TABLE SUMMARIZES THE SIMULATION PARAMETERS.

Object storage bucket	Mode of operation	Result
Add new objects. (Ten objects added)	HARD mode (for all objects)	Datasets added and replicated in destination bucket
Add new objects and update few existing objects (Two new objects added, one existing object updated)	HARD mode for all new objects, HARD or SOFT Mode determined for remaining objects	Source and destination bucket updated with new changes. (backup copy created for updated objects)
Add new objects, update few existing objects, delete existing objects (One new object added, one existing object updated, two objects deleted)	HARD mode for new object, HARD or SOFT Mode determined for remaining objects	Source and destination bucket updated with new changes. (backup copy created for updated, deleted objects)

B. *Results*

Simulation program, using above algorithm resulted in avoiding data loss during accidental data delete and corruption scenarios. Overall cost to store additional data is minimal compared to alternatives like enabling object versioning or versioning in combination with object lifecycle policies on the destination bucket objects.

VI. Storage

A. *Storage tiers*

Storage service is a high-performance, internet-scale storage platform that provides dependable and affordable data durability. A limitless amount of unstructured data, including analytical data and rich content like images and videos, can be stored using the Object Storage service.

You can store or retrieve data directly from the internet or from the cloud platform using Object Storage in a safe and secure manner. You can easily manage storage at scale with Object Storage's many management interfaces. You can start small and scale easily with the platform's elasticity without suffering any performance or service reliability degradation.

A specific compute instance is not required for object storage, which is a regional service. If you have internet access and can access one of the Object Storage endpoints, you can access data from anywhere within or outside the Oracle Cloud Infrastructure.

To meet the demands for effective, frequently accessed "hot" storage, less frequently accessed "cool" storage, and infrequently accessed "cold" storage, cloud infrastructure provides distinct storage class tiers. Storage tiers enable you to optimize access performance when necessary and cut storage costs when practical.

TABLE III. The Following Table Summarizes The Features Of The Standard, Infrequent Access, And Archive Tiers.

Tier	Storage Cost	Minimum Retention Period	Retrieval Fee	Availability SLA
Standard	Highest	None	No	99.9%
Infrequent Access	Cheaper	31 days	Yes	99%
Archive	Lowest	90 days	No	Data is offline and objects must be restored before they can be read.

A storage tier is given to each object that is uploaded to object storage. The object's storage fees, and any related retrieval charges are determined by the storage tier property.

B. *Storage Cost*

Object Lifecycle Management works by taking automated action based on rules that you define. In this section we analyze the storage cost of existing and our proposed system. Considering that each file has an ideal size of 2 MB. We have used a combination of add, update, delete operations on both valid and corrupted datasets and repeated each experiment few times then we have taken average at the end.

TABLE IV. Storage Cost For Buckets Located In A Single Region.

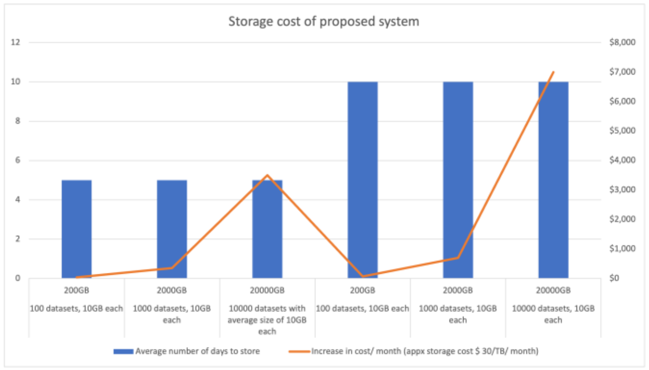
Storage Class	Class A operations (per 1,000 operations)	Class B operations (per 1,000 operations)	Free operations
Standard storage	$0.005	$0.0004	Free
Nearline storage and Durable Reduced Availability (DRA) storage	$0.01	$0.001	Free
Coldline storage	$0.02	$0.01	Free
Archive storage	$0.05	$0.05	Free

TABLE V. Storage Cost For Buckets Located In A Dual-Region Or Multi-Region.

Storage Class	Class A operations (per 1,000 operations)	Class B operations (per 1,000 operations)	Free operations
Standard storage	$0.01	$0.0004	Free
Nearline storage and Durable Reduced Availability (DRA) storage	$0.02	$0.001	Free
Coldline storage	$0.04	$0.01	Free
Archive storage	$0.10	$0.05	Free

Table IV, provides an example of the cost for storing data in same region while Table V, provides the cost for storing data in dual or multi-region. The cost might vary by cloud vendor.

