15-440/15-640: Homework 3

Due: November 8, 2018 11:59pm

Name:
Andrew ID:

1 GFS FTW (25 points)

Part A (10 points)

The Google File System (GFS) is an extremely popular filesystem used by Google for a lot of its everyday tasks. AFS has also gained its fair share of fame especially among educational institutes. State whether AFS or GFS is better for each of the following cases. Give **one** line of explanation for your answer.

1. Tolerating high rate of failure of the file server. **[2 pts]**

   GFS. Because it supports failure recovery by having replicas and WAL and checkpointing.

2. Repeatedly accessing the same file. **[2 pts]**.

   AFS. Because AFS utilizes caching.

3. Large file reads. **[2 pts]**.

   GFS. GFS is specifically meant for this use case.

4. Multiple clients (several hundred) concurrently adding entries to the same todo file. **[2 pts]**.

   GFS. Concurrency is built in for GFS.

5. Low server storage requirements. **[2 pts]**.

   AFS. GFS might store duplicate data.

Part B (5 points)

Recall that the GFS master server is the central repository of a lot of information and serves an important purpose in GFS. The chunkservers store chunks of files.

1. The GFS master stores metadata such as the Namespace (directory hierarchy), the mapping from files to chunks etc. State **one** reason why the master server crashing is especially bad for **recovery**? **[1 pts]**

   Master Server stores metadata in RAM, so it's hard to recover when it crashed.

2. State **two** other tasks that the GFS master server does. **[2 pts]**.

   Delegates Consistency management; Garbage collection of orphaned chunks; Migrates chunks between chunkservers.

3. The Chunkservers store 64MB file chunks on local disk. State **one** disadvantage of choosing a smaller file chunksize. **[1 pts]**.

With a smaller chunksize, the system becomes hard to scale because of GFS overhead per chunk.

4. Explain **one** reason why the GFS client does not cache data. **[1 pts]**.

Alleviate consistency difficulties; Streaming reads don't benefit from caching.

Part C (10 points)

The 440 staff uses GFS with 4 chunkservers (A, B, C, D) and one master server. We only store one file that consists of one chunk and its replicated across A, B and C. Assume that A is the chunk's primary. Zeleena and Amadou don't talk to each other and decide that it's a good idea to edit the same file at the same time. Zeleena wants to fill the chunks with "ZZZZZ...", while Amadou wants to fill the file with "AAAAA...". Zeleena and Amadou operate on the same chunk at the same time. The chunks are initialized to "MMMMM...". Initially, assume that there are no failures.

1. If Zeleena and Amadou use WRITE, is it possible that Amadou writes to a chunk on B "AAAAA..." but Amadou's data gets overwritten by Zeleena with "ZZZZ..."? **[2 pts]**

Yes. Writes are serialized but nothing prevents clients from overwriting each other.

2. If Zeleena and Amadou use APPEND, is it possible that Amadou writes to a chunk on B "AAAAA..." but Amadou's data gets overwritten by Zeleena with "ZZZZ..."? **[2 pts]**.

No. Append cannot overwrite data due to serialization through primary.

3. Now B suddenly fails. **Name** the mechanism the master uses to detect this failure. State the next steps once the master has detected the failure. **[3 pts]**.

Heartbeats. Decrement count of replica for all chunks on dead chunkserver. Re-replicate chunks missing replicas in background.

4. B is back up again and everything is back to normal. Now Zeleena wants to delete the file. In **three short** bullet points explain the steps once the client (Zeleena) has issued the delete request. **[3 pts]**.

(1)Master records deletion in log; (2)Master scans file namespace in background → Removes files with such names if delete for longer than 3 days. In-memory metadata erased; (3) Master removes unreferenced chunks from chunkservers.

2 The Return Of BurgerNet! (20 points)

Part A (15 points)

By popular demand we have decided to change the name of BergerNet to BurgerNet. BurgerNet is a social network that has suddenly gained a lot of popularity. BurgerNet hosts its services in its own proprietary servers located in Pittsburgh. We now need your help to scale BurgerNet.

