CS 61C Spring 2023 — WSC, RAID, Parity & ECC

Discussion 13

1 Precheck

This section is designed as a conceptual check for you to determine if you conceptually understand and have any misconceptions about this topic. Please answer true/false to the following questions, and include an explanation:

1.1 MapReduce is a more general programming model than Spark since it is lower level.

1.2 If a data center has a higher PUE, then it is more energy-efficient.

1.3 We can improve the availability of a service either by increasing MTTF or decreasing MTTR.

1.4 Hamming codes can detect any type of data corruption.

1.5 All RAID levels improve reliability.

2 MapReduce

For each problem below, write pseudocode to complete the implementations using the MapReduce model. Tips:

- The input to each MapReduce job is given by the signature of `map()`.
- `emit(key k, value v)` outputs the key-value pair `(k, v)`.
- `for var in list` can be used to iterate through `Iterables` or you can call the `hasNext()` and `next()` functions.
- Usable data types: **`int`**, **`float`**, `String`. You may also use lists and custom data types composed of the aforementioned types.
- `intersection(list1, list2)` returns a list of the common elements of `list1`, `list2`.

2.1 Given the student's name and course taken, output their name and total GPA.

Declare any custom data types here:

```
CourseData:
    int courseID
    float studentGrade // a number from 0-4
```

```
map(________________, ________________):
```

```
reduce(________________, ________________):
```

2.2 You are given a list of tuples containing people's unique int ID and a list of the IDs of their friends (i.e. each tuple in the list gives the list of friends for *different* person). Compute the list of mutual friends between each pair of friends in a social network, including the pair themselves. You have access to the `intersection` function, which takes in two lists and finds the set of elements that appear in both lists.

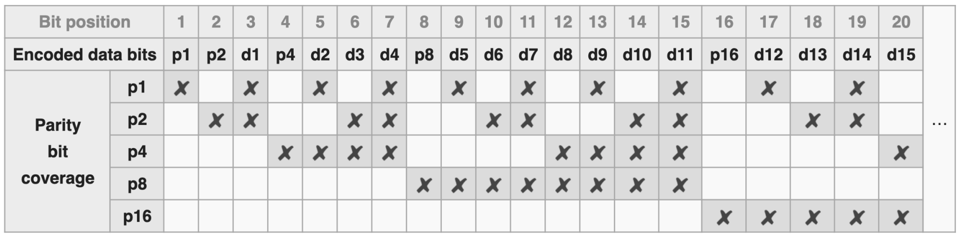
```
FriendPair:
    int friendOne
    int friendTwo
```

```
map(tuple<int, list<int>> info):
```

```
reduce(________________, ________________):
```

3 Hamming ECC

Recall the basic structure of a Hamming code. We start out with some bitstring, and then add parity bits at the indices that are powers of two (1, 2, 4, etc.). We don't assign values to these parity bits yet. **Note that the indexing convention used for Hamming ECC is different from what you are familiar with.** In particular, the 1 index represents the MSB, and we index from left-to-right. The ith parity bit $P\{i\}$ covers the bits in the new bitstring where the *index* of the bit under the aforementioned convention, j, has a 1 at the same position as i when represented as binary. For instance, 4 is `0b100` in binary. The integers j that have a 1 in the same position when represented in binary are 4, 5, 6, 7, 12, 13, etc. Therefore, $P4$ covers the bits at indices 4, 5, 6, 7, 12, 13, etc. A visual representation of this is:



Bit position		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
Encoded data bits		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	d11	p16	d12	d13	d14	d15	
Parity bit coverage	p1	✗		✗		✗		✗		✗		✗		✗		✗		✗		✗		
	p2		✗	✗			✗	✗			✗	✗			✗	✗			✗	✗		...
	p4				✗	✗	✗	✗					✗	✗	✗	✗					✗	
	p8								✗	✗	✗	✗	✗	✗	✗	✗						
	p16																✗	✗	✗	✗	✗	

Source: **`https://en.wikipedia.org/wiki/Hamming_code`**

3.1 How many bits do we need to add to 0011_2 to allow single error correction?

3.2 Which locations in 0011_2 would parity bits be included?

3.3 Which bits does each parity bit cover in 0011_2?

3.4 Write the completed coded representation for 0011_2 to enable single error correction. Assume that we set the parity bits so that the bits they cover have even parity.

3.5 How can we enable an additional double error detection on top of this?

3.6 Find the original bits given the following SEC Hamming Code: 0110111_2.

3.7 Find the original bits given the following SEC Hamming Code: 1000100_2.

4 RAID

4.1 Fill out the following table on how each RAID level lays out data for redundancy, as well as their pros and cons:

	Configuration	**Pro/Good for**	**Con/Bad for**
RAID 0			
RAID 1			
RAID 4			
RAID 5			
RAID 6			

5 Warehouse Scale Computing

Sources speculate Google has over 1 million servers. Assume it has exactly 1 million servers, which draw an average of 200W each, the PUE is 1.5, and that Google pays an average of 6 cents per kilowatt-hour for data center electricity.

5.1 Estimate Google's annual power bill for its data centers.

5.2 Google reduced the PUE of a 50,000-machine data center from 1.5 to 1.25 without decreasing the power supplied to the servers. What's the cost savings per year?