Assignment Tutorial Step-by-Step Guide

Advanced Computing

File and Error Handling

- **Establish an input buffer and read in the data.**
- **Parse data element by element checking for corrupted data (format compliance) and null sets which will need to be filtered. Also checking for inconsistencies such as incorrect codes for airports (can be identified by implementing a lookup between the two files).**
- **Segment the data (file) into chunks to be fed into numerous mappers. i.e. Partition the buffer.**
- **Establish interim results buffer and output buffer.**

Job - Thread Controller

- **Set-up a 'Job' to keep track of the MapReduce process in terms of thread control throughout the different stages (Map, Shuffle, Sort and Reduce etc.).**

- **Consider data structures required for specific objective – What is the required input and outputs from the mappers and reducers?**

Mapping

- **Set-up a thread pool (number of threads dependents on the size of the file (number of blocks) and the number of processors).**
- **‘Map’ Keys and Values into an appropriate data structure (a tuple containing an identifier and a object to store relevant values associated with the key).**
- **Emit these tuples to the interim results buffer ready for combining and shuffling.**

Combining

- **Set-up combiner threads (as many as there are mappers)**
- **Take the buffer from each Mapper and group values for the same key.**
- **Emit results tuples to the interim results buffer ready the Reducers**

Shuffling and Sorting

- **Set-up shuffler and sorting threads**
- **Handle shuffling and sorting.**
- **Emit results tuples to the interim results buffer ready the Reducers**

Reducing

- **Setup Reducer threads (as many as the primary keys to answer the information objective).**
- **Implement the aggregation logic needed to group the values belonging to keys transmitted by all mappers.**
- **Write the result to the output buffer (possibly a file)**
- **(Log any additional logical errors that have now become apparent)**

Output Format and Error Reporting

- **Output Results as well as error reporting (per objective) in the best tabularisation layout for readability.**
- **Display would need to have an inclusive design**
- **Display would have to reduce cognitive load**
- **Display would have to be space-efficient**

Smart Assignment

Do your good assignment execution justice by the way you present your report.

Get Assignment Presentation Smart

Provide an informative structured and concise description setting out the journey travelled in developing and refining the execution and validation of your assignment task

Your excellence should come across in everything you do

In every line and section of your assignment report

It should sow a systematic methodologically guided approach

It should show a commitment to software engineering standards

It should show a commitment to code verification, elegance and validation

It should show you have been mindful of presentation smart principles

Map Reduce Assignment Tutorial **Overview**

Advanced Computing

CS3AC16

Map Reduce ‘Jobs’

- Run by a master ‘node’ (or potentially just the main thread in this case) in order to coordinate a ‘Job’.
- In the Map Reduce paradigm a job encapsulates the entire workflow of the MapReduce process for a single objective from input reading to output writing.
- It is used to set-up necessary parameters such as input and output formats for each step, and also to synchronise the Map and Reduce ‘worker’ nodes.

Input Reader

- The objective is to read in the necessary data from the provided files into a buffer and to process this buffer efficiently using multiple mapper instances each executing a `map()` function in parallel.
- Each mapper should be given a chunk of the input to process.
  - The number of mappers to launch should be equal to the number of records input divided by the number of available processors (nodes or cores). This ensures an even distribution across workers and optimised efficiency with the available resources.

Mapping

- Each mapper should read in line-by-line its portion of the input and should emit intermediate key-value pairs.
- Syntactic error checking should take place in the `map()` function.

[Combining]

- At this stage an optional reduction of the intermediate key-value pairs output as a result of mapping each input chunk maybe launched.

Shuffle

- The next stage is to shuffle the intermediate key-value pairs.
- The process of shuffling is the transfer of data between mappers and reducers.
- Each unique key output from the mappers should be assigned to a unique reducer.
- At this stage the values connected to a key should be collated into a iterable data structure such as a list.
- Once all mappers have processed the partitions assigned to them and the data has been transferred to the necessary reducer the `reduce()` function may be invoked.

[Partitioning and Comparison (Sorting)]

- Should the objective deem it necessary; composite keys comprised of information for sorting and for grouping maybe used throughout.
- If this is the case a partitioner function is required to segment the natural key from the sorting property, and a comparison method is required for implementing the sorting logic.

Reducing and Output Writing

- Each reducer instance should receive a single key and a list of associated values which are aggregated by the `reduce()` function in order to produce output according to the objective.
- The reducer should output a key-value pair in a format optimised for the resolution of the objective.

Error Detection and Handling

- Error detection and handling maybe incorporated into any stage of the Map Reduce process. For example values can be cross-referenced between the two input files in the map phase and duplicates maybe removed in the reduce phase.
- The most basic error handling strategy is to raise and log errors in an error report which can be addressed at a later stage. More advanced strategies may include attempts at error correction.