1. Sam has travelled to London and is trying to access his pics on BurgerNet but its significantly slower to load than when he was in Pittsburgh. Explain the possible reason. **[2 pts]**.

Increase in latency. Server is in Pittsburgh, Sam is in London. Distance is directly proportional to latency. Pics are generally $>$ 500kb.

2. Sam is trying to load his profile picture but it takes X milliseconds when he is in London, while it only takes 5 ms in NYC. The speed of information travel is 10^5 mile/s. The distance between Pittsburgh and London is 3000 miles and the distance between Pittsburgh and NYC is 300 miles. The bandwidth is 1 Gb/s. Give an exact value for X. Assume there is no ACKing and to 'load' a picture you only have to send the picture across the network. **[6 pts]**

Size of pic = (5ms - $(300/(10^4)) * (1Gb/s) = 2ms * (10^9 b/s) = (2 * 10^-3) * (10^9) = 2 * 10^6 = 2Mb$. $X = (3000/10^5) + (2 * 10^6)/(10^9) = 30ms + 2ms = 32ms$

3. We talked in class about something called a CDN. Define the term CDN. In **two short sentences** explain how a large CDN like **Akamai** could potentially help alleviate the problem described above. **[3 pts]**.

CDN = Content Delivery Network. A content delivery network or content distribution network (CDN) is a geographically distributed network of proxy servers and their data centers. Akamai CDNs are great for serving static content. Closer to user = low latency, faster loading times.

4. BurgerNet users normally access the same picture multiple number of times. What other technique could help make BurgerNet even faster? Explain in **one sentence** how. **[2 pts]**.

Caching. Local cache on each user's computer, store recently accessed pics.

5. BurgerNet decides to add a collaborative document and picture editing. This feature is now very popular. Answer and explain if Akamai CDNs would still help? **[2 pts]**.

No: Akamai CDNs are only made for static content delivery, hence would not be useful for editable and dynamic content. OR Yes: CDNs do speed up interactive applications, by terminating TCP, HTTPS early. So, even if caching is not possible, CDNs often lead to a 5-10x speedup.

Part B (5 points)

BurgerNet has decided to adopt the model of Sequential Consistency over its backend infrastructure. One day you want to post comments on a controversial event. You first remove some people from your list of friends (transaction T1), and then post your comments (transaction T2).

1. Is it possible that the people you remove in T1 could potentially see the post in T2? Explain your answer in one sentence. **[2 pts]**.

Yes. Any legal ordering of events is allowed in SC.

2. Define External Consistency. **[1 pts]**

System behaves as though all transactions are done serially on a single machine.

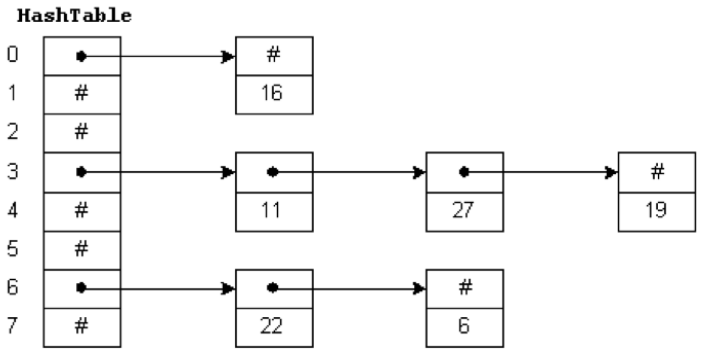
3. Spanner is a globally distributed database that uses "TrueTime" to guarantee external consistency. With Spanner is it possible for the people you remove in T1 to see the post in T2? Explain your answer in **one sentence**. **[2 pts]**.

Not possible, because T1 will have to be done first and T2 will not be able to acquire the locks because its TT.now() will be after T1s.

