CS425 Fall 2020 – Homework 1
(a.k.a. “All the Presidential Candidates’ Interns”)


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, 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.

9. For Fall 2020 semester, we are making the following exceptions:

   a. You can discuss up to 4 (out of 8) problems with a group of at most 2 other students who are in this semester’s CS425 class. This is to encourage remote “group work” among students. (We encourage setting up Zoom sessions to discuss.)

   b. At least 4 (out of 8) problems must be on your own.
c. For each solution you write, please say at the top of the solution, either “Discussed with: Hermione Granger (netid: hgranger1), Harry Potter (netid: hpotter1),” OR say “Solved completely on my own”.
d. We encourage that with each subsequent homework, you try to monotonically increase the number of questions you solve “on your own”. If you can solve all HW4 questions on your own (last homework), that will give you confidence!

Prologue: On November 3rd 2020, the US will hold Presidential Elections. Most Presidential Campaigns today run distributed systems and cloud computing for storage and analytics of data such as the campaign, events, voters, schedules, etc., (e.g., both the Obama and Romney campaigns in 2012 built and used distributed systems, including cloud computing and mobile computing).

This homework uses fictitious stories and characters from the ongoing presidential campaigns to frame the homework problems, and to keep the questions fun and relevant (and also informative!). The choice of candidates or parties in questions, or their frequency across questions, is purely arbitrary, and there is no political message in the question framing. Any resemblance to persons, places, or events, living or dead, past, present or future, is purely coincidental. These stories and this homework are not intended to side with or against any candidate, or make comments about any candidate. They are not at supporting or endorsing, nor at criticizing or disparaging, any candidate, campaign, people in campaigns, political parties or affiliations, or voters or citizens or persons living in the US. All interns are fictional, as are their goals and actions.

Problems:

1. (For this question you can look up appropriate websites) Apart from fixed-price “on-demand” instances, cloud providers also provide preemptible instances (Google Cloud), or spot instances (AWS). An intern in candidate L’s presidential campaign is kinda used to now-on-now off characteristics, and so the intern is wondering about the differences.
   a. What is the key difference between preemptible/spot instances and the fixed-price on-demand instances?
   b. Find at least 2 major differences between Google Cloud’s preemptible instances and AWS’s spot instances. (Cite URLs if the difference you state is minor/not well-known).
c. Give an example of an application you would run with spot/preemptible instances instead of dedicated on-demand instances, and an application where you would not.

2. An intern at the election office thinks MapReduce could be useful for “instant runoff voting” in primaries. Consider an election with three candidates on the ballot – A, B, C. Every voter ranks the three candidates as first preference, second preference, and last preference. Between any two candidates X and Y, if a majority of voters ranked X above Y, then X dominates Y (and vice versa) — note that this only takes into account X and Y’s relative rankings, not where they appear in the preference order, or where the third candidate appears. A Condorcet winner is a candidate that dominates all other candidates (pair wise) on the ballot. By definition an election can have at most one Condorcet winner (however, there may be zero). You are given a dataset of votes from N voters (N is odd and large, and so dataset is sharded), where each vote V has three fields V.1, V.2, V.3, respectively for the first, second, and third preference votes of that voter.), and each line of input is one such voter’s vote V (input to initial Map function). Write a MapReduce program that outputs either the unique single Condorcet winner among the three candidates A, B, or C, or if there is no single Condorcet winner, then it outputs the list of candidate(s) with the highest Condorcet counts (those that dominate the most number of candidates). For background — in MapReduce, one writes a program for Map that processes one input line at a time and outputs zero or more (key, value) pairs; 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 can assume this data is already sharded in HDFS and can be loaded from there. Each line is one vote V and is read as the value and the key is empty (in the first by Map stage). Note that intermediate data from a Map is not available for subsequent stages! (Fun fact: if there is no Condorcet winner with 3 candidates, there is a Condorcet cycle!) For more fun, write your solution for an arbitrary number of candidates (m, not just 3) on the ballot. Correctness is important, efficiency is secondary (but you must have some parallelism). Write either pseudocode, or clear unambiguous descriptions.

3. An intern in the CDC wants to build a MapReduce program to process contact tracing data for Covid-19. 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.

