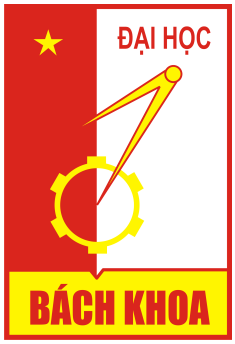
ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG
CÔNG NGHỆ THÔNG TIN VIỆT PHÁP



PROJECT 1

Truy xuất dữ liệu lớn với bộ nhớ nhỏ

GVHD: Nguyễn Đức Anh
SV thực hiện: Nguyễn Hữu Mạnh – 20205213

Hà Nội, Tháng 1/2024

Contents

1 Problem Statement

Tạo một chương trình truy xuất dữ liệu lớn trên bộ nhớ kích thước nhỏ:

1. Tạo dữ liệu nhân tạo bằng cách: Tạo 3,000,000 (3 triệu) mảng interger, kích thước mỗi mảng là một giá trị nguyên ngẫu nhiên từ 100 tới 10000. Lưu trữ tuỳ ý (Gợi ý, lưu trữ qua file).
2. Tạo một hàm truy xuất ngẫu hiên tới mảng thứ i trong 3 triệu mảng trên (Gợi ý, cần lưu thêm giá trị bổ trợ để truy xuất tốt hơn).

Giới hạn bộ nhớ RAM: Dưới 500 MB.

2 Design

Lưu trữ

- Lưu dữ liệu vào file ở dạng nhị phân để bảo toàn dữ liệu.
- Lưu vị trí của mỗi mảng vào 1 file riêng để truy xuất.

Truy xuất

- Sử dụng file lưu vị trí để truy cập đến mảng nhanh.
- In ra mảng đó.

`Để có thể giới hạn bộ nhớ, ta sử dụng java -Xmx500m`

3 Implementation

3.1 Generate Arrays

1. Tạo biến lưu tổng số `array` và lưu giới hạn `elements` trong `array`

```
    int totalArr = 3_000_000;
    int[] rangeArr = new int[] {100, 10_000};
```

2. Thực hiện ghi vào 2 file **data.dat** và **pos.dat**
   File **data.dat** lưu dữ liệu ở dạng nhị phân
   File **pos.dat** vị trí của từng `array` ở dạng nhị phân

```
    try (BufferedOutputStream binout =
            // save data
            new BufferedOutputStream(new FileOutputStream("../bin/data.dat"));
        BufferedOutputStream posout =
            // save position
            new BufferedOutputStream(new FileOutputStream("../bin/pos.dat"))) {
```

   Khai báo 1 biến `pos` để lưu vị trí của từng `array`
   Ta cần biến `pos` là `long` vì mỗi `array` có 100 đến 10000 `elements` nên 3 triệu mảng sẽ có tổng số `elements` $>$ max value `int` có thể lưu là 2^{31}

```
      long pos = 0; // long because total element in 3M > int = 2^31.
```

   Lặp qua từng `array`

```
      for (int i = 1; i <= totalArr; i++) {
```

   Khai báo biến `size` là số `elements` của `array i`. Biến `size` random trong khoảng `rangeArr`

```
        int size = random.nextInt(rangeArr[1] - rangeArr[0] + 1) + rangeArr[0];
```

   Chuyển value `pos` sang `byte` và ghi vào file **pos.dat**

```
        byte[] posBytes = toBytes(pos, 8);
        posout.write(posBytes);
        pos += size;
```

Lặp qua từng `element` trong `array`, lấy giá trị của mỗi `element` là random 1 số `int`
Chuyển value `element` sang `byte` và ghi vào file **data.dat**

```
        for (int j = 0; j < size; j++) {
          int element = random.nextInt();
          byte[] elementBytes = toBytes(element, 4);
          binout.write(elementBytes);
        }
```

Sau khi ghi xong ta sẽ lưu thêm 1 value `pos` cuối cùng nữa để tính được số `element` của `array` cuối cùng

```
        if (i == totalArr) {
          byte[] lastPosBytes = toBytes(pos, 8);
          posout.write(lastPosBytes);
        }
```

3. Hàm chuyển `int` hoặc `long` sang `byte`

```
 private static byte[] toBytes(long value, int a) {
   byte[] result = new byte[a];
   for (int i = 0; i <= (a - 1); i++) {
     result[i] = (byte) (value >> (8 * (a - 1 - i)));
   }
   return result;
 }
```

