BULLETIN of Cherkasy State Technological University

Journal homepage: https://bulletin-chstu.com.ua/en

Vol. 29 No. 1. 2024

UDC 004.738.004.75
DOI: 10.62660/bcstu/1.2024.23

Performance optimisation techniques for Conflict-free Replicated Data Types (CRDT)

Yurii Rabeshko[*]
Postgraduate Student
National University of Water Management and Nature Management
33000, 11 Soborna Str., Rivne, Ukraine
https://orcid.org/0009-0002-0763-7138

Yurii Turbal
Doctor of Technical Sciences, Professor
National University of Water Management and Nature Management
33000, 11 Soborna Str., Rivne, Ukraine
https://orcid.org/0000-0002-5727-5334

Abstract. The research relevance is determined by the need for distributed data handling optimisation that does not cause conflicts during replication, since the introduction of such types of data leads to an increase in the need for their optimisation and performance. The study aims to develop techniques to improve the efficiency of various types of data with conflict-free replication. To achieve this goal, the methods of analysis and experimentation were used. The study results demonstrate that the use of certain optimised methods of conflict-free data types in replication leads to a significant improvement in efficiency and reduction of resource costs. The results show that caching and precomputation, asynchronous synchronisation, and local processing are the most effective methods for improving the efficiency of Conflict-free Replicated Data Types. It is established that the choice of a specific method for improving the performance of Conflict-free Replicated Data Types should be justified and based on a careful analysis of the requirements and characteristics of a particular distributed system. This takes into account user needs, data criticality, edit frequency, etc. The research includes the development and optimisation of synchronisation algorithms that significantly improve the performance of these types of data in distributed systems. The work includes simple software implementations for detailed analysis and study of methods for optimising the performance of data types in conflict-free replications. The introduction of distributed computing has made it possible to optimise the processes of data replication and synchronisation, which helps to reduce overheads and improve system performance. It is concluded that the study is of practical importance since the use of optimised conflict-free data types with replication in real distributed systems can improve their efficiency and reliability. The practical significance of the study lies in the possibility of applying optimised methods of data types with conflict-free replication in real distributed systems, which will increase their performance and ensure efficient operation under conditions of high load and limited resource potential

Keywords: ways to increase performance; data replication; distributed systems; synchronisation efficiency; data synchronisation methods

Article's History: Received: 07.12.2023; Revised: 14.02.2024; Accepted: 18.03.2024.

Suggested Citation:
Rabeshko, Yu., & Turbal, Yu. (2024). Performance optimisation techniques for Conflict-free Replicated Data Types (CRDT). *Bulletin of Cherkasy State Technological University*, 29(1), 10-23. doi: 10.62660/bcstu/1.2024.23.

[*]Corresponding author

INTRODUCTION

Modern information systems require superior performance and reliability for the replication of large data amounts and synchronisation across distributed systems. CRDTs (Conflict-free Replicated Data Types) are known for said reliability, as they combine replication and conflict resolution. They are abstract data structures that can be replicated and synchronised between different nodes in a network without the need for centralised control. However, despite the potential benefits of CRDTs, there are issues related to their performance and scalability in real-world applications. It is these aspects that are of interest to researchers and developers, as improving the performance and efficiency of CRDTs can have a significant impact on distributed systems. In this context, studying ways to improve the performance of CRDTs becomes an urgent problem for researchers and practitioners.

With the rapid growth of data volumes and distributed systems in the modern world, conflict-free data types of research are becoming increasingly relevant. Implementing conflict-free data replication systems requires a constant search for efficient synchronisation and optimisation methods, especially under heavy loads. This topic becomes especially relevant due to the need to ensure performance and reliability in distributed systems, where data access must be ensured even in the absence of stable communication and in the face of conflicts and failures. In addition, innovations in this area can solve the current challenges of the modern Internet of Things, streaming data processing, and improve the reliability and performance of cloud services, making this topic critical for the further development of the information society. In general, the results of the conflict-free CRDT data types of study can improve performance and reduce resource costs when using CRDT in distributed systems, which will be of great practical importance in the modern information environment.

The research problem at this stage is becoming more relevant and complex, as existing CRDT synchronisation and replication methods demonstrate limitations that seriously restrict their performance and scalability. This necessitates the development of more efficient and optimised synchronisation methods to ensure the reliability and stability of distributed systems. In addition, the overheads associated with CRDT replication and synchronisation processes are coming to the fore and are becoming important factors that can affect the performance and speed of distributed applications. Given that even small delays in the synchronisation process can cause conflicts and loss of data compatibility, research in this area is becoming key to ensuring the quality and reliability of distributed systems in the modern information landscape.

Previous research on CRDT determined challenges and limitations, but many aspects remain unexplored. For instance, A. Lutsenko (2020) explored models of conflict-free replicated data types to resolve conflicts between replicas and their use in mobile collaborative offline applications for text documents. The author also developed a software application for creating and editing text documents in a collaborative mode, and analysed experimental data. N. Boliubash and M. Olinyk (2023) investigated certain performance improvement techniques for a progressive web application. To optimise performance, they use analysis and synthesis methods, methods for developing progressive web applications, and methods for optimising their performance. B.O. Biletskyy (2019) discussed the main stages of machine learning, including data collection and storage, training, and evaluation. The study determined that special CRDT data structures are essential in building NoSQL DBMSs (database management systems) to solve the problems of parallel modification of shared resources and always resolve inconsistencies.

Y. Kordiaka and O. Padko (2021) addressed the development of collaborative software for editing text documents in real-time, using conflict-free replicated data types. This study describes in detail the implementation of CRDTs and their application to enable the synchronisation and collaboration of users on shared text documents in a networked environment. V. Ryabushkin (2020) implemented a system for automatic conflict resolution in text using CRDT. The author emphasises that CRDT can be automatically replicated in a parallel application without the need to manually resolve conflicts. V. Shynkarov (2023) reviewed the main optimisation approaches and their applications. The author describes linear programming methods for solving optimisation problems and also considers nonlinear optimisation methods. In contrast to the above studies, which focus on either conflict-free data types or general optimisation methods, this study focuses on methods for improving CRDT and their development.

