Quest Journals
Journal of Software Engineering and Simulation
Volume 8 ~ Issue 5 (2022) pp: 26-33
ISSN(Online) :2321-3795 ISSN (Print):2321-3809
www.questjournals.org

Research Paper

An Efficient Model for Data Consistency during Replication in a Distributed System

[1]ChristyObiaziOsinga,[2]Bennett, E.O., [3]Chizi Michael Ajoku
Department of Computer Science, Rivers State University, Port Harcourt, Nigeria

Abstract
Ensuring data consistency during replication is a critical research issue in a distributed system. Guaranteeing that data replicacontinues to maintain its original status and its consistency while being copied into the distributed system is still an emerging challenge. As a result of this there is an increase in risk operational disruptions in distributed databases, financial losses in database distribution, legal issues in data consistency, passivity penalties and reputational damages to the database. Maintaining the accuracy and consistency of data over its entire life-cycle is another challenge in a distributed computing environment, this prompts intruders to temper and modify data in an invisible manner without permission of its true user thereby leading to data breach and inconsistency. The new model uses Hadoop model in connection with database to create large space memory location that handles all the bulk of unstructured data in the distributed computing environment. The evaluation parameter was taken based on the efficiency of the processing speed, data consistency, memory space and data integrity. The efficiency was recorded in percentages (%).It was observed that the proposed model performed better and achieved more data consistency in the distributed database environment as the efficiency level generateda75 % accuracy.
Keywords:*Replication, Distributed Systems, Data Consistency*

R*eceived 03 May, 2022;* ***R****evised 14 May, 2022;* ***A****ccepted 16 May, 2022 © The author(s) 2022.*
P*ublished with open access at* ***www.questjournals.org***

I. Introduction

The technique of exchanging data across various systems, such as hardware/software systems, to ensure a more fault tolerant, accessible and reliable process is known as replication. It is a method for automatically distributing copies of data and database objects around SQL Server instances while keeping them synchronized. To ensure a much better, dependable, fault tolerant and accessible information, data is replicated between alternate resources, such as hardware / software systems. Data replication, where certain data is saved on multiple storage devices, and computation replication, where the same computer activity is performed multiple times, are two of such instances [1]. In this context, safe information sharing is a difficult problem to solve. There will also be different policies for those who have several sources of data when it comes to data access and dissemination.

Replication protocols are divided into two categories; Dynamic (Active) replication and Passive replication [3].The role of decision support systems in an ambiguous environment, Processing or applying information and knowledge is required for decision making, and the optimal information/knowledge mix is determined by the characteristics of the decision-making context. Most decisions are somewhere in the centre, and in those cases, a decision support system can aid in the support and enhancement of human decision-making. In the decision-making process, the context in which a decision is made, the decision maker's manner of detecting and processing cues, and what the decision maker values or considers significant all play a part [4].

Since each block is unique and has different placement requirements, , the process of replicating or migrating dispersed data blocks to different locations on a chip entails the application of directory or broadcast-based lookup and coherence enforcement mechanisms. The scalability of a storage system determines its utility in various situations. Distributing data across multiple physical computers is a natural architectural decision when a large number of data items must be retained or the number of queries to the store exceeds the capability of stand-alone systems. When a small number of data items are provided to a large number of requests, replication is utilized [5].Stock and client data should be replicated at these multiple locations since it allows for quick access to the local copy and helps to withstand disasters if all machines at a physical location crash. On top of the massive table, the Personalized Search team devised a client-side replication technique that ensured that all clones were eventually consistent. The current system makes advantage of a built-in replication mechanism in the servers [6].

As a result, it's critical to ensure that database systems continue to function properly even in the face of a range of unforeseen situations. Replication is the key to maintaining high accessibility to database systems. While there are a variety of database replication methods, only silent replica crashes, which happen when the system confronts hardware problems, power outages, or other problems, are tolerable by most of them.

II. Related Works

Transactional replication is a sort of replication that allows data changes to be sent sequentially between servers in a distributed environment. Reporting servers and data warehousing settings, as well as Web servers and e-commerce applications, can all benefit from transactional replication. Many of the most popular SQL Server-based Web sites, such as MSN.com, Passport.com, Barnes and Noble.com, and Buy.com, use transactional replication. In high-performance environments, transactional replication offers a scalable and dependable method for data distribution. It is critical to use a Transaction Processing Replication (TP-R) solution that can keep transaction integrity near real time at data copy sites. Transactions are performed initially on one replica in slack replication. Only when a transaction commits, are any updates distributed to additional replicas, resulting in rapid response times.

