15-440/15-640: Homework 3

Due: November 13, 2017 23:59am

Name:	
Andrew ID:	

1 DIG (18 points)

In this problem, we will use dig tool available on Linux and Mac OS to explore DNS servers. You can read about it using `man dig`. Recall that a DNS server higher in the DNS hierarchy delegates a DNS query to a DNS server lower in the hierarchy, by sending back to the DNS client the name of that lower-level DNS server (assuming no recursion is specified). **For each of the following questions, show the sequence of commands that you ran on your shell with dig and the output they generate.** *Hint:* Be sure to use the `+norecurse` option to dig, and remember that you will need to specify different target DNS servers (@) each time. Your queries should look like:
`dig +norecurse @< targetserver> RecordToResolve RecordType`.

1. Starting with a root DNS server (from one of the root servers [a-m].root-servers.net), initiate a sequence of queries using dig for the A-type record for www.blog.xkcd.com without using recursion. Be sure to also show the list of the names of DNS servers in the entire delegation chain starting from the root in answering your query. [**5 points**]

```
a.root-servers.net
$ dig +norecurse @a.root-servers.net www.blog.xkcd.com
a.gtld-servers.net. 172800 IN A 192.5.6.30
$ dig +norecurse @a.gtld-servers.net www.blog.xkcd.com
dns1.p03.nsone.net. 172800 IN A 198.51.44.3
$ dig +norecurse @dns1.p03.nsone.net www.blog.xkcd.com
www.blog.xkcd.com. 3600 IN A 208.118.225.100
```
Depending on the various root servers you chose, your names may differ by a letter or a number, as well as differing IP addresses.

2. Repeat the same procedure as above for www.csd.cs.cmu.edu [**5 points**]

```
a.root-servers.net
$ dig +norecurse @a.root-servers.net www.csd.cs.cmu.edu
a.edu-servers.net. 172800 IN A 192.5.6.30
$ dig +norecurse @a.edu-servers.net www.csd.cs.cmu.edu
nsauth1.net.cmu.edu. 172800 IN A 128.2.1.8
$ dig +norecurse @nsauth1.net.cmu.edu www.csd.cs.cmu.edu
AC-DDNS-1.NET.cs.cmu.edu. 600 IN A 128.2.184.227
$ dig +norecurse @AC-DDNS-1.NET.cs.cmu.edu www.csd.cs.cmu.edu
SCS-MAN-SITES.WEB.CS.CMU.EDU. 3600 IN A 128.2.217.31
```
Depending on the various root servers you chose, your names may differ by a letter or a number, as well as differing IP addresses.

3. Repeat the same procedure for 81.183.132.209.in-addr.arpa. This time, dig for the PTR-type

record. [**5 points**]

```
a.root-servers.net
$ dig +norecurse @a.root-servers.net ptr 81.183.132.209.in-addr.arpa
a.in-addr-servers.arpa. 172800 IN A 199.212.0.73
$ dig +norecurse @a.in-addr-servers.arpa ptr 81.183.132.209.in-addr.arpa
209.in-addr.arpa. 86400 IN NS u.arin.net.
$ dig +norecurse @u.arin.net ptr 81.183.132.209.in-addr.arpa
183.132.209.in-addr.arpa. 86400 IN NS ns1.redhat.com.
$ dig +norecurse @ns1.redhat.com ptr 81.183.132.209.in-addr.arpa
81.183.132.209.in-addr.arpa. 600 IN PTR www.redhat.com.
```

4. Use dig -x IP addr to perform reverse DNS lookup for the IP address 209.132.183.81. What is the domain name associated with this IP address? What is the type of the DNS record in the answer section? Compare the answer section to part 3's answer. [**3 points**]

```
$ dig -x 209.132.183.81
81.183.132.209.in-addr.arpa. 600 IN PTR www.redhat.com.
```

2 Hadoop and MapReduce (18 Points)

1. MapReduce stores the Map output on the local disk of the Map worker. Why don't we write Map output to GFS here? Discuss the advantages and disadvantages. [**5 Points**]

Performance would be reduced, since the Map output would have to be sent twice over the network for GFS replication. Fault-tolerance would be improved, since a Map worker failure before the fetch of all its Map output would not result in the Map output having to be re-computed.