The aims of the study were to develop approaches to improving the performance of conflict-free replication data to reduce overheads and increase the speed of data replication in a distributed system. Synchronisation efficiency, scalability, and optimisation methods development for various algorithms to improve performance and reduce resource costs in systems using CRDT were addressed.

LITERATURE REVIEW

The issue of distributed data optimisation and replication is relevant in the modern information society. The growth of data volumes and distributed computing has led to an increased demand for efficient methods to ensure the availability, reliability and performance of these systems. Previous studies on this topic point to several key aspects that should be considered. These include the importance of ensuring data integrity during replication, minimising resource consumption when synchronising replicas, and ensuring the high performance of distributed systems.

For instance, B. Portela *et al.* (2023) investigated the use of CRDTs in distributed systems. The authors propose an approach to secure data in CRDTs using secure multi-party computation (MPC). The results include extending formal models of CRDT security, proving the security of such designs using MPC, and developing a language and type system for creating secure CRDTs.

Yu. Mao *et al.* (2022) considered the possibility of using the reversal of operations in CRDTs to achieve strong final consistency (SEC) in distributed systems. The authors defined the concept and design of reversible CRDTs that allow for the reversal of changes made. They also proposed different reversal methods for implementing reversal operations and discussed the impact on system performance. D. Adas and R. Friedman (2021) discussed CRDT sketches for storing statistics about data flows and their application in distributed systems. The authors present CRDT algorithms for implementing such sketches and analyse their performance based on real-world tasks.

G. Zakhour *et al.* (2023) described the concept of CRDTs that ensure the compatibility of replicated data and the absence of conflicts between replicas, and propose a programming language Propel with a special type of system for automatic verification of the algebraic properties of CRDTs. They presented successful verification of CRDT implementations and compared it with other verification tools. N. Saquib *et al.* (2021) presented CRDT employment to ensure strong determination of replicated data in distributed systems. CRDTs allow for resolving conflicts between replicas without additional coordination and provide high data availability. The study also discusses the importance of using CRDTs to achieve strong condemnation and demonstrates their effectiveness in realistic distributed system scenarios.

I. David and E. Syriani (2022) addressed the growing need for real-world collaborative modelling in engineering and proposed a framework for this. The framework is based on conflict-free replicated data types that provide scalable and reliable replication mechanisms. The authors demonstrate the benefits of their framework through a model example and compare it with other state-of-the-art modelling solutions. M. Nicolas *et al.* (2022) investigated the use of CRDT in distributed systems to ensure high availability. They point out the problem of the growing size of identifiers in CRDTs and propose a new CRDT sequence with a renaming mechanism that reduces the overhead and improves system performance over time. Experimental results confirm the effectiveness of this mechanism.

D. Brahneborg *et al.* (2022) addressed distributed CRDT systems with read and update operations without synchronous network requests. However, most CRDTs are based on atomic dispatch, which can be excessive for independent CRDT objects. This study proposes a replication protocol, CReDiT (CRDT enhanced with intelligence), which effectively exploits the switchability of CRDTs and reduces the complexity and network load compared to atomic broadcasting. CReDiT uses fewer communication steps and is less sensitive to server failures, ensuring efficient data replication. A. Tranquillini (2022) proposed a method for structuring the state of mobile applications using Redux and strongly typed languages, which allows the use of CRDT to create a coherent state. This enables replicas to edit their state autonomously and merge conflicts. The study also considers the use of a server as a communication channel and analyses the impact of this architecture on the design and optimisation of CRDTs. The analysis of the listed studies indicates the relevance of the CRDT topic. However, the research also highlights the need for further study of data types with conflict-free replication and the creation of methods to optimise their performance.

MATERIALS AND METHODS

The research was conducted using the analysis and experimentation methods. The analysis method was used to address the theoretical aspects of CRDT employment in distributed systems. This method was used to analyse studies of various authors on this topic, the mathematical properties of CRDT, synchronisation algorithms, and other theoretical foundations that underlie data types with conflict-free replication. The analysis is necessary for a deeper understanding of theoretical concepts related to CRDTs and their possible applications in distributed systems. The analysis of the mathematical properties of conflict-free data types determined in detail the mathematical foundations of CRDT, such as commutativity and associativity of operations. The study of synchronisation algorithms allowed to learn CRDT synchronisation algorithms, including their mechanism for resolving conflicts and ensuring data consistency. The analysis of possible applications in distributed systems helped to identify possible areas of application of CRDT in distributed systems and assess their potential impact on data integrity and performance.

The experimental method conducted as part of this study included the implementation of data type-specific approaches using conflict-free replication methods in distributed systems. During the experiment, actual distributed systems were implemented that used CRDT structures to ensure data integrity and avoid potential conflicts between replicas. The results of the comparative analysis of different CRDT implementations determined the advantages and limitations of each approach. The VS Code development environment and the JavaScript programming language were used to implement software solutions and conduct the experiment. Furthermore, the Cross-Platform File Format Apps platform was used to visualise the structural scheme of the local processing method. An important element of the study was the mathematical formula and task for the caching and precomputation method, as well as a table with the analysis and subsequent comparison of the results for asynchronous synchronisation. This made it possible to obtain objective information about the performance and efficiency of the considered methods in practical conditions.

