МРНТИ 81.93.29
УДК 004

10.51889/2959-5894.2023.83.3.017

А.Ж. Карипжанова[1], Қ.Т. Қожахмет[1], Р.З. Жумалиева[1], А.М. Өмірәлі[1], Д.А. Баязитов[1]*

[1]НАО «Университет Нархоз», г. Алматы, Казахстан
[*]e-mail: k.kozhakhmet2023@gmail.com

ПОВЫШЕНИЕ БЕЗОПАСНОСТИ РАСЩЕПЛЕННЫХ ДАННЫХ ПРИ ИСПОЛЬЗОВАНИИ АЛГОРИТМОВ МНОГОМЕРНОЙ ЧЕТНОСТИ

Аннотация

Модель распределенного хранения данных с использованием многомерных кодов с частичной защитой от потерь данных рассматривается как альтернативный способ обеспечения безопасности, который может заменить общепринятый метод множественного резервного копирования. В этой модели избыточные данные генерируются, позволяя восстанавливать частичные потери разделенных частей. Восстановление производится с использованием кодовых файлов, формируемых во время процедуры разделения, основное действие заключается в расчете битовой четности с модулем-2 сложения. Сравнение с рядом моделей, использующих контрольные суммы для восстановления, показывает, что расчетная вероятность потерь в случае сбоев в хранении значительно ниже при использовании многомерных кодов с частичной защитой. Поврежденные файлы могут быть полностью восстановлены с помощью битового сложения неповрежденных файлов. Даже когда теоретически неподвластные восстановлению потери достигаются, всегда есть шанс найти комбинации цепочечного восстановления. Система хранения, использующая многомерные коды с частичной защитой, является более удобной технологией коррекции ошибок. Сложность альтернативных методов, в том числе метода итеративного декодирования, делает систему распределенного хранения расщепленных данных с применением алгоритмов многомерной четности более удобной технологией исправления ошибок.

Ключевые слова: безопасность, распределенное хранение, расщепление данных, многомерная четность, восстановление потерь, файлы четности.

Аңдатпа

А.Ж. Карипжанова[1], Қ.Т. Қожахмет[1], Р.З. Жумалиева[1], А.М. Өмірәлі[1], Д.А. Баязитов[1]
[1]«Нархоз Университеті» КеАҚ, Алматы қ. , Қазақстан

КӨПӨЛШЕМДІ ЕСЕП БЕРУ АЛГОРИТМДЕРІН ПАЙДАЛАНУ КЕЗІНДЕ ЫДЫРАТЫЛҒАН ДЕРЕКТЕРДІҢ ҚАУІПСІЗДІГІН АРТТЫРУ

Деректерді жоғалтудан ішінара қорғалған көпөлшемді кодтарды қолдана отырып, таратылған деректерді сақтау моделі жалпы қабылданған бірнеше сақтық көшірме әдісін алмастыра алатын қауіпсіздікті қамтамасыз етудің балама әдісі ретінде қарастырылады. Бұл модельде артық деректер пайда болады, бұл ыдыратылған бөліктердің ішінара жоғалуын қалпына келтіруге мүмкіндік береді. Қалпына келтіру бөлу процедурасы кезінде қалыптасқан кодтық файлдарды қолдану арқылы жүзеге асырылады, негізгі әрекет-2 қосу модулімен биттік паритетті есептеу. Қалпына келтіру үшін бақылау сомаларын пайдаланатын бірқатар модельдермен салыстыру жартылай қорғалған көп өлшемді кодтарды пайдаланған кезде сақтаудағы ақаулар болған жағдайда шығынның болжамды ықтималдығы айтарлықтай төмен екенін көрсетеді. Бүлінген файлдарды бүлінбеген файлдарды биттік қосу арқылы толығымен қалпына келтіруге болады. Теориялық тұрғыдан қалпына келтірілмеген шығындарға қол жеткізілсе де, тізбекті қалпына келтіру комбинацияларын табуға әрқашан мүмкіндік бар. Жартылай қорғалған көп өлшемді кодтарды қолданатын сақтау жүйесі қателерді түзетудің ыңғайлы технологиясы болып табылады. Баламалы әдістердің күрделілігі, соның ішінде қайталанатын декодтау әдісі, көп өлшемді Паритет алгоритмдерін қолдана отырып, ыдыратылған деректерді үлестірілген сақтау жүйесін қателерді түзетудің ыңғайлы технологиясына айналдырады.

Түйін сөздер: қауіпсіздік, таратылған сақтау, деректерді бөлу, көп өлшемді Паритет, шығындарды қалпына келтіру, паритеттік файлдар.

Abstract

ENHANCING SECURITY OF SHARED DATA USING MULTIDIMENSIONAL PARITY ALGORITHMS

Karipzhanova A.Zh.[1], Kozhakhmet K.T.[1], Zhumalieva R.Z.[1], Omirali A.M.[1], Bayazitov D.A.[1]
[1]NJSC «Narxoz University», Almaty, Kazakhstan

Distributed data storage using multidimensional codes can replace the traditional method of multiple backups to ensure security. Redundant data allows for partial recovery of separated parts, with recovery using code files generated

during separation and calculating parity bits with modulo-2 addition. Compared to checksum models, multidimensional codes with partial protection significantly reduce the probability of data loss due to storage failures. Damaged files can be fully restored using bitwise addition of undamaged files. Even theoretically unrecoverable losses can sometimes be recovered through chain recovery. Compared to iterative decoding and other alternatives, the utilization of multidimensional codes with partial protection in the storage system presents a more convenient fault tolerance technology. These multidimensional error-checking codes provide a simplified and user-friendly approach to error correction within the distributed storage system for partitioned data. By employing this method, the complexity associated with alternative techniques like iterative decoding is significantly reduced, further enhancing the convenience of error correction. It offers a more efficient and reliable means of protecting data against storage failures.

Keywords: security, distributed storage, data splitting, multidimensional parity, loss recovery, parity files.

1. Введение

Поиск новых методов обеспечения IT безопасности идет непрерывно, но пока для сохранения данных повсеместно используется репликация (многократное резервирование). Постоянный рост объема информации приводит к тому, что репликация оборачивается ростом аппаратных, энергетических и, в итоге, финансовых затрат. Избежать дорогостоящего резервного хранения можно попытаться путем операционного восстановления информации (исправление ошибок). В частности, авторами настоящей статьи апробирована возможность хранения файлов в распределенных базах данных (РБД) в расщепленном состоянии, когда расщепление файлов и их восстановление осуществляется с помощью патентованных многомерных кодов с частичной защитой от потерь данных, способные обрабатывать частичные потери мест хранения [1-3] (патенты: GB2467989 «Distributed Storage», UK; GB2463078 «Distributed Storage», UK; GB2463085 «Communication System», UK; GB2463087 «Data Storage», UK; GB2492981 «Data Storage», UK; HO1175001 «Data Storage», Hong Kong; US9026844 «Distributed Data Storage and Communication», USA).

