

Processing Data with MapReduce.: Takeaways



by Dataquest Labs, Inc. - All rights reserved © 2021

Syntax

- Mapper and reducer to calculate the longest word:

```
def map_max_len(words_chunk):  
    return max(words_chunk, key=len)  
  
def reduce_max_len(word1, word2):  
    return map_max_len([word1, word2])
```

- Mapper and reducer to calculate the average word length:

```
def map_average(words_chunk):  
    return sum([len(word) for word in words_chunk]) / len(words)  
  
def reduce_average(res1, res2):  
    return res1 + res2
```

- Mapper and reducer to calculate character frequencies:

```
def map_char_count(words_chunk):  
    char_freq = {}  
    for word in words_chunk:  
        for c in word:  
            if c not in char_freq:  
                char_freq[c] = 0  
            char_freq[c] += 1  
    return char_freq  
  
def reduce_char_count(freq1, freq2):  
    for c in freq2:  
        if c in freq1:  
            freq1[c] += freq2[c]  
        else:  
            freq1[c] = freq2[c]  
    return freq1
```

Concepts

- You can apply MapReduce to many different problems.
- When using MapReduce, the bulk of the work is defining a mapper and a reducer function.
- When designing the mapper function, we need to remember that the results must be mergeable.
- The reducer processes two results at a time. This creates some constraints on the way we design the reducer function.
- Sometimes we need to do post-processing to the MapReduce result. We did this when finding the unique pairs of consecutive character.

Resources

- [MapReduce](#)
- [Hadoop MapReduce](#)

Takeaways by Dataquest Labs, Inc. - All rights reserved © 2021