МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО 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 выполнении работы
по дисциплине «Архитектура суперкомпьютеров»

Разработка приложения, демонстрирующего работу RAID-массива 50 из 6 дисков и заданным размером исходного сообщения – 10 байт.

Обучающийся: _____ Гладков И.А.

Руководитель: _____ Чуватов М.В.

«_____»_____20____г.

Санкт-Петербург, 2024

Содержание

Введение

Цель работы: реализовать программу, демонстрирующую работу RAID-массива из 6 дисков и 10 байтовым размером исходного сообщения.

Условие работы:

1. Подсчет избыточности происходит с помощью операции XOR;
2. Размер входных данных – 10 байт;
3. Количество дисков в массиве - 6 единиц;
4. Количество адрессов в массиве – 64 единицы.

Задачи:

1. Изучить строение дискового массива RAID 50;
2. Изучить организацию записи подсчета избыточности;
3. Реализовать модель, демонстрирующую работу избыточного 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 Математическое описание

1.1 RAID-массив

RAID (Redundant Array of Independent Disks) - это технология, которая используется для объединения нескольких физических дисков в единое логическое устройство с целью повышения производительности, надежности или обоих параметров. Основная идея RAID заключается в том, чтобы скомбинировать несколько дисков таким образом, чтобы данные сохранялись и обрабатывались в более эффективном и надежном формате, чем при использовании отдельных дисков.

Избыточные данные в контексте RAID-массивов относятся к дополнительной информации, которая создается и сохраняется для обеспечения отказоустойчивости и восстановления данных в случае сбоя одного или нескольких дисков. Избыточность позволяет системам продолжать работу и восстанавливать потерянные данные без серьезных потерь или простоев.

1.1.1 Классификация RAID-массивов

RAID-массивы классифицируются по уровню избыточности данных и способу организации дисков. Основные уровни RAID и их характеристики:

RAID 0:

- **Техника хранения данных:** Данные разбиваются на небольшие блоки и записываются на различные диски параллельно, что называется стримингом. Каждый блок данных хранится на отдельном диске.
- **Производительность:** RAID 0 обеспечивает высокую скорость чтения и записи данных за счет распределения данных между дисками. Однако производительность может снизиться при отказе одного из дисков.
- **Избыточность данных:** RAID 0 не предоставляет избыточности данных. Если один из дисков выходит из строя, весь массив становится недоступным, и данные на нем теряются.
- **Применение:** Часто используется в ситуациях, где важна высокая производительность, а сохранность данных не является критической, например, для временных хранилищ или кэширования.

RAID 1:

- **Техника хранения данных:** RAID 1 использует зеркалирование данных. Каждый блок данных записывается на два или более диска, обеспечивая полную копию данных на каждом диске.
- **Производительность:** Чтение данных может быть немного быстрее, так как можно использовать любой из зеркальных дисков. Однако производительность записи обычно ниже, чем у RAID 0.

- **Избыточность данных:** RAID 1 предоставляет полную избыточность данных. Если один из дисков выходит из строя, данные могут быть восстановлены с зеркального диска.
- **Применение:** Используется в приложениях, где важна надежность хранения данных, таких как системы резервного копирования или серверы баз данных.

RAID 5:

- **Техника хранения данных:** RAID 5 использует распределенное хранение четности. Данные разбиваются на блоки, а блоки четности распределяются между дисками.
- **Производительность:** RAID 5 обеспечивает хорошую производительность чтения и некоторое увеличение производительности записи. Он обычно эффективен для случайного доступа к данным.
- **Избыточность данных:** RAID 5 предоставляет избыточность данных. При отказе одного из дисков данные могут быть восстановлены из блоков четности и данных на оставшихся дисках.
- **Применение:** Часто используется в корпоративных серверах, где важны и надежность хранения данных, и производительность.

