1. 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 100 days and a DB server node fails every 50 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 only a single failure at any time, either in the App tier or in the DB tier, and an equal probability of an App or a DB node failure. What is your estimate of the availability of the 2-tier system ?

Solution:

(a) (2 marks)
App tier MTTF = 200
DB tier MTTF = 50
MTTF of system = 1 / (1/200 + 1/50) = 40 days

(b) (2 marks)
There is a 0.5 probability that a failure is an app failure and 0.5 probability that failure is a DB failure.
Only one failure happens at any time and there are no concurrent failures in the system.
When an app node fails the MTTR is 0 because it is an active-active config.
When a DB fails the MTTR is 0.5 days because it takes 12 hours to switch from passive to active.
So Availability(system) = 0.5 x 40 / (40 + 0) + 0.5 x 40 / (40 + 0.5) = 0.5 + .49 = .99 or 99%
First term is for app node failure and the second term for db node failure.

2. Discuss briefly 3 key issues that will impact the performance of a data parallel application and need careful optimization. [Marks: 3]

Solution: Mention 3 relevant issues. 4 example issues given below. (3 x 1 mark)

(i) Data partitioning is critical so that independent computations can be performed by instances of the same code on different compute nodes/processors. Different partitioning strategies can lead to different performance.
(ii) Co-location of compute with data is critical for better LoR. Typically compute is moved to the relevant data location as much as possible by the runtime system.
(iii) Load balancing of various nodes is key for efficiency, this is also closely related to how data is partitioned.
(iv) Latency reduction of the over computation is also key. It depends on various factors such as partitioning, LoR, balancing of nodes, volume of data movement across nodes and hence the choice of network stack matters.

3. Name a system and explain how it utilises the concepts of data and tree parallelism. [Marks: 3]

Solution: (1.5 marks each for map and reduce points)

Hadoop is an example system where
(i) map phase exploits data parallelism so that a problem is broken down into smaller sub-problems by partitioning the data and executing same logic on the partitions in parallel on different compute resources (1.5 marks)
(ii) reduce phase is like inverse tree parallelism where smaller sub-problems are combined to create the final result

In Hadoop there is a single map and reduction phase and an iteration can be built on top through custom programming.

4. A travel review site stores (user, hotel, review) tuples in a data store. E.g. tuple is (“user1”, “hotel ABC”, “<review>”). The data analysis team wants to know which user has written the most reviews and the hotel that has been reviewed the most. Write MapReduce pseudo-code to answer this question. [Marks: 4]

Solution: (1 mark for map and 1 for reduce for each query x 2 queries)

Solution:
In phase 1:
Use a wordcount approach to output a file that has key, count.
Key can be a hotel or a user name.
Note: Prefix the key string with h_ or u_ to have the 2nd phase program find a max for users and a max for hotels separately.
Use a combiner to distribute the counting across nodes.

In phase 2:
Take the phase 1 output file of (key, count) and take the max example approach to find 2 different max, one for hotel and one for user.
The same program can do it because the key is prefixed with h_ or u_ to separate the reducer’s work.
Use a combiner to distribute the max operation across nodes.
The following program uses a single stage map reduce but here all the work is done by a single reducer. You cannot parallelize this program.

Input file: Reviews.csv

abi	Taj	good
abu	Lemeridian	good
Alex	Manor	good
Avery	DelhiInn	good
Asher	Centaur	good
Aiden	SaiVilla	good
Abigail	BestWestern	good
Anthony	Park	good
Bennett	ITDC	good
Bentley	Oberoi	good
Brandon	Orchid	good
Brayden	Lemeridian	good
Braxton	Taj	good
abi	Park	good
Christopher	Taj	good
Caleb	SaiVilla	good
Christian	Lemeridian	good
Cameron	Taj	best
Bennett	Manor	good