Thus, analysis and experimentation methods were used to develop methods of local processing, caching and pre-computation, and asynchronous synchronisation. Their advantages and disadvantages are described. Programme operation containing a local copy of data as an object and data update processing, a method of combining local copies of data from other replicas, identical updates, adding the result to the cache, checking the result, calculating and storing the result in the cache, simulating the calculation of a value by key, obtaining calculated values for users, an asynchronous method of applying an update from another replica, and asynchronous replica merging is described. The task of improving the performance of the online store and the formula for the caching and pre-calculation method are explained. Formula (1) illustrates how reducing data access time affects the speed of query handling:

$$S = k \,/\, T\_access, \tag{1}$$

where S – query handling speed; T_access – data access time; k – proportionality constant that determines the amount of influence of access time on handling speed.

The main components of the structural diagram were also shown, namely the internal structure of the replica, the processes of local processing of changes and saving the results, which include a local copy of the data, a change log, making changes, local processing, conflict resolution, and change results.

RESULTS

Conflict-free data types of performance improvement methods in replication. Modern distributed systems require a reliable and efficient mechanism for synchronising data between different replicas. CRDTs, which provide conflict resolution and data integrity, are a powerful solution for achieving these goals. However, improving their performance and optimising them can create new perspectives for distributed systems.

CRDT performance optimisation is relevant in the context of distributed systems, where data consistency and performance are crucial. The performance of data types with conflict-free replication has certain advantages and disadvantages. The advantages include ensuring conflict resolution and data integrity, increasing replication performance and speed, and reducing resource usage. In other words, optimisation techniques can improve the CRDT conflict handling efficiency, which ensures the reliability and integrity of replicated data. They are designed to reduce the overhead of synchronising data between replicas, which leads to improved system performance. Some methods are designed to speed up the data replication process by reducing transmission and processing time for updates. Optimisation techniques can also reduce the memory and computing resources required to store and synchronise data. The disadvantages are the complexity of implementation, the possibility of new problems, context-specific dependencies, and the possibility of losing some CRDT guarantees. Some optimisation methods may be difficult to implement, requiring additional effort from developers. Optimisation may lead to new problems or deteriorate other aspects of the system, which requires careful analysis and testing. The effectiveness of such methods may depend on the specific use case and system characteristics. Optimisation itself may lead to the loss of some guarantees, such as strong final consistency, which requires careful analysis and balance.

The following methods can be used to optimise the efficiency of data types with conflict-free replication: "Local processing", "Caching and precomputation" and "Asynchronous synchronisation". The Local Processing method is one of the possible methods for improving CRDT performance. It implies the ability to perform most of the CRDT operations locally on each replica without the need for direct network communication with other replicas. The main idea is that replicas can independently process and update their local copies of data based on changes made by users, as long as these changes do not conflict with other changes in the system. This method reduces the amount of network traffic, as not every operation needs to be sent to other replicas immediately. This is especially helpful when network resources are limited, or a large number of replicas are present.

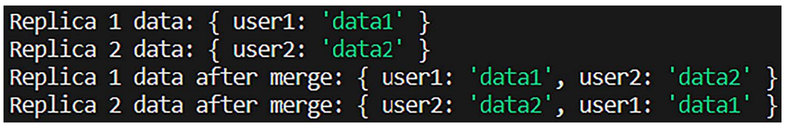
The Local Processing method involves performing certain actions. First, each replica stores its local copy of the data and keeps a log of changes. Next, users can make changes to their local copies of the data, and the replica tries to apply these changes to its local copy of the data. If there are no conflicts between the changes, they are accepted and stored locally. If a conflict occurs, the replica tries to resolve it locally or queries other replicas for a solution.

The advantages of the Local Processing method are reduced network load, reduced communication costs, and localisation. In other words, the method significantly reduces the amount of network data exchange, which can improve system performance, especially in conditions of limited network resources. Fewer data exchange means less network communication costs, which can lead to savings in system maintenance costs. Replicas can operate on a local copy of the data, which keeps data access times low.

Disadvantages of the Local Processing method are conflicts and, the complexity of algorithms. This means that when conflicts arise between changes on different replicas, additional synchronisation may be required to resolve them. Algorithm development that can resolve conflicts locally can be an important and complex task (Fig. 1). While the program output is shown in Figure 2.

```
class CRDT {
  constructor() {
    this.data = {}; // Local copy of data as an object
  }

  applyLocalUpdate(key, value) {
    // Local data refresh handling
    this.data[key] = value;
  }

  merge(otherCRDT) {
    // Local data copy merge method for other replicas
    for (const key in otherCRDT.data) {
      if (!(key in this.data)) {
        this.data[key] = otherCRDT.data[key];
      }
    }
  }

  getData() {
    return this.data;
  }
}

const replica1 = new CRDT();
const replica2 = new CRDT();

replica1.applyLocalUpdate("user1", "data1");
replica2.applyLocalUpdate("user2", "data2");

// Replica 1 and replica 2 have different updates
console.log("Replica 1 data:", replica1.getData());
console.log("Replica 2 data:", replica2.getData());

// Replica merge
replica1.merge(replica2);
replica2.merge(replica1);

// Now both replicas have both updates
console.log("Replica 1 data after merge:", replica1.getData());
console.log("Replica 2 data after merge:", replica2.getData());
```

Figure 1. An example of a simple application of the Local Processing method
Source: compiled by the authors



```
Replica 1 data: { user1: 'data1' }
Replica 2 data: { user2: 'data2' }
Replica 1 data after merge: { user1: 'data1', user2: 'data2' }
Replica 2 data after merge: { user2: 'data2', user1: 'data1' }
```

Figure 2. Local processing method output
Source: compiled by the authors

This code is an example of a CRDT implementation and demonstrates how two replicas merge their data without conflicts. Both replicas end up with the same data after the merge. A diagram of the internal structure of the replica and the processes for processing changes locally can also be drawn (Fig. 3).