Интерфейс фреймворка основан на веб-технологиях и использует простое соглашение HTTP (Hypertext Transfer Protocol), связанное с интуитивно понятными инновациями в AJAX /PHP (WEB 2.0). CMS Word Press используется для реализации фундаментальной системы управления контентом веб-части. Веб-сайт размещен компанией PS Internet Company LLP на выделенном сервере под управлением веб-сервера Apache 2.25 с PHP 7.0 и серверной операционной системы Linux Ubuntu. В качестве базы данных используется MySQL 5.0 [4].

За счет патентованных алгоритмов генерируются избыточные данные, позволяющие восстанавливать частичные утери расщепленных частей. Расщепленные данные не несут осмысленного содержания, и поэтому их можно хранить на любых доступных местах, не опасаясь несанкционированного доступа. Даже собрав все части расщепленных данных, без знания способа расщепления невозможно восстановить информацию. Алгоритм расщепления может использовать бесконечное множество способов расщепления, и поэтому доступ к данным имеет только владелец/создатель информации. При этом в случае частичной потери данных их восстановление производится за счет файлов четности, образующихся при процедуре расщепления, в которой основным действием является вычисление побитовой четности с суммированием по модулю «два». С помощью такой операции сложения между исходными файлами можно восстановить без искажений поврежденный (утерянный) файл.

2. Методы

Алгоритмы четности оперируют бинарной информацией. И суммирование, которое используется нами при расщеплении данных, сводится к определению четности. Причем суммирование ведется строго позиционно («побитово») и по модулю. Получается третий файл, в котором сохраняются просуммированные биты. Суммирование производится по модулю «два», а именно: включается математическая операция «исключающее ИЛИ». Если, например, в первом файле ноль и во втором – ноль, то в третьем будет ноль. Если в первом – ноль, а во втором – единица, то в третьем будет единица. Если в первом единица, во втором ноль, то в третьем тоже будет единица. Если и там, и здесь единица, то в третьем файле получается ноль. Этот третий файл и является файлом четности, а вся операция есть операция четности: из двух файлов получается три. Если удалить любой из этих файлов, то, выполнив операцию сложения между оставшимися двумя, всегда можно восстановить без искажений третий файл. Эта идея, базирующаяся на свойствах четности, и лежит в основе решения задач распределенного хранения информации с расщеплением данных с применением алгоритмов многомерной четности. До сих пор свойство четности использовалось только для того, чтобы обнаружить повреждение файла и

определить, какая информация утеряна, не более того (Redundant Arrays of Inexpensive Disks - далее RAID). В таких системах информация делится на одинаковые блоки, затем вычисляется четность, которая записывается в отдельный файл, содержащий биты четности. И когда, например, при передаче данных бит четности теряется, при сложении данных бит четности не совпадет. Однако этот результат в RAID нужен только для того, чтобы удалить файл, если обнаружится его повреждение. В случае потери при передаче данных делается повторный запрос. Затем утерянный файл восстанавливается из мест резервного хранения (метод репликаций).

Между тем в апробированном нами методе многомерной четности утерянная информация восстанавливается с расчетной вероятностью. Как сказано выше, если поделить файл (носитель данных) на две одинаковые части, вычислить побитовую четность и записать результат в третий файл, то всегда можно восстановить потерю (или повреждение, если известно, какой файл поврежден) любого из трех файлов.

Рассмотрим, для примера, случай трех файлов, когда один из них теряется. Побитово складываем по модулю два оставшиеся файла. Полученный файл четности в точности равен потерянному файлу. Это случай одномерной четности (рис. 1).

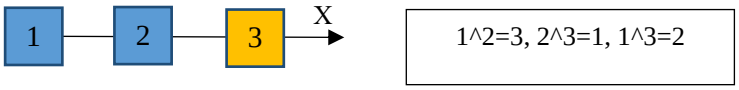

Рисунок 1. Вариант одномерной четности

Если теперь поделить файл на четыре одинаковые части, расположить их в виде квадрата и вычислить побитовую четность по строкам и по столбцам, то получим 4+4+1=9 файлов. Это случай двумерной четности (рис. 2).

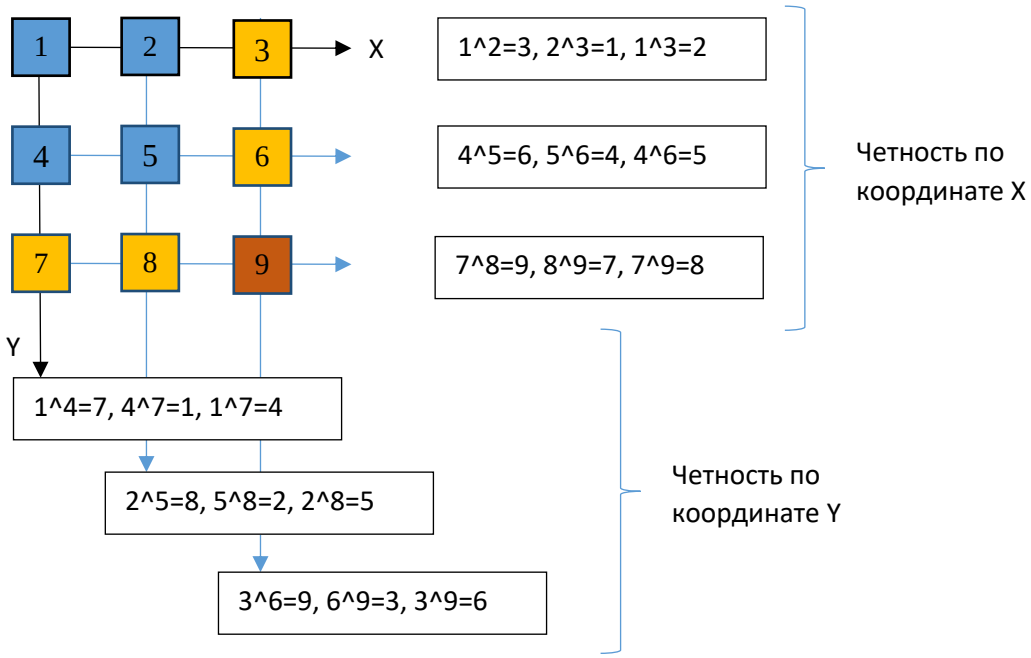

Рисунок 2. Вариант двумерной четности

В случае двумерной четности, очевидно, что мы можем потерять любые три файла из девяти, что равносильно потере файлов по одной из координат, но затем восстановить всю исходную информацию. Чтобы показать это, рассмотрим для простоты случай записи по одному биту (информация хранится на 9 дисках).