2. Suppose we have a cluster of 100 nodes, and 2 of the machines were produced in 2007, while others were produced in 2017. Explain why a massive Hadoop MapReduce job might take longer to complete on 100 nodes, compared to only using the 98 2017-produced machines. [**5 Points**]

Hadoop is a batch-processing system that launches multiple Map and Reduce tasks in parallel that are scheduled across the nodes in a cluster and completes only when every single task of the job has completed. Even if all but 1 task finish in time T and 1 task requires time 2T to complete, the job's execution time is 2T. This is the tail latency problem and tasks scheduled on the slower nodes act as stragglers that increase the overall execution time

3. Sparse Matrix Multiplication using MapReduce:

$$\begin{bmatrix} 1 & 2 & 0 \\ 3 & 4 & 0 \\ 0 & 5 & 6 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 2 & 3 \\ 4 & 0 \end{bmatrix}$$

Input file:

```
1 1 1 A
1 2 2 A
2 1 3 A
```

```
2 2 4 A
3 2 5 A
3 3 6 A
1 2 1 B
2 1 2 B
2 2 3 B
3 1 4 B
```

According to what you were taught in the lecture, write out the output of each Map and Reduce stage of the two stages (four in total) taught in the slides. [**8 Points**]

```
Phase 1 Map:
1 1 1 1 A
2 1 2 2 A
1 2 1 3 A
2 2 2 4 A
2 3 2 5 A
3 3 3 6 A
2 1 2 1 B
1 2 1 2 B
2 2 2 3 B
1 3 1 4 B

Phase 1 Reduce:
1 2 1
2 2 3
1 1 4
1 2 6
2 1 8
2 2 3
2 2 12
3 1 10
3 2 15
3 1 24

Phase 2 Map:
1 2 1
2 2 3
1 1 4
1 2 6
2 1 8
2 2 3
2 2 12
3 1 10
3 2 15
3 1 24

Phase 2 Reduce:
1 1 4
1 2 7
2 1 8
2 2 15
```

```
3 1 34
3 2 15
```

3 GFS and Spanner (15 Points)

1. Discuss the trade offs on selecting the default chunk size in GFS, i.e., list one advantage and one disadvantage of making the default chunk size 64MB instead of 4MB. [**5 Points**]

   Having a large chunk size reduces the amount of metadata to be stored, but on the same time it wastes space due to fragmentation. Larger chunk size also reduces client-master interaction and hence reducing network overhead.

2. Google's GFS design was specialized for their target workload of batch processing of large-scale data, leading to a number of design simplifications that were not appropriate for the broader collection of workloads that later arose. Identify and briefly explain two drawbacks of the GFS design that interfered with effective support for a broader collection of applications. [**5 Points**]

   One drawback is the limitation on number of files, caused by requiring that all metadata fit in the main memory of the one GFS master. Another drawback is the poor tail latency caused by chained delivery of data to the three replica locations, during writes. Another drawback could be the non-POSIX semantics, including duplication and empty spaces with the at-least once append.

3. Suppose in Spanner, one client does a Put, notifies another client, and the client that receives the notification performs a Get on the same key. How does Spanner ensure that the Get will see the Put's value? [**5 Points**]

   Spanner will assign the Put a commit timestamp within [TT.now().earliest, TT.now().latest]. Due to commit wait, Spanner will make sure when Put is completed, true time is after the commit timestamp of the Put. For Get, Spanner will assign it either the commit timestamp of the Put or TT.now().latest, Spanner will then only execute the Get until it sees all writes that may happen before the commit time of the Get, which means Get will see the Put.

