CS425 Fall 2022 – Homework 1

(a.k.a. "Friday Night Plights")

Out: Aug 24, 2022. Due: Sep 21, 2022 (Start of Lecture, 2 pm US Central)

Topics: Clouds, Mapreduce, Gossip, Failure detectors, Membership, Grids, P2P Systems (Lectures 1-8)

Instructions:

1. **Attempt any 8 out of the 10 problems** in this homework (regardless of how many credits you're taking the course for). If you attempt more, we will grade only the first 8 solutions that appear in your homework (and ignore the rest). Choose wisely!
2. Please hand in **solutions that are typed** (you may use your favorite word processor. We will not accept handwritten solutions. Figures and equations (if any) may be drawn by hand (and scanned).
3. **All students (On-campus and Online/Coursera)** – Please submit PDF only! Please submit on Gradescope. [https://www.gradescope.com/]
4. Please **start each problem on a fresh page**, and **type your name at the top of each page**.
5. Homeworks will be **due at the beginning of class on the day of the deadline. No extensions. For DRES students only:** once the solutions are posted (typically a few hours after the HW is due), subsequent submissions will get a zero. **All non-DRES students must submit by the deadline time+date.**
6. Each problem has the same grade value as the others (10 points each).
7. Unless otherwise specified in the question, the only resources you can avail of in your HWs are the provided course materials (slides, textbooks, etc.), and communication with instructor/TA via discussion forum and e-mail.
8. You can discuss lecture concepts and the questions on Piazza and with your friends, but you cannot discuss solutions or ideas on Piazza.

Prologue: Today, almost all professional sports teams (e.g., NFL, NBA, WNBA, Tennis – women's and men's, MLB, NHL, MLS, etc.) use distributed computing to analyze games, give feedback to players (sometimes in real time), and to decide how good players are.

This homework uses fictitious stories and characters from sports teams to frame the homework problems. Any resemblance to persons, places, things, or events, living or dead, past, present, or future, is purely coincidental. These stories and this homework is aimed neither at endorsing, nor criticizing, any league, any sports team or persons associated with these teams or leagues.

Problems:

1. A fictitious wealthy person named L. Skum decides that after trying to acquire a social media company called Qwitter, he should acquire the Houston Rockets. He decides that the Solar system needs to be colonized, and would like to use BitTorrent among users of the 9 planets (Skum believes Pluto is still a planet, go figure!). Anyway, while working through an example, he finds a case where a file has 6 shards: A, B, C, D, E, F. The querying node (peer, leecher) is on a spacecraft, connected to all the planets directly (Skum figured out a way to have high bandwidth, low latency communication that beats the speed of light!). The planet servers at the 9 planets each have the following shards: Mercury (BCD), Venus (CADFE), Earth (DEAB), Mars(ABCDEF), Jupiter (E), Saturn (CDE), Uranus (ACD), Neptune (DEA), Pluto (CA).
   a. Which shard will be fetched *first*?
   b. What is the *order* in which shards will be fetched by the querying node? (assume that no shards are being fetched by any other nodes). If there are ties, you can use the alphabetically lower one.
   c. If one introduced a 10th renegade planet called DeGrasseTyson (and the spacecraft was connected to it), assign a set of shards to DeGrasseTyson that changes the *first* fetched shard by the spacecraft (ties broken by lower letter)?
2. Anthony Fauci and Deborah Birx decide that since the pandemic is over, they will purchase a new sports team. So they do. But being the scientists that they are, they still would like to do contact tracing among all the fans and players in the hometown of their new team. They need to build a MapReduce program to process contact tracing data for Covid-19/Monkeypox. You are given two (private) datasets: D1 is a large dataset contains (in each line of input) triples (Unique_Person_Name, location, start_time, end_time) indicating that Unique_Person_Name was in location from start_time to end_time (times include dates, etc.—you don't need to worry about that). Dataset D2 contains (in each line of input) a Unique_Person_Names of persons who tested positive for Covid. Your goal is to find the list of individuals (unique names, non-duplicated) from D1 who were in the same place and same time as at least one individual in