Результат суммирования приведен в таблице 1.

Таблица 1. Результат суммирования по четности для 9 дисков

№	*Бит*	*№*	*Бит*	*№*	***Бит***
1-диск	*1*	*2-ой диск*	*0*	***7-ой диск, четность по 1-ой строке***	***1***
3-ий диск	*0*	*4-ый диск*	*1*	***8-ой диск, четность по 2-ой строке***	***1***
5-ый диск, четность по 1-ой колонке	***1***	***6-ый диск, четность по 2-ой колонке***	***1***	***9-ый диск, четность четности***	***0***

В такой системе с двумерной четностью могут выйти из строя 3 диска из 9 одновременно, и данные всегда можно восстановить, т.к. восстанавливать биты можно по двум координатам. А это уже равносильно 3-х-кратному копированию при репликации. Более того, можно потерять даже 4 диска из 9, и все еще будет возможность восстановить информацию с вероятностью 92%. Можно потерять даже до 5 дисков и восстановить с вероятностью 67% [5].

В системе с двумерной четностью, конечно, в реальности необходимо больше дисков и большое количество данных, потому что в этом варианте должно быть больше данных четности. Но, в двумерной четности избыточность равна 2,25 (данные записываются на 4 диска и дополнительно еще на 5 дисках: четность 9/4=2,25).

Если сравнить с репликацией, то избыточность двумерной четности получается почти как при двукратной репликации с избыточностью 2,0, когда записываются две копии. А потерять в системе с двумерной четностью можно от 3 до 5 дисков, т.е. надежность – как при 3-5-кратной репликации.

3. Связанные работы

На сегодняшний день стандартным способом обеспечения безопасности является метод многократных репликаций.

Кроме того, были изучены различные подходы к повышению информационной безопасности. К ним относятся использование алгоритма Рида-Соломона [6], использование технологий fountain [7] и включение контрольных сумм в систему RAID5/6 и кодов, обеспечивающих локальную четность LRC [8]. Также были достигнуты успехи в области кодов с исправлением ошибок, в частности в теории каналов со стираниями, первоначально введенной В. Петерсоном [9]. Впоследствии М. Лаби расширил эту модель каналов со стираниями в своих исследованиях [10].

3.1. Метод репликаций

Алгоритмы проверки четности, которые являются одновременно простыми и быстродействующими, приобрели значительную популярность в алгоритмах исправления ошибок. Они особенно широко используются, с разной эффективностью, в широко используемых RAID-системах, реализованных в отказоустойчивых дисковых массивах для серверов. Однако следует отметить, что, хотя эти высокоскоростные и простые алгоритмы проверки четности подходят для обычных приложений, они не обеспечивают возможности восстановления после множественных потерь данных. Например, распространенный вариант RAID5 при отказе одного диска переходит в состояние восстановление, в котором начинается интенсивное чтение со всех дисков массива для восстановления данных отказавшего диска. Если при размерах дисков порядка нескольких гигабайт это не вызывает проблем, то с увеличением объемов дисков до терабайтов процесс может занять значительное время, в течение которого любой отказ еще одного диска приводит к полной потере всех данных.

Проблемы с надежностью стандартных RAID-массивов заставляют использовать более сложные конфигурации RAID с комбинированием различных схем, такие как RAID6, применять дополнительное зеркалирование. Известны коммерческие проприетарные варианты RAID такие как HP EVA с собственной технологией vRAID, RAID m + n с использованием стирающих кодов Рида-Соломона. Но все это приводит к еще большему удорожанию технологии.

Поэтому часто в RAID-массивах выделяют специально один диск для записи четности, с помощью которой можно восстановить данные со сломанного диска.

Чтобы пояснить смысл восстановления по четности, рассмотрим, например, случай RAID из 5 дисков. Пусть для простоты записываются четыре байта на четыре диска: 1, 2, 3 и 4. Результат суммирования по четности записывается на 5-й диск (таблица 2).

Таблица 2. Результат суммирования по четности в системе RAID

№	*Бит*							
1 диск	*1*	*1*	*1*	*0*	*1*	*0*	*1*	*0*
2 диск	*0*	*0*	*1*	*0*	*1*	*1*	*1*	*0*
3 диск	*0*	*1*	*0*	*1*	*1*	*1*	*0*	*1*
4 диск	*1*	*1*	*1*	*0*	*0*	*1*	*0*	*1*
5 диск – четность	***0***	***1***	***1***	***1***	***1***	***1***	***0***	***0***

Суммирование по четности производится вычислением «исключающего ИЛИ» битов между разными дисками. Его еще называют «сложением по модулю 2». При этом 0^0=0, 0^1=1, 1^0=1, 1^1=0. Метод называется суммированием по четности, потому что в результате всегда должно быть четное количество битов вместе с вычисленным.

Из таблицы видно, что если складывается четное количество единиц, то расчетный бит четности должен быть равен нулю: четность не меняется. Например, в первой колонке битов: 1 диск = 1; 2 диск = 0; 3 диск = 0; 4 диск = 1. Две единицы, на 5-й диск записывается 0. Если количество битов нечетное, то бит четности будет 1, превращая таким образом общее количество единиц в четное число. Например, во второй колонке: 1 диск = 1; 2 диск = 0; 3 диск = 1; 4 диск = 1. Три единицы, на 5-й диск записывается 1. Тогда получаем 4 единицы, т.е. «добиваемся» четного числа 4. Если теперь любой из пяти дисков сломается, то всегда можно восстановить пропавшие данные, записывая в новый диск «1» до четности, если количество единиц на оставшихся 4 дисках нечетное, и записывая «0», если количество единиц на оставшихся 4 дисках четное.

Но все же в современных RAID-массивах предпочитают отказываться от выделения диска для записи четности. Почему?

В системах RAID, использующих диск четности, можно восстановить данные при отказе только одного диска из массива. Диск просто меняют на новый, и начинается восстановления данных. Этот процесс называется восстановление, при этом считываются данные одновременно со всех оставшихся дисков для вычисления четности. Если во время восстановление откажет еще один диск, то все данные можно потерять полностью, так как уже невозможно ничего восстановить. Риск двойного отказа тем выше, чем дольше процесс восстановление. При современных объемах, когда они достигают террабайтов и более, это может длиться нескольку дней. Возможна ситуация с наличием так называемого «гнилого» бита (сбойного бита), являющегося одним из заводских параметров жестких дисков. В этом случае, который проявляется именно при больших объемах хранения, риск потери данных в RAID становится неприемлемо высоким.

