Assignment 2 - Databases

Frederik Lassen

2021
April

Contents

1 (Q)Task 1 - Investigation

Produce a small writeup (around 300 words) answering the following questions.

1. What is point of NoSQL databases?
2. What is the CAP theorem?
3. What are ideal use cases of HBase?

2 (A)Task 1 - Investigation

NoSQL databases was created because programmers weren't content with SQL. NoSQL is easier scalable, easy datamodelling and the highest of uptimes. NoSQL stores its data in JSON formatted documents, and doesn't use the conventional columns and rows that SQL uses. But of course NoSQL isn't totally perfect, everything has its flaws. If you want the reliability of a relational database in your NoSQL database, it could be a very complex procedure. NoSQL is also not compatible with SQL, and also generally NoSQl is newer and as such, haven't been as developed to the same extent relational databases have. This leads us to the CAP theorem which is an idea from a computer scientist, Eric Brewer, that explains that in computer systems you can't satisfy all guarantees, those being availability, consistency and partition tolerance. This is explained in an example by the "two out of three" rule, whereas you want availability, consistency and partition tolerance but you can only have 2 of the 3 guarantees. Hbase is a NoSQL database that can store large qualities of sparse data, which is basically a very small amount of information kept in a very large set of not important data. Hbase is best used as a database where very large amounts of data is added consistently and needs to be handled without the database deteriorating. Some examples could be; Hbase is used actively for CDP by many companies and by banking institutions for checking for fraud in transactions. It is also used for long term storage, as well as hen data is generally too large to comfortably scale using traditional technologies.

3 (Q)Task 2 - Bloom filters

Bloom filters are used in hbase as an incredible optimization. Solve the following.

1. What is a bloom filter?
2. What is an advantage of bloom filters over hash tables?
3. What is a disadvantage of bloom filters?
4. Using your language of choice, implement a bloom filter with add and check functions. The backing bit-array can simply be a long (64 bit integer).
5. If you are to store one million ASCII strings with an average size of 10 characters in a hash set, what would be the approximate space consumption?
6. The following equation gives the required number of bits of space per inserted key, where E is the false positive rate. b = 1.44log2(1/E) (1)
7. How many bits per element are required for a 1% false positive rate?
8. How many bits per element are required for a 5% false positive rate?
9. If you are to store one million ASCII strings with an average size of 10 characters in a bloom filter, what would be the approximate space consumption, given an allowed false positive rate of 5% ?.

4 (A)Task 2 - Bloom filters

1. A Bloom filters checks whether an element is a member of a specific set. So it will either return a possible in set or a definitely not in set. It is probabilistic in nature.
2. Bloom filters dont store elements themselves. You dont test whether something is present, but rather if it is certainly not present. Bloom filters are mroe space-efficient than hash-tables: bloom filter uses many hash functions.
3. false negative potential. It can only say yes/no. No memory.

4. implement a bloom filter NOT DONE!

5. Assuming 1 string would be 56 bytes, the space consumption would be 56 000 000 bytes. which is roughly 53MB.

6. Not a question

7. Fewer than 10 bits per element is required for a 1% false positive probability(not rate).

8. Using a calculator assuming 1000000 elements it would be around 19 bits per element for a 5% probability for a false positive(1 hash function)

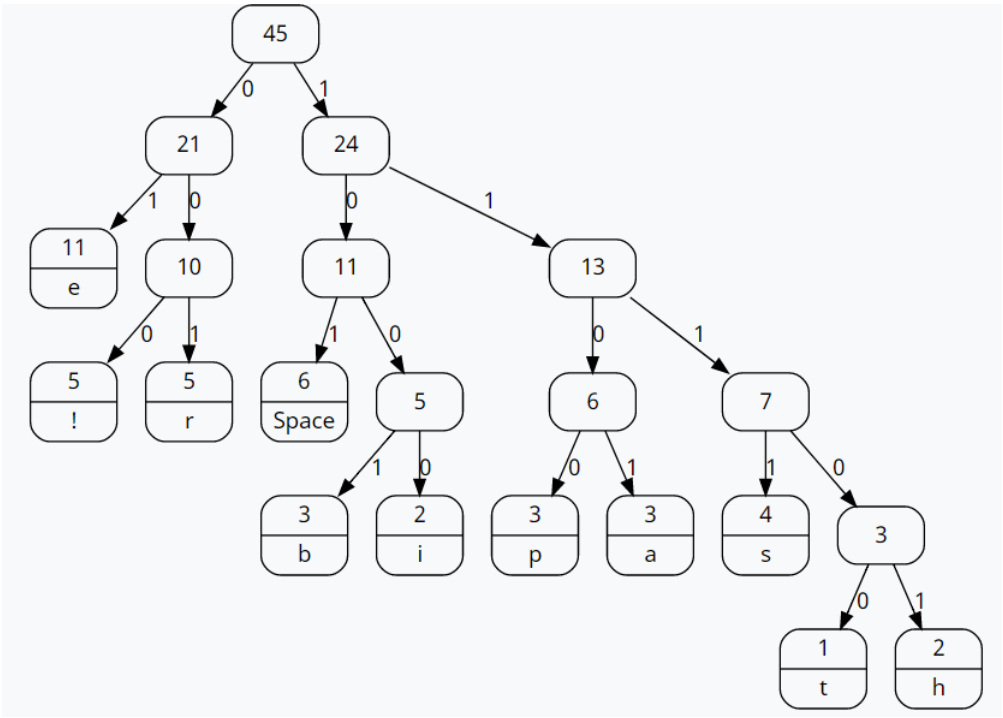
9. 1MB (har brugt denne til 3 af opgaverne: https://hur.st/bloomfilter/)

5 (Q)Task 3 - Huffman coding

HBase internally uses a compression that is a combination of LZ77 and Huffman Coding.

