# 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.

False. While Spark is higher level, you can still do basic map-reduce with Spark. It is easier to express more complex computations in Spark. For more information on higher level vs. lower level, visit https://en.wikipedia.org/wiki/High-_and_low-level

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

False. The ideal and minimum value of PUE is 1.0.

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

True. Increasing MTTF makes the system fail less often, while lowering MTTR makes the downtime shorter if it does fail, both improving availability.

1.4 Hamming codes can detect any type of data corruption.

False. They cannot detect all three bit errors.

1.5 All RAID levels improve reliability.

False. Raid 0 is just storing data across all disks, which doesn't 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
```

```
1 map(_________________, _________________):
```

```
map(String student, CourseData value):
    emit(student, value.studentGrade)
```

```
1 reduce(_________________, _________________):
```

```
reduce(String key, Iterable<float> values):
    totalPts = 0
    totalClasses = 0
    for grade in values:
        totalPts += grade
        totalClasses += 1
    emit(key, totalPts / totalClasses)
```

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):
```

```
map(tuple<int, list<int>> info):
    personID, friendIDs = info
    for fID in friendIDs:
        if (personID < fID):
            friendPair = (personID, fID)
        else:
            friendPair = (fID, personID)
        emit(friendPair, friendIDs)
```

```
reduce(_________________, _________________):
```

```
reduce(FriendPair key,Iterable<list<int>> values):
    # Note: values only has two elements,
    # once for each person in the FriendPair.
    mutualFriends = intersection(
        values[0], values[1]
    )
    emit(key, mutualFriends)
```

# 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 $i$th 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?

$m$ parity bits can cover bits 1 through $2^m - 1$, of which $2^m - m - 1$ are data bits. Thus, to cover 4 data bits, we need 3 parity bits.

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

Using P to represent parity bits: $PP0P011_2$

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

Parity bit 1: 1, 3, 5, 7
Parity bit 2: 2, 3, 6, 7
Parity bit 3: 4, 5, 6, 7

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.

$1000011_2$

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

Add an additional parity bit over the entire sequence. If after correcting the message using the original Hamming code, we have a message whose total parity doesn't match the additional parity bit, then we have detected a double error.

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

Parity group 1: error
Parity group 2: okay
Parity group 4: error
To find the incorrect bit's index, we simply sum up the indices of all the erroneous bits.
Incorrect bit: $1 + 4 = 5$, change bit 5 from 1 to 0: $0110011_2$
$0110011_2 \rightarrow 1011_2$

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

Parity group 1: okay
Parity group 2: okay
Parity group 4: error
Incorrect bit: 4, change bit 4 from 0 to 1: $1001100_2$
$1001100_2 \rightarrow 0100_2$

# 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 | Split data across multiple disks | No overhead, fast read / write | Reliability |
| RAID 1 | Mirrored Disks: Extra copy of data | Fast read / write, Fast recovery | High overhead → expensive |
| RAID 4 | Block-level striping with single parity disk. | Higher throughput for small reads | Still slow small writes (A single check disk is a bottleneck) |
| RAID 5 | Block-level striping, parity distributed across disks. | Higher throughput of small writes | The time to repair a disk is so long that another disk might fail in the meantime. |
| RAID 6 | Block-level striping, parity distributed across disks with two parity blocks per set. | Supports up to two concurrent disk failures. | More overhead needed than RAID 5. |

# 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.

$1.5 \cdot 10^6$ servers $\cdot$ 0.2kW/server $\cdot$ \$0.06/kW-hr $\cdot$ 8760 hrs/yr $\approx$ \$157.68 M/year

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?

$\text{PUE} = \frac{\text{Total building power}}{\text{IT equipment power}} \implies Savings \propto (PUE_{old} - PUE_{new}) * \text{IT equipment power}$
$(1.5 - 1.25) \cdot 50000$ servers $\cdot$ 0.2kW/server $\cdot$ \$0.06/kW-hr $\cdot$ 8760hrs/yr $\approx$ \$1.314 M/year