1. T(1) is the completion time of a Job with 1 processor and T(N) is the completion time of the same Job with N processors. Answer the following questions. [Marks: 4]

a. The speedup S(N) defined as T(1)/T(N) will be always less than N. Why ?

b. What will be the speed up if a program is completely sequential and run on 10 processors?

c. If 10% of a program is sequential and if this program is run on a server with 100 processors, what is the theoretical limit on the speedup if you could not change the workload ?

d. A program has 40% sequential portion and 60% parallel portion. To get an effective speedup greater than 2, it should be run on how many processors. Assume you are able to scale the workload with the number of processors.

Solution: (1 mark x 4)

a. S(N) is ideally N when it is a perfectly parallelizable program. In any program, there will be a sequential portion of the code. Speedup by parallelization is not applicable for the sequential portion. In addition, there will be overheads in task distribution and scheduling. Hence the speedup will be always less than N.

b. The speed will be the same as that of running the program on one processor.

c. Using Amdah's Law, effective speedup S(N) = 1 / (f + (1-f) / N), where f is the sequential portion.

f=0.1

S(100) = 1/(0.1+(1-0.1)/100)

= 1/(0.1+0.009)

= 1/0.1009

= 9.9108 (Effective Speed up)

d. We need to use Gustafson-Barsis Law because workload can scale. By this law,

Speed up S(N) = f + (1-f) * N

f=0.4

S(N) = 2.0

N = (S(N)-f) / (1-f)

N = (2.0-0.4) / (1.0-0.4)

N = 1.6/0.6 = 2.6667

N = 3 (Number of processors cannot be fractional. Approximated to ceiling value, since speedup should be greater than 2.0). The program should be run on 3 processors.

2. Describe the most appropriate scheduling mechanism to be used with YARN in the following scenarios: [Marks: 4]

a. A company wants to run jobs of equal size on a Hadoop cluster. Number of jobs can vary at any point of time.
b. A company wants to run short jobs and long jobs on the same cluster and short jobs cannot always be starved by back to back submission of long jobs.
c. A company wants to divide the Hadoop cluster resources in different proportions between the departments and does not care about wastage of unutilized resources.
d. A company wants to run jobs of different nature concurrently on the same cluster and is worried about wastage of resources.

Solution: (1 mark x 4)

(a) Fair share scheduler with equal weights. Weights are used to determine the ratio of usage. Given the number of jobs can vary, it is hard to use a capacity scheduler.

(b) Fair share scheduler. Works well where shorter jobs are not starved and longer jobs also get to finish without getting unduly delayed when there are many short jobs.

(c) Capacity Scheduler. Allocates resources to departments in a predetermined proportion. If a department is not using the allocated resources, it will be wasted.

(d) Fair share scheduler with a weight of 0 for a queue that can use spare resources. Weight 0 means best effort basis allocation is good enough. Jobs can occupy resources based on requirement and availability.

3. You are given a demographic data set across all countries in the world. The number of records for a country is directly proportional to the population of the country given the sampling technique used to create the data set. You need to process the data in a Hadoop cluster using MapReduce and generate outputs specific to each country. What performance issues do you anticipate and what can be your approach to mitigate them ? [Marks: 4]

Solution:

Since the number of records is proportional to the population, the dataset will have a skewed distribution across countries. Given output has to be generated per country, if partitions are generated by country, there will be an imbalance of the data distribution leading to various performance issues –

(a) Reducers will get different loads and waiting times
(b) The network traffic will be skewed across map and reduce nodes during shuffle

(2 marks if the point about data distribution imbalance across nodes is mentioned)

There are some approaches to mitigating this –

1. There is a need for better localization of reducers to reduce latency.
2. Custom partitioners can be used to make sure there is better data balance. One can relabel the targets based on multiple features such as ethnicity, continent to provide better balancing.
3. Implementation of major compaction across coarse grained distributed data samples to balance the data size before mapper execution.
4. Implementation of minor compaction across fine grained distributed data samples to balance the data size before mapper execution.