The idea of a transaction had been utilized in database systems to make it easier to execute concurrent actions over shared data in a consistent manner. Transactions, on the other hand, have been employed considerably more widely since then, for example, in distributed systems in a variety of application contexts, where they increase data consistency and stability. For replicated databases, this is about committing a transaction. For several years, authors have advocated for the use of abstractions, they are frequently used to build dependable distributed systems (e.g., unanimity, full order broadcast) in order to enable crash-proof database replication or, further extensively, batch processing. Minimization of query execution time was a significant performance target in distributed database design, according to [7]who presented "Sub query Allocations in Distributed Databases Using Genetic Algorithms (GA). While total time for OLTP queries was to be lowered, response time for Decision Support queries was to be minimized. As a result, depending on the question type, different sub query allocations to sites, as well as their execution schedules, were found to be optimal. They present an analytical cost model for those two objective functions and outline the sub-query allocation problem. They used a genetic algorithm to solve the problem because it was NP-hard (GA). The results reveal that query execution plans with a total reduction aim failed to meet the latency goal. Simulated tests employing sophisticated queries with up to 20 joins were used to test the GA method. When compared to exhaustive enumeration, GA delivered optimal solutions in all cases in a fraction of the time.

According to [8], in distributed database systems, databases are frequently divided and copied across multiple sites to reduce connection expenses. Static fragmentation, replication, and allocation, as well as a priori query analysis, have previously been used to handle complicated challenges including fragmentation, replication, and site allocation. Many modern distributed database system applications generate extremely dynamic workloads across multiple sites, with fast fluctuations in access patterns. Systematic re-fragmentation and redistribution can considerably enhance productivity in these situations. They described DYFRAM, a distributed mechanism for dynamic database fragmentation and distribution in distributed systems based on site configurations, in that publication. The strategy divides, replicates, and reallocates data based on recent access history, with the goal of increasing the number of local accesses over remote accesses. They demonstrated the practicality of their strategy through experiments conducted on the DASCOSA distributed database system, demonstrating that it cuts network overhead for primary access patterns significantly.

[9]Demonstrated that fragmenting various tables and distributing fragmented data over network sites was a key component of any Relational Distributed Database Query Optimizer in their paper. They proposed a Genetic Algorithm (GA) for establishing a relatively close fragmentation plan for picking nodes or sites for iteratively adding vertically fragmented data attributes in two components. They examined the benefits of adopting the suggested Genetic Algorithm (PGA) over other commonly used algorithms as well as the un-partitioned scenario. The use of PGA over other strategies yields positive outcomes in a replicated Distributed Database across a Wide Area Network, according to experimental results.

[10] Deals with the modified ratio estimator, which uses a linear different combination of the median and the co-efficient of the auxiliary variable, as well as the use of auxiliary data in practice to improve the efficiency of the parameter for the distribution of consisted data.

Authentication, cryptographic algorithms, access control, and other innovations have all contributed to the creation of safe and trusted data distribution systems, according to [11]. Path authentication and entity tracking in distributed systems have been presented for heterogeneous distributed systems, which are significantly required in various distributed applications. A security-driven scheduling system has been devised.

In a dispersed context, remote client authentication is the most essential. This is explained using a three-factor authentication mechanism.

[12]Reflects the conceptual properties of these databases and proposes a realistic distributed system approach for implementing polyglot persistence in data-intensive systems in the big data era. The authors performed an excellent job. However, in terms of time complexity and cost-benefit analysis, there was no meaningful comparative system study of the Polyglot Persistence Model with other models.

[13]Employed deep reinforcement learning in a distribution scenario to understand how to optimize join queries. According to the author, most optimizers avoid full enumeration of all possible join orders and instead use heuristics to prune the search area. However, the described query optimization could not be implemented without a specific relational database and trained datasets.

III. Methodology

The proposed system design methodology is Object Oriented Design Methodology (OODM). The Object-Oriented and Design Methodology (OODM) is a technical system approach to application analysis and design. This development methodology splits the entire system into subsystems and modules using a recursive procedure and object-oriented development style.

Stage 1(Linking and updating the dataset): This step includes middleware components that ensure that the dataset is treated as a single logical entity with a Logical Dataset Id, as well as updating and linking other replica datasets. When it wants to amend the dataset, it visits one of the replicas, modifies its contents, and then asks the replica consistency server to propagate the change to all of the other replicas.