RAID 6:

- **Техника хранения данных:** RAID 6 аналогичен RAID 5, но использует два блока четности для обеспечения избыточности данных.
- **Производительность:** RAID 6 обеспечивает производительность чтения и записи данных, сравнимую с RAID 5. Однако производительность записи может быть немного медленнее из-за расчетов двух блоков четности.
- **Избыточность данных:** RAID 6 предоставляет двойную избыточность данных. Он может выдержать отказ до двух дисков без потери данных.
- **Применение:** Используется в критически важных средах, где требуется высокий уровень надежности, например, в больших хранилищах данных или при хранении ценной корпоративной информации.

1.2 Комбинированные уровни RAID-массивов

RAID 50:

1. **Исходные диски:** Массив RAID 50 начинается с минимум шести физических дисков, объединяемых в два или более наборов RAID 5.
2. **Создание наборов RAID 5:** Каждый набор RAID 5 содержит минимум три диска и обеспечивает избыточность данных с использованием распределенной четности.
3. **Объединение наборов RAID 5 в RAID 0:** Наборы RAID 5 объединяются в массив RAID 0 путем стриминга данных, что обеспечивает высокую производительность записи и чтения.
4. **Преимущества:**
   - RAID 50 обеспечивает высокую производительность и избыточность данных, объединяя преимущества RAID 0 и RAID 5.
   - Он предоставляет высокую производительность благодаря RAID 0 и защиту данных благодаря RAID 5.
5. **Недостатки:**
   - Создание массива RAID 50 требует минимум шести дисков, что может быть дорого и сложно для некоторых систем.
   - Обновление или замена дисков в массиве RAID 50 может быть сложным из-за его сложной структуры.
6. **Применение:** RAID 50 идеально подходит для сред и приложений, где требуется высокая производительность и избыточность данных, например, для серверов баз данных, виртуализации серверов и видеопотоков.

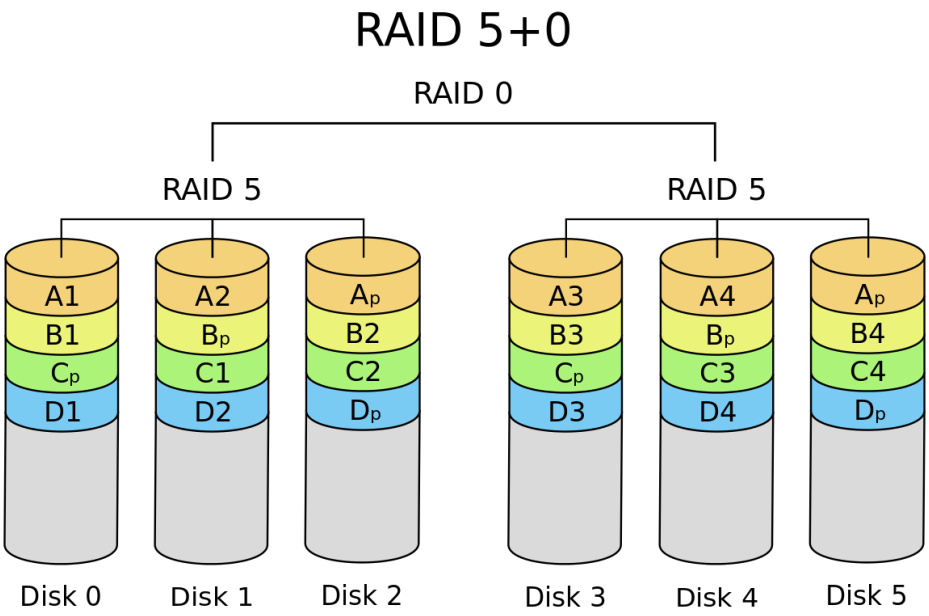

Рис. 1. RAID 50

На рисунке 1 представлена схема RAID 50 для 6 дисков. Данные распределяются на четыре части, оставшиеся две части отведены на избыточность. Рассмотрим первую строчку данных: данные разделяются на четыре блока – A1, A2, A3, A4. Данные блоки разделяются на определенное количество частей, в данном случае, на две. Для первой из них находится блок избыточности с помощью операции XOR – A_p. Данные из первой части записываются на диски, также записывается избыточнось. Аналогично записывается вторая часть данных.

