10th International Conference on Control Engineering &Information Technology (CEIT-2025)
Proceedings Book Series –PBS- Vol 23, pp.59-63

Hybrid Data Duplication to Ensure Adaptability and Availability

Kahina Benhabiles [#1], Djamel Eddine Zegour [#2]

[#] *Laboratoire de la Communication dans les Systèmes Informatiques, Ecole nationale Supérieure d'Informatique ESI Oued Smar, Algiers, 16309, Algeria*

[1] k_benhabiles@esi.dz

[2] d_zegour@esi.dz

***Abstract*— Data replication is a long-standing research focus, especially challenging when servers use different data structures. This leads to *hybrid data duplication*, where data is stored across varied structures to ensure adaptability (efficient structures) and availability (resilience to failure). Unlike traditional replication, hybrid duplication explicitly considers structural adaptability. The proposed system models clients and heterogeneous servers using concurrency techniques from data replication. A RAM-based Java simulation was used to test the setup, laying the groundwork for future work on PBST (Partitioned Binary Search Trees), which generates balanced search trees. The paper also defines hybrid duplication, contrasts it with replication, and outlines replication techniques in distributed systems.**

***Keywords*— Replication, Hybrid duplication, PBST, Concurrency.**

I. INTRODUCTION

Data replication involves saving and maintaining copies of data on different storage nodes or servers. Unlike replication, data duplication is not necessarily a synchronous or periodic process between the source and the target. Data duplication refers to having one or more copies of data. While hybrid data duplication shares the same basic definition, it differs in that the duplicated data are stored across different data structures based on application needs and are transformed to match the target data structures.

Various replication algorithms have been proposed in distributed systems to address data placement issues, such as availability, reliability, high latency, security and more [1]–[6], [8], [10]. The process of replication should not negatively impact system performance or user experience. Choosing an appropriate data placement method ensures optimal performance and resource utilization. It is crucial to adapt the replication strategy based on four key questions: Which data to replicate? When to replicate? Where should the new replicas be placed? And how many copies should be made [8]–[10]?

In the next section, we present a literature review on data replication in distributed systems.

A. Literature Review

Various replication techniques have been proposed across edge, fog, cloud, and distributed systems. In edge computing, data and processing occur on edge servers [2], while fog computing enables localized processing for IoT with enhanced security frameworks [1]. A dynamic resource and replica allocation model for edge environments was also introduced to minimize node rental costs [2]. In P2P networks, [3] proposed a strategy to reduce delay and increase query success.

For cloud environments, [4] presented EnE-Rep to reduce energy use via optimized resource allocation. A machine learning-based approach [5] was introduced to dynamically generate hybrid replication strategies tailored to specific scenarios. Reference [10] emphasized the need to balance energy consumption and profit in cloud databases.

In grid systems, [6] proposed EA2-IMDG, using in-memory data storage for faster access and scalability. A multi-objective replica placement strategy in HDFS was developed to balance storage and network load [7].

Hybrid heuristic methods like PSGWA [8] (combining PSO and GWO) and AOEHO [9] (combining AO and EHO) were introduced to optimize replica placement, reduce access time, and enhance cost-efficiency in fog and distributed systems.

B. Projected Work

Today, computer memory is abundant, widely available, and increasingly inexpensive. This makes it advantageous to duplicate data, whether on the same machine or across distributed systems. Hybrid duplication allows data to be represented in multiple forms (e.g., different types of trees), thus ensuring high availability.

ISSN: 2961-6611

PBST(n) [19] is a framework that provides a family of binary search trees whose balance level is controlled by a partitioning parameter. This framework generalizes the well-known Red-Black Tree structure: in fact, PBST(2) is exactly a Red-Black Tree. The balance quality of PBST(n) is inversely proportional to the parameter *n*—the lower the *n*, the more balanced the tree.