3 A Side of Consistent Hashing (10 points)

You are the primary software engineer who manages and maintains a new CDN. You're responsible for choosing where files reside in the servers and therefore, how to direct user requests. You decide to distributed the content based on the hash value of the file ID (int), with the goal of having a good load balancer that doesn't have to move too many files on server failures. Initially you decide to use separate chaining as the underlying technique, as shown below:

Part A (4 points)



The hash function for the table is simply f(x) = x. Hence a number k will get placed at index f(k) % sizeOfArray. Example: f(22) = 22. 22 % 8 = 6. Hence 22 is placed at index 6.

1. Due to increased load, we decide to increase the size of the hash-table. A re-size causes the array of the hashtable to be increased by size 1. Assume that there is a re-size operation that takes place right after the state shown above. How many keys would need to be remapped? **[2 pts]**.

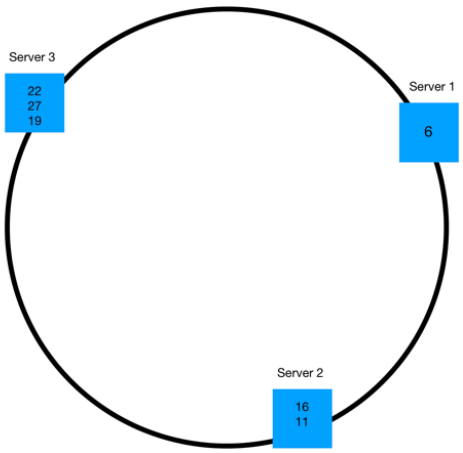
   16, 11, 27, 19 and 22 would need to be re-mapped.

2. State the array index where numbers 11 and 6 would go after resizing. **[2 pts]**.

   11 would go to index 2. 6 would remain at same index.

Part B (6 points)

Your colleague who took 15-440 mentions that you could load balance in a better way, and recommends using consistent hashing, as seen below:



The hashing function is f(x) = (x*10)° mod 360°. The ring above represents a 360° circle. Each number is mapped to the closest server whose corresponding degree is great than or equal to its own angle. Server 1 is at 65°, Server 2 is at 170° and Server 3 is at 270°. Example: f(6) = 60° and hence it would be mapped to Server 1.

In each question below assume the state starts off as shown in the figure above.

1. Which servers would the numbers 17, 0, 28 and 18 be mapped to? **[2 pts]**.

17 → Server 2. 0 → Server 1. 28 → Server 1. 18 → Server 3.

2. Now assume that Server 3 is overloaded. Hence we decide to add a new Server 4 at 240°. State the numbers that need to be remapped to the new server? **[2 pts]**.

22, 19.

3. Suddenly Server 2 fails. State the numbers that need to remapped and which server would they be mapped to. **[2 pts]**.

16 and 11 would now be mapped to Server 3.

4 MoCkInG sPoNgEbOb (20 points)

Chelsea wants to de-stress before her exam and she checks out the newest SpongeMock memes online by directing the browser to **spongemock.memes.com**. The local DNS server then performs an iterative lookup. The diagram below shows some of the DNS records contained in each DNS server. Note that DNS responses are cached in the local DNS server.

localdns.cmu.edu (S1)

Record Number	Name	Value	Type	TTL
R1	.	a.root.net	NS	1 day
R2	a.root.net	198.40.4.8	A	1 day

a.root.net (S2)

Record Number	Name	Value	Type	TTL
R3	com.	b.gtld.net	NS	12 hours
R4	b.gtld.net	198.31.2.90	A	12 hours

b.gtld.net (S3)

Record Number	Name	Value	Type	TTL
R5	memes.com.	ns-9.memes.com	NS	2 hours
R6	ns-9.memes.com	83.102.188.3	A	2 hours

ns-9.memes.com (S4)

Record Number	Name	Value	Type	TTL
R7	spongebob.memes.com	83.102.188.4	A	30 minutes
R8	spongemock.memes.com	83.102.188.5	A	30 minutes
R9	spoderman.memes.com	83.102.188.6	A	30 minutes