Connor	Taj	good

<u>my_mapper.py</u>

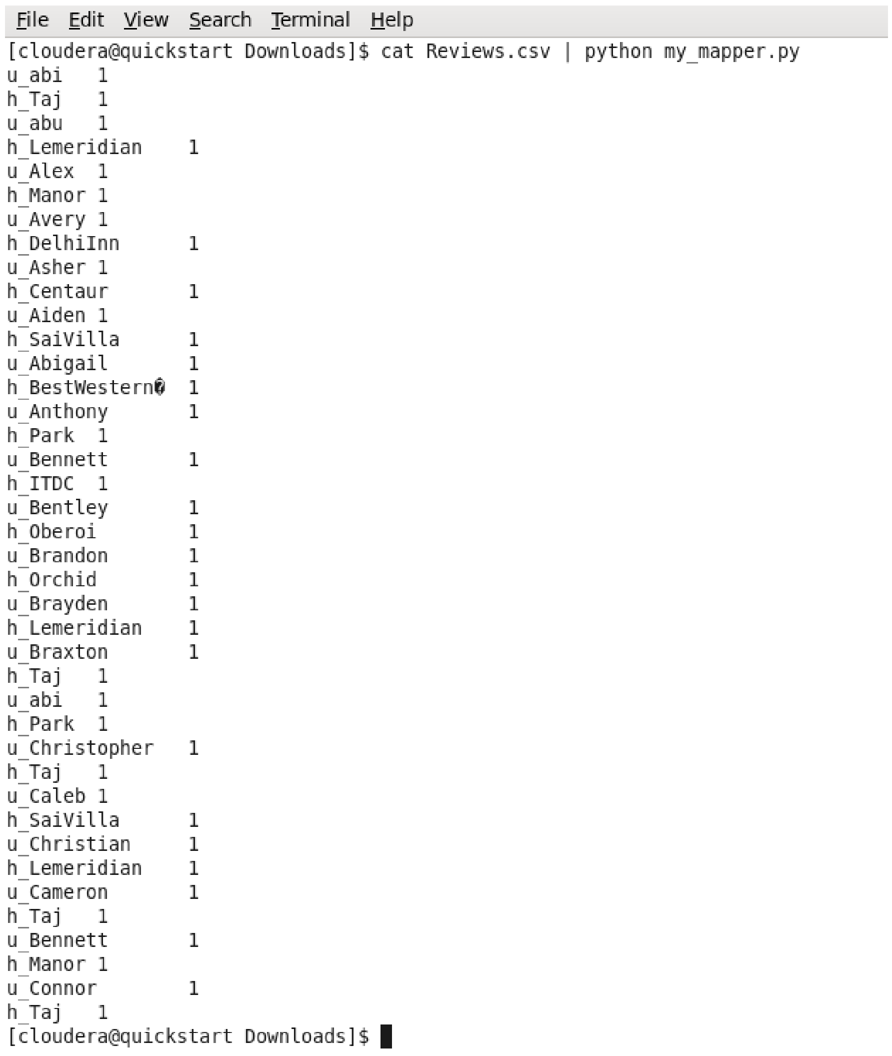
```python
import csv
rows=[]

# csv file name
filename = "Reviews.csv"

# reading csv file
with open(filename, 'r') as csvfile:
  # creating a csv reader object
  csvreader = csv.reader(csvfile)

  # extracting each data row one by one
  for row in csvreader:
    rows.append(row)

# map function
for row in rows:
  print '%s\t%d' % ('u_'+row[0],1)
  print '%s\t%d' % ('h_'+row[1],1)
```



```
[cloudera@quickstart Downloads]$ cat Reviews.csv | python my_mapper.py
u_abi   1
h_Taj   1
u_abu   1
h_Lemeridian    1
u_Alex  1
h_Manor 1
u_Avery 1
h_DelhiInn      1
u_Asher 1
h_Centaur       1
u_Aiden 1
h_SaiVilla      1
u_Abigail       1
h_BestWestern�  1
u_Anthony       1
h_Park  1
u_Bennett       1
h_ITDC  1
u_Bentley       1
h_Oberoi        1
u_Brandon       1
h_Orchid        1
u_Brayden       1
h_Lemeridian    1
u_Braxton       1
h_Taj   1
u_abi   1
h_Park  1
u_Christopher   1
h_Taj   1
u_Caleb 1
h_SaiVilla      1
u_Christian     1
h_Lemeridian    1
u_Cameron       1
h_Taj   1
u_Bennett       1
h_Manor 1
u_Connor        1
h_Taj   1
[cloudera@quickstart Downloads]$
```