D2 (note that D2's names will also appear in D1). Write a (chained) MapReduce program for this. You can assume times can be compared (< > =). Feel free to assume appropriate library functions (e.g., test if two time intervals overlap), and you don't need to use fancy datastructures. As usual, all data is sharded in HDFS, and you can chain MapReduces (but be judicious), you should have some parallelism, etc. Each line is read as the value and the key is empty (in the first Map stage). To keep it simple, we recommend starting with two MapReduce programs, one that reads from D1 and one that reads from D2. Later MapReduces can read from output of previous MapReduces, as well as D1 and/or D2. Note that intermediate data from a Map is not available for subsequent stages! (The datasets are kept private within CDC, so you can ignore privacy concerns for this question!) Correctness is important, efficiency is secondary (but you must have some parallelism). Write either pseudocode, or clear unambiguous descriptions. You can chain Mapreduces if you want (but only if you must, and even then, only the least number). Be sure to output each pair at most once (e.g., in sorted order). You don't need to write code – pseudocode is fine as long as it is understandable. Hint: Think about the "key" in Map output.

3. The Chicago Bears have a deep conversation about why they've been such a failure of late, and they decide it's because they don't have a large enough social media reach. They would like to increase their Twitter presence to find big donors. As a warmup exercise they wish to their heavy-hitter followers on social media. They would like to use MapReduce for this. In MapReduce, one writes a program for Map that processes one input line at a time and outputs a (key, value) or nothing; and one writes a program for Reduce that processes an input of (key, all values for key). The iteration over input lines is done automatically by the MapReduce framework. You are given an input file containing information from an asymmetrical social network (e.g., twitter) about which users "follow" which other users. If user a follows b, the entry line is (a, b), and this line appears exactly once. Write a MapReduce program (Map and Reduce separately) that outputs the list of all users U who satisfy the following three conditions simultaneously: U has at least 4 million followers, and U herself/himself follows at least 500 users, and U follows at least one user V who in turn has at least 2 million followers (e.g., @AaronRodgers12 would be such a U). You can chain Mapreduces if you want (but only if you must, and even then, only the least number). Be sure to output each pair at most once (e.g., in sorted order). You don't need to write code – pseudocode is fine as long as it is understandable. Hint: Think about the "key" in Map output.

4. (Read the previous question) Speaking of Aaron Rodgers, the Green Bay Packers, being a more democratic ownerless (and some say nonprofit team) laugh at the Chicago Bears (like they always do) and decide to one-up them by finding pairs. They want to output all pairs (a,b) such that both a and b follow @packers, and a and b follow each other (there is no restriction on how many followers a and b have), and neither a nor b follow the @ChicagoBears. Your input is the same as question 3. You can chain Mapreduces if you want (but only if you must, and even then, only the least number). Be sure to output each pair at most once (e.g., in sorted order). You don't need to write code – pseudocode is fine as long as it is understandable. Hint: Think about the "key" in Map output.
5. (You can use other websites for this question) Unlike many men's sports, the history-making US Women's Soccer team thinks "out of the box" (a major reason for their consistent success!). While they have been running their services on AWS EC2 instances, because they are used to "passing the ball efficiently" they recently became aware of serverless cloud services. Can you help them? (Please limit your answer for each part to less than 50 words. Be concise!)
    a. What is the key difference between AWS Lambda and AWS EC2?
    b. Given the same cost ($) budget, name two (classes of) applications for which AWS Lambda would be a better choice than EC2? Justify your answer, don't just give the name.
    c. What is the key difference between AWS Lambda and AWS spot instances?
6. One of the poorest teams in sports, The Champaign-Urbana Cricketers, would like to run a membership protocol among their fans' phones. Because they have so many fans but so little memory on their fans' phones, they wish to maintain a *partial membership list* at each phone. The membership protocol is gossip-style. The membership list at each process is selected uniformly at random across the entire group and is of size k (much smaller than N the total number of processes in the system. Don't worry about the protocol part.). For this question you can assume processes don't fail or join. Each message (containing the entire local membership list) is gossiped to m randomly selected neighbors (from the membership list), where $m < k$, and $m = O(\log(N))$, with the latter needed to ensure spread of gossip. The team argues that due to random selection, the overall "behavior" of this protocol (in terms of dissemination time of gossips, etc.) is the same as in the case where all processes might have had full membership lists (known everyone in the group), and each gossip (partial membership list) was sent to m neighbors. Are they right or is their reasoning as poor as their team's coffers? If yes, then give a proof. If no, show why.