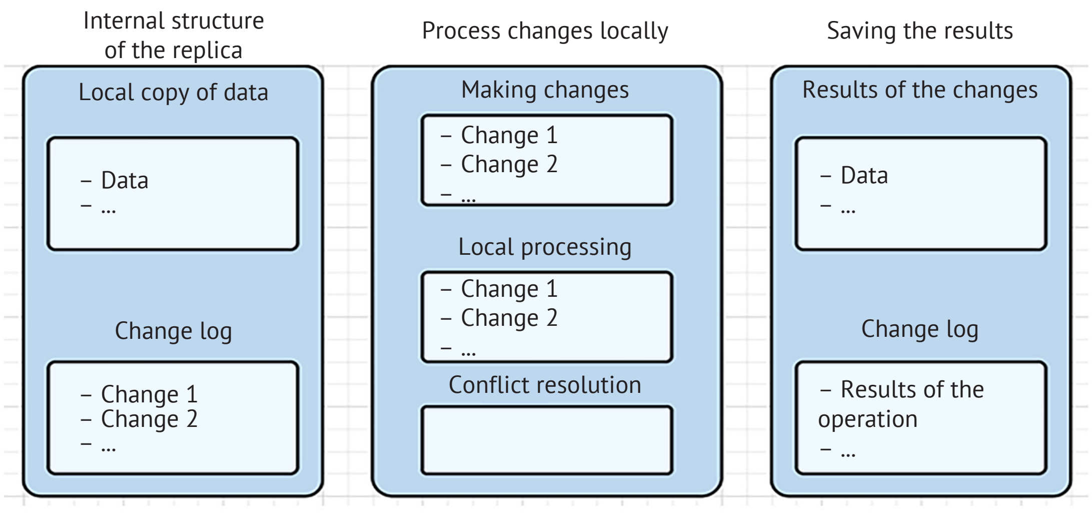


Figure 3. Local processing method diagram

Source: compiled by the authors

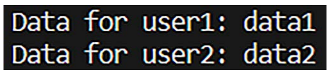
The main components of this diagram include the internal structure of the replica, the processes of processing changes locally, and storing the results. With the internal replica structure, each replica has its local copy of the data, and the log records all changes that are made to the local copy of the data. The processes of processing changes locally contain certain stages. When a user makes changes to their local copy of data, these changes are first recorded in the change log. Then the replica tries to apply these changes to its local copy of the data directly. If there are conflicts between changes, the replica tries to resolve them locally using conflict resolution algorithms. Saving the results is done as follows: if the changes are successfully processed and conflicts are resolved, the results are saved locally, and the local copy of the data is updated. Local copies of data can also pre-calculate the results of operations and store them for later use, which helps to improve performance and reduce network data exchange.

The Cache and Precompute method are another key approach to optimising CRDT performance. It involves the use of a mechanism for caching data and results of operations on replicas. This method provides local copy storage of certain data and results of operations on replicas. Local data copies are used for quick access to information without the need for network exchange. It also involves the pre-calculation of some transaction results that can be known in advance and stored on the replicas. The Local Processing method also involves specific procedures. Each replica maintains its local copies of data and operation results. When performing operations where the results can be pre-calculated, the replica cheques whether it already has the result of this operation. If the result is already prepared, the replica uses it without operating again. If the result is not known, the operation is performed, and the result is stored in the cache for later use. The advantages of this method are improved access time, network communication savings, and localisation. This method significantly reduces the time to access data and transaction results, as most requests can be handled locally. The reduction in network communication leads to more efficient use of network resources. Replicas can operate on local data and results, which improves efficiency and reduces network load.

The disadvantages of the Cache and Precomputation method include cache management, data relevance, memory consumption, and precomputation. Data management is required for the effective operation of this method, as a faulty cache of data and results can lead to incorrect results or increased memory consumption. Keeping local copies of data can lead to data out-of-date issues if other replicas have changed the data, requiring additional synchronisation to keep the information up to date. Caching can increase memory consumption on replicas, especially if the data or results of operations are voluminous. Furthermore, not all operations can be pre-computed in advance, and for some operations, this method may not make sense (Fig. 4). While the program output is shown in Figure 5.

```
class CRDTWithCaching {
  constructor() {
   this.data = {}; // Local copy of data as an object
   this.cache = {}; // Cache for storing pre-calculated results
  }

  applyLocalUpdate(key, value) {
   // Local data update processing
   this.data[key] = value;

   // Add the result to the cache
   this.cache[key] = value;
  }

  getFromCache(key) {
   // Check if the result is in the cache
   if (key in this.cache) {
    return this.cache[key];
   } else {
    // If the result is not in the cache, calculate it
    const result = this.computeValue(key);
    // Save the result to the cache
    this.cache[key] = result;
    return result;
   }
  }

  computeValue(key) {
   // Simulate calculating the value by key
   return `Computed value for ${key}`;
  }

  getData() {
   return this.data;
  }
 }

 const replica1 = new CRDTWithCaching();

 replica1.applyLocalUpdate("user1", "data1");
 replica1.applyLocalUpdate("user2", "data2");

 // Retrieving data from the cache
 console.log("Data for user1:", replica1.getFromCache("user1")); // Calculated value for user 1
 console.log("Data for user2:", replica1.getFromCache("user2")); // Calculated value for user 2
```

Figure 4. Implementation of a simple programme of the Cache and Precomputation method
Source: compiled by the authors



```
Data for user1: data1
Data for user2: data2
```

Figure 5. The result of the caching and pre-calculation method
Source: compiled by the authors

This Cache and Precomputation implementation is a simple CRDT performance optimisation programme. It provides storage and use of local copies of data to improve access to it. The result of executing the code outputs the merged data from the replicas that contain the data for the users.

Certain formulae and tasks can also be used to show how methods for optimising the efficiency of conflict-free data types in replications work. For example, for the caching and precomputation method, a simple formula that shows how reducing data access time can affect the speed of query service (formula (1)) can be used. Different formulae can be applied to this method: data access time, the ratio of access time reduction, memory consumption, and other performance indicators.