1. Fill in following table to indicate the sequence of queries and responses exchanged among the servers. **[8 pts]**

	Sender	Receiver	Type (Query/Response)	Data
1	Chelsea's PC	S1	Query	spongemock.memes.com
2	S1	S2	Query	spongemock.memes.com
3	S2	S1	Response	R3, R4
4	S1	S3	Query	spongemock.memes.com
5	S3	S1	Response	R5, R6
6	S1	S4	Query	spongemock.memes.com
7	S4	S1	Response	R8
8	S1	Chelsea's PC	Response	R8

2. Fill in any new DNS records in the local DNS server right after the sequence of queries and responses in (a). Label the new records with record numbers starting from R10. **[4 pts]**.

Record Number	Name	Value	Type	TTL
R1	.	a.root.net	NS	1 day
R2	a.root.net	198.40.4.8	A	1 day
R10	com.	b.gtld.net	NS	12 hours
R11	b.gtld.net	198.31.2.90	A	12 hours
R12	meme.com.	ns-9.memes.com	NS	2 hours
R13	ns-9.memes.com	83.102.188.3	A	2 hours
R14	spongemock.memes.com	83.102.188.5	A	30 minutes

3. After six hours, Chelsea is done with her exam and checks spongemock.memes.com again to see if there are any updates. Fill in the DNS records in the local DNS server right before any queries and responses are performed for her second request. **[4 pts]**.

Record Number	Name	Value	Type	TTL
R1	.	a.root.net	NS	18 hours
R2	a.root.net	198.40.4.8	A	18 hours
R10	com.	b.gtld.net	NS	6 hours
R11	b.gtld.net	198.31.2.90	A	6 hours

4. Once again, fill in following table to indicate the sequence of queries and responses exchanged among the servers for Chelsea's second request. **[4 pts]**.

	Sender	Receiver	Type (Query/Response)	Data
1	Chelsea's PC	S1	Query	spongemock.memes.com
2	S1	S3	Query	spongemock.memes.com
3	S3	S1	Response	R5, R6
4	S1	S4	Query	spongemock.memes.com
5	S4	S1	Response	R8
6	S1	Chelsea's PC	Response	R8

5 MapReduce (10 points)

Freddie just bought a MapReduce compute cluster and wanted to run the following tasks on his new cluster. For each of the following tasks, state whether it is a good fit for MapReduce and explain your answer. If MapReduce is a good fit, describe how you will define the mappers and reducers for the task; otherwise, explain why MapReduce does not work well.

1. Support a Bitcoin mining service that is similar to the one you implemented in Project 1. There are over 10^{10} requests, and each request is consists of message M and unsigned integers N1, N2. The Bitcoin service needs to find the unsigned integer pair (n1, n2) that generates the largest hash value for all $0 <= n1 <= N1$, $0 <= n2 <= N2$ when concatenated with M. **[2 pts]**.

   Yes. There are multiple ways to model this problem. For each request, we can do the following: If N1 and N2 are large, split N1 into multiple ranges: (0, p1) (p1, p2), ... (pi, N1). Similarly for N2, we have multiple ranges (0, q1) (q1, q2), ... (qj, N2). Then we can split the request into smaller sub-requests in the form of (message, pi, pi+1, qj, qj+1).
   Each mapper takes in a sub-request, computes the maximum hash value in the given range, and returns (maxHash, n1, n2). Here (n1, n2) is the pair of integers in the given range that gives the maximum hash. Each record in the mapper output is a key-value pair: (message, N1, N2) -¿ (maxHash, n1, n2). For each request (message, N1, N2), the reducer collects all key-value pairs from the mappers and finds the maximum hash among these outputs.
   Note: Some of you found a single maximum value among all messages instead of finding one maximum hash value for each message. Since we did not explain it clearly in the problem, we accepted both interpretations.

2. Solve an object recognition task using stochastic gradient descent method. This involves over 10^5 iterations. **[2 pts]**.

