ТЕХНИЧЕСКИЕ НАУКИ

УДК 519.725

Имитационная модель распределенного отказоустойчивого хранилища данных на основе системы остаточных классов

А. С. Михайлишин, Н. С. Могилевская
Институт механики, математики и компьютерных наук им. И.И. Воровича Южного федерального университета, г. Ростов-на-Дону, Российская Федерация

Аннотация. Пороговые методы разделения данных используются для построения отказоустойчивых хранилищ данных. Цель работы состоит в исследовании свойств порогового метода разделения данных, основанного на избыточной системе остаточных классов. Отличительная особенность метода состоит в возможности контроля целостности долей, локализации и исправлении ошибок в долях. Для исследования метода построено программное средство, реализующее имитационную модель отказоустойчивого распределенного хранилища данных. В процессе создания программного средства решен ряд специфических задач. Одна часть из этих задач связана с ускорением работы программного средства, другая — с повышением эффективности использования памяти. В работе приведены примеры разделения и восстановления данных. Сделаны выводы о связи числа ошибок, повредивших доли, и вероятности корректного восстановления данных. Обсуждаются возможные способы улучшения корректирующей способности метода. Полученные результаты могут быть полезны исследователям и проектировщикам систем распределенного хранения данных.

Ключевые слова: система распределенного хранения, избыточная система остаточных классов, отказоустойчивые хранилища данных

Model of a Distributed Fault-Tolerant Data Storage Based on a System of Residual Classes

Andrei S. Mikhailishin, Nadezhda S. Mogilevskaya
Institute of Mechanics, Mathematics and Computer Sciences named after I.I. Vorovich of the Southern Federal University, Rostov-on-Don, Russian Federation

Abstract. Threshold data separation methods are used to build fault-tolerant data storages. The aim of the work was to study the properties of a threshold method for data separation based on a redundant system of residual classes. A distinctive feature of the method is the ability to control the integrity of the shares, localize and correct errors in the shares. To study the method, a software tool was built. It implemented a simulation model of a fault-tolerant distributed data storage. In the process of creating the software, several specific problems were solved. One part of these tasks was related to speeding up the operation of the software, the other was related to increasing the efficiency of memory use. The work provides examples of data separation and recovery. Conclusions are drawn about the relationship between the number of errors that damaged shares and the likelihood of correct data recovery. Possible ways to improve the corrective ability of the method are discussed. The results obtained can be useful to researchers and designers of distributed data storage systems.

Keywords: distributed storage system, redundant residual class system, fault-tolerant data stores

Введение. Одним из способов обеспечения надежности, отказоустойчивости и доступности данных является использование методов порогового разделения данных. Такие методы позволяют создавать новые модели хранения и передачи данных [1–3]. Основная идея, лежащая в основе таких систем, состоит в том, что исходные данные разделяются специальным образом на несколько частей. Затем эти части могут храниться или передаваться по отдельности. А сборка исходных данных возможна даже при потере или повреждении некоторых из частей.

В данной работе рассматривается метод разделения данных на основе избыточной системы остаточных классов [1]. Важнейшая его особенность состоит в возможности контроля целостности долей, локализации и исправления ошибок в долях. Цель работы состоит в создании программного средства, реализующего модель

распределенной отказоустойчивой системы хранения данных на основе избыточной системы остаточных классов и предназначенного для проведения экспериментального исследования метода разделения данных.

Для достижения цели необходимо было построить программное средство (имитационную модель на языке программирования C++), реализующее метод разделения данных, и провести эксперименты с реализацией метода.

