CSCI/ECEN 5673: Distributed Systems
Spring 2022
Homework 3 Solutions
Due Date: 04/21/2021

Please submit a PDF copy of your answers via the submission link on Canvas by April 21, 2022. Write your answers in the space provided. Please DO NOT use any extra space. The space provided is sufficient for answering the questions.

Honor Code Pledge: On my honor, as a University of Colorado at Boulder student, I have neither given nor received unauthorized assistance on this work.

Name:

1. What are Resilient Distributed Datasets? Explain the fault recovery method used and the kinds of operations in RDDs?

Resilient Distributed Datasets (RDDs), a distributed memory abstraction that lets programmers perform in-memory computations on large clusters in a fault-tolerant manner. They are read only, partitioned collection of records. They can be built through coarse grained operations called as transformations. RDDs provide fault recovery through "lineage". All the transformations are logged in the lineage so that whenever the RDD fails or partitions of an RDD failed they are recomputed using lineage.

RDDs provide two kinds of operations namely transformations and actions.
Transformations are used to build RDDs whereas actions are performed on the RDD to compute and output a value to the application or to export data to a storage system.

2. What are the workload and technology characteristics GFS assumed in its design and what are their corresponding design choices? What design choices GFS adopts to keep the master from being a bottleneck?

Workload and Technology characteristics:
100s to 1000s of inexpensive commodity hardware and software => frequent failures
Multi-GB files are common for Google. Billions of such objects, and TBs of data.
Most files modified via append, not random write
Most files read sequentially in large batches

Design Choices:
Larger chunk size: 64 MB
Replication of chunks
No caching of file data at client or chunk server, other than standard Linux buffer caching
Efficiency of Append. Small writes are supported but not efficiently

Master is not a bottleneck:
Master is not in the way for data transfer
Caching of meta data on the client side
Delegation of replica consistency management to the chunk servers

3. For each of the following problems describe how you would solve it using map-reduce. You should explain how the input is mapped into (key, value) pairs by the map stage, i.e., specify what is the key and what is the associated value in each pair, and, if needed, how the key(s) and value(s) are computed. Then you should explain how the (key, value) pairs produced by the map stage are processed by the reduce stage to get the final answer(s). If the job cannot be done in a single map-reduce pass, describe how it would be structured into two or more map-reduce jobs with the output of the first job becoming input to the next one(s).

   (a) The input is a list of housing data where each input record contains information about a single house: (address, city, state, zip, value). The output should be the average house value in each zip code.

   Map output: from each individual input record, key = zip; value = value
   Reduce output: key = zip; value = average of all input values with the same zip

   (b) The input contains two lists. One list gives voter information for every registered voter: (voter-id, name, age, zip). The other list gives disease information: (zip, age, disease). For each unique pair of age and zip values, the output should give a list of names and a list of diseases for people in that zip code with that age. If a particular age/zip pair appears in one input list but not the other, then that age/zip pair can appear in the output with an empty list of names or diseases, or you can omit it from the output entirely, depending on which is easier.
   (Hint: the keys in a map/reduce step do not need to be single atomic values.)

   Map output: key = (age, zip) from each input record from either file; value = either a name or disease with some information to distinguish which it is, or a pair of lists one of which is empty and another that contains a single name or a disease as appropriate.

   Reduce output: key = (age, zip); value = lists of all names and diseases concatenated from all input records with the same (age, zip) key.

4. What Consider a DHT using ring mapping such as Chord

a) Explain how a key is mapped to a node.

Keys and nodes map to points in a ring space. A node handles all keys mapped between it and its predecessor.

b) In the most basic form, nodes simply track their predecessor and successor. Very briefly state what problem this leads to as the network grows larger.

Performance problems as nodes forward a key resolution request. The request may need to be forwarded over a long sequence of nodes (O(N) complexity)

c) The Chord scheme overcomes this problem by having each node maintain a "finger table". Explain how this scheme works.

- Maintain a pointer to nodes at 2^i distances ($0 <= i < \log N$).
- The finger table stores the ids of these nodes along with a range of keys managed by this nodes and the successive nodes until the next node in the finger table.

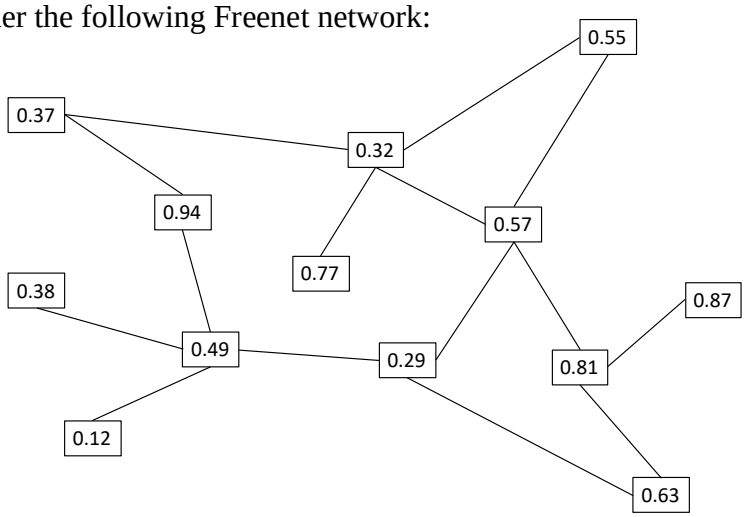
d) In a Chord structured DHT with N nodes, how many hops would a lookup operation require?

log N

e) Assume a Chord network in which node q is the successor of node p. During operation, node p discovers that its successor link is no longer consistent because q has updated its predecessor link. What does this change imply must have happened in the network?

A new node joined the network between p and q.

5. Consider the following Freenet network:



a) a) Suppose <0.62, 100> is stored at node 0.57. A client requests node 0.37 to retrieve the value associated with 0.62. What sequence of nodes will this get request be routed through in the network?

0.37→0.32→0.57

b) What sequence of nodes will the response be routed through?

0.57→0.32→0.37

c) Next, another client requests node 0.38 to retrieve the value associated with 0.62. What sequence of nodes will this *get* request be routed through in the network?

0.38→0.49→0.94→0.37

d) A client requests node 0.55 to store <0.1, 300>. What sequence of nodes will this put request be routed through and where will <0.1, 300> be stored? Assume that <0.1, 300> is not currently stored in the network.

0.55→0.32; it will be stored at 0.32 and 0.55

e) Is it possible to store conflicting <key, value> pairs (pairs that have the same key but different values) in Freenet where no single node is aware of the conflict? Explain your answer.

Yes, it is possible. The two put( ) requests may be routed through a completely different sets of nodes. This can happen due to TTL expiry.