2 Особенности реализации

В качестве дисков приложение использует текстовые файлы (один диск – один файлы, один блок сообщения – одна строка файла). Программа сама обеспечивает присутствие файлов дисков в правильном формате, пересчитывая поврежденные или удаленные файлы дисков, если это возможно,иначе она заново инициализирует все файлы дисков. Каждый диск может хранить до 64 строк данных.

Размещение блоков внутри строки для 6 дисков выглядит следующим образом:

Disk 0	**Disk 1**	**Disk 2**	**Disk 3**	**Disk 4**	**Disk 5**
A_{P1}	A_1	A_2	A_{P2}	A_3	A_4
B_1	B_{P1}	B_2	B_3	B_{P2}	B_4
C_1	C_2	C_{P1}	C_3	C_4	C_{P2}
D_{P1}	D_1	D_2	D_{P2}	D_3	D_4

Сообщение подается в виде 10-байтовой строки. A_i является байтом данного сообщения. Например передается сообщение 12345678, представим по блокам: $A_1 = 123$, $A_2 = 45$, $A_3 = 678$, $A_4 = 90$. После сообщение делится на две части A_1 - A_2 и A_3 - A_4.Каждый блок дополняется до 3 байт незначащими нулями. После для каждой из частей находится блок избыточности по байтово (например для A_1 и A_2 получается A_P1). После все записывается на диски как показано в таблице.

2.1 Реализованные функции

Функция xor

Возвращает результат побитового XOR двух строк.

```
1  def xor(a: str, b: str) -> str:
2    return ''.join([chr(ord(x) ^ ord(y)) for x, y in zip(a, b)])
3
```

Функция recovery

Восстанавливает данные на одном из дисков, используя оставшиеся данные и избыточные данные с других дисков группы RAID 5. Проверяет, какой RAID (RAID_5_1 или RAID_5_2) имеет проблему, считывает данные с остальных дисков и восстанавливает утраченные данные.

```
1  def recovery(disks_index: int) -> None:
2  global index_of_disk
3  global RAID_5_1
4  global RAID_5_2
5
6  if disks_index < 3:
7  problem_raid = RAID_5_1
8  problem_index = disks_index
9  else:
```

```
    problem_raid = RAID_5_2
    problem_index = disks_index - 3

    lost_data = []
    data = []
    for i in range(len(problem_raid)):
    if i != problem_index:
    with open(problem_raid[i], 'r') as file:
    data.append([x for x in file.readlines() if x != '_\n'])
    else:
    data.append(["*" * 3] * 64)

    if data[0][0] != "***":
    cond = data[0]
    else:
    cond = data[1]
    for i in range(len(cond)):
    disk_data = []
    for j in range(len(data)):
    disk_data.append(data[j][i][:3])

    indexes = [ind for ind in range(len(disk_data)) if disk_data[ind] != "***"]

    if len(indexes) == 2:
    a = disk_data[indexes[0]]
    b = disk_data[indexes[1]]
    buf = xor(disk_data[indexes[0]], disk_data[indexes[1]])
    lost_data.append(buf)

    ind = 0
    with open(disks[disks_index], 'w') as file:
    for i in range(64):
    if i in index_of_disk:
    file.write(lost_data[ind] + '\n')
    ind += 1
    else:
    file.write("_\n")

    print("Диск {} был восстановлен.".format(disks_index))
```

