Big Data Analytics

Dr Sirintra Vaiwsri | Email: sirintra.v@itm.kmutnb.ac.th

MapReduce

MapReduce Filter

- The filtering is used for filtering a subset of records based on defined criteria (Bahga and Madisetti, 2019).
- Map function is only required to process filtering while the Reduce function is not necessary (Bahga and Madisetti, 2019).
- Use fb_live_thailand.csv as an input
- Assume we would like to see the number of reactions for each status type in the year 2018 where the number of reactions must be more than 2,000.

MapReduce Filter

```
class MapReduceFilter(MRJob):
    def mapper(self, _, line):
        # Data is a list of values in each line of a file
        # Get the status type
        # Get the number of reactions
        # Get the year
        # Check the year and number of reactions
            # Keep status type and number of reactions
```

MapReduce Distinct

- Distinct uses for selecting distinct values from the dataset (Bahga and Madisetti, 2019).
- Map function groups records with the same key while the value can be None (Bahga and Madisetti, 2019).
- Reduce function (Bahga and Madisetti, 2019)
  - Reduce function uses a list of values grouped by key received from the Map function.
  - The value can be returned as None.
- Use fb_live_thailand.csv as an input
- Assume we would like to find the date in the year 2018 from the input file.

MapReduce Distinct

```
class MapReduceDistinct(MRJob):
    def mapper(self, _, line):
        # Data is a list of values in each line of a file
        # Get the year
        # Get the date
        # Check the year
            # Keep the date
    def reducer(self, key, values):
        # Show the result grouped by key
```

MapReduce Binning

- Binning splits records into bins or categories.
- Map function is only required to process binning while the Reduce function is not necessary (Bahga and Madisetti, 2019).
- Use fb_live_thailand.csv as an input
- Assume we would like to categorise records into groups of status type for the year 2018.

MapReduce Binning

```
class MapReduceBinning(MRJob):
    def mapper(self, _, line):
        # Data is a list of values in each line of a file
        # Get the status type
        # Get the year
        # Check the year
            # Keep status type and line of record
```

MapReduce Inverted Index

- Inverted index data structure stores content such as words in a document and location such as document ID or filename (Bahga and Madisetti, 2019).
- Map function uses a field as a key index (Bahga and Madisetti, 2019).
- Reduce function is a document ID or any value (Bahga and Madisetti, 2019)
- Use fb_live_thailand.csv as an input
- Assume we would like to generate an inverted index where each status type is a key and a list of the number of reactions is a value.

MapReduce Inverted Index

```
class MapReduceInvertedIndex(MRJob):
    def mapper(self, _, line):
        # Data is a list of values in each line of a file
        # Get the status type
        # Get the number of reactions
        # Keep status and number of reactions in memory
    def reducer(self, key, values):
        # Create a list
        # Append value into the created list
        # Show the key and the list of values
```

MapReduce Sorting

- Sort the records based on the field
- Map function uses a field as a key to group-by and uses a value required for computing the average (Bahga and Madisetti, 2019).
- Reduce function (Bahga and Madisetti, 2019)
  - Reduce function uses a list of values grouped by key received from the Map function.
  - It then uses the Python sort function to sort the list of values.
- Use fb_live_thailand.csv as an input
- Assume we would like to sort the number of reactions with more than 3000 for each year

MapReduce Sorting

```
class MapReduceInvertedIndex(MRJob):
    def mapper(self, _, line):
        # Data is a list of values in each line of a file
        # Get the year
        # Keep the year and number of reactions in memory
    def reducer(self, key, values):
        # Show the key and the sorted list of values
```

MapReduce Join

- Join uses for combining records in two or more files based on a field.
- Inner join returns the intersection or common values.
- Full outer join returns the union or common and not-common values where it returns nothing for a table with no record matches.
- Left outer join returns all rows in the left table (file) and returns nothing for unmatched columns in the right table (file).
- Right outer join returns all rows in the right table (file) and returns nothing for unmatched columns in the left table (file).

MapReduce Inner Join

- Use fb_live_thailand2.csv and fb_live_thailand3.csv

```
class MapReduce(MRJob):
    def mapper(self, _, line):
        # Data is a list of values in each line of a file
        # Keep the ID column as key and line as value
    def reducer(self, key, values):
        # Create lists for the first and second files
        # Loop over value in values
            # Check and append the value of the first file into the first list
            # Do the same for the second list
        # Loop over value1 in the first list
            # Loop over value2 in the second list
                # Use None as a key and (value1+value2) as the value
```

MapReduce Left Outer Join

- Use fb_live_thailand2.csv and fb_live_thailand3.csv

```
def reducer(self, key, values):
    # Create lists for the first and second files
    # Loop over value in values
        # Check and append the value of the first file into the first list
        # Do the same for the second list
    # Loop over value1 in the first list
        # Check if the length of the second list is more than 0
            # Loop over value2 in the second list
                # Use None as a key and (value1+value2) as the value
        # If the length of the second list is 0
            Use None as a key and value1 as the value
```

MapReduce Right Outer Join

- Use fb_live_thailand2.csv and fb_live_thailand3.csv

```
def reducer(self, key, values):
    # Create lists for the first and second files
    # Loop over value in values
        # Check and append the value of the first file into the first list
        # Do the same for the second list
    # Loop over value2 in the second list
        # Check if the length of the first list is more than 0
            # Loop over value1 in the first list
                # Use None as a key and (value2+value1) as the value
        # If the length of the first list is 0
            Use None as a key and value2 as the value
```

MapReduce Full Outer Join

```
def reducer(self, key, values):
    # Create lists for the first and second files
    # Loop over value in values
        # Check and append the value of the first file into the first list
        # Do the same for the second list
    # Check if the length of the first list is more than 0
        # Loop over the first list
            # Check if the length of the second list is more than 0
                # Loop over the second list
                    # Use None as a key and (value1+value2) as the value
            # If the length of the second list is 0
                # Use None as a key and value1 as the value
    # If the length of the first list is 0
        # Loop over the second list
            # Use None as a key and value2 as the value
```

Assignment (2 points)

- Please implement the inner join, left outer join, right outer join, and full outer join.
- Please run your code and show the results to get 2 points.

References

- Bahga, A., & Madisetti, V. (2019). Big Data analytics: A hands-on approach.