7. Since they became very popular and successful, the US Women's Gymnastics team has been trying to build a peer to peer network to connect their current players with aspiring gymnasts. They want to use Chord, but they want to modify the Chord DHT rules to make it topologically aware of the underlying network latencies (like Pastry is). Design a variant of Chord that is topology-aware and yet preserves the O(log(N)) lookup cost and O(log(N)) memory cost. Use examples or pseudocode – whatever you chose, *be clear*! Make the least changes possible. You should only change the finger selection algorithm to select "nearby" neighbors, but without changing the routing algorithm. Show that (formal proof or informal argument):
    a. Lookup cost is O(log(N)) hops.
    b. Memory cost is O(log(N)).
    c. The algorithm is significantly more topologically aware than Chord, and almost as topology aware as Pastry.
8. One of the oldest teams in NBA, Sacramento Kings, also thinks very old-fashioned,. They structure their Gnutella system as a perfect balanced binary tree with $N=2^m-1$ processes. (m large). All leaves are at the same level. Additionally, all the leaf nodes (only) are joined together in a ring (when nodes are arranged topologically). You can assume m is quite large ($m > 20$).
    a. What is the minimum TTL required so that a query is received by all, no matter who its sender is?
    b. How many processes (apart from the sender) receive the query if a child of the root is the querying node, and the TTL=m-2?
    c. What is the TTL required by a child of the root for everyone to receive its query?
    d. When m is small and equals 2, 3, or 4, re-calculate part (a) for these 3 values of m.
9. Although the NE Patriots team did not win the last Superbowl, their coach is still paranoid. Due to their paranoia they decide to design a failure detector for an asynchronous system of *N* processes (*N* very large). A cornerback has designed a ring-based failure detection protocol that works as follows: each process *i* selects a set of target processes (once selected, these targets don't change) and asks these processes to send <u>to</u> it (i.e., to process *i*) heartbeats directly. Targets are selected as follows. All processes are organized in a virtual ring. Targets of a process *i* include three subsets: its *k* predecessors, its *k* successors, and *k* further processes chosen randomly among all non-predecessor/successor/*i* processes. Once the list of targets is selected, it is not changed (including the randomly selected members). *k* is a fixed number much smaller than *N*, and known to all. Heartbeats are not relayed (so this is not gossip, but more like ring failure

detection, except there is no ring), and process *i* times out if it doesn't receive heartbeats. A process is detected as failed if any of its heartbeat receivers do not receive expected heartbeats within a pre-specified timeout.

   a. When is completeness violated? That is, find the value *M* so that if there are (*M-1*) simultaneous failures, all of them will be detected, but if there are *M* simultaneous failures then not all of them may be detected.
   b. Is the algorithm 100% accurate?
   c. If the period is fixed (say 1 s) at all processes, what is the load on each process in terms of heartbeats that it needs to send? Calculate the worst case, best case, and average load.

10. (You can use other websites for this question) The Chicago Bulls realize that they've sucked since the 90s because they have relied on one start player for too long (Jordan, Rose), and that Machine learning is the way to go. But they want to do it right this time, and use machine learning to discover the best combinations of players on the court. They realize that machine learning applications commonly use GPUs. (Please limit your answer for each part to less than 50 words. Be concise!)
    a. They want to find the single GPU type across all major cloud providers (AWS, Azure, Google cloud) that has the *highest* available memory. Can you find it for them? (Note this refers to single GPU, not cloud instance)
    b. Give one advantage of using GPUs instead of CPUs for machine learning applications?
    c. Give one disadvantage of using GPUs instead of CPUs for machine learning applications?

====== END OF HOMEWORK 1 =====