Stage 2 (Synchronized Replication): This stage employs a synchronous propagation policy, which helps ensure that no replicas will be accessible until all the originals have been revised, and the policy of asynchronous propagation, which enables replicas with outmoded content to be viewed while others are being revised. This means that the data consistency constraint is satisfied in the time required to complete updating transmission, which is more feasible in this case.

Stage 3 (Global Replica Consistency Service (GRCS)implementation): The principal point of contact for requests for updates from all users' replicas that will be uploaded into the distributed database is GRCS. Other sub-modules are involved, which execute different functions such as data updating when the replication process is running.

Stage 4 (Registration of customers in the Local Replica Consistency Service): the Local Replica Consistency Service is included in this stage (LRCS). It is in charge of registering new local duplicates that it will control directly, as well as keeping these replicas up to date. Before accessing the distributed system, each client must first register. Before adding the replica, the first database is created, as well as a replicated server that houses all duplicated clients. When a replica is added to a distributed database and a query is run, the database in the client is updated based on the query. The database on the server is automatically updated when the tables on the client are updated.

The replica consistency catalogue stores data that has previously been replicated to a few sites and their locations were reproduced.

3.1 System Architecture

The design is built on breaking down a system's tasks into distinct, re-implementable, and self-contained objects, each with its own set of data and behaviour as shown in figure 3.1.

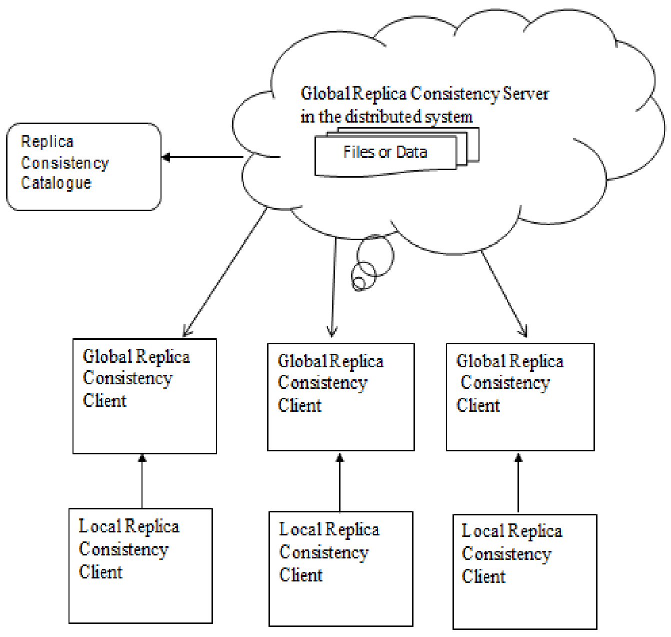
Global Replica Consistency Server in the distributed system
Files or Data
Replica Consistency Catalogue
Global Replica Consistency Client
Global Replica Consistency Client
Global Replica Consistency Client
Local Replica Consistency Client
Local Replica Consistency Client
Local Replica Consistency Client

Figure 3.1: Architecture of the proposed system

The Global Replica Consistency Server: This component provides consistency for all the client replica or user using the application in the distributed database system. It also provides authentication and security to all the user in the local replica consistency service and ensure encryption of information. The client or the users in the distributed database system register their details and send to the Network security Interface for accessing. The users most register into the interface to upload his information. The user registration pages are user name, password and the confirm password. All the uploaded coping files from each user are send to the Global Replica Consistency Server. And the content of the sent files remains consistent during review. This replica can be downloaded from the Global Replica Consistency Server to the replica consistency catalogue for update and modification. The files in the Global Replica Consistency Server contain several documents and their title including description of each file.

Global Replica Consistency Client: While the system is made up of numerous distributed components, the Global Replica Consistency Client is the principal entry point for update requests. Individual files will be sent to the distributed system in this folder. The files in the global consistency client are encrypted which cannot be accessible by other clients.

Local Replica Consistency Service (LRCS): The Local Replica Consistency Service is in charge of the registration and update of new local replicas that it handles directly. The RCS uses the Replica Consistency Catalogue (RCC) to record metadata such as each replica's location and status, as well as LRCS information. The algorithms for the consistency and replication is shown in Algorithm 3.1