Функция check_disks

Проверяет, какие диски отсутствуют. Если отсутствует более одного диска в одной группе RAID, данные восстановить невозможно. В случае, если данные могут быть восстановлены, вызывает функцию **recovery** для отсутствующих дисков.

```
    def check_disks():
    disk_indexes = []
    can_recovery = True
```

```
    for i in range(len(disks)):
    if not os.path.isfile(disks[i]):
    disk_indexes.append(i)

    if len(disk_indexes) == 2:
    if not ((0 <= disk_indexes[0] <= 2 and 3 <= disk_indexes[1] <= 5) or
    (0 <= disk_indexes[1] <= 2 and 3 <= disk_indexes[0] <= 5)):
    can_recovery = False
    elif len(disk_indexes) > 2:
    can_recovery = False

    if can_recovery:
    for disk in disk_indexes:
    print("\nДиск {} отсутствовал.".format(disk))
    recovery(disk)
    else:
    output = "\nДиски "
    for i in range(len(disk_indexes)):
    output += str(disk_indexes[i])
    if i < len(disk_indexes) - 1:
    output += ', '

    output += " отсутсвуют"
    print(output)
    print("\nДиски с данными невозможно востановить")
    return "Данные невозможно востановить"
```

Функция read

Читает данные из строки с указанным индексом. Проверяет наличие дисков и их целостность. Восстанавливает данные, если необходимо, и выводит содержимое строки.

```
    def read() -> None:
    global index_of_disk
    if len(index_of_disk) == 0:
    print('\nДиски пусты.\n')
    return

    if check_disks() != "Данные невозможно востановить":
    while True:
    index = input("\nВведите индекс строки которую хотите прочитать [0;63]: ")
    if int(index) > 63 or int(index) < 0:
    print("индекс вне диапазона [0;63].")
    elif int(index) not in index_of_disk:
    print("Данных нет по данному адресу.")
    else:
    break

    data = []
```

```
18
19  for i in range(len(disks)):
20  file = open(disks[i], 'r')
21  data += [file.readlines()]
22  file.close()
23
24  result = ''
25  index = int(index)
26  for i in range(len(data)):
27  if int(index % 3) != int(i % 3):
28  if len(data[i][int(index)][:-1]) == 3:
29  buf = str(data[i][int(index)][:-1])
30
31  if ''.join(chr(0)) in buf:
32  buf = buf.replace(''.join(chr(0)), '')
33  result += buf
34  else:
35  if i < len(data)//2:
36  indexes = [ind for ind in range(len(data)//2) if ind != int(index % 3) and ind != i]
37  recover_string = data[int(index % 3)][index]
38  else:
39  indexes = [ind for ind in range(len(data)//2, len(data)) if ind != int(index % 3) + 3
      and ind != i]
40  recover_string = data[int(index % 3) + 3][index]
41  result += xor(data[indexes[0]], recover_string)
42
43  print("Данные по адресу {}: ".format(index))
44  print(f"{result}\n")
```

Функция write

Записывает данные в строку с указанным индексом. Разбивает введенные данные на блоки, вычисляет избыточные данные и записывает их на соответствующие диски. Обновляет **index_of_disk** для отслеживания изменений.

```
1   def write() -> None:
2   if check_disks() != "Данные невозможно востановить":
3   global index_of_disk
4
5   while True:
6   index = int(input("\nВведите индекс строки для записи в диапазоне[0;63]: "))
7   if  int(index) > 63 or int(index) < 0:
8   print("введенный индекс вне диапазона [0;63].")
9   else:
10  break
11
12  while True:
13  input_data = str(input("Введите строку (10 байт): "))
14  if len(input_data) != 10:
```

```
15  print("Длина строки должна состоять из 10 байт.")
16  else:
17  break
18
19  blocks = [input_data[:3], input_data[3:5], input_data[5:8], input_data[8:10]]
20  for i in range(len(blocks)):
21  while (len(blocks[i]) < 3):
22  blocks[i] = ''.join(chr(0)) + blocks[i]
23
24  # Обновляем данные для первой и второй группы
25  excess_data1 = xor(blocks[0], blocks[1])
26  excess_data2 = xor(blocks[2], blocks[3])
27
28  index_of_disk.append(index)
29  index_of_disk = list(set(index_of_disk))
30  l1 = ['', '', '']
31  l2 = ['', '', '']
32
33  l1[index % 3] = excess_data1
34  l2[index % 3] = excess_data2
35  indexes = [x for x in range(len(l1)) if x != index % 3]
36
37  l1[indexes[0]] = blocks[0]
38  l1[indexes[1]] = blocks[1]
39
40  l2[indexes[0]] = blocks[2]
41  l2[indexes[1]] = blocks[3]
42
43  result_data = []
44  for x in disks:
45  file = open(x, "r")
46  result_data.append(file.readlines())
47  file.close()
48
49  for i in range(len(RAID_5_1)):
50  result_data[i][index] = l1[i] + '\n'
51  result_data[i+3][index] = l2[i] + '\n'
52
53  for i in range(len(disks)):
54  file = open(disks[i], "w")
55  for j in range(len(result_data[0])):
56  file.write(result_data[i][j])
57  file.close()
58
59  print('Данные записаны в строку под индексом {}.\n'.format(index))
```