The main idea of this work is to duplicate data on servers' RAMs across *p* different trees (or servers) using various PBST forms (e.g., PBST(2), PBST(3), ..., PBST(p+1)). We then propose the design of a library that provides data operations with tunable performance, depending on application needs. These operations are executed in parallel across the different tree types.

For example, one application may require fast updates with slower searches, while another may prefer the opposite. Applications can also combine several trees to leverage the combined benefits of each type, using synchronous and asynchronous replication protocols to maintain consistency.

In this short paper, we used simple data structures on the servers, namely an unsorted array, a sorted array, and a binary search tree (BST). In a future extended version of this work, we plan to incorporate the PBST family of trees.

Technically, this work addresses efficient data access management, focusing on concurrency control using locks and the ROWA (Read One Write All) technique. Locks ensure thread-safe operations by allowing multiple readers but only one writer at a time [13]–[15]. ROWA improves read performance and availability by permitting reads from any replica, while writes must update all replicas—leading to higher write costs [11], [12]. The proposed system was tested via a RAM-based simulation using Java, enabling multi-threaded communication between nodes.

The paper covers data replication techniques (Section 2), hybrid duplication (Section 3), a preliminary test platform (Section 4), and concludes in Section 5.

II. Replication Techniques

An overview of the relevant studies is presented in the following references.

Reference [1] proposed a secure data replication model for edge and fog computing, combining RBAC and ABAC to protect data integrity, confidentiality, and availability. Only authorized entities can access replicas, and integrity checks ensure data consistency.

[2] focused on balancing storage overhead and user experience. They modeled and optimized replica placement using real-world datasets, improving access efficiency while reducing overhead.

[3] introduced a replication strategy in peer-to-peer networks using NSGA-II, aiming to minimize access delay and maximize query success rate, though it faces limitations like scalability and dynamic conditions.

[4] presented EnE-Rep, an energy-efficient replication strategy for cloud systems. It minimizes energy use by selecting energy-efficient nodes and avoiding under/overloaded ones, reducing VM migrations.

[5] developed a machine learning-based hybrid replication approach that evolves over time. It generates adaptable strategies optimized at runtime and stores the best solutions for reuse.

[6] used In-Memory Data Grid (IMDG) for replication and scheduling in grid environments. IMDG improves performance via parallel processing and reduced latency, outperforming centralized models in several metrics.

[7] proposed a replica placement method for HDFS using an improved multi-objective memetic algorithm (IMOMAD), optimizing storage and network load while outperforming other algorithms.

[8] introduced PSGWA, combining PSO and GWO to find efficient replica placements that reduce data access time and processing cost. It excels in static environments.

[9] presented AOEHO for fog computing, merging AO and EHO to replicate popular files near users. It outperforms other methods in varied conditions.

[10] evaluated replication strategies in cloud databases, highlighting the trade-off between energy use and profit. They proposed managing replicas to lower energy consumption and operational costs.

The next section will define data replication, duplication, and their distinctions.

III. Replication vs Hybrid Duplication

Data duplication refers to creating one or more copies of data, either in the same or different locations. It can occur intentionally (e.g., for backup) or unintentionally (due to human or system error) [16],[17],[18]. Unlike replication, duplication is not necessarily synchronous or periodic, and duplicated data is independent of the original. In contrast, data replication is a continuous, coordinated process aimed at ensuring fault

ISSN: 2961-6611

tolerance, availability, and load balancing, with replicas kept up to date. The term "data duplication" is less commonly used in literature compared to "data replication" or "deduplication." The next section introduces hybrid duplication.

A. Hybrid Duplication

Hybrid duplication involves storing data across different data structures on servers based on application needs. The data is transformed according to the target data structures. Hybrid duplication is an interesting approach as it ensures both availability and adaptability. It offers higher performance and query flexibility at the cost of complexity and storage. Replication is easier to implement, lighter, and consistent.

IV. TEST PLATFORM

To evaluate our proposal, we ran a RAM-based Java simulation on a single machine using a test platform with three components: AppGenerator, clients, and servers.