Tasks can also be of different types, for example, the task "Improve the performance of an online store when loading the product page and calculating prices with discounts". First, an online store with many products and various discounts that can be applied to the products, should be assumed. Users visit the product page to view the assortment and prices. This task can be approached from different angles: without caching and with caching. In the first case, every time a user opens the product page, the server must calculate prices for each product, accounting for current discounts. This may require database queries and many calculations. It can take a long time to load the page, especially if with a large assortment of products and many users. In the second case, the first calculated prices for products are stored in the cache on the server. When the product page is loaded, the server cheques whether the cache contains relevant results for a given query (for example, for a given user and their selected region). If the results are in the cache, the server returns them to the client without calculating them. This renders the product page quickly and reduces the load on the server and database.

Task results:

- the time it takes to load the product page from the cache is reduced, no repeated calculations are required;
- the server processes fewer database queries and saves computing resources;
- users get quick access to product prices, which increases their satisfaction and increases conversion.

This approach helps improve the performance of the online store and provides a positive user experience.

The asynchronous synchronisation method is also a strategy for optimising CRDT performance. This method increases concurrency and reduces latency while synchronising data between replicas. It involves allowing replicas to communicate and synchronise data asynchronously, without explicitly waiting for responses from other replicas. This improves system performance in distributed environments and enables faster data exchange.

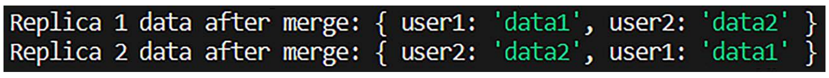
This method operates as follows. Each replica stores its local copy of data as an object. Replicas can asynchronously merge their data with other replicas. During the merge, replicas access the data of other replicas and apply to their local data what they do not already have. If the data is already local or known from previous merges, the replica ignores it. Since the operation is asynchronous, replicas can continue operating even if the join has not yet been completed.

The advantages of the asynchronous synchronisation method include increased parallelism, reduced latency, and performance. That is, replicas can interact and synchronise data in parallel, which allows for better use of the system's computing resources. The absence of blocking synchronisation reduces waiting time during data exchange and operations. The method allows the system to maintain stable performance even during intensive data exchange and heavy workloads.

"Asynchronous synchronisation" has the following disadvantages: the complexity of implementation, the possibility of conflicts, and task specificity. Developing asynchronous synchronisation can be complex and requires careful management of conflicts and the order in which operations are performed. Asynchronous synchronisation can lead to conflicts that require additional processing and resolution. Moreover, not all types of data and operations can be effectively synchronised asynchronously, and for some tasks, other methods may be more appropriate (Fig. 6). While the program output is shown in Figure 7.

```
class CRDT {
  constructor() {
    this.data = {}; // Local copy of data as an object
  }

  applyLocalUpdate(key, value) {
    // Local data update processing
    this.data[key] = value;
  }

  async mergeAsync(otherCRDT) {
    // Asynchronous method for merging local copies of other replica data
    for (const key in otherCRDT.data) {
      if (!(key in this.data)) {
        await this.applyRemoteUpdate(key, otherCRDT.data[key]);
      }
    }
  }

  async applyRemoteUpdate(key, value) {
    // Asynchronous method for applying updates from another replica
    if (!(key in this.data)) {
      this.data[key] = value;
    }
  }

  getData() {
    return this.data;
  }
}

const replica1 = new CRDT();
const replica2 = new CRDT();

replica1.applyLocalUpdate("user1", "data1");
replica2.applyLocalUpdate("user2", "data2");

// Asynchronous replica merging
(async () => {
  await replica1.mergeAsync(replica2);
  await replica2.mergeAsync(replica1);

  // Both replicas have both updates
  console.log("Replica 1 data after merge:", replica1.getData());
  console.log("Replica 2 data after merge:", replica2.getData());
})();
```

Figure 6. An example of a simple application of the asynchronous synchronisation method
Source: compiled by the authors



```
Replica 1 data after merge: { user1: 'data1', user2: 'data2' }
Replica 2 data after merge: { user2: 'data2', user1: 'data1' }
```

Figure 7. Asynchronous synchronisation method result
Source: compiled by the authors

This code illustrates the principle of asynchronous data synchronisation between CRDT replicas. It shows the process of asynchronous data merging of replicas and outputs the result of this synchronisation. There are two options for different approaches to data synchronisation to compare the main characteristics of the asynchronous synchronisation method (Table 1).

Table 1. Comparison of different data synchronisation approaches

Characteristic	Variant 1	Variant 2
Core idea	Asynchronous exchange without interruptions.	Interrupting change exchange with interruption of other replicas
Approach principle	Replicas maintain local data and a change log. Changes made on one replica are asynchronously sent to other replicas	Replicas maintain local data and a change log. Changes made on one replica are sent to other replicas with interruption
Conflict resolution	Conflicts are resolved asynchronously	Conflicts are resolved by locking
Advantages	Increased concurrency and performance, reduced network load, and localisation	Ensure data is up-to-date, and reduce the risk of conflicts
Disadvantages	Possible conflicts that need to be resolved, the complexity of developing synchronisation mechanisms	Blocked flows and expectations, possible conflicts, less concurrency

Source: compiled by the authors

Thus, the table presents a comparison of the two options for data synchronisation in the asynchronous synchronisation method and highlights their main characteristics, advantages and disadvantages. Using the table helped to visualise the difference between the two approaches to data synchronisation and help to understand their impact on the system. Thus, each of the CRDT types of performance improvement methods has its advantages and disadvantages, and the choice of a particular method will depend on the needs and requirements of a particular distributed system. The study results demonstrate the importance of the context and project requirements when selecting a method to ensure data integrity and improve performance in a distributed system.