Функция files

Инициализирует файлы дисков, если они не существуют, или очищает их, если уже существу-

ют. Записывает 64 строки со знаком "_"в каждый файл.

```
  def files() -> None:
  global disks
  for x in disks:
  if not os.path.exists(x):
  with open(x, "w") as file:
  for i in range(64):
  file.write("_\n")
  else:
  with open(x, "r") as file:
  lines = file.readlines()
  if len(lines) == 0:
  with open(x, "w") as file_append:
  for i in range(64):
  file_append.write("_\n")
  else:
  with open(x, "w") as file:
  for i in range(64):
  file.write("_\n")
```

2.2 Пример записи и чтения данных

В начале всегда прописывается индекс строки, с которой необходимо работать, после пишется либо команда **write** либо команда **read**. После команды **write** необходимо написать строку в шестнадцатеричной системе счисления, состоящую из 12 байт.

Пример:

```
----- Меню действий -----
[1] - Записать данные.
[2] - Прочитать данные.
[0] - Выход.
--------------------------
Выберите действие:
Такой команды нет.
----- Меню действий -----
[1] - Записать данные.
[2] - Прочитать данные.
[0] - Выход.
--------------------------
Выберите действие: 1

Введите индекс строки для записи в диапазоне[0;63]: 0
Введите строку (10 байт): 1234567890
Данные записаны в строку под индексом 0.

----- Меню действий -----
[1] - Записать данные.
```

```
[2] - Прочитать данные.
[0] - Выход.
--------------------------
Выберите действие: 2

Введите индекс строки которую хотите прочитать [0;63]: 0
Данные по адресу 0:
1234567890
```

Заключение

В результате работы была реализована модель, демонстрирующая работу избыточного массива независимых дисков RAID 50 на основе 6 дисков для ввода 10-байтовой строки с подсчетом избыточности через операцию XOR. Полученная модель позволяет записывать, читать данные и восстанавливать один из дисков в случае его выхода из строя.

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

1. был произведен анализ источников информации, содержащие сведения об избыточного массива независимых дисков. Этот анализ позволил понять основные принципы работы RAID 50.

2. Была реализована модель, демонстрирующая работу избыточного массива независимых дисков RAID 50 на основе 6 дисков для ввода 10-байтовой строки с подсчетом избыточности через операцию XOR. Полученная модель позволяет записывать, читать данные и восстанавливать один из дисков в случае его выхода из строя.

3. Произведено тестирование реализованной модели, что позволило убедиться в корректности реализации программы.

Источники

1. RAID 50 (RAID 5+0) [электронный ресурс] URL:
https://www.techtarget.com/searchstorage/definition/RAID-50-RAID-50
(Дата обращения 24.05.2024)
2. RAID — Википедия [электронный ресурс] URL:
https://ru.wikipedia.org/wiki/RAID
(Дата обращения 24.05.2024)
3. RAID Calculator [электронный ресурс] URL:
https://www.raid-calculator.com/default.aspx
(Дата обращения 24.05.2024)