УДК 004.056

МЕТОДЫ СОЗДАНИЯ РАСПРЕДЕЛЕННЫХ БАЗ ДАННЫХ ИНФОРМАЦИОННОЙ СИСТЕМЫ С МНОГОМЕРНЫМ КОНТРОЛЕМ ЧЕТНОСТИ

Карипжанова А.Ж.
докторант 3 курса специальности «Информационные системы»
Евразийского национального университета
им. Л.Н. Гумилева, г. Астана, Республика Казахстан
Научный руководитель Сагиндыков К.М. к.т.н., доцент
Евразийского национального университета
им. Л.Н. Гумилева, г. Астана, Республика Казахстан

Проблема безопасности баз данных – одна из основных на пути внедрения облачных технологий в финансовом секторе, на предприятиях, а также в рамках государственного сектора. Необходимо надежная безопасность хранимой информации.

Именно решение задачи обеспечения гарантированной безопасности составляет суть настоящей работы, которая заключается в применении разработанной нами в составе исследовательской группы в Казахском гуманитарно-юридическом инновационном университете (г. Семей) в течение последних 7 лет метода обеспечения безопасности по технологии распределенного хранения с использованием технологии многомерной четности [1]. В рамках этой системы нами разработаны гибридные облачные архитектуры, защищенные каналы связи и иные высокоэффективные, надежные и безопасные от внешних атак решения по хранению/обмену данных [2].

Кроме того, новый метод повышения надежности требуются в условиях резкого возросших объемов хранения и широкого распространения многодисковых СХД. Рассматривается нами способ хранения отвечает всем этим противоречивым требованиям к избыточности и надежности. Ниже приводятся теоретическое обоснование параметров надежности и преимуществ перед существующими методами распределенного хранения.

В многодисковых системах хранения вероятность отказов дисков увеличивается пропорционально количеству используемых дисков. Для повышения надежности хранения информации в таких СХД применяют различные методы восстановления данных с отказавших дисков. Самый распространённый метод повышения надежности, метод многократных репликаций, несмотря на высокую надежность, имеет высокую избыточность.

Существуют методы опирающиеся на алгоритмы обработки данных называемые корректирующим кодированием (кодирование с исправлением ошибок), которые позволяют обнаруживать и восстанавливать повреждения. К этому классу относятся многочисленные реализации методов, основанных

на известном алгоритме Рид-Соломон [3], фонтанных технологиях [4] и с использованием контрольных сумм — RAID5/6, кодах с локальной четностью LRC [5] и т.д.

В рамках модели исправляющих кодов разработана теория каналов со стираниями, которую предложил В. Питерсон в работе [6]. Модель каналов со стираниями, впоследствии, была развита в работе М. Лаби [7]. Стиранием называется ошибка, позиция которой известна. Предлагаемый метод работает именно в рамках данной модели, где под стиранием подразумевается полный отказ одного или более мест хранения в массиве дисков хранилища, позиции которых известны.

1. Корректирующие коды с контрольной суммой

Наиболее распространённым и простым методом восстановления потерь является использование кодов с контрольными суммами. Пусть имеется набор данных $D=\{d_0, d_1, \dots d_n\}$,, разбитый на блоки одинаковые по размеру. Если мы знаем сумму $C_D=d_0+d_1 \dots d_j \dots +d_n$,, то любую пропажу блока d_i можно восстановить пересчитав сумму оставшихся данных C_D, разность которой с файлом суммы и будет пропавшим блоком $d_{1^i}=C_1 D-C_1 D^{\uparrow}$.. При суммировании используется операция сложения по модулю, так как для целей контроля требуется знать только различия в блоках данных, т.е., в общем случае, объем данных, необходимый для контроля сумм не превышает размер блока. В бинарной логике вычисление контрольных сумм сводится к суммированию по модулю 2, которое реализуется одной операцией исключающее ИЛИ (XOR) и превращается в контроль четности. В дальнейшем мы будем использовать термин контроль четности, так как описываем технологии, связанные обработкой двоичных данных.

В применении к многодисковым системам такая технология впервые была реализована в системах RAID. В этой технологии (в частности, в RAID 3/4/5/6) данные делятся на блоки, вычисляется четность и блоки данных с блоком четности раскидываются по разным дискам массива. Такой массив может восстановить единичные отказы дисков. При хранении больших объемов данных RAID не может обеспечить приемлемый уровень надежности. Восстановление после отказа — восстановление данных с отказавшего диска и запись их на исправный новый диск — при возросших объемах начинает занимать недопустимо много времени, так как для проведения расчетов требуется считать данные со всех дисков. В процессе восстановления (Rebuild) система хранения данных (СХД), фактически, не защищен от повторных сбоев и, в случае отказа еще одного диска, теряются все данные безвозвратно.