DISCUSSION

Y. Zhang *et al.* (2023) discussed the importance and complexity of developing conflict-free replicated data types to ensure the efficient operation of distributed systems. The researchers pointed out the need for formal specification and verification of CRDT design, as well as systematic testing of the implementation. The authors proposed the MET (Model Evaluation Tools) framework, which combines model checking at the design level and exploratory testing at the CRDT implementation level. This approach detects errors and improves the reliability of CRDTs in distributed systems. The common aspects between the present study and the aforementioned one are that both consider the use of conflict-free replicated data types. However, this study uses programmes written in C++ in the VS Code environment for methods to improve CRDT performance. Moreover, Y. Zhang *et al.* (2023) focused on the design and testing of conflict-free data types using the MET framework.

N. Saquib *et al.* (2022) investigated extensions to CRDT data types that enable their use in distributed systems such as the Internet of Things (IoT), even in the face of resource constraints and diversity. The methods are designed to ensure robustness and strong replica consistency and to avoid the limitations of standard CRDTs. The authors studied various conflict-free data types in replicas and found that their methods allow for increased performance of operations compared to conventional CRDTs for the workloads considered. The common aspect of the studies is the investigation of CRDT performance optimisation techniques. However, N. Saquib *et al.* (2022) used IoT to achieve this goal, whereas the study did not have specific distributed systems.

Yu. Ou and J. Zhou (2023) highlighted that CRDTs are a popular approach for group editing in applications, but they have the problem of inefficient data fetching. To solve this problem, the paper proposes a new data fetching algorithm, Relative Distance Skip List (RDSL), which is an efficient and stable solution. RDSL uses a probabilistic structure of identifiers based on the relative distances between CRDT nodes. This algorithm improves the efficiency of CRDT by providing fast access to data. Both studies develop certain approaches to improve the efficiency of CRDT types. However, to implement this improvement, the study proposed methods of local processing, caching and precomputation, and asynchronous synchronisation. Yu. Ou and J. Zhou (2023) proposed an RDSL algorithm.

Sh. Laddad *et al.* (2022a) highlighted the challenges in developing reliable distributed applications and proposed the use of CRDT as a promising approach to achieve coordination in distributed systems. The authors emphasised the limitations of CRDT in providing secure observations of data state. They proposed the extension of conflict-free data types in replications with a query model based on monotonicity from the CALM (Consistency as Logical Monotonicity) theorem to provide more guarantees and secure interaction with the replicated state of applications. This study creates new possibilities for improving the reliability and efficiency of CRDT in distributed systems. Both studies share the idea of using conflict-free data types in replication in distributed systems, considering their advantages and disadvantages, as well as the possibility of making CRDT more efficient. However, Sh. Laddad *et al.* (2022a) used the CALM theorem, while the present study contains certain tasks and formulae, but no theorems.

F. Guidec *et al.* (2021) considered the use of conflict-free replicated data types in opportunistic networks (OppNets), where information is transmitted through temporary radio contacts between mobile nodes. A delta-state-based algorithm for synchronising CRDT replicas in OppNets is proposed, and experimental results confirm the effectiveness of this algorithm, especially when working with container CRDTs. This approach exploits CRDT functionality to withstand asynchronous communication and can find practical application in distributed networks with unpredictable connections between nodes. This study, similar to the work of F. Guidec *et al.* (2021), focused on conflict-free data types in replications and does some experiments on the effectiveness of CRDT. However, the aforementioned paper used the OppNets network, and this paper considers CRDT in different distributed systems without being tied to a specific network.

S.E. Brynjulfsen (2023) addressed replicated conflict-free data type efficiency optimisation using an SQLite database. CRDTs enable data replication without the need for coordination and provide certain guarantees regarding data consistency. However, there are performance issues due to the use of triggers in SynQLite, which leads to slower performance compared to SQLite. The study proposes various approaches to improve the performance of CRDT types and identifies a solution that significantly improves their performance. The common aspects between the two papers are the optimisation of CRDT performance and the practical implementation of these types. However, this article uses different C++ programmes, while the study by S.E. Brynjulfsen (2023) used an SQLite database.

S. Rostad (2020) investigated distributed CRDTs that guarantee strong end-to-end logic (SEL) and have important properties such as switchability and idempotency. The authors have considered delta CRDTs, which are state-based, and state-based CRDTs, where their instances are synchronised by sending their state to each other. The paper explores existing types of CRDTs and proposes new designs, including the "Causal Length Set" (CLSet), a simple and efficient delta state-based CRDT, and the "Multiple Value Map" (MVMap), a CRDT designed with user-friendliness in mind. Both articles discuss the effectiveness of CRDTs. However, the article by S. Rostad (2020) uses CLSet and MVMap designs for this purpose. The present study did not address the designs but rather used methods for the performance of distributed data types.

G. Litt *et al.* (2022) considered the possibility of applying CRDT to multi-text data with formatting that allows editing and sharing of data without the need for coordination. The authors created a model for preserving user intent in multi-text editing and developed a CRDT algorithm that satisfies this model. The basic idea of the algorithm is to store formatting next to the text and output the final formatted text from these areas in a deterministic way. The algorithm has been prototyped, validated by testing, and integrated into the editor's interface, and its properties of preserving correctness and user intent have been proven. G. Litt *et al.* (2022) focused on conflict-free data types in replications and their testing, as well as on the convenience of CRDT for users. However, this paper developed methods for optimising CRDT types, and another paper developed a CRDT algorithm for preserving user intentions in multi-text editing.

Sh. Laddad *et al.* (2022b) explored the types of CRDTs, which are a powerful tool for creating distributed systems without the need for coordination. However, the researchers emphasised that their proper design is a challenge. This study introduces Katara, a system that automatically generates validated CRDTs from consistent implementations of data types. It provides a simplified and reliable way to create CRDTs that can be useful for developers of distributed systems. Both studies focus on CRDTs, but Sh. Laddad *et al.* (2022) used the Katara system to create CRDTs. This article did not use specific systems but studied CRDTs in general.