Метод ИСОК

Представление числа A в системе остаточных классов. Рассмотрим систему остаточных классов (СОК) [1], которая позволяет представлять произвольное число A в виде специальной последовательности чисел, каждое из которых значительно меньше A. Для этого задается набор рабочих оснований, т.е. последовательность взаимно простых чисел $p_i,\ i = 1,\dots,k$, а число A представляется в виде следующего набора:

$$A = (a_1, a_2, \dots, a_k),\ a_i = A\ mod\ p_i,\ \text{где } i = 1,\dots,k.$$

Согласно китайской теореме об остатках [4] (КТО), такое представление $A \in [0, P_k)$ уникально лишь в том случае, если все p_i взаимно простые. Назовем рабочим диапазоном представления чисел в СОК величину:

$$P_k = \prod_{i=1}^{k} p_i.$$

Добавив к системе рабочих оснований $\{p_1, p_2, \dots, p_k\}$ избыточные основания $p_{k+1}, \dots, p_n$ и расширив представление A значениями $a_{k+1}, \dots, a_n$:

$$A = (a_1, a_2, \dots, a_k, a_{k+1}, \dots, a_n), \quad (1)$$

где $a_j = A\ mod\ p_j,\ j = 1,\dots,n$, получаем избыточную (k,n)-систему остаточных классов (ИСОК). ИСОК обладает свойством, полезным для для создания отказоустойчивой системы хранения данных, а именно, потеря или искажение любых $n-k$ остатков не лишает возможности восстановить исходное число A.

Восстановление числа A из неискаженного набора остатков. Для восстановления числа $\boldsymbol{A}$ из произвольных k остатков из набора (1) используем метод ортогональных базисов [1]:

$$\boldsymbol{A} = \left|\sum_{i=1}^{n} \boldsymbol{a_i} \cdot \boldsymbol{B_i^{(k)}}\right|_{\boldsymbol{P_k}},\ \textbf{где } \boldsymbol{B_i^{(k)}} = \frac{\boldsymbol{p_i}}{\boldsymbol{P_k}}\left(\frac{\boldsymbol{p_i}}{\boldsymbol{P_k}}\ \boldsymbol{mod}\ \boldsymbol{p_i}\right). \quad (2)$$

Заметим, что ортогональные базисы $B_i^{(k)}$ зависят только от системы оснований и не зависят от числа A. Следовательно они могут быть вычислены заранее.

Восстановление числа A из набора остатков с искажениями. Если набор (1) во время его хранения и передачи по линиям связи был поврежден, т.е. получен набор:

$$\boldsymbol{A'} = (\boldsymbol{a'_1},\ \boldsymbol{a'_2},\ \dots,\ \boldsymbol{a'_n}), \quad (3)$$

то можно попытаться восстановить число A с использованием метода проекций с максимальным правдоподобием [5]. Пусть дана ИСОК с рабочими основаниями $\{p_1,\ p_2,\ \dots,\ p_k\}$, избыточными основаниями $\{p_{k+1},\ p_{k+2},\ \dots,\ p_n\}$, рабочим диапазоном $P_k = p_1 \cdot\ p_2 \cdot \dots \cdot p_k$ и полным диапазоном ИСОК :

$$P_n\ = \prod_{i=1}^{n} p_i.$$

Максимальное количество ошибок, при котором возможно корректное восстановление A, не превышает половины количества контрольных значений [1]

$$q\ = \left\lfloor \frac{n-k}{2} \right\rfloor.$$

Для восстановления A достаточно использовать произвольные k остатков из имеющихся n остатков разложения (3). Далее наборы из k остатков будем называть проекциями. Заметим, что произвольный остаток a_i может быть включен в несколько проекций. Если в проекции, используемой для восстановления A, использованы неискаженные остатки, то результат применения к проекции формулы (2) должен лежать в диапазоне $[0, P_k)$, иначе, если среди используемых k остатков есть искаженные, то результат вычисления (2) будет лежать вне диапазона $[0, P_k)$.