С другой стороны, RAID массивы, весьма дорогие устройства потому, что основная область их применения, под которое все оптимизировано, это высокоскоростные устройства оперативного хранения и обмена данными. Контроллер RAID устройства может одновременно читать и записывать по всем дискам повышая общую скорость обмена данными. Для хранилищ

большой емкости разрабатываются специализированные системы использующие относительно медленные, но высокоемкие диски, объединенные технологией распределенного хранения устойчивого к отказам дисков. Такие СХД также называют RAID, хотя и не используется «фирменные» технологии четности. Встречаются различные обозначения технологий многодисковых хранилищ, как открытых, так и проприетарных, например, RAID 7.3 или RAID m + n, которые не имеют никакого отношения к традиционным RAID системам и отсылают нас к первоначальному смыслу аббревиатуры RAID — Redundant Array of Inexpensive Disks, — избыточный массив недорогих дисков.

2. Применение кодов Рид-Соломон

Высокая избыточность репликаций, дороговизна и низкая надежность систем RAID заставляет разрабатывать технологии хранения на более сложных методах. Известны различные системы, обычно обозначаемые как RAID n+m, с алгоритмом помехоустойчивого кодирования данных Рид-Соломон (РС). Наиболее известные это RAID 7.3 компании RAIDIX из Санкт-Петербурга, СХД от IBM на базе разработок компании Cleversafe купленных IBM.

Алгоритм РС позволяет задавать требуемую отказоустойчивость в виде количества блоков потерю которых можно восстановить. РС может обнаруживать повреждения или утери блоков и вычислять местоположение блоков, требующих восстановления. Теоретически, алгоритм позволяет обнаружить ^{t} ошибок и восстанавливать информацию с использованием избыточных данных. Общее количество блоков тогда будет $^{n+2t}$, где ^{n} — это количество блоков оригинальных данных.

Способ РС наиболее экономно увеличивает общий объем хранимых данных, избыточность практически близка к теоретическому пределу. Однако алгоритм ресурсоемкий потому, что использует сложные расчеты, а при интенсивном обмене информацией с СХД способен сильно повысить латентность системы. Также, алгоритм плохо масштабируется, любое увеличение мест хранения требует полного перерасчета данных всего массива, так как РС имеет жесткую алгебраическую структуру. В условиях современных многодисковых хранилищ с большой частотой отказов высок риск потери данных при малейшем случайном превышении порога отказов потому, что отказоустойчивость РС носит пороговый характер — малейшее превышение приводит к полному отказу всего хранилища. Еще одно свойство алгоритма, в применении к распределенному хранению, вызывает большие проблемы — это то, что РС требуются данные со всех дисков массива для расчетов и всплески трафика в сети СХД будут сильно влиять на доступность данных.

3. Многомерная четность

Существуют различные методы помехоустойчивого кодирования, называемые многомерными кодами коррекции ошибок, применяющиеся в помехоустойчивом кодировании [8].

Разработан простой не ресурсоемкий алгоритм со сравнимой с РС надежностью, но свободный от его недостатков. Избыточность при этом, хотя и выше чем у РС, но гораздо меньше чем при использовании репликаций.

3.1. Одномерная четность

Разобьём данные на два блока равные по размеру и вычислим четность

$$D \to \{d_0, d_1\}, \qquad d_2 = d_{0 \infty d_1} \tag{1.}$$

где $\oplus$ обозначает операцию XOR, реализующую побитовое сложения по модулю 2.

Обратим внимание, что, исходя из свойств операции XOR, полученные блоки данных обладают свойством восстановления любой пропавшей части. Покажем, что, например, $d_0 = d_{1 \infty d_2}$..