Code hoàn chỉnh

```
import java.io.BufferedOutputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Random;

public class GenerateArrays {

  public static void main(String[] args) {

    int totalArr = 3_000_000;
    int[] rangeArr = new int[] {100, 10_000};

    try (BufferedOutputStream binout =
            // save data
            new BufferedOutputStream(new FileOutputStream("../bin/data.dat"));
        BufferedOutputStream posout =
            // save position
            new BufferedOutputStream(new FileOutputStream("../bin/pos.dat"))) {
      long pos = 0; // long because total element in 3M > int = 2^31.

      for (int i = 1; i <= totalArr; i++) {
        int size = random.nextInt(rangeArr[1] - rangeArr[0] + 1) + rangeArr[0];

        byte[] posBytes = toBytes(pos, 8);
        posout.write(posBytes);
        pos += size;

        for (int j = 0; j < size; j++) {
          int element = random.nextInt();
          byte[] elementBytes = toBytes(element, 4);
          binout.write(elementBytes);
        }

        if (i == totalArr) {
          byte[] lastPosBytes = toBytes(pos, 8);
          posout.write(lastPosBytes);
        }

        System.out.println("Array: " + i);
      }
    } catch (IOException e) {
      System.err.println("Error write file: " + e.getMessage());
```

```
    }
  }

  private static final Random random = new Random();

  // (int || long ) -> byte (a = total byte)
  private static byte[] toBytes(long value, int a) {
    byte[] result = new byte[a];
    for (int i = 0; i <= (a - 1); i++) {
      result[i] = (byte) (value >> (8 * (a - 1 - i)));
    }
    return result;
  }
}
```

3.2 Access Arrays

1. Nhập vào thứ tự của `array` muốn truy xuất

```
    Scanner sc = new Scanner(System.in);
    System.out.print("Input index: ");
    int index = sc.nextInt();
    sc.close();
```

2. Thực hiện truy xuất `array`

```
    getArray(index);
```

3. Hàm truy xuất `getArray`
   Đọc dữ liệu từ 2 file **data.dat** và **pos.dat**
   Ta sử dụng `RandomAccessFile` để có thể truy nhập file bằng `byte`.

```
  public static void getArray(int index) throws IOException {
    try (RandomAccessFile pos = new RandomAccessFile("../bin/pos.dat", "r");
        RandomAccessFile dat = new RandomAccessFile("../bin/data.dat", "r")) {
```

   Thực hiện lấy vị trí của `array` cần truy xuất
   Dùng `seek` để có thể truy nhập file vào đúng vị trí cần truy nhập

```
      index--;
      pos.seek(index * 8); // pos.dat luu du lieu long
```

   Đọc vị trí và kích thước của `array`

```
      long posData = pos.readLong();
      long size = pos.readLong() - posData;
```

   Truy nhập vào vị trí của `array` cần truy xuất

```
      dat.seek(posData * 4); // data.dat luu du lieu int
```

   Dùng vòng lặp `for` để đọc từng `element` của `array` cần truy xuất và in ra

```
      for (long i = 1; i <= size; i++) {
        int element = dat.readInt();
        System.out.print(element + " ");
      }
```

Code hoàn chỉnh

```
import java.io.IOException;
import java.io.RandomAccessFile;
import java.util.Scanner;

public class AccessArrays {

  public static void main(String[] args) throws IOException {
    Scanner sc = new Scanner(System.in);
    System.out.print("Input index: ");
```

```
    int index = sc.nextInt();
    sc.close();

    getArray(index);
  }

  public static void getArray(int index) throws IOException {
    try (RandomAccessFile pos = new RandomAccessFile("../bin/pos.dat", "r");
        RandomAccessFile dat = new RandomAccessFile("../bin/data.dat", "r")) {

      index--;
      pos.seek(index * 8); // pos.dat luu du lieu long
      long posData = pos.readLong();
      long size = pos.readLong() - posData;

      // System.out.print(posData + " " + size + "\n");
      dat.seek(posData * 4); // data.dat luu du lieu int
      for (long i = 1; i <= size; i++) {
        int element = dat.readInt();
        System.out.print(element + " ");
      }
    }
  }
}
```

3.3 Code

Souce code toàn bộ của project[1]

4 Testing

4.1 Compile Code

Thư mục **src/** chứa source code, thư mục **bin/** chứa các file `.class` và các file dữ liệu.

Giả sử ta đang ở thư mục cha của thu mục **src/** và thư mục **bin/**

```
javac -d ./bin src/*.java
```

Vào thư mục `bin`

```
cd ./bin
```

4.2 Generate Arrays

```
java GenerateArrays
```

Sau khi chạy ta được 2 file **data.dat** và **pos.dat** được lưu ở cùng thư mục `bin`

4.3 Access Arrays

Chạy AccessArrays với flag `-Xmx500m` để giới hạn bộ nhớ còn $500Mb$

```
java -Xmx500m AccessArrays
Input index: // nhập 1 số i bất kì
```

> In ra console `array i` cần truy xuất

Thời gian truy xuất với 3 triệu mảng, mỗi mảng từ 100 đến 10000 nghìn được thực hiện chỉ tính bằng mili giây, ta có thể kiểm tra bằng

```
time java -Xmx500m AccessArrays
```

lệnh này sẽ in ra thời gian thực hiện câu lệnh.