Для восстановления A выполним следующие действия для каждой проекции. По проекции $a_{g(1)}, a_{g(2)}, \dots, a_{g(k)}$, полученной по основаниям $p_{g(1)}, p_{g(2)}, \dots, p_{g(k)}$, вычислим:

$$A^{\Delta}\ = \left|\sum_{i=1}^{k} a'_{g(i)} \cdot B_i^{\Delta}\right|_{P_{\Delta}}, \quad (4)$$

где

$$B^{\Delta} = \{B_1^{\Delta}, \dots, B_k^{\Delta}\},\ B_i^{\Delta} = \frac{p_i}{P_{\Delta}}\left(\frac{p_i}{P_{\Delta}} mod\ p_i\right),\ P_{\Delta}\ = \prod_{i=1}^{k} p_{g(i)}. \quad (5)$$

Если $A^{\Delta} \notin [0, P_k)$, то текущая проекция отбрасывается. Если $A^{\Delta} \in [0, P_k)$, то предполагаем, что искомое A совпадает с найденным значением A^{Δ}. Далее проверяем целостность всех имеющихся остатков, которыми записано число A. Для этого A^{Δ} преобразуется в набор остатков $(a_1^{\Delta},\ a_2^{\Delta},\ \dots,\ a_n^{\Delta})$ и вычисляется расстояние Хемминга d между векторами $(a_1^{\Delta},\ a_2^{\Delta},\ \dots,\ a_n^{\Delta})$ и $(a'_1,\ a'_2,\ \dots,\ a'_n)$. Если $d \le q$ тогда A^{Δ} объявляется исправленным числом, иначе текущая проекция отбрасывается.

Программная реализация

Методы разделения и восстановления данных на основе метода избыточных классов программно реализованы. Программное средство было создано на языке C++ в компиляторе Clion. Для работы с длинными числами использовали открытую библиотеку LongInt [6]. В процессе создания программного средства был решен ряд специфических задач. Часть из этих задач была связана с ускорением работы программного средства, другая — с повышением эффективности использования памяти. Рассмотрим некоторые разработанные алгоритмы.

Подбор оснований для представления числа A в ИСОК. Пусть A имеет разрядность b бит, тогда для корректного представления в СОК рабочий диапазон должен удовлетворять условию:

$$P_k \geq 2^b. \quad (6)$$

Очевидно, что чем меньше разрядность каждого основания, тем меньше требуется аппаратных затрат для реализации операций по данному основанию. Таким образом при подборе элементов набора оснований необходимо выбирать их взаимно простыми, согласно требованию алгоритма, и минимальной возможной разрядности с учетом (6). Дополнительно сократить аппаратные затраты на выполнение операций можно за счёт использования одного из оснований равного степени двойки, т.е. 2^l, так как для нахождения остатка по данному модулю достаточно взять l младших разрядов исходного числа. Построен алгоритм, который может выдавать наборы оснований двух видов: набор без числа, являющимся степенью двойки, и с таким числом. В первом случае сначала по формуле $(2^{\lceil b/k \rceil}) - 1$ вычисляется первое значение из набора оснований, которое помещается в середину набора рабочих оснований. Затем заполняются все остальные значения рабочих оснований по принципу: выбирается первое взаимно простое для всех уже выбранных оснований слева в $p^1_{\lceil k/2 \rceil - i}$, первое взаимно простое для всех уже выбранных оснований справа в $p^1_{\lceil k/2 \rceil + i}$, i = 1, …, $\lfloor k/2 \rfloor$). Затем все выбранные основания смещаются в начало массива, и справа подбираются избыточные основания. При подборе избыточных оснований отыскиваются минимально возможные значения взаимно простые с уже построенными. На выходе алгоритма получаем первый набор оснований $\{p_i^1\}_{i=1}^n$. Заметим, что по случайности в нем может оказаться число, являющееся степенью двойки. Для второго набора оснований $\{p_i^2\}_{i=1}^n$, где степень двойки является обязательной, сначала вычисляется $b(p_n^1)$ — количество бит, необходимых для записи самого большой основания p_n^1 из первого набора. Затем условному среднему рабочих оснований присваивается значение $2^{b(p_n^1)-1}$ и повторяются те же шаги, что и при нахождении первого набора оснований.