Произведем операцию $d_{1\infty}c$ с обеими сторонами равенства. Учитывая инволютивность операции сокращаем $d_{1\infty}d_1$ и получаем верное равенство (1).

$$d_{1\infty(d_0} = d_{1\infty d_2}) \to_{d_{0\infty d_1}} = d_2 \tag{2.}$$

Аналогично доказывается равенство $d_1 = d_{0\infty d_2}$ с использованием операции $^{d_0}\infty$

$$d_{0\infty(d_1} = d_{0\infty d_2}) \to_{d_{0\infty d_1}} = d_2 \tag{3.}$$

Отсюда имеем тройку соотношений, из которых можно восстановить потерю любого блока данных из тройки

$$\begin{cases} d_0 = d_{1\infty d_2} \\ d_1 = d_{0\infty d_2} \\ d_2 = d_{0\infty d_1} \end{cases} \tag{4.}$$

Полученную тройку данных можно представить, как одномерный вектор по координате x данные которой связаны между собой соотношениями четности

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

где $^{i=\{0,1,2\}}$ вычисляется по модулю 3. Это означает, что в формуле четностей (5) $^{0-1=3}$ и $^{3+1=0}$. .

3.2. Двумерная четность

Разобьем оригинальные данные на четыре одинаковых по размеру блока. Впишем блоки в двумерную матрицу 2x2. Преобразуем матрицу в 3x3 добавив пустой столбец справа и пустую строку снизу изначальной матрицы. Вычислим контрольные суммы по строкам и столбцам, а результаты впишем в пустые места.

$$\begin{bmatrix} d_{0_0} & d_{0_1} \\ d_{1_0} & d_{1_1} \end{bmatrix} \rightarrow \begin{bmatrix} d_{0_0} & d_{0_1} & d_{0_1} \\ d_{1_0} & d_{1_1} & d_{0_1} \\ d_{2_0} & d_{2_1} & \end{bmatrix} \quad (6.)$$

Полученная матрица данных теперь позволяет восстановить любую потерю данных уже двумя способами — как по столбцам, так и по строкам. Полученная матрица демонстрирует двумерную четность Multidimensional parity-check code [9].

Дополним матрицу вычислив четности четностей. Мы получим два равенства — $d_{0_0} \infty d_{2_1} = d_{2_2}$ и $d_{0_2} \infty d_{1_2} = d_{2_2}$ Докажем, что $d_{2_2} = d_{2_2}$. Если раскрыть значения четностей и учесть коммутативность операции XOR

$$\begin{aligned} d_{2_0} \infty d_{2_1} &= (d_{0_0} \infty d_{1_0}) \infty (d_{0_1} \infty d_{1_1}) = d_{2_2} \\ d_{0_2} \infty d_{1_2} &= (d_{0_0} \infty d_{0_1}) \infty (d_{1_0} \infty d_{1_1}) = (d_{0_0} \infty d_{1_0}) \infty (d_{0_1} \infty d_{1_1}) = d_{2_2} \end{aligned} \quad (7.)$$

получим $d_{2_2} = d_{2_2}$.

Результирующая двумерная матрица 3x3 будет матрицей, все данные, в столбцах и строках которой, связаны друг с другом путем взятия операции XOR между соседними парами данных. Каждый блок данных, при этом, можно получить двумя способами – операцией XOR как по строкам, так и по столбцам

$$D \rightarrow \begin{bmatrix} d_{0_0} & d_{0_1} & d_{0_2} \\ d_{1_0} & d_{1_1} & d_{1_2} \\ d_{2_0} & d_{2_1} & d_{2_2} \end{bmatrix} \qquad \begin{cases} d_{i,j} = d_{i+1,j} \infty d_{i-1,j} \\ d_{i,j} = d_{i,j+1,j+1} \infty d_{i,j-1} \end{cases} \quad (8.)$$

где значения индексов вычисляются по модулю 3, т.е. являются кольцом вычетов $^{\{0,1,2\}}$. Эти соотношения можно интерпретировать как двумерную четность по координатам ^{x,y}, соответствующих одномерным векторам из строк и столбцов.

3.3. Трехмерная четность

По описанному алгоритму можно получить трехмерную матрицу, расщепляя данные на 8 одинаковых по размеру блоков и генерируя данные четности уже по трем координатам x, y, z

$$D\rightarrow \begin{bmatrix} d_{00_0} & d_{00_1} & d_{00_2} \\ d_{01_0} & d_{01_1} & d_{01_2} \\ d_{02_0} & d_{02_1} & d_{02_2} \end{bmatrix}, \begin{bmatrix} d_{10_0} & d_{10_1} & d_{10_2} \\ d_{11_0} & d_{11_1} & d_{11_2} \\ d_{12_0} & d_{12_1} & d_{12_2} \end{bmatrix}, \begin{bmatrix} d_{20_0} & d_{20_1} & d_{20_2} \\ d_{21_0} & d_{21_1} & d_{21_2} \\ d_{22_0} & d_{22_1} & d_{22_2} \end{bmatrix} \quad (9)$$