В связи с этим в современных многодисковых системах хранения Big Data отказываются от RAID-массивов, и единственным методом защиты данных, де-факто, остается многократное копирование (репликация). Причем репликация идет не только между узлами, но и внутри узлов. Даже внутри DATA-центра приходится делать репликацию от одного сервера к другому.

3.2. Коды для исправления ошибок, включающие контрольную сумму

Коды исправления ошибок, включающие контрольные суммы, обычно используются в качестве простого и эффективного метода восстановления после потери данных.

Когда данные $D = \{d_0, d_1, \ldots d_n\}$ разделены на блоки одинакового размера, сумма данных может быть вычислена с использованием контрольной суммы $C_D = d_0 + d_1 + \ldots d_i \ldots + d_n$. В случае потери блока d_i он может быть восстановлен путем пересчета суммы оставшихся данных C'D, и разница между файлом суммы и пересчитанной суммой будет равна отсутствующему блоку, т.е. $d_i = C_D - C'_D$. Суммирование по модулю используется во время сложения для определения различий между блоками данных, гарантируя, что объем данных, необходимый для контроля контрольной суммы, не превышает размер блока. В двоичной логике вычисление контрольной суммы упрощается до суммирования по модулю 2, реализуемого с помощью одной операции XOR, которая известна как контроль четности в контексте обработки двоичных данных.

Первоначально эта технология применялась в RAID-системах для многодисковых систем. В RAID 3/4/5/6 данные делятся на блоки и вычисляется четность, при этом блоки данных и блоки контроля четности распределяются по разным дискам внутри массива. Такой массив может восстановиться после сбоя одного диска. Однако при работе с большими объемами данных RAID сам по себе может не обеспечить приемлемый уровень надежности. Восстановление после сбоя, которое включает в себя восстановление данных с вышедшего из строя диска и запись их на новый функциональный диск, становится неприемлемо трудоемким, поскольку для расчета требуются данные со всех дисков. Во время процесса восстановления (перестройки) система хранения данных недостаточно защищена от последующих сбоев, и потеря другого диска может привести к безвозвратной потере данных.

С другой стороны, RAID-массивы - это дорогостоящие устройства, предназначенные в первую очередь для высокоскоростного онлайн-хранения и обмена данными. RAID-контроллер внутри устройства обеспечивает одновременное чтение и запись на все диски, тем самым увеличивая общую скорость обмена данными. Для удовлетворения потребности в хранилищах большой емкости разрабатываются специализированные системы, которые сочетают относительно более медленные, но емкие диски с технологиями распределенного хранения, обеспечивающими устойчивость к сбоям дисков. Эти системы хранения данных также называются RAID, хотя в них не используются традиционные проприетарные технологии контроля четности. Для этих многодисковых технологий хранения данных существуют различные обозначения, такие как RAID 7.3 или RAID m+n, как в открытой, так и в проприетарной формах, которые отличаются от обычных RAID-систем и восходят к первоначальному значению аббревиатуры RAID — избыточный массив недорогих дисков, означающий массив доступных дисков [11].

3.3. Использование кодов Рида-Соломона

Из-за недостатков RAID-систем, связанных с высокой избыточностью, стоимостью и низкой надежностью, существует необходимость в изучении технологий хранения, использующих более продвинутые методы. Различные системы, часто называемые RAID n+m, используют помехоустойчивый алгоритм кодирования данных Рида-Соломона (RS). Примечательными примерами являются RAID 7.3 от RAIDIX из Санкт-Петербурга и решение IBM для хранения данных, основанное на технологии Cleversafe, приобретенное IBM.

Алгоритм RS позволяет настроить желаемую отказоустойчивость путем указания количества блоков, которые могут быть восстановлены после потери. Он может обнаруживать поврежденные или отсутствующие блоки и определять местоположение блоков, требующих восстановления. Теоретически алгоритм может обнаруживать t ошибок и восстанавливать информацию, используя избыточные данные, в результате чего получается в общей сложности $n+2t$ блоков, где n представляет собой количество блоков данных.

Метод RS максимизирует емкость хранилища при сохранении избыточности, близкой к теоретическому пределу. Однако это требует больших вычислительных затрат из-за сложных вычислений и может привести к значительной задержке системы при обработке обширного обмена информацией с системой хранения. Кроме того, алгоритму не хватает масштабируемости, поскольку любое увеличение емкости хранилища требует полного пересчета данных по всему массиву, учитывая жесткую алгебраическую структуру алгоритма RS. В современных многодисковых системах хранения данных с высокой частотой отказов даже незначительное отклонение от порогового значения отказа может привести к полной потере данных, поскольку отказоустойчивость алгоритма RS имеет пороговую природу. Другая проблема, связанная с распределенным хранилищем, заключается в том, что алгоритм RS требует для вычислений данных со всех дисков массива, что делает доступность данных уязвимой для скачков сетевого трафика хранилища.

Многомерные коды коррекции ошибок, известные как помехоустойчивое кодирование, предлагают различные подходы для преодоления этих проблем [12]. Мы разработали простой, ресурсосберегающий алгоритм, который обеспечивает надежность, сравнимую с алгоритмом RS, но лишенный его недостатков. Хотя он обеспечивает более высокую избыточность, чем алгоритм RS, она значительно ниже, чем избыточность, достигаемая с помощью методов репликации [13].

4. Результаты

4.1. Четность в одном измерении.

Давайте разделим данные на два блока одинакового размера и вычислим их четность.

$$D \to \{d_0, d_1\}, \quad d_2 = d_0 \oplus d_1 \tag{1}$$

где $\oplus$ обозначает операцию XOR, которая выполняет побитовое сложение по модулю 2, стоит отметить, что результирующие блоки данных обладают способностью восстанавливать любую недостающую часть. Покажем, что, например, $d_0 = d_1 \oplus d_2$. Это можно продемонстрировать, например, выполнив операцию с обеими сторонами уравнения d_1. Используя свойство инволюции операции XOR, мы упрощаем уравнение $d_1 \oplus d_1$, чтобы получить правильное равенство (1).

$$d_1 \oplus (d_0 = d_1 \oplus d_2) \to d_0 \oplus d_1 = d_2 \tag{2}$$

Аналогично доказывается равенство $d_1 = d_0 \oplus d_2$ с использованием операции $d_0 \oplus$:

$$d_0 \oplus (d_1 = d_0 \oplus d_2) \to d_0 \oplus d_1 = d_2 \tag{3}$$

В результате мы получаем набор из трех соотношений, которые позволяют восстановить любой потерянный блок данных в пределах набора:

$$\begin{cases} d_0 = d_1 \oplus d_2 \\ d_1 = d_0 \oplus d_2 \\ d_2 = d_0 \oplus d_1 \end{cases} \tag{4}$$

Результирующая тройка данных может быть выражена в виде одномерного вектора, выровненного по оси x, где элементы данных связаны между собой отношениями четности:

$$D \to \begin{bmatrix} d_0 \\ d_1 \\ d_2 \end{bmatrix}, \; d_i = d_{i+1} \oplus d_{i-1} \tag{5}$$

где $i = \{0, 1, 2\}$ вычисления выполняются с использованием операции по модулю 3. Это подразумевает, что формула четности работает в контексте арифметики по модулю (5) 0–1=3 и 3+1=0.