   No. Because each iteration depends on the results from previous iterations. These iterations can only be executed sequentially, and therefore will not fit in the MapReduce model.

3. Calculate the average amount of time spent on coursework in the past week among all CMU students. **[2 pts]**.

   No. There is no need to use MapReduce here since computing the average for 20000 numbers can be simply done on a single computer.

4. Analyze the similarities of 10^9 websites. Specifically, for each website A, find the website B that is most similar to A. **[2 pts]**.

Yes. There are multiple ways to model the scenario as a MapReduce problem. For example, in the first phase, each mapper takes a portion of the websites, preprocesses these websites and returns the data for these websites; the reducer collects these results. In the second phase, split the websites into multiple groups. Each mapper takes in the data for all websites and a group G of websites. It then finds the most similar websites for every website in G, and outputs a **map[website]website** that stores all the most similar pairs.
Note: The above example only works well when we don't need a large amount of data for each website. Otherwise, the I/O operations will be too expensive.

5. Maintain a trading simulation app that updates stock prices every minute based on 10^8 real-time transactions. **[2 pts]**.

No. Because MapReduce is not good at processing real-time data. It is designed for existing datasets on disks. Also, MapReduce is not good for low-latency computations like updating prices every minute.

6 MapReduce's Limitations & Spark (7 points)

1. A key performance challenge in MapReduce is called "stragglers". What's a practical technique to alleviate the straggler problem? (One sentence) **[1 pts]**

Redundancy: reschedule slow tasks.

2. List and explain in a single sentence two additional limitations of MapReduce that are addressed by Spark (one sentence each). **[2 pts]**

(1) Doesn't work for interactive data exploration, framework focuses on throughput. Spark focuses on interactive queries and fast response time;
(2) MapReduce is very I/O intensive (every MR phase read/writes to HDFS). Power of MapReduce relies on iterating MapReduce phases, so that 90% of time might be spend on just waiting for I/O to complete very inefficient. Spark minimizes I/O;
(3) MapReduce is somewhat limited and hard to write. Spark offers much broader set of actions that can be done on a dataset.

3. Spark enables fault-tolerance in-memory cluster computation using Resilient Distributed Datasets (RDD). **[2 pts]**

   (a) Explain the idea of lineage in enabling fault tolerance in two sentences.

   Lineage keeps track of how transformations are applied to RDDs. This enables faster and cheaper recomputation upon failure.

   (b) Why are RDDs immutable? Please give a one-sentence example.

   Because it is safe to share immutable data across multiple processes. Also, RDDs are designed to be immutable so they are deterministic for the same inputs, which enables lineage when RDDs need to be recreated.

4. Give a specific scenario where Spark is preferred over Hadoop MapReduce. Briefly explain your answer. **[1 pts]**.

One possible scenario: Many distributed ML algorithms perform a lot of Map/Reduce iterative steps. For these iterative steps, Hadoop MapReduce spends a lot of time on I/O to disks and over network, whereas Spark with in-memory computations is more efficient;
Other possible answers: Spark overcomes I/O bottlenecks in general, Spark lowers latency and thus is better for interactive data exploration, etc.

5. Give a specific scenario where Hadoop MapReduce is preferred over Spark. Briefly explain your answer. [**1 pts**].

   Possible Answers: Large datasets cannot fit into memory and can only be stored on disks, Hadoop is in general cheaper because disk space costs less than memory space, etc.

7 We are Industrial Fans (8 points)

The purpose of this question is to expose you to some of the famous distributed systems used in industry. These questions are intended to be **short** answers. Discuss in 2-3 sentences what each one of these technologies is and **one famous use-case**. The links to the papers have been provided to you.

1. BigTable [**4 pts**].

   Used by Google. Spanner was an improvement over BigTable. Built on top of GFS. Used a compressed, high performance data storage. Did not support transactions.

2. DynamoDB [**4 pts**].

   Used by Amazon, Netflix and many others. Dynamo is a highly available key-value store. To achieve high availability it sacrifices consistency under certain failure scenarios.