CT-LSD A5

Albin Jansfelt, Amanda Allgurén, Lukas Martinsson

May 2022

1 Problem 2

a) Discuss the computational efficiency of the MapReduce code for detecting duplicates from the lecture on this data. Assume the data set is distributed across 16 nodes

As we can see in the implementation from the lecture of detecting duplicates with MapReduce, we use a combiner to partially sum counts of a specific item. Then, in the reducer we sum all partial sums of item counts and yield every item that has frequency greater than 1.

If we use 16 nodes then we will distribute the data in the mapper and combiner into 16 parts. If we have one billion data this would mean that each node have approximately 62500000 items to work with. We will therefore speed up the process by 16 compared to with no parallelization.

However, since the number of unique items are very high, MapReduce might not be the most efficient approach to solve this problem. Very few items will be mapped to the same key and thereby reduce the number of operations much. The mapper will need to do as many mappings as the number of items (1 B), as for the combiner and reducer the number of calculations will be as many as unique keys existing. For MapReduce to be suitable, the number of keys should be greater than one to make any difference and to split up operations on different nodes but also not too many for each item to not be handled seperately. When the number of keys gets close to the number of items, this approach is therefore not suitable. In this case, the combiner may not add much to improve the efficiency of the operations since the number of unique keys are very many.

b) Sketch a serial solution using Bloom filters to accelerate this counting; please determine k, the number of hash functions, and b the size of Bloom filters. Choose error rates to obtain accurate counts.

We can calculate optimal $|b|$ and k using the following formulas:

$$|b| = -\frac{n * ln(p)}{(ln2)^2} \tag{1}$$

$$k = -\frac{ln(p)}{ln(2)} \tag{2}$$

Where p is the chosen error rate. For different p we thus get the parameter values as

p	b	k
0.1	4 792 529 189	3
0.01	9 585 058 378	7
0.001	14 377 587 567	10
0.0001	19 170 116 755	13
0.00001	23 962 645 944	17

Table 1: b and k values given different error rates p

The idea to solve this problem is to loop though all elements in the collection of items. Checking if this item has already been encountered is done using a Bloom filter (using the hashed values to check if bits are selected). The idea is to take use of serial Bloom filters and thereby add the item to a new Bloom filter if encountered in the previous, and by this way keep track of the counts and easily look up elements.

If an item is already in the first Bloom filter (f1), it is added to the next one (f2) which is indicating a count of two. If it is already in (f2) it is added to (f3), following this pattern up to (f4).

If an item has not yet been encountered by a Bloom filter, the item should be added to this specific Bloom filter. Since we are only interested in the frequency of items appearing four or more times, a counter can be incremented by one each time it is added to (f4). This counter indicates the number of items appearing four or more times.

$X \leftarrow$ items
$f_i \leftarrow$ Bloom Filter i with error-rate p, $i \in \{1, 2, 3, 4\}$
$c \leftarrow$ counts items appearing 4 or more times
for $x \in X$ **do**
  if x **not in** f_1 **then**
    f_1.add(x)
  else if x **not in** f_2 **then**
    f_2.add(x)
  else if x **not in** f_3 **then**
    f_3.add(x)
  else if x **not in** f_4 **then**
    f_4.add(x)
    c++
  end if
end for

2 Problem 3

2.1 (a) Probability of not causing false negative during query if deleting

A false negative result occurs if two items has been hashed to the same bit. We have:

$$P(\textit{hitting one of the hashed bits}) = \frac{k}{b} \tag{3}$$

$$P(\textit{avoiding all of the hashed bits}) = 1 - \frac{k}{b} \tag{4}$$

$$P(\textit{avoiding all bits when inserting one items}) = (1 - \frac{k}{b})^k \tag{5}$$

$$P(\textit{avoiding all bits when inserting n items}) = (1 - \frac{k}{b})^{kn} \tag{6}$$

Thus the probability of not causing a false negative during query when deleting an item is the last equation above.

2.2 (b)

Let $A_i = 1$ if item $i \in \{1, \ldots, n\}$ is affected by the deletion. From (a) we got the result that

$$P(\textit{avoiding all bits when inserting one items}) = (1 - \frac{k}{b})^k$$

And thus

$$\begin{aligned} P(A_i = 1) &= P(\textit{hitting at least one bit when inserting one item}) = \\ &= 1 - (1 - \frac{k}{b})^k \end{aligned}$$

Let the indication function $I_{A_i} = 1$ with probability $P(A_i = 1)$ as above, 0 otherwise. Then $E(I_{A_i}) = P(A_i = 1)$. Further, let

$$I_A = \sum_{i=1}^{n} I_{A_i}$$

then by the linearity of expectation we have

$$E(I_A) = E(\sum_{i=1}^{n} I_{A_i}) = \sum_{i=1}^{n} E(I_{A_i}) = \sum_{i=1}^{n} P(A_i = 1) = nP(A_i = 1) = n(1-(1-\frac{k}{b})^k)$$

So the expected number of items in B affected by one deletion will be
$n(1 - (1 - \frac{k}{b})^k)$