МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение
высшего образования «Санкт-Петербургский политехнический
университет Петра Великого»

Институт компьютерных наук и кибербезопасности

Высшая школа технологий искусственного интеллекта

Направление: 02.03.01 Математика и компьютерные науки

ОТЧЕТ О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

Реализация модели, демонстрирующей работу избыточного массива независимых дисков RAID 50

Преподаватель _____ Чуватов М. В.

«_____»_____ 2024г.

Санкт-Петербург, 2024

СПИСОК ИСПОЛНИТЕЛЕЙ

Создание модели _____ Д.В. Губковский

РЕФЕРАТ

Отчёт 13 с., 1 кн., 1 рис., 3 источн.

ИЗБЫТОЧНЫЙ МАССИВ НЕЗАВИСИМЫХ ДИСКОВ, RAID, RAID 50.

Объектом исследования является избыточный массив независимых дисков RAID 50.

Цель работы – реализация модели, демонстрирующей работу избыточного массива независимых дисков RAID 50 на основе 6 дисков для ввода 14 битового шестнадцатиричного значения с подсчетом избыточности через операцию XOR.

В процессе работы были проанализированы источники информация, содержащие сведения о избыточных массивах независимых дисков. Также была составлена модель работы RAID 50.

В результате исследования была реализована модель, демонстрирующая работу избыточного массива независимых дисков RAID 50 на основе 6 дисков для ввода 14 битового шестнадцатиричного значения с подсчетом избыточности через операцию XOR.

Содержание

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ 5

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ 6

ВВЕДЕНИЕ 7

1 ОСНОВНАЯ ЧАСТЬ 8
- 1.1 RAID-массив . . . 8
- 1.2 Классификация RAID-массив . . . 9
- 1.3 Комбинированные уровни RAID . . . 11

2 ОСОБЕННОСТИ РЕАЛИЗАЦИИ 12
- 2.1 Функция check_disks . . . 12
- 2.2 Функция reconstruction . . . 13
- 2.3 Функция write . . . 14
- 2.4 Функция read . . . 16

ЗАКЛЮЧЕНИЕ 18

СПИСОК ИСТОЧНИКОВ 19

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

В настоящем отчете о НИР применяют следующие термины с соответствующими определениями:
Избыточный массив самостоятельных дисков — это технология объединения двух и более накопителей в единый логический элемент с целью повышения производительности и (или) отказоустойчивости отдельно взятого элемента массива..

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящем отчете о НИР применяют следующие сокращения и обозначения:
RAID - Redundant Array of Independent Disks.

ВВЕДЕНИЕ

Цель работы: реалировать модель, демонстрирующую работу избыточного массива независимых дисков RAID 50. Условия работы:

1. Подсчет избыточности идет через операцию XOR;
2. Размер входный данных — 14 байтов;
3. Количество дисков в массиве — 6 единиц;
4. Количество адресов в массиве — 64 единицы.

Задачи:

1. Изучить строение дискового массива RAID 50.
2. Изучить организацию записи подсчета избыточности.
3. Реализовать модель демонстрирующую работуизбыточного массива независимых дисков RAID 50.
4. Протестировать полученную модель.

Условия работы модели:

1. Модель записывать данные по определенному адресу;
2. Модель предоставлять пользователю возможность прочесть данные по определенному адресу;
3. Модель должна восстановить один из дисков, если тот выйдет из строя.

1 ОСНОВНАЯ ЧАСТЬ

1.1 RAID-массив

RAID (Redundant Array of Independent Disks — избыточный массив самостоятельных (независимых) дисков) — это технология объединения двух и более накопителей в единый логический элемент с целью повышения производительности и (или) отказоустойчивости отдельно взятого элемента массива.

Избыточные данные по-другому называют дополнительными. Для их хранения нужен какой-то объем. Выделяемый объем для этих данных в массивах называется аналогично — избыточным. Избыточность массива — это наличие такого объема, и чем он больше, тем больше у массива избыточность, что в свою очередь обеспечивает отказоустойчивость.

Простыми словами, RAID-массив — это несколько дисков соединенные в единый диск с выделением места под хранение информации для исправления ошибок в данных.

RAID-массивы классифицируются по следующим параметрам:

1. по исполнению RAID-контроллера;
2. по типам поддерживаемых интерфейсов накопителей;
3. по поддерживаемым уровням RAID.

1.2 Классификация RAID-массив

RAID 0

Данный массив еще называется «чередованием», так как при записи информации на него она разбивается на части (блоки) фиксированного размера и записывается поочередно на все собранные в массив диски. Применяется для библиотеки игр, видеомонтажа или рендеринга.

