CS425 Fall 2018 – Homework 1

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

Out: Aug 28, 2018. Due: Sep 25, 2018 (Start of Lecture)

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. **MCSDS (online/Coursera) students** – Please submit Word doc only! Please submit on Coursera.
4. **On-campus students**: Please submit PDF only! Please submit on Gradescope.
5. Please **start each problem on a fresh sheet (not just page)**, and **type your name at the top of each sheet**.
6. 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.**
7. Each problem has the same grade value as the others (10 points each).
8. Unless otherwise specified, 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.
9. You can discuss lecture concepts and the questions on Piazza and with your friends, but you cannot discuss solutions or ideas. All work must be your own.

Prologue: Today, almost all professional sports teams (e.g., NFL, NBA, MLB, NHL, MLS, etc.) use distributed computing to analyze games, give feedback to players (sometimes in real time), and to decide whom to trade and when.

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. The Chicago Bears NFL team would like to increase their Twitter presence to find groups of their fans who know each other. 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 – you can assume this data is already sharded in HDFS and can be loaded from there. Write a MapReduce program (Map and Reduce separately) that outputs all quadruplets (sets of 4) of users (a,b,c,d) so that every pair follows each other (i.e., if you were to draw a directed graph of "follows" relations, it would be a complete graph with 12 directed edges). Every such quadruplet must be output only once (i.e., no duplicates in the final list output by your program). You don't need to write code – pseudocode is fine as long as it is understandable. Hint: Think about the "key" in Map output. You can chain Mapreduces if you want (but only if you must, and even then, only the least number).
2. (Read the previous question) The Green Bay Packers team are a more democratic ownerless (and some say nonprofit) team. They really don't like the Chicago Bears. So they would like to find all pairs (a,b) such that both a and b follow @packers, but neither a follows b, nor b follows a. Each such pair should be output exactly once (i.e., no duplicates in the final list output by your program). Your input is the same as Question 1. 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.
3. (Read the previous two questions) But LeBron James, fresh from his move to the LA Lakers, snickers at both the above teams and says he by himself is more popular than either of those teams (it's true! @KingJames has over 41 M Twitter

followers!). King James would like to know who are the popular Twitter users who don't follow him, so he can make some new friends in LA. He would like to use Hadoop for this. The input is the same as Question 1. Write a MapReduce program (Map and Reduce separately) that outputs the list of all users u who satisfy the following three conditions simultaneously: i) u follows @KingJames, ii) u follows at least one other user v (other than @KingJames) such that v has over 10 million followers, and iii) v does NOT follow @KingJames.

4. Although the NE Patriots team did not win the last superbowl, everyone still hates them. 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 to it (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.
5. (For this question, slides discussed in lecture ought to suffice. You can feel free to refer to the original SWIM pre-paper from PODC: http://dl.acm.org/citation.cfm?id=384010 . You can also feel free to look up mathematical limit theorems online.)
   The Chicago White Sox would like to detect failures quickly in their team with N players. They plan to use a SWIM-like failure detection protocol, but with only direct pinging (i.e., no indirect pings and acks). Each process picks m processes at random, every T time units, and pings them. The rest of the protocol is similar to SWIM. (So this SWIM-like protocol essentially replaces the "ping 1 per T" in SWIM with "ping m per T", and does not use indirect pinging).

a. Is this protocol complete, i.e., is every failure detected eventually? (assuming membership lists are complete).
b. Calculate the expected detection time in terms of N, m, and T. Is it still independent of N?

6. Since the Cleveland Cavaliers parted from LeBron James, they will have an incomplete team forever. The team manager, depressed, decides to build a partial membership protocol (for a system of N processes) where each process has a membership list of size $k << N$, and $k = \Omega(\log(N))$ (the latter means you can assume that the membership graph is strongly connected). The membership list at each process is selected *uniformly at random* from across the entire group. Each message is gossiped to m randomly selected neighbors (from the membership list), where $m < k$, and $m = \Omega(\log(N))$. The manager claims 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 was sent to B neighbors. What is the value of B in terms of k, m, N? Hint: Focus on gossip target selection probabilities.
7. The French soccer team, fresh from their win at the FIFA World Cup 2018, realize that they are a truly multi-national team, and its team players have relatives all over the globe. So they build a p2p system to help players connect and chat with their relatives and friends. To design a variant of Chord that is topology-aware, they do the following: for the i-th finger table entry at node n, instead of selecting the first peer beyond $n+2^i$, select among all peers between $n+2^i$ and $n+2^{i+1}$, that peer which is closest in (network) round trip distance. Routing remains unchanged. Show that (formal proof or informal argument):
    a. Show that Memory cost is O(log(N)).
    b. Show that Lookup cost is O(log(N)) hops.
    c. In the network space (RTT or latency space), does the routing algorithm make "big" jumps in its initial hops or "small" jumps?
8. The Washington Nationals ice hockey team is overjoyed with its Stanley Cup win in 2018, and in order to have their players and fans share celebration videos with each other, they decide to use a peer to peer system. They use the Gnutella system, except that they change it so that duplicate queries are also forwarded (but with TTL decrement in place as usual). At one point of time, the Gnutella topology looks like a virtual ring with 20 processes, with each process connected to its immediate **two** clockwise neighbors and immediate **two** anticlockwise neighbors (thus each process has four neighbors). All links are bidirectional. Answer three parts:
    a. A process sends a Query message with TTL=3. How many of the processes receive this Query message? Include the sender in this count.

b. What is the minimum TTL in a Query message to ensure all nodes in the system receive the Query?
c. If we add a 21st process (the coach) who is connected to all the 20 processes, and anyone can be sender, what is the minimum TTL in a Query message to ensure all nodes in the system receive the Query?

9. In order to win back to back championships (after the century-plus drought) the Chicago Cubs realize they need to stop losing fans. They decide that they need to get their fans excited. They decide to use a Chord-like peer to peer system to connect the mobile phones of their fans with each other. The players also start using this system – after a particular game, the players' mobile phones are in a Chord ring using m = 12, nodes with the following peer ids (or node ids) join the system: 1992, 1996, 2000, 2004, 2008, 2012, 2016, 2020, 2024 (those are all years that the Presidential Elections are/will be held). Then, answer the following questions:
    a. Show or list all finger table entries for node 2016.
    b. When all finger tables and successors have converged, show the path taken by a search (or query) message originating from node 2016 intended for the key 1999.
    c. Node 2020 fails. List all the nodes whose finger tables need to be updated.
10. The St. Louis Cardinals, really unhappy with the move of the Rams NFL team away from St. Louis further West, decide that they need to be more efficient. They do a lot of data analytics using Hadoop, but the barrier in between Map and Reduce phases in Hadoop is really bogging them down. They ask you to think of ways to "break the barrier". What kinds of data structures would you need to maintain? What changes will you need to make to the way shuffle data is sent? The output of the barrier-less Hadoop should be identical to default Hadoop.