AppGenerator produces insert, delete, and search operations, sent to clients, who forward them to servers holding different data structures (unsorted array, sorted array, BST). Communication uses sockets (AppGenerator ↔ Clients ↔ Servers) and threads (between servers). Updates use ROWA with locks for concurrency, and a verification step ensures data consistency across servers. A timer signals the end of the simulation and displays results. The architecture is illustrated in Fig. 1.

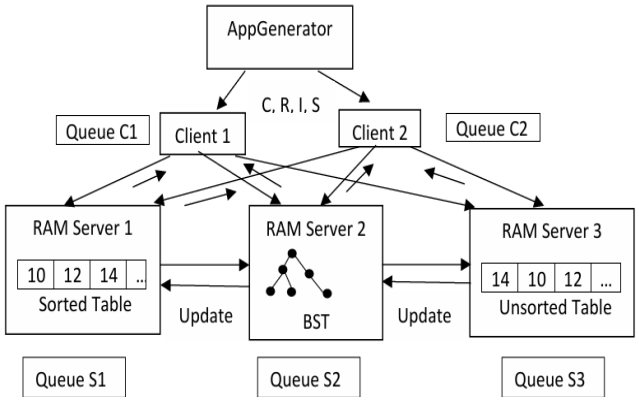

Fig. 1 architecture of the test platform: using ROWA method and locks

We ran two simultaions with data samples of size n = 5000 and 8000, respectively. For each server, we calculate: the number of writes (cw) and reads (cr), the total time write time (Wtime) and read time (Rtime) in milliseconds, and the average time of writes (avgW) and reads (avgR), also in milliseconds. The results are shown in tables I and II.

TABLE I
N = 5000

Server	Cw	Cr	Wtime	Rtime	avgW	avgR
S1	2592	521	157	10	0.061	0.019
S2	2592	561	99	19	0.038	0.034
S3	2592	530	113	22	0.044	0.042

TABLE III
N = 8000

Server	Cw	Cr	Wtime	Rtime	avgW	avgR
S1	4040	854	219	7	0.054	0.008
S2	4040	863	116	27	0.029	0.031

ISSN: 2961-6611

S3	4040	883	132	23	0.033	0.026

The results show that there is no concurrency, and they are coherent. The number of writes is the same for all servers, indicating that each server contains the same data. However, the number of reads varies. Servers exhibit different execution times depending on the data structure used and the order of operations.

V. CONCLUSION

This paper presents various replication techniques used in distributed systems. It defines hybrid duplication, introduces a test platform, and concludes with findings and perspectives.

To evaluate our proposal, we conducted a RAM-based simulation of our implemented test platform using a Java program on the same computer. To manage replicated data, we applied the ROWA strategy, and employed locks to ensure proper concurrency control. In our future work, we plan to implement a hybrid duplication approach for a specific data structure called PBST (Partitioned Binary Search Trees), which can generate a family of binary search trees based on specified balance degree [19].

ACKNOWLEDGMENT

We thank Pr W.K Hidouci for his help and suggestions.

REFERENCES

[1] H. Alalibo, E.O. Bennett, N.D. Nwiabu, and D. Matthias, “Secure Data Replication in Edge and Fog Computing Environments,” *International Journal of Computer Science and Mathematical Theory (IJCSMT)*, vol. 10, no.3, pp. 187–201, 2024.

[2] C. Li, J. Bai, Y. Chen, and Y. Lu, “Resource and replica management strategy for optimizing financial cost and user experience in edge cloud computing system,” *Information Sciences*, vol. 516, pp.33–55, 2020.

[3] A. Samimi, and M. Goudarzi, “A novel strategy for optimal data replication in peer-to-peer networks based on a multi-objective optimisation – NSGA-II algorithm,” *IET Networks*, pp. 1–17, 2024.

[4] M. Alghobiri, “EnE-Rep: An Energy-Efficient Data Replication Strategy for Clouds,” *Baltic J. Modern Computing*, vol. 12, no.3, pp. 304-326, 2024.