4 P2P (17 Points)

You've recently been hired as the distributed systems guru at a new game development company. The company's flagship game, *Luke Flukem Whoever*, is supposed to be a massively multiplayer first-person-shooter game. The online world is expected to consist of thousands of players distributed around the world, and because the company is eliminating dedicated servers to save money, you are responsible for designing the peer-to-peer system for online play.

The game's initial design is to store data, such as other players and in-game objects, in a peer-to peer system that a player will query dynamically. Your manager hears about a cool new idea called Chord, and suggests that you look into it, arguing that it provides a distributed hash lookup primitive and is robust to frequent node arrivals and departures. Your manager argues that you can store in-game objects on peers (with replication to handle node departures) using Chord for scalable lookup of these in-game objects.

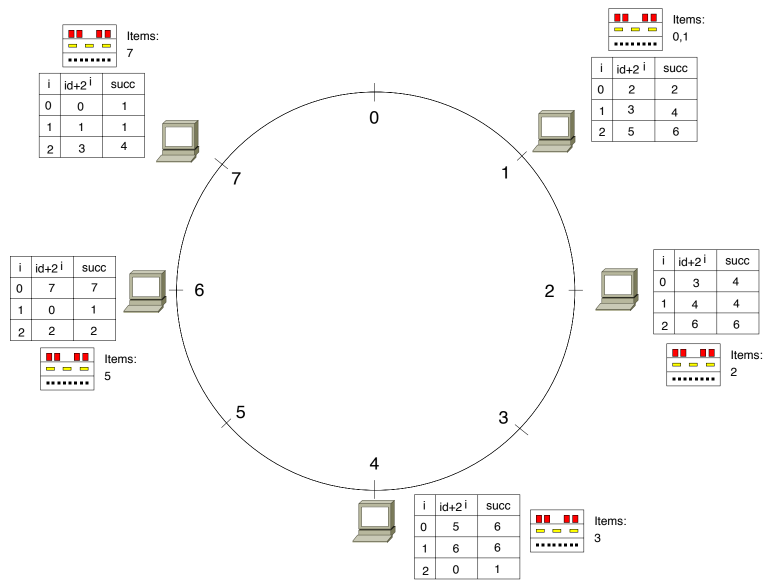
1. After giving it some thought, you decide that this is a poor solution for a first-person-shooter game where the peer-to-peer system may have thousands of geographically distant players in it. Explain briefly the rationale for your decision. [**4 Points**]

Chord provides log(N) hops for finding an item, but with 1000 nodes, this will require 10 hops to reach the node that has the item. Because the nodes are distributed around the world, each hop is likely to incur a latency of about 50-100ms. Thus, the lookup time for an object will often by 500ms, which is far too much for an interactive first-person-shooter.

The publisher, Eccentric Arts, decides that the game release date needs to be pushed up by 3 months. Consequently, the game design slightly changes, and instead of all the players being in one very large world, players only connect with players in the same geographic city. You now decide that Chord is a reasonable design choice for storing certain static in-game objects on peers.

You implement this system with ease and begin a small scale deployment consisting of only four peers. The peers contain the listed items (e.g., *Node 4* only has *Item 3*), and have successor tables as shown (the $id + 2^i$ column is there to remind you how the successor table is set up).

2. List the nodes that will receive a query from *node 1* for *item 7*. [**5 Points**]



i	id+2^i	succ
0	0	1
1	1	1
2	3	4

i	id+2^i	succ
0	2	2
1	3	4
2	5	6

i	id+2^i	succ
0	7	7
1	0	1
2	2	2

i	id+2^i	succ
0	3	4
1	4	4
2	6	6

i	id+2^i	succ
0	5	6
1	6	6
2	0	1

The query will begin at node 1 and go to the highest node in the successor table that does not exceed the the number of the item—node 6. Node 6 will then see that it has an entry in its table for 7 and forward it on to 7.