4.2. Вычисление четности в двух измерениях

Давайте разделим исходные данные на четыре блока одинакового размера. Мы расположим эти блоки в виде двумерной матрицы 2x2. Чтобы преобразовать матрицу, мы расширим ее до матрицы 3x3, добавив пустой столбец справа и пустую строку внизу. Далее мы вычислим контрольные суммы для каждой строки и столбца и поместим результаты в пустые места.

$$\begin{bmatrix} d_{00} & d_{01} \\ d_{10} & d_{11} \end{bmatrix} \to \begin{bmatrix} d_{00}\, d_{01}\, d_{02} \\ d_{10}\, d_{11}\, d_{12} \\ d_{20}\, d_{21} \end{bmatrix} \tag{6}$$

Результирующая матрица данных обладает двумерной четностью, что позволяет восстанавливать любые потерянные данные с помощью вычислений как по столбцам, так и по строкам. Это демонстрирует применение многомерного кода проверки четности [11].

Чтобы завершить матрицу, мы вычислим четность четности. Это приведет к двум равенствам – $d_{20} \oplus d_{22} = d_{22}$ и $d_{02} \oplus d_{12} = d_{22}$, где $\oplus$, напомним, что этот знак представляет операцию XOR, которая выполняет побитовое сложение по модулю 2. Мы демонстрируем это, выявляя значения четности и рассматривая коммутативность операции XOR:

$$d_{20} \oplus d_{21} = (d_{00} \oplus d_{10}) \oplus (d_{01} \oplus d_{11}) = d_{22}$$

$$d_{02} \oplus d_{12} = (d_{00} \oplus d_{01}) \oplus (d_{10} \oplus d_{11}) = (d_{00} \oplus d_{10}) \oplus (d_{01} \oplus d_{11}) = d_{22}' \tag{7}$$

получим $d_{22}' = d_{22}$.

Результирующая матрица 3x3 в двух измерениях будет содержать данные в своих столбцах и строках, соединенных с помощью операции XOR, применяемой к соседним парам данных. В результате каждый блок данных может быть получен с помощью операции XOR двумя способами, как по горизонтали, так и по вертикали:

$$D \to \begin{bmatrix} d_{00}\, d_{01}\, d_{02} \\ d_{10}\, d_{11}\, d_{12} \\ d_{20}\, d_{21}\, d_{22} \end{bmatrix}, \begin{cases} d_{i,j} = d_{i+1,j} \oplus d_{i-1,j} \\ d_{i,j} = d_{i,j+1} \oplus d_{i,j-1} \end{cases} \tag{8}$$

где числовое значение индекса определяются по модулю 3, что показывает кольцо вычетов {0, 1, 2}. Эти соотношения можно понимать как двумерную четность в координатах x и y, которые соответствуют одномерным векторам, полученным из строк и столбцов соответственно.

4.3. Вычисление четности в трёх измерениях

Следуя описанному алгоритму, становится возможным сгенерировать трехмерную матрицу путем разделения данных на 8 блоков одинакового размера и вычисления данных о четности по трем координатам *x, y, z*:

$$D \to \left\{ \begin{bmatrix} d_{000}\, d_{001}\, d_{002} \\ d_{010}\, d_{011}\, d_{012} \\ d_{020}\, d_{021}\, d_{022} \end{bmatrix}, \begin{bmatrix} d_{100}\, d_{101}\, d_{02} \\ d_{110}\, d_{111}\, d_{12} \\ d_{120}\, d_{121}\, d_{22} \end{bmatrix}, \begin{bmatrix} d_{200}\, d_{201}\, d_{202} \\ d_{210}\, d_{211}\, d_{212} \\ d_{220}\, d_{221}\, d_{222} \end{bmatrix} \right. \tag{9}$$

путем установления трех взаимосвязей между блоками данных:

$$\begin{cases} d_{i,j,k} = d_{i+1,j,k} \oplus d_{i-1,j,k} \\ d_{i,j,k} = d_{i,j+1,k} \oplus d_{i,j-1,k}, \text{ где } (i,j,k)\ mod\ 3 = \{0,1,2\} \\ d_{i,j,k} = d_{i,j,k+1} \oplus d_{i,j,k-1} \end{cases} \tag{10}$$

Если мы рассматриваем двумерную матрицу, описанную ранее, как состоящую из одномерных элементов (5), то трехмерная матрица формируется путем последовательного сложения наборов из трех двумерных матриц (9). Третья матрица, представленная координатной плоскостью z=2, генерируется на основе вычисленных данных о четности двух верхних плоскостей. В общем, любой блок данных со значением индекса 2 является блоком четности.

В результате мы получаем куб размером 2x2x2, состоящий из исходных блоков данных со значениями индекса 0 или 1, а также трех граничных плоскостей, содержащих данные о четности, одна из которых имеет значение индекса 2.

4.4. N-мерная четность

Описанную модель расщепления данных на блоки и способ генерации блоков четности можно расширить на любую *n*-мерную четность. При этом оригинальные данные будут расщепляться на 2^n блоков, после обработки алгоритмом генерации четностей мы получим 3^n блоков, где n – это размерность четности. Полученные блоки будут связаны между собой соотношениями четности:

$$\begin{cases} d_{i,j,k,\dots,n} = d_{i+1,j,k,\dots,n} \oplus d_{i-1,j,k,\dots,n} \\ d_{i,j,k,\dots,n} = d_{i,j+1,k,\dots,n} \oplus d_{i,j-1,k,\dots,n} \\ d_{i,j,k,\dots,n} = d_{i,j,k+1,\dots,n} \oplus d_{i,j,k-1,\dots,n}, \text{где индексы } mod\ 3 = \{0,1,2\} \\ \qquad\qquad \dots \\ d_{i,j,k,\dots,n} = d_{i,j,k,\dots,n+1} \oplus d_{i,j,k,\dots,n-1} \end{cases} \tag{11}$$

Ключевые характеристики многомерной четности включают:

1. По мере увеличения размерности четности увеличивается и количество методов восстановления потерянных блоков данных. В n-мерном пространстве фундаментальное свойство одномерного вектора четности преобразуется в перекрестный контроль четности в каждом из n направлений, позволяя восстановить потерянный блок, используя оставшиеся два.