Для выбора между $\{p_i^1\}_{i=1}^n$ и $\{p_i^2\}_{i=1}^n$ используем следующее соображение: если разрядности наибольших элементов наборов совпадают $b(p_n^1) = b(p_n^2)$, тогда используем диапазон со степенью двойки $\{p_n^2\}_{i=1}^n$, иначе: используем набор $\{p_n^1\}_{i=1}^n$.

Преобразование числа A в ИСОК. Разбиение $\boldsymbol{A}$ на остатки по формуле (1) с помощью обычной операции деления является довольно неэффективным из-за сложности операции. Более приемлемым является метод непосредственного суммирования с использованием табличной арифметики для числа $\boldsymbol{A}$ [1]. Пусть число $\boldsymbol{A}$ записано в позиционной системе счисления с основанием $\boldsymbol{N}$, то есть:

$$A = A_0 \cdot N^0 + \ldots + A_m \cdot N^m = \sum_{i=0}^{m} A_i N^i, \text{ где } 0 \leq A_i \leq (N-1), i = 0, \ldots, m.$$

Представим степени основания N^i и коэффициенты A_i в системе остаточных классов с основаниями $\{p_1, p_2, \ldots, p_n\}$:

$$N^i = \left\{N_1^{(i)}, \ldots, N_n^{(i)}\right\}, A_i = \left\{A_1^{(i)}, \ldots, A_n^{(i)}\right\}$$

получим:

$$A = \left(\sum_{i=0}^{m} A_i^{(1)} N_i^{(1)} \bmod p_1, \ldots, \sum_{i=0}^{m} A_i^{(n)} N_i^{(n)} \bmod p_n\right) = (a_1, \ldots, a_n). \quad (7)$$

Пример. Рассмотрим СОК с основаниями:

$$\{p_1, p_2, \ldots, p_6\} = \{4, 5, 7, 9, 11, 13\}. \quad (8)$$

Десятичное число $A = 1446$ запишем в виде:

$$1446 = 1 \cdot 10^3 + 4 \cdot 10^2 + 4 \cdot 10^1 + 6 \cdot 10^0.$$

Теперь используем формулу (7) для представления числа A. Подготовим вспомогательные таблицы. В таблице 1 представлены остатки степеней $10^i, i = 0, \ldots, 4$, по модулям (8), а в таблице 2 представлены остатки всех возможных коэффициентов A^i по модулям (8).

Таблица 1

Остатки степеней $10^i, i = 0..4$ по модулям СОК

	10^0	10^1	10^2	10^3
$p_1 = 4$	1	2	0	0
$p_2 = 5$	1	0	0	0
$p_3 = 7$	1	3	2	6
$p_4 = 9$	1	1	1	1
$p_5 = 11$	1	10	1	10
$p_6 = 13$	1	10	9	12

Таблица 2

Остатки коэффициентов A^i по модулям СОК

		1	2	3	4	5	6	7	8	9
$p_1 = 4$	0	1	2	3	0	1	2	3	0	1
$p_2 = 5$	0	1	2	3	4	0	1	2	3	4
$p_3 = 7$	0	1	2	3	4	5	6	0	1	2
$p_4 = 9$	0	1	2	3	4	5	6	7	8	0
$p_5 = 11$	0	1	2	3	4	5	6	7	8	9
$p_6 = 13$	0	1	2	3	4	5	6	7	8	9

Используя (7), получаем:

$$A \bmod 4 = (1 \cdot 0 + 0 \cdot 0 + 0 \cdot 2 + 2 \cdot 1) \bmod 4 = 2,$$
$$A \bmod 5 = (1 \cdot 0 + 4 \cdot 0 + 4 \cdot 0 + 1 \cdot 1) \bmod 5 = 1,$$
$$A \bmod 7 = (1 \cdot 6 + 4 \cdot 2 + 4 \cdot 3 + 6 \cdot 1) \bmod 7 = 4,$$
$$A \bmod 9 = (1 \cdot 1 + 4 \cdot 1 + 4 \cdot 1 + 6 \cdot 1) \bmod 9 = 6,$$
$$A \bmod 11 = (1 \cdot 10 + 4 \cdot 1 + 4 \cdot 10 + 6 \cdot 1) \bmod 11 = 5,$$
$$A \bmod 13 = (1 \cdot 12 + 4 \cdot 9 + 4 \cdot 10 + 6 \cdot 1) \bmod 13 = 3,$$

то есть

$$A = (2, 1, 4, 6, 5, 3).$$

Конец примера.