my_reducer1.py

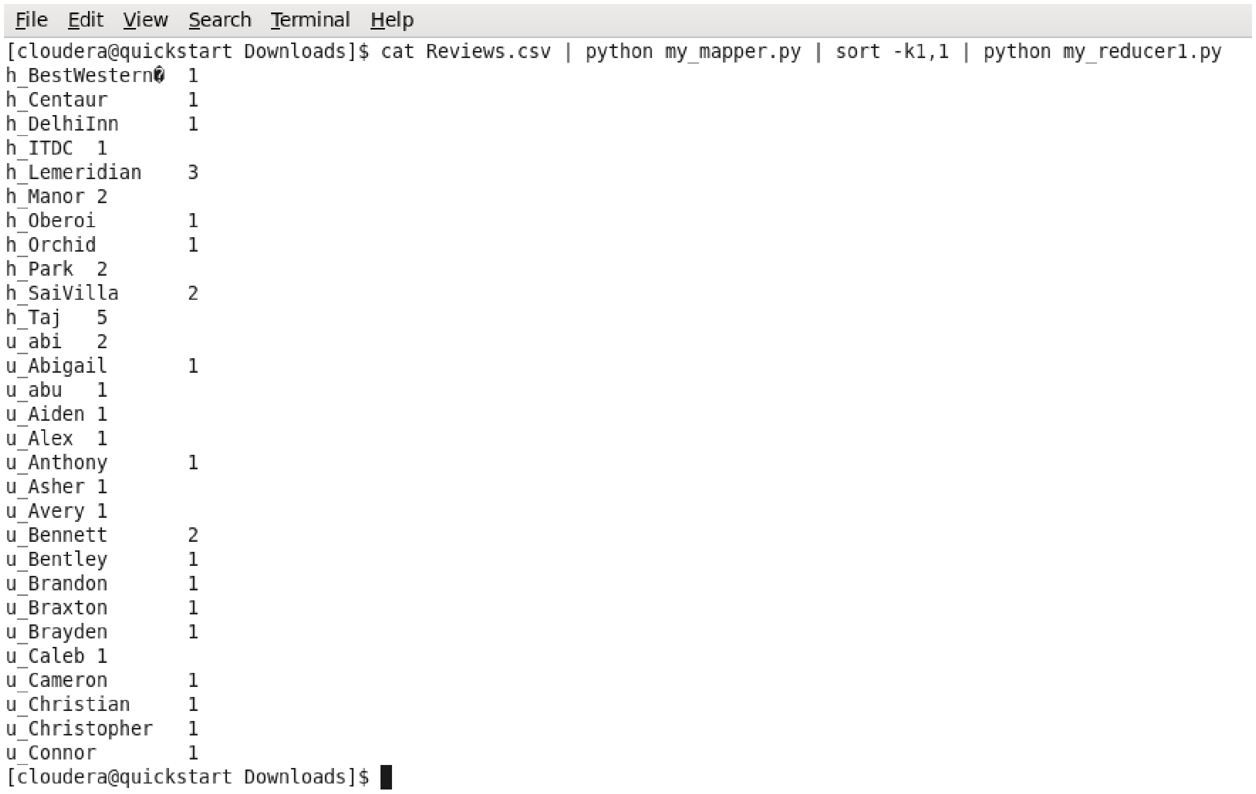
```python
from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

# read the entire line from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # splitting the data on the basis of tab we have provided in mapper.py
    word, count = line.split('\t', 1)
    # convert count (currently a string) to int
    count = int(count)

    # by key (here: word) before it is passed to the reducer
    if current_word == word:
        current_count += count
    else:
        if current_word:
            # write result to STDOUT
            print '%s\t%s' % (current_word, current_count)
        current_count = count
        current_word = word

# do not forget to output the last word if needed!
if current_word == word:
    print '%s\t%s' % (current_word, current_count)
```



```
[cloudera@quickstart Downloads]$ cat Reviews.csv | python my_mapper.py | sort -k1,1 | python my_reducer1.py
h_BestWestern�  1
h_Centaur       1
h_DelhiInn      1
h_ITDC  1
h_Lemeridian    3
h_Manor 2
h_Oberoi        1
h_Orchid        1
h_Park  2
h_SaiVilla      2
h_Taj   5
u_abi   2
u_Abigail       1
u_abu   1
u_Aiden 1
u_Alex  1
u_Anthony       1
u_Asher 1
u_Avery 1
u_Bennett       2
u_Bentley       1
u_Brandon       1
u_Braxton       1
u_Brayden       1
u_Caleb 1
u_Cameron       1
u_Christian     1
u_Christopher   1
u_Connor        1
[cloudera@quickstart Downloads]$
```

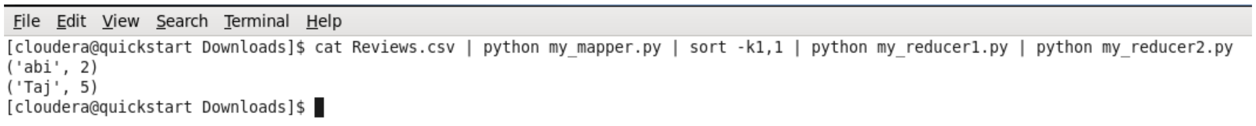
my_reducer2.py

```
# import sys to read and write data
import sys
```

```
u_max = 0
h_max = 0
User = ''
Hotel = ''

# reading data from stdin
for line in sys.stdin:
  # to remove leading and trailing whitespace
  line=line.strip()
  # split the line into words
  row=line.split()
  n_row = row[0].split('_')
  if n_row[0] == 'h':
    if int(row[1]) > h_max:
      h_max = int(row[1])
      Hotel = n_row[1]
  elif n_row[0] == 'u':
    if int(row[1]) > u_max:
      u_max = int(row[1])
      User = n_row[1]

print (User, u_max)
print (Hotel, h_max)
```



```
[cloudera@quickstart Downloads]$ cat Reviews.csv | python my_mapper.py | sort -k1,1 | python my_reducer1.py | python my_reducer2.py
('abi', 2)
('Taj', 5)
[cloudera@quickstart Downloads]$
```