FIGURE 4.



The storage cost usage of the proposed system using new solution is depicted in Figure 4 using the data from Table VI. The cost of storing is slightly higher than using the current process due to backup copies created by our new algorithm. Due to low backup data storage costs and lifecycle policies to delete the unwanted datasets, overall impact on cost is low.

TABLE VI. Storage Cost For New Proposed System

Scenario	Average number of data change operations/ day	Average number of days to store	Increase in cost/ month (appx storage cost $ 30/TB/ month)
100 datasets, 10GB each	200GB	5	$35
1000 datasets, 10GB each	2000GB	5	$350
10000 datasets with average size of 10GB each	20000GB	5	$3,500
1000 datasets, 10GB each	2000GB	10	$7,000

The overall cost is much lower than using other alternatives like enabling object versioning or using it in combination with object lifecycle management policies to achieve the same results. Other approaches were evaluated but not thoroughly discussed in the scope of current research work.

VII. Conclusion

In this paper, we address the issue of accidental data loss during data replication and propose an AI programming-based algorithm that takes the replication decisions in advance by monitoring and acting before the data loss occurs. Numerical example and experiments illustrate the benefits of our algorithm to solve the problem of accidental data loss during data replication. We have also analyzed accidental data loss scenarios and discussed alternatives to solve the problem. The outcome of implementing propose solution may be dependent on other factors, including storage cost of backup data, size of datasets, and frequency of data change operations. Evaluating impact of other alternative approaches will be part of our future work.

References

[1] B. Rajkumar, V. Christian, and S. S. Thamarai, Mastering Cloud Computing, Mc Graw Hill, 2013.

[2] K. Swaroopa, A. S. P. Kumari, N. Manne, et al., “An efficient replication management system for HDFS management,” Science Direct Material Proceedings, July 2021.

[3] A. Shakarami, M. Ghobaei-Arani, A. Shahidinejad, et al., “Data replication schemes in cloud computing: A survey,” Cluster Compute, vol. 24, pp. 2545-2579, 2021.

[4] Y. Su and W. Zhang, “A multi-index evaluation replication placement strategy for cloud storage cluster,” in Proc. of the 4th International Conference on Cloud and Big Data Computing, August 2020, pp. 20-26.

[5] M. Ghobaei-Arani, “A workload clustering based resource provisioning mechanism using biogeography based optimization technique in the cloud based systems,” Soft Compute, vol. 25, pp. 3813-3830, 2021.

[6] AWS Cloud Infrastructure (AWS) Documentation, architecture blogs [Online]. https://aws.amazon.com/blogs/architecture/

[7] Oracle Cloud Infrastructure (OCI) Documentation [Online]. https://docs.oracle.com/en-us/iaas/Content/home.htm

[8] E. Torabi, M. Ghobaei-Arani, and A. Shahidinejad, “Data replica placement approaches in fog computing: A review,” Cluster Compute, pp. 1-29, 2022.

[9] S. Gopinath and E. Sherly, “A dynamic replica factor calculator for weighted dynamic replication management in cloud storage systems,” Procedia Computer Science, vol. 132, pp. 1771-1780, 2018.

[10] S. N. John and T. T. Mirnalinee, “A novel dynamic data replication strategy to improve access efficiency of cloud storage,” Information System and e-Business Management, vol. 18, pp. 405- 426, 2020.

[11] I. A. Ibrahim, W. Dai, and M. Bassiouni, “An intelligent data placement mechanism for replica distribution in cloud storage system,” in Proc. IEEE International Conference on Smart Cloud, December 2016.

[12] D. Sun, G. Chang, S. Gao, et al., “Modelling a dynamic data replication strategy to increase system availability in cloud computing environments,” Journal of Computer Science and Technology, vol. 27, pp. 256-272, 2012.

Nilesh Suresh Jain has been writing since high school, when his elder sister gave him a journal in which to write down his stories. Nilesh and his family live in Virginia, USA. A Virginian since 2013, Nilesh received his Master of Computer Engineering degree from Mumbai University (India), and, after which he practiced as a software engineer for, CITI group, The Home Depot, Capital One before joining Oracle America Inc. in 2018.

He is currently working at Oracle, leading technology innovations on Oracle cloud(OCI), for utility companies across United States. In addition to being a prolific writer, Nilesh is a devoted philanthropist, and his greatest efforts are dedicated to support child and adult literacy programs in the United States and India. (Email: technoNilesh@gmail.com/ nilesh.s.jain@oracle.com)