Отметим, что использование формулы (7) значительно сокращает затраты на вычисления, так как построение вспомогательных таблиц можно выполнить единожды и затем обращаться к ним по необходимости.

Локализация искажений. Локализация искажений производится с помощью функции выбора оснований для проекций, предложенной в материалах [1]. Для данной (k, n) избыточной СОК, где $k < \lfloor (n-k)/2 \rfloor$, проекции строятся следующим образом:

$$A = a_1, \dots, a_k, a_{k+1}, \dots, a_{2k}, \dots, a_{\lfloor \frac{n}{k} \rfloor k-k+1}, a_{\lfloor \frac{n}{k} \rfloor k-k+2}, \dots, a_{\lfloor \frac{n}{k} \rfloor k}, \dots, a_n,$$
$$\text{СОК}_1 = a_1, \dots, a_k,$$
$$\text{СОК}_2 = a_{k+1}, \dots, a_{2k},$$
$$\dots$$

$$\text{СОК}_{\lfloor \frac{n}{k} \rfloor} = a_{\lfloor \frac{n}{k} \rfloor k-k+1}, a_{\lfloor \frac{n}{k} \rfloor k-k+2}, \dots, a_{\lfloor \frac{n}{k} \rfloor k}.$$

При таком построении проекций если n не кратно k, то основания $a_{\lfloor \frac{n}{k} \rfloor k+1}, \dots a_n$ не используются. С ростом n и k вероятность восстановить число при фиксированной вероятности ошибок уменьшается, так как растет вероятность появления ошибки в каждой проекции. Для примера оценим вероятность нахождения таким методом хотя бы одной неискаженной проекции при разном количестве ошибок. Для проведения такой оценки подготовлен скрипт для программы Maple, который моделировал n произвольных числовых значений, в которых расставлял заданное количество ошибок. Искажения моделировались независимыми равномерно распределенными. Для каждой пары n и вносимого числа ошибок выполнено 10 000 экспериментов. На рис. 1 для параметров $k = 4,\ n = 24$ представлен график зависимости вероятности появления хотя бы одной неискаженной проекции от количества ошибок в n элементах. Из анализа графика можно сделать вывод, что рассматриваемый метод гарантирует корректное восстановление числа из набора остатков только если число ошибок, повредивших используемый набор, меньше числа проекций (на рисунке такие случаи выделены зеленым фоном). Вертикальная фиолетовая линия на рисунке соответствует числу ошибок, которые метод может теоретически исправить (в рассматриваемом случае — это 10 ошибок). Из рисунка видно, что 10 ошибок исправляются методом только в половине случаев. Розовая зона рисунка соответствует числу ошибок, исправление которых методом не гарантируется.