2. По мере увеличения размерности расширяется не только контроль перекрестной четности, но и становятся доступными дополнительные опции для восстановления цепочки. Более высокая размерность предоставляет возможность восстановления "отсутствующих" данных для любой заданной координаты (в случае потенциально критической потери) путем использования других файлов. Это позволяет осуществлять пошаговый процесс восстановления, при котором требуемый файл восстанавливается после последовательного восстановления других файлов.

4.5. Иллюстрация пошагового восстановления

Допустим, что в матрице двух измерений отсутствуют 5 блоков данных. В случае с файлом d_{00} это представляет собой потенциально необратимую ситуацию, поскольку ни одна из координат не позволяет его восстановить:

$$\begin{bmatrix} d_{00} & d_{01} & d_{02} \\ d_{10} & d_{11} & d_{12} \\ d_{20} & d_{21} & d_{22} \end{bmatrix} \rightarrow \begin{bmatrix} & d_{11} & d_{12} \\ d_{20} & d_{21} & \end{bmatrix} \tag{12}$$

Но есть два варианта цепочечного восстановления:

$$\begin{gathered} d_{10} = d_{11} \oplus d_{12} \rightarrow d_{00} = d_{10} \oplus d_{20} \text{ – в два шага} \\ d_{22} = d_{20} \oplus d_{21} \rightarrow d_{02} = d_{22} \oplus d_{12} \rightarrow d_{01} = d_{11} \oplus d_{21} \rightarrow d_{00} = d_{01} \oplus d_{02} \text{ – в четыре шага} \end{gathered} \tag{13}$$

Наличие опций восстановления цепочки в многомерных кодах четности устраняет пороговый характер вероятности восстановления данных. Это означает, что даже в сценариях, где происходят теоретические безвозвратные потери, всегда существует возможность поиска комбинаций для восстановления цепочки.

В двумерной матрице четности, состоящей из 9 файлов, каждый файл соединен с двумя соседними файлами через паритетные связи, что позволяет осуществлять восстановление с их поддержкой. Это приводит к следующим наблюдениям:

а) потери от 1 до 3 мест хранения файлов всегда гарантированно будут восстановлены.

б) потери в 4-5 местах обеспечивают варианты восстановления цепочки, обеспечивая ненулевую вероятность восстановления.

в) естественно, наиболее серьезным сценарием потери файлов является отсутствие 6 из 9 файлов.

4.6. Различные сценарии потери данных, вызванные отказами хранилища

Давайте определим критическую комбинацию сбоев DL^{kn} как сценарий, при котором исходный файл не может быть восстановлен при потере k из n мест хранения. Например, в случае двумерной матрицы, если 4 из 9 ячеек хранения будут потеряны, набор критических комбинаций будет следующим:

$$DL^{4,9} = \left\{ \begin{bmatrix} & d_{02} \\ & d_{12} \\ d_{20} d_{21} d_{22} \end{bmatrix}, \begin{bmatrix} & d_{02} \\ d_{10} d_{11} d_{12} & \\ & d_{22} \end{bmatrix}, \begin{bmatrix} d_{01} & \\ d_{11} & \\ d_{20} d_{21} d_{22} & \end{bmatrix}, \begin{bmatrix} d_{00} & \\ d_{10} & \\ d_{20} d_{21} d_{22} & \end{bmatrix}, \begin{bmatrix} d_{00} & d_{01} & d_{02} \\ & d_{11} & \\ & d_{21} & \end{bmatrix}, \begin{bmatrix} d_{00} & d_{01} & d_{02} \\ d_{10} & d_{11} & d_{12} \\ d_{20} & & \end{bmatrix}, \begin{bmatrix} d_{00} & d_{01} & d_{02} \\ & & d_{12} \\ & & d_{22} \end{bmatrix}, \begin{bmatrix} & d_{01} & \\ d_{10} & d_{11} & d_{12} \\ & d_{21} & \end{bmatrix} \right\} \tag{14}$$

Суммарное количество потенциальных комбинаций из k местоположений из n эквивалентно количеству комбинаций из n выбранных k [12]:

$$C_k^n = \frac{n(n-1)\ldots(n-k+1)}{k(k-1)\ldots 1} = \frac{n!}{(n-k)!k!} \tag{15}$$

Невозможность восстановления после потери четырех блоков возникает, когда вышедшие из строя места хранения, расположенные в вершинах двумерной матрицы, приводят к пересекающимся файловым координатам, которые формируют парные потери.

Для каждой комбинации из 2 из 3 будет 2 комбинации из 2 из 3, в результате чего в общей сложности получится $DL^{4,9} = 9$ комбинаций.

Невозможность восстановления после потери пяти блоков возникает, когда места хранения расположены таким образом, что любые четыре из них образуют вышеупомянутую комбинацию.

Учитывая, что доступно пять позиций, общее количество комбинаций равно $DL^{5,9} = DL^{4,9} \cdot 5 = 45$.

Теперь давайте подсчитаем вероятность потери данных при случайном отказе четырех дисков из девяти. Каков шанс получить комбинацию $DL^{4,9}$, которая приведет к потере данных?

Вероятность потери данных $Q^{4,9}$ при выходе из строя четырех мест хранения:

$$Q^{4,9} = \frac{\text{число } DL^{4,9}}{\frac{n!}{(n-m)!m!}} = \frac{9}{\frac{9!}{(9-4)!4!}} = \frac{9}{126} = 0{,}0714 \tag{16}$$

Предположим теперь, что случайным образом отказывают 5 дисков. Какова вероятность, что мы получим при этом комбинацию $DL^{5,9}$ ведущую к потере данных?

Вероятность $Q^{4,9}$ потери данных при отказе пяти мест хранения:

$$Q^{5,9} = \frac{\text{число } DL^{5,9}}{\frac{n!}{(n-m)!m!}} = \frac{45}{\frac{9!}{(9-5)!5!}} = \frac{45}{126} = 0{,}3571 \tag{17}$$

Вероятность потери данных с комбинациями $Q^{6,9}$, $Q^{7,9}$, $Q^{8,9}$, $Q^{9,9}$ при потере 6 или более мест хранения равна 1,0. Другими словами, такие потери мест хранения являются невосполнимыми и приводят к полной потере данных.