5. An e-commerce site stores (user, product, rating) tuples for data analysis. E.g. tuple is ("user1", "product_x", 3), where rating is from 1-10 with 10 being the best. A user can rate many products and products can be rated by many users. Write MapReduce pseudo-code to find the range (min and max) of ratings received for each product. So each output record contains (<product>, <min rating> to <max rating>). [Marks: 4]

Solution: (2 marks for map and 2 marks for reduce to calculate min and max for each product)

Write pseudo code for finding min and max. Combine the same map and reduce logic where max and min are both produced by map and reduce functions. Accept if a student has created 2 different sets for max and min and then a post-processor that creates the desired single output for each product. Deduct 1 mark for second approach given it is inefficient and creates 2 passes through the data. Refer to max/min examples for core logic.

```
#map function

def map(user, product, rating)
yield (product, rating)
```

```
# reduce function
# initialize min and max to 0   #for loop
def reduce(product, ratings):
min_rating=min(MinR,ratings)
max_rating=max(MaxR,ratings)
yield (product, (min_rating, max_rating))
```

6. In the following application scenarios, point out what is most important - consistency or availability, when a system failure results in a network partition in the backend distributed DB. Explain briefly the reason behind your answer. [Marks: 4]
(a) A limited quantity discount offer on a product for 100 items at an online retail store is almost 98% claimed.
(b) An online survey application records inputs from millions of users across the globe.
(c) A travel reservation website is trying to sell rooms at a destination that is seeing very few bookings.
(d) A multi-player game with virtual avatars and users from all across the world needs a set of sequential steps between team members to progress across game milestones.

Solution: (4 x 1 mark)

(a) Consistency is critical because only a few items are left on sale and the system should avoid overbooking.
(b) Availability is critical because users should be able to submit the survey but there is no relationship between multiple user writes. Writes can be appended to the data store.
(c) Given few bookings and high availability of rooms, consistency is not critical because there is less chance of overbooking. Availability of the system is more critical for accepting user requests.
(d) Given dependencies between user requests, consistency is critical.

7. The CPU of a movie streaming server has L1 cache reference of 0.5 ns and main memory reference of 100 ns. The L1 cache hit during peak hours was found to be 23% of the total memory references. [Marks: 4]

a. Calculate the cache hit ratio h.
b. Find out the average time (Tavg) to access the memory.
c. If the size of the cache memory is doubled, what will be the impact on h and Tavg.
d. If there is a total failure of the cache memory, calculate h and Tavg.

Solution:
(a) Hit + Miss = Total CPU Reference
Hit = 23%, Miss = 77%
Hit Ratio h = Hit / (Hit + Miss), that is 23 / 100= 0.23
(1 mark)
(b) Tavg = Average time to access memory
Tavg = h * Tc + (1-h) * (Tm + Tc)
= 0.23*0.5 + (0.77)*(0.5+100)
= 0.115 + 77.385
= 77.5 ns
(1 mark)
(c) When the size of the cache is doubled, the L1 cache hit also will double.

So the new L1 cache hit = 46%
h = 0.46
Tavg = 0.46*0.5 + (0.54)*(0.5+100.0)
= 0.23 + 54.27
= 54.5 ns
(1 mark)

(d) When there is total failure of cache memory, cache hit ratio h = 0.0
Since the cache has failed, data will be fetched from the main memory bypassing cache.
Therefore, Tavg = 0.0*0.5 + (1.0)*(100.0)
= 100.0 ns
(1 mark)

8. Assume that you have a NoSQL database with 3 nodes and a configurable replication factor (RF). R is the number of replicas that participate to return a Read request. W is the number of replicas that need to be updated to acknowledge a Write request. In each of the cases below explain why data is consistent or in-consistent for read requests. [Marks: 4]
1. RF=1, R=1, W=1.
2. RF=2, R=1, W=Majority/Quorum.
3. RF=3, R=2, W=Majority/Quorum.
4. RF=3, R=Majority/Quorum, W=3.

Solution: (4 x 1 mark)

1. Consistent because there is only one replica and the same replica is used for read and write acknowledgements.
2. Write will go to both replicas and read will use one of these 2 replicas. Hence read will be consistent.
3. Read will go to 2 replicas and write also to 2 replicas. So there is a common replica, hence data read will be consistent.
4. Consistent because both read uses 2 replicas and write uses all replicas.