4.4 Test

Vì số lượng `array + element` quá lớn nên không thể kiểm tra được truy xuất có chính xác không nên ta sẽ tạo 1 file **test.java** với số `array` 100000, mỗi array từ 10-20 `element` để có thể kiểm tra việc truy xuất có đúng không

```
    int totalArr = 100000;
    int[] rangeArr = new int[] {10, 20};
```

Ghi dữ liệu dạng text vào file **data.txt**

```
    try (BufferedWriter writer = new BufferedWriter(new FileWriter("../bin/data.txt"));
```

Trong vòng lặp `for` khi thực hiện random `element` và ghi vào **data.dat** ta cũng sẽ lưu dữ liệu vào **data.txt**, mỗi `array` sẽ được lưu vào 1 dòng

```
        for (int j = 0; j < size; j++) {
          int element = random.nextInt();
          byte[] elementBytes = toBytes(element, 4);
          binout.write(elementBytes);

          writer.write(String.valueOf(element) + " ");
        }
        writer.newLine();
```

Code hoàn chỉnh

```
import java.io.BufferedOutputStream;
import java.io.BufferedWriter;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Random;

public class test {

  public static void main(String[] args) {

    int totalArr = 100000;
    int[] rangeArr = new int[] {10, 20};

    try (BufferedWriter writer = new BufferedWriter(new FileWriter("../bin/data.txt"));
        BufferedOutputStream binout =
            new BufferedOutputStream(new FileOutputStream("../bin/data.dat"));
        BufferedOutputStream posout =
            new BufferedOutputStream(new FileOutputStream("../bin/pos.dat"))) {
      long pos = 0; // long because total element in 3M > int = 2^31.

      for (int i = 1; i <= totalArr; i++) {
        int size = random.nextInt(rangeArr[1] - rangeArr[0] + 1) + rangeArr[0];

        // writer.write(String.valueOf(i) + " ");

        byte[] posBytes = toBytes(pos, 8);
        posout.write(posBytes);
        pos += size;

        for (int j = 0; j < size; j++) {
          int element = random.nextInt();
          byte[] elementBytes = toBytes(element, 4);
          binout.write(elementBytes);

          writer.write(String.valueOf(element) + " ");
        }
        writer.newLine();

        if (i == totalArr) {
```

```
          byte[] lastPosBytes = toBytes(pos, 8);
          posout.write(lastPosBytes);
        }
      }
      System.out.println("Done");
    } catch (IOException e) {
      System.err.println("Error write file: " + e.getMessage());
    }
  }

  private static final Random random = new Random();

  // (int || long ) -> byte (a = total byte)
  private static byte[] toBytes(long value, int a) {
    byte[] result = new byte[a];
    for (int i = 0; i <= (a - 1); i++) {
      result[i] = (byte) (value >> (8 * (a - 1 - i)));
    }
    return result;
  }
}
```

Biên dịch tương tự giống như trên

Tạo dữ liệu nhỏ hơn với 100000 `arrays`, mỗi `array` từ 10->20 `element`

```
java test
```

Sau khi chạy ta được 3 file **data.dat**, **pos.dat** và **data.txt** lưu dữ liệu ở dạng văn bản có thể đọc, mỗi mảng được lưu 1 dòng cách nhau bởi " ". Được lưu ở cùng thư mục `bin`

Truy xuất thử 1 `array` bất kì, ví dụ với `array` thứ `100000`

```
java -Xmx500m AccessArrays
Input index: 100000
```

> In ra console `array 100000` cần truy xuất

Để kiểm tra truy xuất có chính xác không ta kiểm tra file **data.txt**

Ta sử dụng command `` `sed' `` để in ra màn hình `array 100000`

```
sed -n '100000p' data.txt
```

Lệnh này sẽ in ra dòng thứ `100000` trong file **data.txt**

> Đối chiếu kết quả của 2 lệnh thấy giống nhau => truy xuất chính xác.

5 Conclude

Qua bài trên em đã học được cách tổ chức lưu trữ 1 dữ liệu lớn để có thể tối ưu trong việc truy xuất.

Cách để có thể truy xuất vào 1 vị trí bất kì trong 1 file dữ liệu lớn trong thời gian ngắn gần như bằng `0` với bộ nhớ cực kì nhỏ.