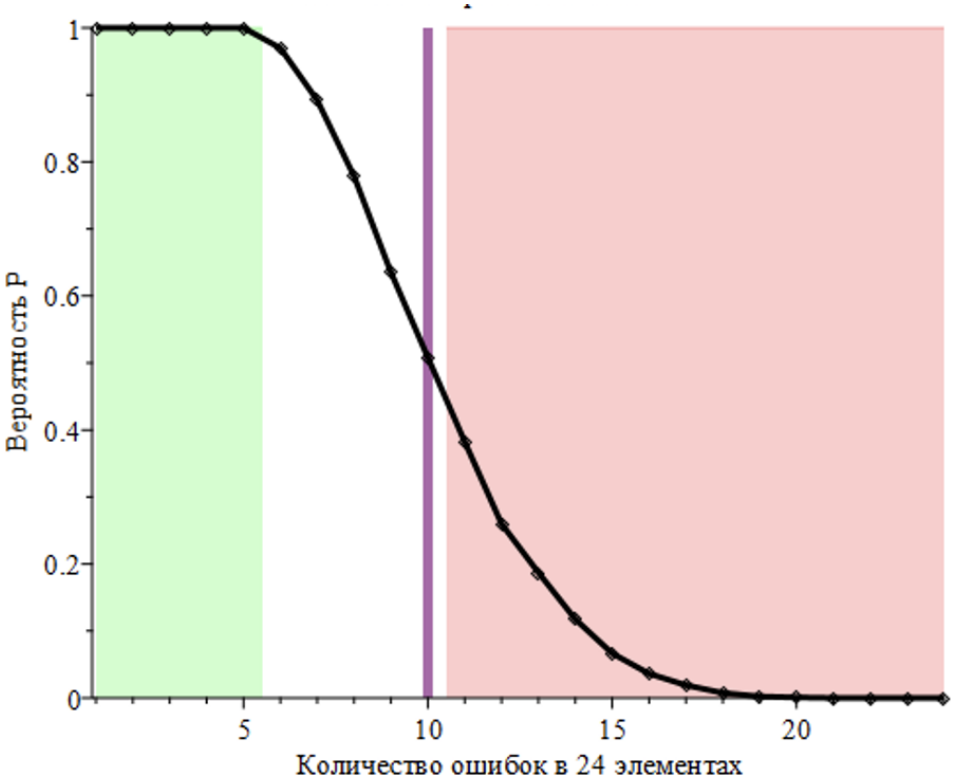

Рис. 1. Вероятности нахождения хотя бы одной неискаженной проекции

Для ускорения работы программы использован описанный выше метод проекций, но очевидно, что процедуру исправления ошибок можно улучшить за счет использования большего количества проекций.

Экспериментальные исследования метода

Рассмотрим два примера работы программы, демонстрирующие возможности метода разделения данных на основе остаточных классов.

Пример 1. Пусть необходимо представлять в ИСОК числа из диапазона [0, 15]. Зафиксируем параметры метода $b = 4,\ k = 2,\ n = 6$ и представим $A = 6$ в виде системы остатков. Для представления произвольного числа из диапазона [0, 15] в двоичном виде необходимо 4 бита. Согласно (6) условию метода, рабочий диапазон должен быть не меньше, чем 2^4. Подбирая возможные наборы оснований, методом, описанным выше, получаем два одинаковых набора:

$$P = P1 = P2 = [4, 5, 7, 9, 11, 13].$$

Разложение числа A на ИСОК: $[2, 1, 6, 6, 6, 6]$. Внесем ошибки в части разделенного на остатки числа A, используя генератор ошибок. Результат работы этой части программы приведен на рис. 2.

```
================================================
Errors: 2 # число вносимых ошибок
Error_Is_Here = [0, 0, 1, 1, 0, 0] # позиции ошибок
------------------------------------------------
Corrupt Result = [2, 1, 4, 4, 6, 6] # искаженное число A
================================================
```

Рис. 2. Внесение ошибок в разделенное число

Для восстановления числа вычислим ортогональные базисы (рис. 3) согласно (5). Для проверки искаженного числа A' используем ортогональные базисы по всему диапазону оснований `P` по формуле аналогичной (4):

$$A' = \left|\sum_{i=1}^{k} a_{g(i)}^{\cdot} \cdot B_i^{\Delta}\right|_{P_\Delta}.$$

```
================================================
B = [45045, 36036, 25740, 140140, 16380, 97020]
B[1…2] = [5, 16, 0, 0, 0, 0]
B[3…4] = [0, 0, 36, 28, 0, 0]
B[5…6] = [0, 0, 0, 0, 78, 66]
================================================
```