Количество дисков для сборки RAID 0 — минимум две штуки. При использовании такого уровня RAID возрастают скорости записи и чтения информации, поскольку операции происходят параллельно на всех дисках. Чем больше дисков в массиве, тем он производительней. Доступен объем всех дисков. Но самый главный минус в том, что поломка одного из дисков такого массива будет означать полную потерю информации, поскольку данные хранятся разбитыми по дискам.

RAID 1

Массив известный как «зеркалирование». Представляет собой полную копию информации с одного диска массива на другой. RAID 1 подойдет для важных данных, сохранность и доступность которых в приоритете.

Предполагается использование двух накопителей. За счет дублирования обеспечивается высокая надежность — работа продолжится при повреждении одного диска. Но в таком случае необходима его срочная замена, при этом данные восстанавливаются с «зеркального» диска. Распараллеливанием запросов обеспечивается высокая скорость чтения.

Поскольку диски являются клонами друг друга, то для использования доступен объем одного диска. Логическим продолжением этого является двухкратная цена за гигабайт памяти. А еще скорость записи остается либо такой же, как в случае использования одного диска, либо даже может быть меньше.

RAID 2

Эти массивы используют чередование дисков и коды коррекции ошибок (код Хэмминга). Из-за этого диски в нем распределяются на две группы: для самих данных и для указанных выше кодов.

За счет чередования достигается высокая скорость операций с данными по сравнению с одним диском. А Код Хэмминга позволяет обнаруживать и исправлять ошибки при операциях с файлами без снижения скорости операций с данными. Также при выходе из строя одного накопителя массива восстанавливаться данные будут по хранящимся кодам коррекции ошибок.

RAID 3

Так же, как и RAID 2, использует чередование дисков, но без кодов Хэмминга. Вместо этого хранятся контрольные суммы, и они-то и используются для восстановления, а данные разбиваются на байты. Данный RAID-массив наиболее удачен для работы с большими файлами, потоковым мультимедиа, но на практике не встречается ввиду невысокой надежности.

Порог вхождения в этот тип массива — три диска. Скорость операций чтения высокая, скорости операций записи высоки только для больших файлов. RAID 3 предлагает хороший компромисс между доступным объемом и ценой. Хранящаяся информация теряется, если из строя выйдет больше одного диска.

К недостаткам. Могут наблюдаться проблемы со скоростью при работе с данными небольшого объема. Да и не все контроллеры поддерживают этот массив. Плюс высокая нагрузка на диск, хранящий контрольные суммы, сокращает срок его службы относительно дисков с данными.

RAID 4

Можно сказать, это тот же самый RAID 3, только файл бьется не на однобайтные блоки, за счет чего удалось несколько повысить скорость записи мелких файлов. Остальные характеристики соответствуют рейду третьего уровня.

RAID 5

Здесь используются контрольные суммы и чередование как в RAID 3 и RAID 4, но объем под хранение сумм распределяется по всему массиву. Это дает прирост в скорости записи, так как операции теперь можно производить параллельно. Количество накопителей в массиве начинается с трех. Для хранения контрольных сумм выделяется объем, равный объему одного накопителя. RAID 5 наиболее распространен и используется в файловых серверах, серверах общего хранения, серверах резервного копирования, работе с потоковыми данными и других средах, требующих хорошей производительности.

Полезный объем накопителей зависит от количества накопителей в массиве — чем больше накопителей, тем больше объем. Скорость чтения — высокая, относительно RAID 4 выше и скорость записи. А за счет распределения сумм по массиву и нагрузка на все диски равномерная.

Но когда один диск выходит из строя, надежность массива существенно падает, и он переходит в критическое состояние. Восстановление — очень длительный процесс, вызывающий падение производительности массива и сильно увеличивающий нагрузку на накопители, поскольку с них производится продолжительное интенсивное чтение.

RAID 6

Контрольные суммы на этот массив записываются в двойном размере, что требует и увеличения объема в два раза. Такая функция приводит к повышению отказоустойчивости, при этом не сильно увеличивая стоимость.

Минимальное количество дисков — четыре. Ко всем достоинствам RAID 5 прибавляется повышенная надежность, за счет допустимого отказа двух дисков одновременно. Но скорость записи может быть значительно ниже, чем у RAID 5.

1.3 Комбинированные уровни RAID

RAID 50

Массивы представляют собой кооперирование уровней 0 и 5. RAID 05 — это разбиение двух массивов RAID 5. Находит применение в серверах очень большой емкости.