J. Bauwens and E. Gonzalez Boix (2020) addressed CRDTs, which are special data types designed for highly available systems and guarantee a certain level of consistency. However, the implementation of CRDTs requires keeping track of additional metadata, which can be inefficient at scale. The authors analysed the metadata problem in CRDT and proposed a new optimisation strategy that helps to reduce the memory overhead. They also proposed a solution to improve the responsiveness of CRDTs built on robust causal language. The results of the study show that this approach can significantly ease metadata management compared to existing methods. Both studies considered approaches to optimise conflict-free replicated data types. However, unlike this article, the study by J. Bauwens and E. Gonzalez Boix (2020) focused on the introduction of metadata in CRDT. This article discusses the use of CRDT with different data types.

This study uses a specific programming language, tables, flowcharts and formulae. However, unlike previous research, it is not limited to specific frameworks, systems, algorithms, theorems, networks, databases, or designs. This makes it more versatile and easier to use. Nevertheless, both this study and all the articles listed above make an important contribution to the development of CRDTs and their application in distributed systems. They provide a variety of approaches to solving problems and improving the functionality of CRDTs, which extends their practical use in modern applications and systems.

CONCLUSIONS

This research aimed to improve conflict-free replicated data types, which are key components in distributed systems. A wide range of aspects related to these data types were covered, and the goal was to develop new methods to ensure their efficiency and applicability.

The study analysed various aspects of CRDTs, including their theoretical basis, synchronisation mechanisms, and potential challenges associated with their application in real-world distributed systems. The possibilities of improving the performance of CRDTs were investigated using analysis and experimentation methods. To implement the study, simple programmes were written and various examples, a table, a formula, and a block diagram were used. The C++ language and VS Code environment were used for the programmes, and the Cross-Platform File Format Apps platform was used for the block diagram.

To optimise the efficiency of CRDT, local processing, caching and pre-computation, as well as asynchronous synchronisation, are deemed most effective. The study results show that the local processing method improves system performance by reducing network load and reducing network communication costs, as well as providing low data access time. By using the caching and precomputation method, it is possible to reduce data access time, reduce network communication, and increase overall network efficiency. The asynchronous synchronisation method makes better use of the system's computing resources, reduces the waiting time during data exchange, and maintains stable performance even under heavy load. The results also confirm that these methods significantly improve performance and reduce resource consumption. However, each of the methods for improving CRDT performance has not only advantages but also limitations, and the choice of a particular method should be determined by the requirements and needs of a particular distributed system. Based on the results of the study, it is possible to conclude that CRDTs have great potential in improving the consistency and performance of distributed systems. However, they also require additional research and optimisation to be used in practise.

Based on this study, several recommendations can be made for further research and development in the field of conflict-free replicated data types. For example, further research could include the development of new applications of CRDTs in different industries. Developers can continue to research methods to improve the performance of these data types. They should also create more powerful tools for implementing CRDTs and testing their performance and reliability. Future research should focus on security and privacy issues when using CRDTs. In summary, the results of this study indicate the importance and prospects of using CRDTs in distributed systems and provide specific methods to improve their performance and efficiency.

ACKNOWLEDGEMENTS

None.

CONFLICT OF INTEREST

None.

REFERENCES

[1] Adas, D., & Friedman, R. (2021). Sliding window CRDT sketches. In *2021 40th international symposium on reliable distributed systems (SRDS)* (pp. 288-298). Chicago: IEEE. doi: 10.1109/SRDS53918.2021.00036.

[2] Bauwens, J., & Gonzalez Boix, E. (2020). From causality to stability: Understanding and reducing meta-data in CRDTs. In *MPLR '20: Proceedings of the 17 th international conference on managed programming languages and runtimes* (pp. 3-14). New York: Association for Computing Machinery. doi: 10.1145/3426182.3426183.

[3] Biletskyy, B.O. (2019). Horizontal and vertical scalability of machine learning methods. *Problems in Programming*, 2, 69-80. doi: 10.15407/pp2019.02.069.

[4] Boliubash, N., & Olinyk, M. (2023). Methods for increasing the performance of the library's progressive web application based om the RAIL model. *Information Technology and Society*, 1(7), 13-20. doi: 10.32689/maup.it.2023.1.2.

[5] Brahneborg, D., Afzal, W., & Mubeen, S. (2022). Resilient conflict-free replicated data types without atomic broadcast. In *Proceedings of the 17th international conference on software technologies ICSOFT* (pp. 516-523). Lisbon: SciTePress. doi: 10.5220/0011314500003266.

[6] Brynjulfsen, S.E. (2023). *Improving the performance of a Conflict-Free Replicated Relational Database System*. Tromsø: Arctic University of Norway.

[7] David, I., & Syriani, E. (2023). Real-time collaborative multi-level modeling by conflict-free replicated data types. *Software and Systems Modeling*, 22, 1131-1150. doi: 10.1007/s10270-022-01054-5.

[8] Guidec, F., Mahéo, Y., & Noûs, C. (2021). Delta-state-based synchronization of CRDTs in opportunistic networks. In *2021 IEEE 46th conference on local computer networks (LCN)* (pp. 335-338). Edmond: IEEE. doi: 10.1109/LCN52139.2021.9524978.

[9] Kordiaka, Y., & Padko, O. (2021). A collaborative diagram editor. In *Thesis statements of the V international scientific and practical conference "Mechatronic systems: Innovation and engineering"* (pp. 230-231). Kyiv: Kyiv National University of Technology and Design.

[10] Laddad, Sh., Power, C., Milano, M., Cheung, A., Crooks, N., & Hellerstein, J.M. (2022a). Keep CALM and CRDT on. *Proceedings of the VLDB Endowment*, 16(4), 856-863. doi: 10.14778/3574245.3574268.

[11] Laddad, Sh., Power, C., Milano, M., Cheung, A., & Hellerstein, J.M. (2022b). Katara: Synthesizing CRDTs with verified lifting. *Proceedings of the ACM on Programming Languages*, 6(OOPSLA2), 1349-1377. doi: 10.1145/3563336.