1. Generate Huffmann Code (and draw the Huffmann Tree) based on the following string: "beebs beepps!!!!! their eerie ears hear pears"
2. How many bits is the compressed string? How many bits is the raw ASCII string?
3. Compress "pete is here" with the Huffmann tree from before.
4. Write your own 10 word sentence. Generate the Huffmann Code (a new Huffmann Tree), and write a new compressed message (ie. in binary). Swap with one of your fellow students, and decompress each other's message.

6 (A)Task 3 - Huffman coding

1. 

2. There are:

e = 01, ! = 000 ,r = 001 , _ = 101, b =1001, i = 1000,
p = 1100, a = 1101, s = 1111, t = 11100, h = 11101
So compressed it uses 145 bits.
As a raw Ascii string it uses 352 bits.

3. pete is here =
   1100 | 01 | 11100 | 01 | 101 | 1000 |
   1111 | 101 | 11101 | 01 | 001 | 01

   Compressed 39 bits
   Raw ASCII 96 bits

7 (Q)Task 4 - Map and Reduce

Solve the following using Javascript, for example in your browser's developer console.

1. Map the list of numbers to a list of their square roots: [1, 9, 16, 100]
2. Map the list of words so each is wrapped in a ¡h1¿ tag: [“Intro”, “Requirements”, “Analysis”, “Implementation”, “Conclusion”, “Discussion”, “References”]
3. Use map to uppercase the words (all letters): [“i’m”, “yelling”, “today”]
4. Use map to transform words into their lengths: [“I”, “have”, “looooooong”, “words”]
5. Get the json file comics.json from the course site. Paste it into your browser's Javascript console. Use map to get all the image urls, and wrap them in img-tags.
6. Use reduce to sum the array of numbers: [1,2,3,4,5]
7. Use reduce to sum the x-value of the objects in the array: [x: 1,x: 2,x: 3]
8. Use reduce to flatten an array of arrays: [[1,2],[3,4],[5,6]]
9. Use reduce to return an array of the positive numbers: [-3, -1, 2, 4, 5]
10. Optional: The accumulator function can obviously use objects outside of itself. Use reduce to implement groupBy. For example: p e o pl e = [name : ’ Rikke ’ , age : 4 6 , name : ’ Michael ’ , age : 4 7 , name : ’ Mathias ’ , age : 46] ; should be turned into g roupedPeople = groupBy (pe ople , ’ age ’) ; /* g roupPe ople : 4 6: [name : ’ Rikke ’ , age : 4 6 , name : ’ Mathias ’ , age : 4 6] , 4 7: [name : ’ Michael ’ , age : 4 7] */

8 (A)Task 4 - Map and Reduce

1. 1

```
>let arr = [1,9,16,100]
>let result = arr.map(Math.sqrt)
>result [1, 3, 4, 10]
```

2. 2

```
>let arr = ["Intro","Requirements", "Analysis","Implementation",
"Conclusion","Discussion","References"]
>let h1 = arr.map(function(word){return "<h1>" + word +
"<h1>"})
>h1 ["<h1>Intro<h1>", "<h1>Requirements<h1>",
 "<h1>Analysis<h1>", "<h1>Implementation<h1>",
 "<h1>Conclusion<h1>", "<h1>Discussion<h1>",
 "<h1>References<h1>"]
```

3. 3

```
>let arr = ["i'm", "yelling", "today"]
>let up = arr.map(function(word){return word.toUpperCase()})
>up ["I'M", "YELLING", "TODAY"]
```

4. 4

```
>let arr = ["I","have","looooooong","words"]
>let length = arr.map(function(word){return word.length})
>length [1, 4, 10, 5]
```

5. 5

Couldn't find the file on the course page.

6. 6

```
>let arr = [1,2,3,4,5]
>let sum = arr.reduce(function(summed,current){
return summed + current}
>sum 15
```

7. 7

```
>let arr = [{x:1},{x:2},{x:3}]
>let sumobj = arr.reduce(function(summed,current){
return {x: summed.x + current.x}})
>sumobj {x: 6}
```

8. 8

```
>let arr = [[1,2],[3,4],[5,6]]
>let flat = arr.reduce(function(a,b){return a.concat(b)})
>flat [1, 2, 3, 4, 5, 6]
```

9. 9 Er det meningen denne opgave skal løses med reduce og ikke filter? Er også usikker på om svaret skal være [1,2,3,4,5] eller [2,4,5]. Opgaven kan forstås på begge måder.

```
>let arr =  [-3,-1,2,4,5]
>let posa = arr.filter(function(val){return val > 0})
posa [2, 4, 5]
```