Algorithm 3.1: Data Consistency Algorithm

```
Step 1: Initialize number of nodes (n):
Step 2: Ensure records are split and represented in a fixed
size blocks stored in data node
Step 3: Each data node is sorted and shuffled, use to stores
attribute
step 4: apply $n = H/d = c*r*S/(1-i)/d$
Step 5: for each time step of the node do Select a random
action at with probability
otherwise select at = argmaxa $Q(\varphi(st), a; \omega$
Step 6: Calculate the number of nodes using the formula.
Number of nodes (n) = $M/d = C * R*S/ (1 – i) / d$ .
Step 7: Establish a connection between different files.
Step 8: Pass the connection using the steps below:
Load the files and split into given split ratio
Step 11: Summarize data
Step 12: Separate data by its class: each of the data is
separated by its class.
Step 15: display result as required
End
```

Database Design

Tables are used in database architecture to examine objects such as entities and attributes. Data in a table is logically structured rows and columns arrangement, similar to a spread sheet, each field shows a property, and each row provides a different record. Table 3.1 shows different categories of objects used in the proposed system. In the database table there are six fields (Attributes) namely: Serial Number, Primary Key, Field-Name, field-Size, Data-Type and Description. The primary key is a reference to the database. The field names include: user-id, username which is the User Name for the user, user password, user email address and logged in for Time and date of last login. The data type refers to the type of information that will be stored in the field by the user. Each user's user-id is a unique identifier

.**Table *3*.1: Database Table for the Proposed System**

S/N	Key	Field Name	Data Type	Field Size	Description
1	PK	User-ID	Int	20	Unique identifier for each user
2		User-Name	Varchar	30	User Name for the user
3		User-Password	Varchar	20	Password for the user
4		User-Email Address	Varchar	30	Email for each user
5		Last-Logged-in	Date/Time	20	Time and date of last login
6		Consistency View	Varchar	20	View the consistency of Data
7		Assign Privilege	Varchar	40	Assign privilege to users
8		Replica Consistency Catalogue	Varchar	40	Store or hold all the replica

IV. Results and Discussion

The goal of this study was to design and implement an efficient model for data consistency during replication in a distributed system that will improve on the consistent coping of replica and data integrity in the existing system. The proposed system provides a consistent replication of files in the distribution of various files or information uploaded by different client or users into a computing environment or distributed system. The model contained a single name node which comprises cluster of different size blocks that are stored in data nodes, the data nodes are sorted and shuffled. Each of the cluster nodes are encrypted with a security authentication code that are managed by the administrator.

System Requirements

i. JavaScript is used for the system's interface design and coding.
ii. PHP Hypertext Pre-processor: PHP was used to create the system's backend (database connection).
iii. Structured Query Language: Structured Query (SQL) Management System is used to design the database.
iv. Hadoop model: The Hadoop is connected to the MYSQL database to create large memory location for any kind of unstructured data that will be uploaded or downloaded in the distributed computing environment.

System Testing

The Efficient Model for Data Consistency during Replication in a Distributed System that was created was put to the test using:
i. Unit Test
ii. Validation Test
iii. Verification Test

The various patterns and tests that were carried out on the developed are displayed in Tables 4.1, 4.2 and 4.3 respectively.

Table 4.1: Test Case 1.0 (Login)

Test Case Number	**1.0**
CaseName	Login
Precondition	The login form has been loaded and the application has been launched.
Case Input	Enter username and password
Case Expected Output	The dashboard (Home Page) as been successfully loaded.
Case Steps/Description	**Input:** In the corresponding textboxes, type the username and password, then click the login button. **Output:** If the login and password are correct, the Dashboard (Home Page) will load successfully. Otherwise, a failure warning for Invalid username or password appears.

Table 4.2: Test Case 1.1 (Security Key)

Test Case Number	**1.1**
CaseName	**Security Key**
Precondition	The user selects the security key and enters it into the application to be authenticated by the system.
Case Input	create an account and enter the correct security code for login before clicking Register submit
Case Expected Output	Successful result of the key are displayed
Case Steps/Description	**Input**: Create a user account and provide the relevant security code. **Output**: Displayed successful authentication security code

Table 4.3: Test Case 1.2 (Access Files)

Test Case Number	**1.2**
CaseName	**Access Files**
Precondition	Files to be accessed have been selected
Case Input	Before selecting open file, make a list of all the files you want to access.

