Questions from the Paper "MapReduce: Simplified Data Processing on Large Clusters" by Dean and Ghemawat

Paper can be downloaded from eLearning or
http://research.google.com/archive/mapreduce.html

1. Introduction:

a. What prompted the need for a new abstraction of parallel computation? What types of "messy details" does it hide from a programmer?

2. Programming Model:

a. What are the inputs and outputs for the map and reduce functions? What is the function of the MapReduce library?

b. The paper mentions several examples where MapReduce model can be applied. The first one is the famous WordCount problem and the paper lists the map and reduce pseudo-code. Try to write pseudo-code for the remaining examples in a similar format.

3.1. Implementation:

3.1. Execution Overview:

a. What is the role of the partitioning function in MapReduce?

b. In MapReduce, where is the output of the Map task buffered? What happens if there is an overflow? What would happen if the node running the Map task fails? (Hint: Read points 3 and 4)

c. Where is the sorting of the intermediate keys done? Why is sorting necessary?

d. After the reduce phase, how many output files are produced? Generally, what is done with these files?

3.2. Execution Overview:

a. What type of data structures does the master store for each task? What information does the master convey to the reduce tasks?

3.3. Fault Tolerance:

a. In case of node failure, which type of tasks (map or reduce) will need to be re-executed? Explain the reason.

b. Why does large scale worker failure not affect MapReduce operation?

3.4. Locality:

a. How does MR master try to preserve network bandwidth?

3.5. Task Granularity:

a. In practice, what are some good choices for M and R?

3.6. Backup Tasks:

a. What are stragglers? What is a general mechanism to alleviate this problem?

4. Refinements:

4.1. Partitioning Function:

a. What is the default partitioning function? When would you want to override it?

4.2. Ordering Guarantees:

a. What guarantee is made by the framework about the keys arriving at a partition? How is this useful?

4.3. Combiner Function:

a. What is the advantage of a combiner function? Where is it executed? When properties does the reduce function need to have before a combiner can be used?

4.4. Input and Output Types:

a. If you are using a "text" mode input reader, what will be the input (K, V) values?

4.9. Counters:

a. What are some uses of counter objects?

8. Conclusions:

a. According to the authors, what are the three main reasons why MapReduce programming model has been successful at Google?

b. What are the three main things that the authors have learned from this work?