с блоками данных связанных между собой уже тремя соотношениями

$$\begin{cases} d_{i,j,k} = d_{i+1,j,k} \infty d_{i-1,j,k} \\ d_{i,j,k} = d_{i,j+1,k} \infty d_{i,j-1,j,k} \\ d_{i,j,k} = d_{i,j,k+1} \infty d_{y,j,k-1} \end{cases} \text{ где } (i,j,k) \bmod 3=\{0,1,2\} \quad (10)$$

Если описанная выше двумерная матрица состоит из одномерных (5), то трехмерная матрица, соответственно, состоит из наборов трех плоскостей двумерных матриц (9), расположенных последовательно друг под другом. Третья матрица — плоскость с координатой $^{z=2}$, образуется из вычисленных данных четности верхних двух плоскостей. В общем случае, блок данных имеющий значение любого индекса равное 2 является блоком четностей.

Таким образом, мы получаем куб 2x2x2 с оригинальными данными индексы блоков которых имеют значения 0 или 1 и три граничных плоскости, на которых расположены данные четностей, одно из значений индекса которых равно 2.

4. Основные свойства многомерной четности

1. С ростом размерности четности растет и число способов восстановления потерянных блоков данных.

В n -мерном пространстве основное свойство одномерного вектора четности: восстановление потери одного из трех блоков с помощью двух оставшихся, превращается в перекрестный контроль четности по каждому из n -направлений.

2. С ростом размерности одновременно растут варианты не только перекрестного контроля четности, но и дополнительных вариантов цепочечного восстановления.

С ростом размерности появляются, дополнительные варианты восстановления, когда «недостающие» данных по какой-либо из координат — условно фатальном случае потери для этого файла, в свою очередь, можно восстановить с помощью других файлов, с последующим восстановлением, на следующем шаге, требуемого файла.

4.1. Пример цепочечного восстановления

В двумерной матрице потеряны 5 блоков данных. Для файла d_{00} это условно фатальный случай, так как ни по одной из координат его невозможно восстановить

$$\begin{bmatrix} d_{0_0} & d_{0_1} & d_{0_2} \\ d_{1_0} & d_{1_1} & d_{1_2} \\ d_{2_0} & d_{2_1} & d_{2_2} \end{bmatrix} \qquad \begin{bmatrix} d_{1_1} & d_{1_2} \\ d_{2_0} & d_{2_1} \end{bmatrix} \tag{9.}$$

Но есть два варианта цепочечного восстановления
$d_{10=}\, d_{1_{1\infty d_{1_2}}} \rightarrow\ d_{0_{0=}}\, d_{10\infty d_{2_0}} \rightarrow$ в два шага
или

$$d_{22} = d_{2_{0\infty d_{2_1}}} \rightarrow d_{0_2} = d_{2_{2\infty d_{1_2}}} \rightarrow d_{0_1} = d_{1_{1\infty d_{2_1}}} \rightarrow d_{0_0} = d_{0_{1\infty d_{0_2}}} \tag{10.}$$

Наличие вариантов цепочечного восстановления означает, что:

3. Коды многомерной четности не имеют порогового характера вероятности восстановления данных.

Это свойство, означает, что даже при достижении «теоретически» невосстанавливаемых потерь, всегда есть вероятность найти комбинации цепочечного восстановления.

5. Свойства СХД с двумерной четностью

В матрице двумерной четности 9 файлов, в которой каждый файл связан соотношениями четности с двумя соседними и может быть восстановлен с их помощью. Это означает, что:

- *потери от 1 до 3 любых мест хранения файлов всегда гарантированно восстанавливаются;*
- *потери от 4 до 5 мест имеют варианты цепочечного восстановления, поэтому имеется ненулевая вероятность восстановления;*
- *безусловно фатальный сценарий утерь файлов – это утеря 6 файлов из 9.*

5.1. Комбинации отказов мест хранения приводящих к потере данных

Обозначим фатальную комбинацию потерь DL^{kn} , когда оригинальный файл не может быть восстановлен при сбое ^{k} мест хранения из ^{n}. Например, для двумерной матрицы, при потере 4 мест хранения из 9, перечень фатальные комбинаций будет выглядеть так:

$$d_{00}\&d_{01}\&d_{02}@d_{10}\&\&d_2 \tag{11.}$$

Общее число возможных комбинаций по ^{k} мест из ^{n} равно сочетаниям ^{n} по ^{k} [10]

$$C_k^n = \frac{n(n-1)\ldots(n-k+1)}{k(k-1)\ldots 1} = \frac{n!}{(n-k)!k\quad !} \tag{12.}$$

1. Потеря четырех блоков не имеют вариантов цепочечного восстановления, если координаты отказавших мест хранения расположены в вершинах двумерной матрицы таким образов, что образуются парные потери по пересекающимся координатам файлов.