(2 marks if 2 relevant points are mentioned)

4. You are asked to design a Decision Tree algorithm on a MapReduce cluster. It requires an iterative design where map() will perform split and reduce() will compute the information gain. Give an outline of your design approach in pseudo-code. [Marks: 4]

Solution:

/First MapReduce
//count the attribute for specific class label

Map Phase input:<k1, v1>

where k1-line no, v1-records
// Extract attribute and class label from instance of the record
output(k2, v2) where k2-attribute with class label, v2-1

Reduce Phase input<k2, List<v2>>
//Counts number of occurrences of combination for attribute with class label

output(k3, v3) where k3-attribute with class label, v3-frequency

//Second MapReduce
//Find the best splitting attribute (decision node)
//Calculate Gain Ratio for each attribute

Map Phase input: <k3, v3>
Compute Entropy(k3)
Compute InformationGain(k3)
Compute SplitInfo(k3)
Output(k4,v4) where k4-attribute, v4-Entropy Information and Split Information

Reduce Phase input:<k4, List<v4>>
//Calculate Information Gain Ratio for each attribute
Highest Gain ratio is the best splitting attribute (decision node)
Output(k5,v5) where k5-decision node, v5–Information Gain Ratio

//Third MapReduce
//Tree construction

Map Phase input:<k5, v5>
//Reads the record of best attribute
//Compute the node id for highest attribute
//Every Node is represented by a list of attributes and its values.
This process recursively calls for creating non leaf branches, until all data is classified.
Output: <k6,v6> where K6-node id, V6-Elements (attribute values)

5. You have to design a microblogging application where a user can have various types of interactions - e.g. signup, connect to users as friends, read posts from friends and others, create a new post, reply to an existing post or reply to a comment on a post. The application needs to run at a large scale with users across the globe. Discuss your choice of database (e.g. RDBMS, NoSQL) and the type(s) of configuration you will choose as per CAP Theorem. Given the example user interactions listed, it is possible that different interaction scenarios need unique configurations. [Marks: 4]

Given the scale requirements of the application and no strict multi-data item transaction level consistency requirements, it is better to use a NoSQL database. (2 marks if NoSQL is a choice with at least 1 reason)

None of the user interactions have consistency issues between multiple readers and writers. Sign-up, connect requests are independent of each other. Reading and writing of posts do not have any consistency requirements. Even if one replies to a post or replies to an existing comment, one can only write once the read includes a particular comment or post. It is possible that the latest posts and comments are not visible to all. So as per CAP theorem, it is better to use an AP configuration for the database. (2 marks if AP is the choice with reason given)

6. An e-commerce application records an anonymised user profile, item code and purchase quantity for every item in a sales transaction. The user profile includes age, gender etc. information. An individual user is mapped to a profile and only the profile ID is stored. E.g. tuple is ("user_profile1", "item_ABC12", 5). The data analysis team wants top 3 profiles that buy the most quantity of a particular item code. Write MapReduce pseudo-code to answer this question. [Marks: 4]

Solution:
Mapper:
Input: profileID, itemcode, purQty
Output:
    If itemCode == specifieditemcode:
        Emit profileID, purQty
    Else
        Skip the input record
Reducer:
Input: profileID, purQty
Output:
    ProfileID, totpurQty
Sort:
    Output records on totpurQty in descending order
Emit:
    ProfileID of first 3 sorted records

7. An enterprise application consists of a 2 node active-active application server cluster connected to a 2 node active-passive database (DB) cluster. Both tiers need to be working for the system to be available. Over a long period of time it has been observed that an application server node fails every 30 days and a DB server node fails every 60 days. A passive DB node takes 12 hours to take over from the failed active node. Answer the following questions. [Marks: 4]

a. What is the overall MTTF of the 2-tier system ?
b. Assume that there are no simultaneous failures of nodes in the App tier and an app node can always accommodate work from the failed node. What is your estimate of the availability of the 2-tier system ?

Solution:
(a)
MTTF of 2 node app cluster = 30 x 2 = 60 days
MTTF of 2 node DB cluster = 60 days
MTTF of 2 tier system = 1/(1/60 + 1/60) = 30 days

(b)
Availability of app tier = 1 or 100% given active-active config and no simultaneous failure of the nodes
Availability of DB tier = 30 / (30 + 0.5) = 0.9836 or 98.36 %o
Availability of the 2 tier system = 1 x 0.9836 = .9836 or 98.36%

8. You are asked to design a data processing logic that computes aggregates at various time granularities on timestamped data in a Big Data store. For example, hourly, monthly, quarterly aggregates of certain metrics. Provide arguments why your logic should exploit data and task parallelism. [Marks: 2]

Solution:
The data can be partitioned by time ranges, say by quarter, and each partition can be processed in parallel to compute arguments. This is data parallelism. (1 mark)
Within each partition, the aggregates can be computed in a pipelined fashion. Each aggregate computation can be a task - e.g. hourly, monthly, quarterly are separate tasks. However, a quarterly aggregate can be computed from monthly and monthly from hourly. These tasks can run in parallel once the pipeline is full. This is task parallelism. (1 mark)