С увеличением количества дисков в массиве, надежность многодисковых систем снижается. Путем использования статистической модели, учитывающей вероятность отказов дисков, можно утверждать, что частота отказов λ в массиве из n дисков увеличивается примерно пропорционально n [13]. Вероятность отказа диска Q_{dev} в течение заданного временного интервала t может быть выражена следующим образом:

$$Q_{dev} = 1 - e^{-\lambda t}, \tag{18}$$

Для массива, состоящего из n дисков, вероятность сбоя диска Q^n_{arr} будет определяться как:

$$Q^n_{arr} = 1 - e^{-n\lambda t} \tag{19}$$

Заводской показатель, представляющий качество диска, MTBF (среднее время между отказами), соответствует параметру λ, который указывает ожидаемое количество часов работы диска до первого отказа.

В текущем контексте наше внимание сосредоточено на определении вероятности одновременных отказов k дисков из заданного набора n. Хотя такие сбои могут возникать в массиве из n дисков, только сценарий фатального $DL^{k,n}$, приведет к сбою хранилища$C^n_k = \frac{n!}{(n-k)!k!}$. Это подразумевает, что нам нужно учитывать совместную вероятность k отказов диска из n, наряду с вероятностью возникновения фатального сценария $Q^{k,n}$. Предполагая, что отказ отдельных дисков является случайным и независимым событием, мы можем рассчитать вероятность сбоя хранилища k следующим образом:

$$Q^{k,n}_{arr} = Q^{k,n} \cdot (Q^n_{arr})^k \tag{20}$$

4.7. Вероятность отказа СХД с двумерной четностью

Приведем примерный расчет вероятности отказа СХД с двумерной четностью, использующей типовые жесткие диски с MTBF ≈ 800 тыс. часов (Mean Time Between Failure – время между сбоями или наработка на отказ). Вероятность сбоя диска в течение года с MTBF будет:

$$Q_{dev} = 1 - e^{-\lambda t} = 1 - e^{-\frac{8760}{800\,000}} \cong 0{,}0109 \quad (21)$$

где λ = 1/MTBF, а t = 1 год (8760 часов). Диски с таким MTBF имеют вероятность отказа в течение года 1,09%.

Вероятность сбоя какого-либо диска в течение года в массиве из 9 дисков, соответственно будет уже 9,38%:

$$Q_{dev} = 1 - e^{-9\lambda t} = 1 - e^{-9\cdot\frac{8760}{800\,000}} \cong 0{,}0938 \quad (22)$$

Вероятность отказа СХД при отказе 4 дисков, т.е. получение фатального сценария будет 0,00055273%:

$$Q_{arr}^{4,9} = Q^{k,n} \cdot (Q_{arr}^{9})^4 \cong 0{,}0714 \cdot (0{,}0938)^4 \cong 5{,}5273 \cdot 10^{-6} \quad (23)$$

Вероятность отказа СХД при отказе 5 дисков будет 0,0002593%:

$$Q_{arr}^{5,9} = Q^{5,9} \cdot (Q_{arr}^{9})^5 \cong 0{,}3571 \cdot (0{,}0938)^5 \cong 2{,}5930 \cdot 10^{-6} \quad (24)$$

5. Обсуждение

Избыточность СХД, использующей технологию многомерной четности для повышения надежности хранения в многодисковых массивах, определяется соотношением оригинальных файлов к общему числу файлов вместе с избыточностью.

Как указывалось при обсуждении свойств многомерной четности, она равна соотношению $R = (3/2)^n$, где n – размерность четности.

Для СХД с двумерной четностью $R = 2{,}25$, что практически равносильно двукратной репликации или зеркалированию. Вероятность отказа СХД с зеркалированием на два диска будет равна вероятности отказа двух дисков одновременно. Учитывая формулу (19) для вероятности отказа диска в массиве и считая события отказа независимыми и совместными, получаем вероятность отказа 2,166%:

$$Q_{mirror} = (1 - e^{-2\lambda t})^2 = (1 - e^{-2\cdot\frac{8760}{800\,000}})^2 \cong 0{,}02166 \quad (25)$$

В СХД с технологией двумерной четности при сравнимой избыточности мы получаем вероятности отказа от 0,00055273% до 0,0002593%.

Таким образом, расчетная вероятность потерь при отказе мест хранения при использовании кодов многомерной четности значительно меньше, чем при использовании иных вариантов СХД. Многомерные коды четности по своей сути не имеют определенной пороговой вероятности восстановления данных. Это означает, что даже в сценариях, где теоретически происходят безвозвратные потери, всегда существует возможность обнаружения комбинаций восстановления цепочки. Однако важно отметить, что надежность многодисковых систем обычно снижается по мере увеличения количества дисков в массиве. Но избыточность СХД, использующих технологию многомерной четности для повышения надежности хранения в многодисковых массивах, теоретически эквивалентна многократной репликации.

6. Выводы

Способ хранения и передачи файлов без использования шифрования, а только за счет расщепления файлов по алгоритмам многомерной четности обеспечивает конфиденциальность и гарантирует безопасность содержимого от несанкционированного доступа. Главная особенность этой технологии, в отличие от существующих методов обеспечения безопасности, заключается в ее способности создавать внутреннюю модель хранения и обработки данных, которая является непротиворечивой и актуальной, обладая высоким уровнем защиты от внешних вторжений [14].

Во-первых, система не позволяет воспользоваться информацией в случае несанкционированного доступа. Разделенные данные не содержат существенной информации. Кроме того, процесс восстановления может быть выполнен с использованием только части разделенных данных, что приводит к регенерации мест хранения [13]. При этом в отличие от метода резервирования (многократного копирования), система устроена так, что чем больше мест хранения используется, тем большее количество информации расщепляется. Фрагменты информации хранятся в разных местах. Желающему добыть информацию, придется собирать фрагменты со всех мест хранения. Но даже если он соберет то, что расщеплено, он должен знать, каким образом соединить части. Способ же расщепления/восстановления записан в мета-файле, который предоставляет доступ к информации. В шифровании есть ключ, в нашем методе – мета-файл. Это файл, в котором описан способ расщепления, именно с его помощью можно собрать то, что расщеплено.

Конечная цель технологии подразумевает функционирование базы данных, в которой частные пользователи, компании и организации могут хранить свои файлы, будучи уверенными в их безопасности.

Благодарность. *Исследование выполнено в рамках грантового финансирования МНиВО по проекту AP13068084 «Разработка технологии детектирования аномального (обманчивого) поведения респондента с использованием алгоритмов искусственного интеллекта (ИИ) на основе изменения характеристик голоса и речи» (конкурс молодых ученых по научным и (или) научно-техническим проектам на 2022-2024 годы).*

Список использованной литературы:

1 Сыргабеков И., Задаулы Е., Курманбаев Е. Защита информационных баз по методу распределенного хранения // Доклады Национальной академии наук Республики Казахстан. – №5. – 2014. – С. 141-153.