На каждую комбинацию 2 из 3 придется по 2 комбинации 2 из 3

всего комбинаций $DL^{4,9}=(C_2^2)^2=9$ (13.)

2. Потеря 5 блоков не имеет вариантов цепочечного восстановления, если координаты мест хранения расположены таким образом, что любые четыре из них образуют комбинацию, описанную в п.1.

Учитывая, что свободных позиций пять, то получаем всего комбинаций

$$DL^{5,9}=DL^{4,9}*5=45 \quad (14.)$$

5.2. Вероятность потерь данных при отказе мест хранения

5.2.1. Комбинаторная вероятность

Предположим, что случайным образом отказывают 4 диска из 9.

Какова вероятность, что мы получим при этом комбинацию $DL^{4,9}$ ведущую к потере данных?

1. Вероятность $Q^{4,9}$ потери данных при отказе четырех мест хранения.

$$Q^{4.9} = \frac{\text{число } DL^{4.9}}{\frac{n!}{(n-m)!m!}} = \frac{9}{\frac{9}{(9-4)!4!}} = \frac{9}{126} = 0.0714 \quad (16.)$$

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

2. Вероятность $Q^{5,9}$ потери данных при отказе пяти мест хранения

$$Q^{5.9} = \frac{\text{число}DL^{5.9}}{\frac{n!}{(n-m)!m!}} = \frac{45}{\frac{9!}{(9-5)!5!}} = \frac{45}{126} = 0.3571 \quad (17.)$$

3. Вероятность $Q^{6,9}$, $Q^{7,9}$,$Q^{8,9}$, $Q^{0,9}$ потери данных при потере от 6 мест хранения и выше равна 1,0. Т.е., такие потери мест хранения восстановить невозможно и они являются стопроцентно фатальными.

5.2.2. Статистическая вероятность

Надежность многодисковых систем снижается с увеличением количества дисков в массиве. Используя статистическую модель вероятности отказов дисков можно утверждать, что частота отказа диска λ в массиве из n дисков увеличивается, практически, пропорционально в n раз. Вероятность отказа Q_{dev} диска в течение времени t будет

$$Q_{dev} = 1 - e^{-\lambda t} \quad (15.)$$

Для массива из n дисков вероятность отказа диска Q_{arr}^{n} будет

$$Q_{arr}^{n} = 1 - e^{-n\lambda t} \quad (16.)$$

За параметр λ отвечает заводская характеристика качества диска MTBF (Mean time between failures) — условное количество часов работы диска до первого отказа.

В рассматриваемом случае нас интересует вероятность одновременного отказа k дисков из n. Таких отказов в массиве из n дисков будет $C_n^k = \frac{n!}{(n-k)!k!}$, но приведет к отказу СХД только фатальный сценарий $DL^{k,n}$. Это

означает, что мы имеем совместную вероятность отказа ^{k} дисков из ^{n} с вероятностью $Q^{k,n}$ возникновения при этом фатального сценария. Считая вероятность отказа ^{k} дисков одновременно случайными и независимыми событиями получаем вероятность отказа СХД

$$Q_{arr}^{n} = Q^{k,n} * (Q_{arr}^{n})^{k} \quad (17.)$$

Список литературы:

1. 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.

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

4. Shokrollahi A. // *Raptor Codes,* Vol. 52, IEEE Transactions on Information Theory, 2006, pp. 2551-2567.

5. WEAVER Codes: Highly Fault Tolerant Erasure Codes for Storage Systems // *FAST-2005: 4th Usenix Conference on File and Storage Technologies*, 2005.

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

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

8. Wong T.F. Multidimensional Codes // *The Wiley Encyclopedia of Telecommunications*, 2016.

9. Wikipedia, «Multidimensional parity-check code,» [В Интернете]. Available: https://en.wikipedia.org/wiki/Multidimensional_parity-check_code.

10. Виленкин Н.Я., Виленкин А.Н., Виленкин П.А. Комбинаторика, Москва: "Фима" МЦНМО, 2017. – 400 с.