Допускает потерю по одному диску в разных массивах RAID 5 без утраты данных, соответственно, чем больше чередуется массивов, тем больше дисков допустимо потерять.

К достоинствам RAID 5 прибавляется повышение скорости записи и повышение скорости восстановления. Допускаемый минимум — шесть дисков.

2 ОСОБЕННОСТИ РЕАЛИЗАЦИИ

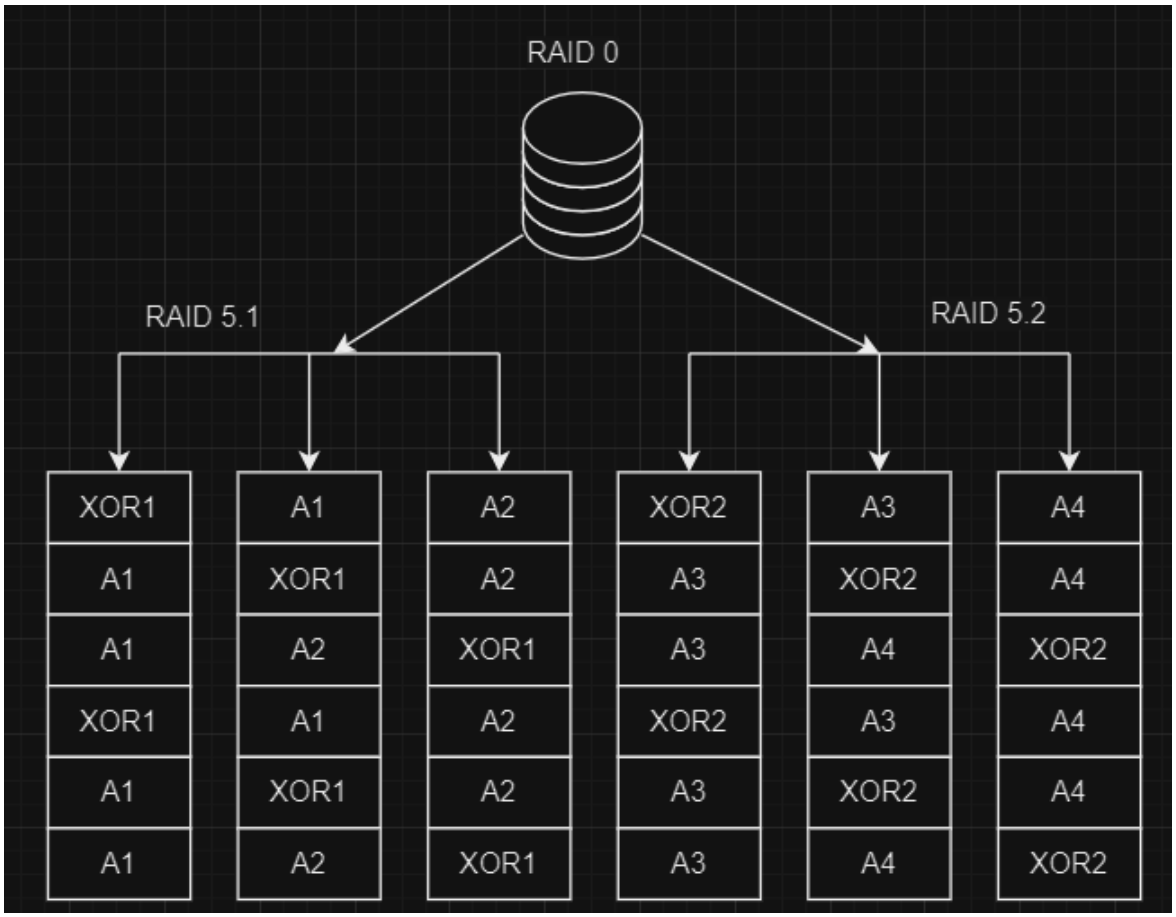

Рис. 1.Модель RAID 50

На рисунке 1 представлена схема RAID 50 для 6 дисков. Данные разделены на 4 части: A1, A2, A3 и A4. В первую часть массива записывается первая часть данных A1, A2 и расчитывается избыточность XOR1. По такому же принципу записывается вторая часть данных во второй массив: A3, A4 и избыточность XOR2 для этих данных.

2.1 Функция check_disks

Вход: система дисков.

Выход: восстановленный диск, если произошла потеря одного из дисков.