[5] S.M.A. Bokhari, and O. Theel, “A Genetic Programming-Based Multi-Objective Optimization Approach to Data Replication Strategies for Distributed Systems,” in *2020 IEEE Congress on Evolutionary Computation (CEC)*, 2020, Glasgow, UK, pp. 1-9.

[6] A.H. Guroob, “EA2-IMDG: Efficient Approach of Using an In-Memory Data Grid to Improve the Performance of Replication and Scheduling in Grid Environment Systems,” *Computation*, vol. 11, no. 65, 2023.

[7] Y. Li, M. Tian, Y. Wang, Q. Zhang, D.K. Saxena, and L. Jiao, “A new replica placement strategy based on multi-objective optimisation for HDFS,” *Int. J. Bio-Inspired Computation*, vol. 16, no. 1, pp.13–22, 2020.

[8] B. Arasteh, S.S. Sefati, S. Halunga, O. Fratu, and T. Allahviranloo, “A Hybrid Heuristic Algorithm Using Artificial Agents for Data Replication Problem in Distributed Systems,” *Symmetry*, vol. 15, no. 487, 2023.

[9] A.a. Mohamed, L. Abualigah, A. Alburaikan, and H.A.E.-W. Khalifa, “AOEHO: A New Hybrid Data Replication Method in Fog Computing for IoT Application,” *Sensors*, vol. 23, no. 2189, 2023.

[10] M. Seguela, R. Mokadem, and J-M. Pierson, “Comparing energy-aware vs. cost-aware data replication strategy,” in *10th International Green and Sustainable Computing Conference (IGSC 2019)*, 2019, Alexandria, United States.

[11] M. Rabinovich, and E. D. Lazowska, “An efficient and highly available read-one write-all protocol for replicated data management,” in *Proceedings of the Second International Conference on Parallel and Distributed Information Systems*, 1993, San Diego, CA, USA, pp. 56-65.

[12] N. Ahmad, A.N. Abdalla, and R.M Sidek, “Data Replication Using Read-One-Write-All Monitoring Synchronization Transaction System in Distributed Environment,” *Journal of Computer Science*, vol. 6, no. 10, pp. 1095-1098, 2010.

[13] Y. Zhang, S. Shao, H. Liu, J. Qiu, D. Zhang, and G. Zhang, “Refactoring Java Programs for Customizable Locks Based on Bytecode Transformation,” *IEEE Access*, vol. 7, pp. 66292-66303, 2019.

[14] Oracle Documentation. (2025) Java.Util.Concurrent.Locks API specification. [Online]. Available: https://docs.oracle.com/en/java/javase/22/docs/api/java.base/java/util/concurrent/locks/package-summary.html

[15] Oracle Documentation. (2025) Java.Util.Concurrent.Locks API specification. [Online]. Available: https://docs.oracle.com/en/java/javase/22/docs/api/java.base/java/util/concurrent/locks/ReentrantReadWriteLock.html

[16] Oracle. (2025) Data duplication Implications and Solutions. [Online]. Available: https://www.oracle.com/dz/data-duplication/

[17] Y. Guo, Q. Zhuge, J. Hu, J. Yi, M. Qiu, and E. H-M. Sha, “Data Placement and Duplication for Embedded Multicore Systems with Scratch Pad Memory,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 6, pp. 809-817, 2013.

[18] L. Li, Y. Zhang, and Y. Ding, “MT-DIPS: a new data duplication integrity protection scheme for multi-tenants sharing storage in SaaS,” Int. J. Grid and Utility Computing, vol.9, no. 1, pp. 26–36, 2018.

[19] S. Zouana, and D.E. Zegour, “Partitioned binary search trees: a generalization of red black trees,” *Computación y Sistemas*, vol. 23, no. 4, pp. 1375–1391, 2019.

ISSN: 2961-6611