Рис. 3. Вычисление ортогональных базисов

В результате восстановления числа A' по набору всех ортогональных базисов (верхняя строка рис. 2) получаем число $A' = 28606$. Очевидно, что число восстановлено с ошибкой, т.к. $A' \notin [0, 15]$. Восстановим искомое число, используя различные проекции (рис. 4). На рисунке `CurrentA` обозначает число, восстановленное с использование данной проекции, `CorruptA` — это число A', d — расстояние Хэмминга между `CurrentA` и `CorruptA`. Далее, если d не превышает максимальное количество ошибок q, то остатки по текущим основаниям считаются верными (неискаженными).

```
============================================================
P=[4, 5]        A=6   d(CorruptA, CurrentA)= 2, (2<=2) = True
P=[7, 9]        A=4   d(CorruptA, CurrentA)= 4, (4<=2) = False
P=[11, 13]      A=6   d(CorruptA, CurrentA)= 2, (2<=2) = True
============================================================
```

Рис. 4. Вычисление исходного числа с использованием проекций

Из рис. 4 видно, что число A верно восстановлено по двум проекциям $P = [4,\ 5]$ и $P = [11,\ 13]$. Восстановление по проекции $P = [7, 9]$ дало ошибочный результат. Отметим, что именно этой проекции соответствуют ошибочные элементы числа, разложенного по ИСОК (рис. 2). В заключение работы программа выдает исходное число A и его верное разложение по ИСОК (рис. 5).

```
================================================
A исправлено успешно
A = 6 = [2, 1, 6, 6, 6, 6]
Ошибка в 3м фрагменте
Ошибка в 4м фрагменте
================================================
```

Рис. 5. Итоговое сообщение программы

Пример 2. Параметры метода $b = 80$, $k = 8, n = 16$, число$A = 12345678900098765432 1$ занимает 64 бита. Результат генерации набора оснований показан на рис. 6.

```
==================================================
P1 = [1021, 1023, 1024, 1025, 1027, 1031, 1033, 1037,
      1039, 1043, 1049, 1051, 1061, 1063, 1069, 1073]
P2 = [512, 2045, 2047, 2049, 2051, 2053, 2057, 2059,
      2063, 2069, 2071, 2077, 2081, 2083, 2087, 2089]
P = [1021, 1023, 1024, 1025, 1027, 1031, 1033, 1037,
      1039, 1043, 1049, 1051, 1061, 1063, 1069, 1073]
==================================================
```

Рис. 6. Результаты генерации набора оснований

Разложение числа A на ИСОК: [699, 720, 921, 696, 279, 631, 73, 875, 265, 591, 595, 741, 591, 1050, 803, 633]. Результат моделирования ошибок в частях A представлены на рис. 7.

```
==================================================
Errors: 1 # число вносимых ошибок
# позиции ошибок
Error_Is_Here = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
# искаженное число A
[699, 720, 921, 696, 279,  631,  73, 875,
 265, 591, 595, 741, 591, 1045, 803, 633]
==================================================
```

Рис. 7. Результаты работы генератора ошибок

Вычисление ортогональной системы для большого A дает длинные числа. Не будем приводить их полностью. Для примера:

$$B_1 = 771783758616969303035487461090858245515249408000.$$

Процедура восстановления числа дает результат:

$A' = 157117309053793313252629055909593994151432798 8121,$

Напомним, что $A = 12345678900098765721$. Результаты процедуры восстановления A и локализации ошибок показаны на рис. 8.

```
==================================================
P = [1021, 1023, 1024, 1025, 1027, 1031, 1033, 1037]
A = 12345678900098765721
d(CorruptA, CurrentA)= 1, (1 <= 4) = True
P = [1039, 1043, 1049, 1051, 1061, 1063, 1069, 1073]
A = 14860224657507460519979 63
d(CorruptA, CurrentA) = 16, (16 <= 4) = False
--------------------------------------------------
A восстановлено успешно
A = 12345678900098765721 =
[699, 720, 921, 696, 279,  631,  73, 875,
 265, 591, 595, 741, 591, 1050, 803, 633]
Ошибка в 14м фрагменте
d(CorruptA, CurrentA) = 1, (1 <= 4) = True, (q=4)
==================================================
```