Данная функция проверяет все диски на работоспособность. Если же какой-то из дисков вышел из строя, запускается процесс восстановления.

```
1 def check_disks() -> None:
2     for i in range(len(disks)):
3         if not os.path.isfile(disks[i]):
4             print("The disk {} was lost.".format(i))
5             reconstruction(i)
6             break
```

Листинг 1. Функция check_disks

2.2 Функция reconstruction

Вход: индекс потерянного диска.

Выход: полностью восстановленный диск.

Данная функция функция использует информацию из других дисков и восстанавливает содержимое потерянного диска. Для этого происходит полный сбор информации. Далее проходит проверка по строчно, если в данной строке в утерянном диске были данные, то они восстанавливаются через функцию XOR.

```
1 def reconstruction(disks_index: int) -> None:
2     global disks_indexes
3     global RAID_5_1
4     global RAID_5_2
5
6     if (disks_index < 3):
7         problematische_Raid = RAID_5_1
8         problematische_index = disks_index
9     else:
10        problematische_Raid = RAID_5_2
11        problematische_index = disks_index - 3
12
13    lost_data = []
14    list_of_data = []
15    for i in range(len(problematische_Raid)):
16        if i != problematische_index:
17            file = open(problematische_Raid[i], 'r')
18            list_of_data += [[x for x in file.readlines() if x != '_\n']]
19            file.close()
20        else:
21            list_of_data += ["*" * len(disks_indexes)]
22
23
24
25    for i in range(len(list_of_data[0])):
26        small_list_of_data = []
27        for j in range(len(list_of_data)):
28            small_list_of_data.append(list_of_data[j][i])
29
30        indexes = [x for x in range(len(small_list_of_data))
31             if small_list_of_data[x] != '*']
32
33        lost_data.append(xor_two_str(small_list_of_data[indexes[0]][:-1:],
34            small_list_of_data[indexes[1]][:-1:])[2::])
35
36    for i in range(len(lost_data)):
37        while (len(lost_data[i]) < 3):
38            lost_data[i] = '0' + lost_data[i]
```

```
39
40      ind = 0
41      with open(disks[disks_index], 'w') as file:
42          for i in range(64):
43              if i in disks_indexes:
44                  file.write(lost_data[ind] + '\n')
45                  ind += 1
46              else:
47                  file.write("_\n")
48
49      print("The disk {} was recovered.".format(disks_index))
```

Листинг 2. Функция reconstruction

2.3 Функция write

Вход: адрес ячейки для записи и значение для записи.

Выход: измененый дисковый массив с новыми данными.

Для начала проиходит двойная проверка на наличие всех дисков в двух подмассивов. После чего исходное значение для записи разбивается на 2 части по 7 байтов. Далее еще на две части по 4 и 3 байта соответственно. Для каждой пары расчитывается своя избыточность. После чего происходит запись данных так, чтобы блоки по 4 байта, 3 байта и избыточность шли в порядке, как указанно на рисунке 1.

```
1 def write() -> None:
2     check_disks()
3     check_disks()
4     global disks_indexes
5
6     while (True):
7         input_index = input("Enter index to write: ")
8         try:
9             if int(input_index) > 63 or int(input_index) < 0:
10                print("Out of addressing limits .")
11            else:
12                break
13        except:
14            print("Wrong index.")
15
16
17    while (True):
18        input_data = str(input("Enter the string: "))
19        if len(input_data) != 14:
20            print("String length is not equal to 14 bytes.")
21        else:
22            break
23
24
```

```
25
26      blocks = [input_data[:4], input_data[4:7],
27          input_data[7:11], input_data[11:14]]
28      for i in range(len(blocks)):
29          while (len(blocks[i]) < 3):
30              blocks[i] = '0' + blocks[i]
31
32      excess_data1 = xor_two_str(blocks[0], blocks[1])[2::]
33      excess_data2 = xor_two_str(blocks[2], blocks[3])[2::]
34
35      disks_indexes.append(int(input_index))
36      disks_indexes = list(set(disks_indexes))
37      l1 = ['', '', '']
38      l2 = ['', '', '']
39
40      l1[int(input_index) % 3] = excess_data1
41      l2[int(input_index) % 3] = excess_data2
42      indexes = [x for x in range(len(l1)) if x != int(input_index) % 3]
43
44      l1[indexes[0]] = blocks[0]
45      l2[indexes[0]] = blocks[2]
46
47      l1[indexes[1]] = blocks[1]
48      l2[indexes[1]] = blocks[3]
49
50
51      data_for_write = []
52      for x in disks:
53          file = open(x, "r")
54          data_for_write.append(file.readlines())
55          file.close()
56
57
58
59      for i in range(len(RAID_5_1)):
60          data_for_write[i][int(input_index)] = l1[i] + '\n'
61
62          data_for_write[i+3][int(input_index)] = l2[i] + '\n'
63
64
65      for i in range(len(disks)):
66          file = open(disks[i], "w")
67          for j in range(len(data_for_write[0])):
68              file.write(data_for_write[i][j])
69          file.close()
70
71
72      print('The data were recorded with an index of {}.'.format(input_index))
```

73

Листинг 3. Функция write

2.4 Функция read

Вход: адрес ячейки чтения.