2 Задаулы Е., Курманбаев Е., Сыргабеков И. Инновационная система безопасности на базе распределенного хранения информации с расщеплением данных // Patriot Engineering. – №2 (7). 2015. С. 111-119.

3 Карипжанова А.Ж., Гудов А.М. Организация распределенных баз данных информационных систем методом расщепления данных // Материалы XIV (XLV) Международной научной конференции студентов и молодых ученых «Образование, наука, инновации: вклад молодых исследователей», г. Кемерово, Россия, 25 апреля 2019 г. – Кемерово, 2019.

4 Kurmanbaev E.A., Syrgabekov I. N., Zadauly E. Karipzhanova A.Zh., Urazbaeva K.T. Information Security System on the Basis of the Distributed Storage with Splitting of Data // International Journal of Applied Engineering Research. – 2017. – Vol. 12. – № 8. – pp. 1703-1711.

5 Karipzhanova A., Sagindykov K., Dimitrov K. Justification of the method and algorithm of multidimensional parity control in distributed databases of information systems // Proc. X National Conference with International Participation «Electronica 2019», May 16-17, 2019, Sofia, Bulgaria. – https://ieeexplore.ieee.org/xpl/conhome/8816819/proceeding. – DOI: 10.1109/ELECTRONICA.2019.8825600.

6 Plank J.S. Optimizing Cauchy Reed-Solomon Codes for Fault-Tolerant Storage Applications. – Tennessee, 2005.

7 Shokrollahi A. Transactions on Information Theory // Raptor Codes. – 2006. – Vol. 52. – P. 2551-2567.

8 Lee J.H. WEAVER Codes: Highly Fault Tolerant Erasure Codes for Storage Systems, in FAST-2005: 4th Usenix Conference on File and Storage Technologies, 2005.

9 Peterson W.W., Weldon E.J. Error-Correcting Codes / 2nd edition.Cambridge, Massachusetts: MIT Press, 1972.

10 Luby M. LT Codes // Proc. of the 43rd Annual IEEE Symp. on Foundations of Computer Science (FOCS), 2002.

11 Patterson D.A., Gibson G., Katz R.H. A Case for Redundant Arrays of Inexpensive Disks (RAID) // Proceed. of the 1988 ACM SIGMOD conf. on Management of Data. – Chicago IL, 1988. – P. 109-116.

12 Wong J., Shea M., Tan F. Multidimensional Codes. – The Wiley Encyclopedia of Telecommunications, 2016.

13 Карипжанова А.Ж. Тестирование системы хранения информации с применением алгоритмов многомерной четности, устойчивых к частичным потерям мест хранения // Вестник ЕНУ. Серия Математика. Компьютерные науки. Механика. – 2019. – Т. 129. – №4. – С. 67-76.

14 Карипжанова А.Ж., Сагындыков К.М. Способы повышения надежности хранения информации в базах данных // Вестник КазГЮИУ. – 2018. – №3(39). – С. 265-270.

References:

1 Syrgabekov I., Zadauly E., Kurmanbaev E. (2014) Zashhita informacionnyh baz po metodu raspredelennogo hranenija [Protecting infobases using the distributed storage method]. Doklady Nacional'noj akademii nauk Respubliki Kazahstan. №5. 141-153. (In Russian)

2 Zadauly E., Kurmanbaev E., Syrgabekov I. (2015) Innovacionnaja sistema bezopasnosti na baze raspredelennogo hranenija informacii s rasshhepleniem dannyh [Innovative security system based on distributed information storage with data splitting]. Patriot Engineering. №2 (7), 111-119. (In Russian)

3 Karipzhanova A.Zh., Gudov A.M. (2019) Organizacija raspredelennyh baz dannyh informacionnyh sistem metodom rasshheplenija dannyh [Organization of distributed databases of information systems by data splitting method]. Materialy XIV (XLV) Mezhdunarodnoj nauchnoj konferencii studentov i molodyh uchenyh «Obrazovanie, nauka, innovacii: vklad molodyh issledovatelej». (In Russian)

4 Kurmanbaev E.A., Syrgabekov I. N., Zadauly E. Karipzhanova A.Zh., Urazbaeva K.T. Information Security System on the Basis of the Distributed Storage with Splitting of Data // International Journal of Applied Engineering Research. – 2017. – Vol. 12. – № 8. – pp. 1703-1711.

5 Karipzhanova A., Sagindykov K., Dimitrov K. Justification of the method and algorithm of multidimensional parity control in distributed databases of information systems // Proc. X National Conference with International Participation «Electronica 2019», May 16-17, 2019, Sofia, Bulgaria. – https://ieeexplore.ieee.org/xpl/conhome/8816819/proceeding . – DOI: 10.1109/ELECTRONICA.2019.8825600.

6 Plank J.S. Optimizing Cauchy Reed-Solomon Codes for Fault-Tolerant Storage Applications. – Tennessee, 2005.

7 Shokrollahi A. Transactions on Information Theory // Raptor Codes. – 2006. – Vol. 52. – pp. 2551-2567.

8 Lee J.H. WEAVER Codes: Highly Fault Tolerant Erasure Codes for Storage Systems, in FAST-2005: 4th Usenix Conference on File and Storage Technologies, 2005.

9 Peterson W.W., Weldon E.J. Error-Correcting Codes / 2nd edition. Cambridge, Massachusetts: MIT Press, 1972.

10 Luby M. LT Codes // Proc. of the 43rd Annual IEEE Symp. on Foundations of Computer Science (FOCS), 2002.

11 Patterson D.A., Gibson G., Katz R.H. A Case for Redundant Arrays of Inexpensive Disks (RAID) // Proceed. of the 1988 ACM SIGMOD conf. on Management of Data. – Chicago IL, 1988. – pp. 109-116.

12 Wong J., Shea M., Tan F. Multidimensional Codes. – The Wiley Encyclopedia of Telecommunications, 2016.

13 Karipzhanova A.Zh. (2019) Testirovanie sistemy hranenija informacii s primeneniem algoritmov mnogomernoj chetnosti, ustojchivyh k chastichnym poterjam mest hranenija [Testing of the information storage system using multidimensional parity algorithms that are resistant to partial loss of storage locations]. Vestnik ENU, Serija "Matematika. Komp'juternye nauki. Mehanika". №4, 67-76. (In Russian)

14 Karipzhanova A.Zh., Sagyndykov K.M. (2018) Sposoby povyshenija nadezhnosti hranenija informacii v bazah dannyh [Ways to increase the reliability of information storage in databases]. Vestnik KazGJuIU. №3(39), 265-270. (In Russian)