4. An intern in the S Party campaign designs a SWIM/ping-based failure detection
protocol, for an asynchronous distributed system, that works as follows. Assume
there are N=M*K processes in the system (M, K are positive integers, each > 2).
Arrange these N processes in a MxK matrix, with M processes in each column
and K processes in each row. All processes maintain a full membership list,
however pinging is partial. Each process P_{ij} (in i-th row and j-th column)
periodically (every T time units) pings both: i) all the other processes in its own
row P_i*, and ii) all the processes in in its own column P_j* (once selected, these sets
don’t change). Pings are responded to by acks, just like in the SWIM protocol.
There are no indirect pings or indirect acks. A pinging process (pinger) times out
if it doesn’t receive acks by the end of the round and marks the pinged process
(pingee) as failed. This is run in an asynchronous distributed system.

a. How many failures does it take to violate completeness? That is, find the
value L so that if there are (L-1) simultaneous failures, all of them will be
detected, but if there are L simultaneous failures then not all of them may
be detected.

b. Your date claims that for K=2, this algorithm provides completeness for
all scenarios with up to (and including) 9 simultaneous failures. You
gently respond that they are wrong and that it also depends on M. What
are all the values of M (given K=2) for which your date’s claim above is
true? Justify your answer clearly.

c. Your date also claims this algorithm satisfies accuracy for R simultaneous
failures or fewer. Find the value of R (as a function of K, M, N, etc.).
5. An intern in the H Party Campaign is suspicious of election meddling. Naturally they set out to look at SWIM’s suspicion protocol. They are dealing with a datacenter of N machines where in each SWIM period 1 machine is mistakenly suspected of being failed (but is subsequently pinged correctly and rejuvenated with an incremented incarnation number). What is the “steady state” number of entries that are piggybacked atop a SWIM message, in this system? You can give your answer in O() notation.

6. In order to coordinate all the other interns in the S Party campaign, an intern decides to use a peer to peer system. The intern uses the Gnutella system. 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, apart from the sender, receive this Query message?
   b. What is the minimum TTL in a Query message to ensure all nodes in the system receive the Query (no matter the timing of the Query forwarding)?
   c. If we add a 21st process that is directly connected to all the old 20 processes, what is the minimum TTL in a Query message to ensure all nodes in the system receive the Query?

7. The E party instead structures their Gnutella system as a perfect balanced binary tree with N=2^m-1 processes. (m large). All leaves are at the same level.
   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?

8. The candidate from C Party starts playing around with a Chord DHT ring using m = 7, nodes with the following peer ids (or node ids) having joined the system in this order: 75, 14, 10, 30, 19, 15, 25, 49, 3. Anyway, after the Chord ring has converged, answer the following questions:
   a. Show or list all finger table entries for node 14.
   b. When all finger tables and successors have converged, show the path taken by a search (or query) message originating from the following querying node for the following key.
      i. Querying node 14, Key 90
      ii. Querying node 75, Key 26
   c. Node 30 fails. List all the nodes whose finger tables need to be updated.
9. Candidate L seeks the help of his friend F, who also manufactures rockets. While chilling together, they decide that when the Solar system is colonized, they would like to use BitTorrent among users of the 8 planets. While working through an example, they find a case where a file has 5 shards: A, B, C, D, E. The querying node (peer, leecher) is on a spacecraft, connected to all the planets directly. The planet servers at the 8 planets each have the following shards: Mercury (BCD), Venus (CDE), Earth (DEAB), Mars (AC), Jupiter (E), Saturn (CDE), Uranus (ACD), Neptune (DEA).

   a. 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 list them together (use tied ranking).
   b. If one introduced a 9th renegade planet called Pluto (and the spacecraft connected to it), what set of shards assigned to it will change the first fetched block by the spacecraft?

10. At the Presidential debate, the candidates decide to not talk about politics for a few minutes. The candidates from the S Party, E Party, C Party, and L, all sit down and design a new version of Kelips, called NoLips. NoLips has a large number of nodes N (N large), but only 10 (fixed) affinity groups. Otherwise, internal protocols of NoLips are similar to Kelips (gossip, affinity groups, contacts—all of it, i.e., the works!). Assuming no failures:

   a. What is the worst case lookup cost in NoLips in \( O() \)?
   b. What is the memory usage in NoLips? (\( O() \) notation)
   c. Give one advantage of NoLips over Kelips, and one advantage of Kelips over NoLips.