Выход: значени, зранящиеся по данному адресу.

Для начала проиходит двойная проверка на наличие всех дисков в двух подмассивов. Далее происходит чтение данных по адресу. Елси же есть необходимость в дополнение строки нулями, то происходит дописывание.

```
1 def read() -> None:
2     global disks_indexes
3     if (len(disks_indexes) == 0):
4         print('Disks are empty.')
5         return
6 
7     check_disks()
8     check_disks()
9     while (True):
10        input_data = input("Enter the index of the line you \
11        want to read or enter \"-1\" to leave: ")
12        try:
13            if int(input_data) == -1:
14                return
15            elif int(input_data) > 63:
16                print("Wrong index.")
17            elif int(input_data) not in disks_indexes:
18                print("No data is available for this address.")
19            else:
20                break
21        except:
22            print("Wrong index.")
23    list_of_data = []
24 
25    for i in range(len(disks)):
26        file = open(disks[i], 'r')
27        list_of_data += [file.readlines()]
28        file.close()
29 
30    res = ''
31    for i in range(len(list_of_data)):
32        if (int(input_data) % 3) != i and (int(input_data) % 3) + 3 != i:
33            if (list_of_data[i][int(input_data)] == ""):
34                res += "0000"
35            else:
36                list_of_data[i][int(input_data)] =
```

```
37              list_of_data[i][int(input_data)][:-1]
38                 if len(res) == 0 or len(res) == 7:
39                     while (len(list_of_data[i][int(input_data)]) < 4):
40                         list_of_data[i][int(input_data)] = "0" +
41                  list_of_data[i][int(input_data)]
42                     res += list_of_data[i][int(input_data)]
43                 else:
44                     while (len(list_of_data[i][int(input_data)]) < 3):
45                         list_of_data[i][int(input_data)] = "0" +
46             list_of_data[i][int(input_data)]
47                     res += list_of_data[i][int(input_data)]
48
49     print("Data on the address {}:".format(input_data))
50     print(res)
51
```

Листинг 4. Функция read

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы была реализована модель, демонстрирующая работу избыточного массива независимых дисков RAID 50 на основе 6 дисков для ввода 14 битового шестнадцатиричного значения с подсчетом избыточности через операцию XOR.

Исходя из поставленных целей и задач, были выполнены следующие этапы:

1. Был проведен анализ источников информации, содержащих сведения о избыточного массива независимых дисков. Этот анализ позволил понять основные принципы работы RAID 50.

2. Была разработана и реализована модель, демонстрирующая работу избыточного массива независимых дисков RAID 50 на основе 6 дисков для ввода 14 битового шестнадцатиричного значения с подсчетом избыточности через операцию XOR. Эта модель способна записать пользовательские данные по введенному адресу, читать данные по введенному адресу. При выходе из строя одного из дисков модель способна восстановить утерянные диск.

3. Проведено тестирование полученной модели, что позволило убедиться в корректности ее работы и соответствии поставленным требованиям.

Таким образом, выполнение поставленных целей и задач позволило успешно реализовать модель, которая демонстрирует работу избыточного массива независимых дисков RAID 50 на основе 6 дисков для ввода 14 битового шестнадцатиричного значения с подсчетом избыточности через операцию XOR. Полученные результаты могут быть использованы для дальнейших исследований и проектов.

СПИСОК ИСТОЧНИКОВ

[1] Что такое RAID-массив и зачем он нужен. [Электронный ресурс] URL:https://club.dns-shop.ru/blog/t-419-diskovyie-hranilischa-das/29764-chto-takoe-raid-massiv-i-zachem-on-nujen/?utm_referrer=https%3A%2F%2Fwww.google.com%2F2 (дата обращения: 20.05.2024).

[2] RAID CALCULATOR. [Электронный ресурс] URL: https://www.icc-usa.com/raid-calculator (дата обращения: 20.05.2024).

[3] RAID level summary. [Электронный ресурс] URL: https://www.ibm.com/docs/en/power8?topic=overview-raid-level-summary2 (дата обращения: 20.05.2024).