3. List the nodes that will receive a query from *node 2* for *item 5*. [**4 Points**]

Node 2 does not have an entry for item 5, so the query is forwarded to the highest success in the table that does not exceed 5 which is node 4. Node 4 has an entry for 5 in its table which is at node 6, so it will now be forwarded to 6.

During larger-scale testing, you notice that the popularity of objects in your game is skewed, and that a small set of peers get overloaded with requests for the most popular items. Like any good distributed systems student, you pull out a valuable tool from your distributed systems belt to reduce this load: caching.

4. Your manager likes this idea, and suggests that once an object is found, it should be propagated back down the path in the Chord ring taken to lookup the object on the forward path, with the item cached at each node along the way. Your manager argues that this is an effective choice of nodes to replicate on because the nodes caching the object in this path will never have seen this object before. Why is this true? [**4 Points**]

Assume that a node along the lookup path had a cached version of the object that the querying node was looking for. In this case, that node can immediately respond with the cached version of the data. In other words, if a query gets to the node that has the original version of the data, it's a guarantee that no node on the current path had a cached version of that data.

5 Hashing (12 Points)

1. Assume there are M keys distributed across N servers in perfect balance. For conventional hashing (divide as hash function) and consistent hashing (each server has K virtual nodes) respectively, what's the number of keys migrated when one server is leaving the cluster? [**4 Points**]

Conventional Hashing: $\frac{M}{2}$
Consistent Hashing: $\frac{M}{N}$
Given perfect balance assumption, the number of virtual nodes does not have any impact on the result.

2. Virtual node is a common variant of consistent hashing. Unlike what we are taught on 15-440 class, where each physical server is mapped to a random position on circle, a server keeps multiple (or even hundreds of) virtual nodes. Each virtual node is mapped to the circle and act as bucket individually. List *two* potential advantages for keeping virtual nodes when using consistent hashing. [**4 Points**]

1. Better load balancing, especially when the number of servers is small.
2. Data migration time may become shorter, since it can be performed in parallel when a node is joining or leaving.

3. Alice is developing her own cloud storage system and considering implementing deduplication to reduce costs. She decided to compute the hash value for each block (4KB) to see if the same block has already existed. Her friend, Cheryl, who has taken 15-440, suggested using Rabin fingerprints to decide block boundaries. What supports Cheryl's argument? [**4 Points**]

Using Rabin fingerprints to decide block boundaries instead of fix-size chunking helps improve shift-resistance, i.e., when inserting or deleting some bytes, it's highly probable that only one hash value needs to be recompute.

6 CDN (12 Points)

1. David loves reading news from `cnn.com`, which is accelerated by Akamai. When he just traveled from Pittsburgh to the west coast, he noticed that it took longer to load all images on web pages. What may be the cause of the phenomenon? Also briefly describe a way to verify it. Assume David travels very fast (maybe by supersonic aircraft) and Akamai serves content as described

in 15-440 lectures. [**7 Points**]

Since David just moved a long distance in very short time, the DNS records for Akamai's low-level DNS server have not expired yet. So when he loads images, he is actually fetching data from content servers near Pittsburgh. In the case, both latency and bandwidth is worse than that of local servers. It can be verified by 1. clearing local DNS cache to see whether it becomes faster, or, 2. use dig to resolve DNS with both cache on and off, compare them.

2. Kevin has developed a super marvelous online collaborative editor (similar to Google Docs) on a server located in CMU campus. However, his friends reported that user experience was poor when using it outside US. Kevin remembered that he had been taught on 15-440 course that CDN can help accelerate his website, and decided to buy the service from Akamai. Do you agree with his decision? If yes, describe how Akamai can improve the user experience; Otherwise, tell Kevin what are the challenges. [**5 Points**]

Both Yes and No are correct, given reasonable justification:
No - Collaborative editor is a very dynamic web application, so using traditional CDN described in class to accelerate loading static content doesn't help a lot.
Yes - Akamai's Web Application Accelerator works as app engine to deliver the application code and run on edge servers. If the users are located nearby, larger bandwidth and shorter latency can be achieved.