Case Expected Output	Successful result of displayed files will be open
Case Steps/Description	**Input**: select file for accessing **Output**: Displayed updated and modified file

Table 4.4: Performance Evaluation of both Proposed and Existing Systems

Attribute Parameter	**Proposed System** **Efficiency Rate (%)**	**Old System 1 (Wiesmann et al. 2015) Efficiency Rate (%)**	**Old System 2 (Chihoub, 2013)** **Efficiency Rate (%)**
Data Consistency	77	50	55
Data Integrity	65	50	48
Memory Space	80	70	65
Processing Speed	50	30	55

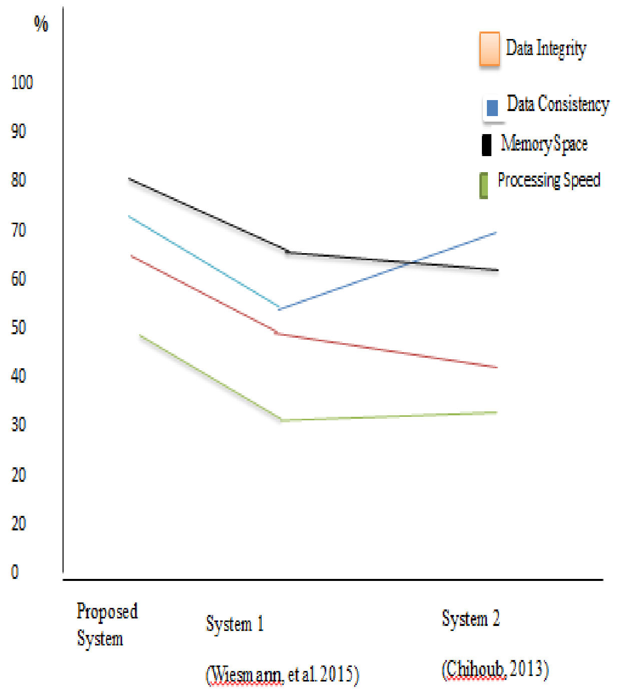

Fig 4.1: Performance Evaluation Graph (Existing vs. Proposed Systems)

V. Conclusion

In distributed systems, replication is critical. Due to the unintended scalability and fault tolerance, systems failures are common in replicated data. With the use of a partial replication and an analytical model, the proposed system is used to clear the database faults. It figures out how partial replication compares to full replication in terms of scalability. When compared to the existing systems, it performs better in terms of performance efficiency. System failures were eliminated, and distributed systems with databases were able to access many replications with no fault tolerance. The outcome of the experiment shows that for distributed databases, data replication in the new system will improve performance by reducing false errors. The developed model for data replication in a distributed system has proffered solution to the replica challenges in a distributed database system.

References

[1]. Sanjay, J. (2017). Regional climate change scenarios. Observed Climate Variability and Change over the Indian Region, 35(26), 285-304.

[2]. Alawari, I. &Bennett E. O. (2018). Hybrid Technique for Optimization of Query Processing in aDistributed Database, International Journal of Computer Science and Mathematical Theory (IJCSMT), 4(2), 19 – 27

[3]. Michael H. Zack (2012). Strategic Pretext for Knowledge Management. Proceeding Third Eur. Conf. Organ. Knowledge, Learn, 12(43), 1137–1148.

[4]. Manjula K.A. (2010). A Study on Applications of Grid Computing in Bioinformatics, International Journal of Engineering Technology (IJET), 4(7), 1 – 8

[5]. Carlo, B. (2013). Schism: A Workload-Driven Approach to Database Replication and Partitioning," In Proc of the Conference on very large database Endowment, 3(1):231-323.

[6]. Ghemawat, C. (2015).The globalization of business education: Through the lens of semiglobalization, 27(4):391-414.

[7]. Abiye-Suku, Ominini Monima & E. O. Bennett (2019): Efficient Algorithms for Data Extraction in Big Data. International Journal of Computer Science and Mathematical Theory. Vol 5 (1); 75-86

[8]. Olav Hauglid (2015).Dynamic fragmentation and replica Management in distributed database systems, Distributed and Parallel Databases, 28 (2-3), 157-185.

[9]. Rajinder S. (2011). Optimizing Access Strategies for a Distributed Database Design using Genetic Fragmentation," International Journal of Computer Science and Network Security, 11(6): 180- 183.

[10]. Nitu, Mehta (2013).A Modified Ratio-Cum-Product Estimator of Finite Population Mean Using Ranked Set Sampling, 45(2), 321-542.

[11]. Vijay, Prakash(2012).A Review on Security Issues in Distributed Systems14(5), 300- 304.

[12]. Jaja, Azunna Ibimina & Bennett, E. O. (2018). Enforcement of Partial Referential Integrity in a Database System. International Journal of Computer Science and Mathematical Theory. 4 (2), 27-33.

[13]. Wordu, C. N. & Bennett, E. O. (2018): Automatic Extraction of Semantic Information from XML Documents. International Journal of Computer Science and Mathematical Theory. 4 (1), 55-62.