[12] Litt, G., Lim, S., Kleppmann, M., & van Hardenberg, P. (2022). Peritext: A CRDT for collaborative rich text editing. *Proceedings of the ACM on Human-Computer Interaction*, 6(CSCW2), article number 531. doi: 10.1145/3555644.

[13] Lutsenko, A. (2020). *Experimental study of structures of conflict-free replicated data types in collaborative offline applications*. Kharkiv: Zhukovsky National Aerospace University "Kharkiv Aviation Institute".

[14] Mao, Yu., Liu, Z., & Jacobsen, H.A. (2022). Reversible conflict-free replicated data types. In *Middleware '22: Proceedings of the 23rd ACM/IFIP international middleware conference* (pp. 295-307). New York: Association for Computing Machinery. doi: 10.1145/3528535.3565252.

[15] Nicolas, M., Oster, G., & Perrin, O. (2022). Efficient renaming in sequence CRDTs. *IEEE Transactions on Parallel and Distributed Systems*, 33(12), 3870-3885. doi: 10.1109/TPDS.2022.3172570.

[16] Ou, Yu., & Zhou, J. (2023). RDSL: An efficient retrieval algorithm for group editing CRDT. *Research Square*, 1, 1-18. doi: 10.21203/rs.3.rs-3316287/v1.

[17] Portela, B., Pacheco, H., Jorge, P., & Pontes, R. (2023). General-purpose secure conflict-free replicated data types. In *2023 IEEE 36th computer security foundations symposium (CSF)* (pp. 521-536). Dubrovnik: IEEE Computer Society. doi: 10.1109/CSF57540.2023.00030.

[18] Rostad, S. (2020). *Towards improved support for conflict-free replicated data types*. Tromsø: UiT The Arctic University of Norway.

[19] Ryabushkin, V. (2020). *Automatic conflict resolution in the text*. Kyiv: National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute".

[20] Saquib, N., Krintz, C., & Wolski, R. (2021). Log-structured conflict-free replicated data types: UCSB Technical Report 2021-01. *Santa Barbara: UCSB Computer Science*.

[21] Saquib, N., Krintz, C., & Wolski, R. (2022). Log-based CRDT for edge applications. In *2022 IEEE international conference on cloud engineering (IC2E)* (pp. 126-137). Pacific Grove: IEEE. doi: 10.1109/IC2E55432.2022.00021.

[22] Shynkarov, V. (2023). *Optimisation methods*. Kyiv: National Aviation University.

[23] Tranquillini, A. (2022). *Handling of mobile applications state using Conflict-Free Replicated Data Types*. Stockholm: KTH Royal Institute of Technology.

[24] Zakhour, G., Weisenburger, P., & Salvaneschi, G. (2023). Type-checking CRDT convergence. *Proceedings of the ACM on Programming Languages*, 7(PLDI), 1365-1388. doi: 10.1145/3591276.

[25] Zhang, Y., Huang, Y., Wei, H., & Ma, X. (2023). Model-checking-driven explorative testing of CRDT designs and implementations. *Journal of Software: Evolution and Process*, 36(4), article number e2555. doi: 10.1002/smr.2555.

Методи оптимізації продуктивності Conflict-free Replicated Data Types (CRDT)

Юрій Рабешко
Аспірант
Національний університет водного господарства та природокористування
33000, вул. Соборна, 11, м. Рівне, Україна
https://orcid.org/0009-0002-0763-7138

Юрій Турбал
Доктор технічних наук, професор
Національний університет водного господарства та природокористування
33000, вул. Соборна, 11, м. Рівне, Україна
https://orcid.org/0000-0002-5727-5334

Анотація. Актуальність досліджуваної проблеми полягає у необхідності оптимізації роботи з розподіленими даними, що не викликають конфліктів при реплікації, оскільки введення таких типів даних призводить до збільшення потреби у їхній оптимізації та підвищенні продуктивності. Метою дослідження є розробка способів підвищення ефективності різних типів даних з реплікацією без конфліктів. Для досягнення мети були використані методи аналізу та експерименту. Результати дослідження демонструють, що застосування певних оптимізованих способів безконфліктних типів даних у реплікаціях призводить до значного покращення ефективності та зниження ресурсних витрат. За результатами було визначено, що методи кешування і попереднього обчислення, асинхроної синхронізації та локальної обробки є найефективнішими для покращення ефективності Conflict-free Replicated Data Types. Встановлено, що вибір конкретного методу для покращення продуктивності Conflict-free Replicated Data Types має бути обгрунтованим і здійснюватися на основі уважного аналізу вимог та характеристик конкретної розподіленої системи. Це враховує потреби користувачів, ступінь критичності даних, частоту редагування тощо. Дослідження включає розроблення та оптимізацію алгоритмів синхронізації, що значно підвищує продуктивність таких типів даних у розподілених системах. У роботі є прості програмні реалізації для детального аналізу та вивчення методів оптимізації продуктивності типів даних у реплікаціях без конфліктів. Впровадження розподіленого обчислення дозволило оптимізувати процеси реплікації та синхронізації даних, що сприяє зниженню накладних витрат та підвищенню швидкодії систем. Зроблено висновок про практичну важливість дослідження, оскільки використання оптимізованих безконфліктних типів даних з реплікацією у реальних розподілених системах може покращити їхню ефективність та надійність. Практичне значення дослідження полягає в можливості застосування оптимізованих методів типів даних з безконфліктною реплікацією у реальних розподілених системах, що дозволить підвищити їхню продуктивність та забезпечити ефективну роботу в умовах великої навантаженості та обмеженого ресурсного потенціалу

Ключові слова: способи збільшення результативності; реплікація даних; розподілені системи; ефективність синхронізації; методи синхронізації даних