Рис. 8. Результаты исправления ошибок в примере 2

Заключение. В работе создана имитационная модель распределенной отказоустойчивой системы хранения данных на основе системы избыточных остаточных классов. Модель реализована в виде программного средства, построенного с применением языка программирования C++. В ходе выполнения работы была обнаружена возможность улучшить корректирующую способность метода, изменяя способ выбора проекций.

К дальнейшим направлениям работы относится улучшение программы с точки зрения ускорения работы, а также сравнение данного метода разделения данных с другими аналогичными методами.

Список литературы

1. Назаров А.С. *Разработка методов и алгоритмов построения отказоустойчивых распределенных систем хранения данных на основе модулярной арифметики*: дис. … канд. техн. наук: 05.13.18. Северо-Кавказский федеральный университет, Ставрополь; 2019. 272 с.

2. Деундяк В.М., Могилевская Н.С. Схема разделенной передачи конфиденциальных данных на основе дифференцирования полиномов нескольких переменных над простыми полями Галуа. *Вопросы кибербезопасности*. 2017;5(24):64–71. URL: https://cyberleninka.ru/article/n/shema-razdelennoy-peredachi-konfidentsialnyh-dannyh-na-osnove-differentsirovaniya-polinomov-neskolkih-peremennyh-nad-prostymi-polyami (дата обращения: 17.10.2023).

3. Могилевская Н.С., Кульбикаян Р.В., Журавлёв Л.А. Пороговое разделение файлов на основе битовых масок: идея и возможное применение. Advanced Engineering Research (Rostov-On-Don). 2011;11(10):1749-1755. URL: https://cyberleninka.ru/article/n/porogovoe-razdelenie-faylov-na-osnove-bitovyh-masok-ideya-i-vozmozhnoe-primenenie (дата обращения: 17.10.2023).

4. Пилиди В.С. Математические основы защиты информации: учеб. пособие. Ростов-на-Дону – Таганрог. Изд-во ЮФУ, 2019. 308 с.

5. Акушский И.Я., Юдицкий Д.И. *Машинная арифметика в остаточных классах*. Москва: Советское радио; 1968. 440 с.

6. Работа с очень длинными числами на C++. URL: https://habr.com/ru/articles/172285/ (Дата обращения: 23.09.2023 г.)

Об авторах:

Михайлишин Андрей Сергеевич, студент кафедры алгебры и дискретной математики Института механики, математики и компьютерных наук им. Воровича Южного федерального университета (344003, РФ, г. Ростов-на-Дону, ул. Мильчакова 8А), mikhailishin@sfedu.ru

Могилевская Надежда Сергеевна, кандидат технических наук, доцент кафедры алгебры и дискретной математики Института механики, математики и компьютерных наук им. Воровича Южного федерального университета (344003, РФ, г. Ростов-на-Дону, ул. Мильчакова 8А), nmogilevskaya@sfedu.ru

About the Authors:

Andrei S. Mikhailishin, Student of the Algebra and Discrete Mathematics Department, Institute of Mechanics, Mathematics and Computer Science in the name of I.I. Vorovich of the Southern Federal University (8A Milchakov Str., Rostov-on-Don,344003, RF), mikhailishin@sfedu.ru

Nadezhda S. Mogilevskaya, Cand. Sci. (Eng.), Associate Professor of the Algebra and Discrete Mathematics Department, Institute of Mechanics, Mathematics and Computer Science in the name of I.I. Vorovich of the Southern Federal University (8A Milchakov Str., Rostov-on-Don,344003, RF), nmogilevskaya@sfedu.ru